Two-Sided Fairness for Repeated Matchings in Two-Sided Markets: A Case Study of a Ride-Hailing Platform

Tom Sühr
Max Planck Inst. for Software Systems

Asia J. Biega*
Microsoft Research Montréal

Meike Zehlike
Max Planck Inst. for Software Systems

Krishna P. Gummadi
Max Planck Inst. for Software Systems

Abhijnan Chakraborty
Max Planck Inst. for Software Systems

ABSTRACT

Ride hailing platforms, such as Uber, Lyft, Ola or DiDi, have traditionally focused on the satisfaction of the passengers, or on boosting successful business transactions. However, recent studies provide a multitude of reasons to worry about the drivers in the ride hailing ecosystem. The concerns range from bad working conditions and worker manipulation to discrimination against minorities. With the sharing economy ecosystem growing, more and more drivers financially depend on online platforms and their algorithms to secure a living. It is pertinent to ask what a *fair distribution of income* on such platforms is and what power and means the platform has in shaping these distributions.

In this paper, we analyze job assignments of a major taxi company and observe that there is significant inequality in the driver income distribution. We propose a novel framework to think about fairness in the matching mechanisms of ride hailing platforms. Specifically, our notion of fairness relies on the idea that, spread *over time*, all drivers should receive benefits proportional to the amount of time they are active in the platform. We postulate that by not requiring every match to be fair, but rather distributing fairness over time, we can achieve better overall benefit for the drivers and the passengers. We experiment with various optimization problems and heuristics to explore the means of achieving *two-sided fairness*, and investigate their caveats and side-effects. Overall, our work takes the first step towards rethinking fairness in ride hailing platforms with an additional emphasis on the well-being of drivers.

CCS CONCEPTS

• **Social and professional topics → Economic impact**;

ACM Reference Format:
Tom Sühr, Asia J. Biega, Meike Zehlike, Krishna P. Gummadi, and Abhijnan Chakraborty. 2019. Two-Sided Fairness for Repeated Matchings in Two-Sided Markets: A Case Study of a Ride-Hailing Platform. In *The 25th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '19), August 4–8, 2019, Anchorage, AK, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3292500.3330793

*Work done while the author was at MPI-INF.

1 INTRODUCTION

Two-sided sharing economy platforms, such as Uber, Lyft, or AirBnB, have brought in disruptions in multiple business landscapes [25, 31]. There are three stakeholders in the two-sided economy: (i) *providers* of goods and services, (ii) *customers* who pay for them, and (iii) the *platform* which performs the *matching* between the providers and customers. Crucially, the platform lies at the center of the framework, mediating transactions between providers and customers, and essentially deciding on the distribution of the service provider income and the quality of customer experience. Most matching platforms today try to maximize the *utility* for the customers in line with the *customer is always right* philosophy [22]. The underlying idea is that the party more directly contributing to the revenue of the platform, such as the passengers in ride-hailing platforms, should be most satisfied.

While there have been reports and studies focusing on exploitation of providers in the on-demand and marketplace platforms [6, 14, 30], the distribution of provider income has received little attention so far. According to a 2016 report by the United States Department of Commerce, the total spending in the sharing economy was 5% of the total spending, and this ratio is predicted to grow to 50% by 2025 [29]. With increasingly many people depending on the sharing economy to earn a living, it becomes crucial to ask the question of what a *fair distribution of income* on such platforms is and what power and means the platform has in shaping these distributions. Even from the platform's perspective, ensuring fair income distribution for providers might prove beneficial to secure a long-term stable platform operation.

The problem of *fair matching* has been considered in a number of scenarios, including the problem of so-called marriage matching, or the problem of matching medical residents to hospitals [5, 21]. While very well studied in these traditional set-ups, these matching models make a variety of simplifying assumptions. Most notably, the matchings are static, performed once and often have long-term effects. In contrast, modern online matching platforms make thousands of matchings per day with repeated sets of providers and customers, often with each individual match having a limited duration. We are thus specifically interested in investigating the *two-sided fairness of providers and customers in a platform performing repeated matchings of providers and customers over time.*

The interplay between drivers and passengers has been investigated in ride-hailing platforms before [19, 23]. The main focus, however, has been on optimizing the total volume (or earnings) of the market or minimizing the distance the drivers have to cover without passengers on board. While both of these objectives are desirable, the experimental studies have shown that the resulting income distributions might be inequitable [19]. In this paper, we focus on studying whether it is possible to optimize the matchings for *equitable income distributions*.

The study we perform in this paper is grounded in yet another observation: when we focus on *minimizing income inequality*, we can *amortize* equality over longer periods of times, such as weeks or months. If the driver does not get her fair share of income on a given day, she may still make up for it on subsequent days, and still get a fair weekly or monthly income as a result. The advantage of introducing a time dimension into the notion of fairness is that the notion becomes more relaxed than the constrained requirement of being fair with respect to every matching. Similar ideas of fairness amortization have been explored in the context of ranking [3, 27].

Minimizing income inequality for the drivers, however, might have a number of side effects on other platform components like the waiting time of the passengers. However, our experiments show that the waiting time of the passengers only increases within reasonable bounds. The acceptable increase can be explained by the fact that the passenger waiting time is a function of the distance between the passenger and a matched driver, which is partially correlated with the driver's effective income. We also observe in certain cases that naively modifying the system to try to achieve fairness goals might counterintuitively lead to a decrease in the total income and equality on the platform.

In summary, this paper focuses on following research questions:

- What are the (amortized) fairness notions for equitable distribution of income in matching platforms?
- When implementing a matching algorithm with amortized fairness objectives in a ride-hailing platform, are there trade-offs between a fair distribution of income for drivers and the waiting time for the customers?
- What are the caveats and side-effects of such fair matching algorithms?

While investigating these questions, we make the following salient contributions:

- We study two notions of amortized fairness for fair distribution of income for providers in ride-hailing platforms. The first notion posits absolute income equality, while the other posits income equity, where income is proportional to the time a driver is active on the platform.
- We design matching mechanisms that minimize income inequality while distributing the resulting increases in the customer waiting time fairly between the customers.
- We perform an empirical study with data from a large Asian taxi company, exploring the effectiveness, caveats, and unexpected side-effects of the proposed fairness mechanisms.

2 BACKGROUND AND RELATED WORK

2.1 Issues in Ride Hailing Platforms

There have been growing research interests on the different components of ride-hailing platforms and their issues. From the passengers' perspective, there have been several attempts to analyze the impact of the *surge pricing* algorithm of Uber [10], and how it positively affects the demand for ride requests [15]. On the other hand, Zoepf *et al.* [32] surveyed over 1, 100 Uber drivers and found that 74% of them earn less than the minimum wage in their respective US states. Moreover, 30% of the drivers earned negative income considering vehicle expenses [32]. Chaudhari *et al.* [9] analyzed how strategic driving can help better placement of the drivers and in turn, increase their income. Allon *et al.* [1] studied how drivers make labor decisions in such on-demand services e.g., when to work and for how long, and attempted to design better financial incentives for drivers to increase their number of work hours.

There have been some works on increasing the overall efficiency of ride hailing services. Ozkan *et al.* [23] proposed strategies for better utilization of drivers, considering (i) the differing arrival rates of customers and drivers in different areas of the city, (ii) how long customers are willing to wait for driver pick-up, and (iii) the time-varying nature of all the aforementioned parameters. Jia *et al.* [19] presented an optimization framework to maximize overall profits of the drivers. However, they showed that with the number of drivers increasing, their proposed scheme encounters the congestion effect, reducing individual driver's income [19].

However, none of these works have considered the fairness associated with distributing the income opportunities among the drivers. In the position paper [7], we introduced the notions of two-sided fairness in sharing economy platforms. Following [7], in this paper, we establish fairness notions related to matchings in ride hailing platforms and propose practical implementations to achieve the same. Except a concurrent work by Bokanyi *et al.* [4], who propose mechanisms to incorporate fairness for drivers in a simulated taxi-passenger environment, to our knowledge, there is no other work to consider fairness in ride hailing platforms.

2.2 Fairness in Matching Markets

Matchings in two-sided markets such as marriage or school admission have received significant attention over the years. Assuming that the actors in both sides have preferences over actors on the other side, Gale and Shapley proposed the seminal *stable matching* algorithm [13]. Alvin Roth [24] applied the stable matching algorithm on the allocation of medical residents to different hospitals. Masarani and Gokturk [21] established four fairness axioms for the Gale-Shapley stable marriage problem, and showed that for certain preference profiles there exists no matching algorithm that can satisfy all axioms. Many recent works have also looked at stability of matchings in different contexts [12, 18].

Most of the works on matching markets have focused on *static markets* (with one-time matching scenarios). Even the works on *dynamic matching markets* [11, 17] consider the scenario where the set of actors on every side is static, but their preference ranking over the actors from the other side change over time. Some algorithms further require complete preferences to guarantee stability [11]. However, such assumptions do not hold in real-world ride hailing platforms, where the matches are for a very short duration (thus there is no equivalent notion of stability). The set of passengers changes very rapidly with new passengers making request every interval, and many of them do not come back to the platform again. While the set of drivers remains relatively similar over a long period, all of them do not remain available for a match at all time instants. Due to these differences, in this paper, we establish different notions of fairness for ride hailing platforms.

3 FAIRNESS OF REPEATED MATCHINGS

3.1 Notation

We model a ride-hailing system with two sides (i.e., two groups of users): Drivers D and Passengers (henceforth referred as Customers) C, where the platform produces a sequence of matches between drivers and customers over time. A customer requests a ride at any point in time. We group such requests coming into the system within a short period of time, thus forming *matching rounds*. Let us assume that we have a total of T such matching rounds. We use binary variables R_i^t to denote customer C_i making a ride request in round t. We can match all requesting customers in a given round t to a subset of drivers who are available at t (that is, active in the system and not busy serving another request.) We use binary variables A_j^t to denote the availability of driver D_j in round t.

We set the binary variable $M_{i,j}^t$ to 1 if driver D_i is matched to customer C_j in round t and 0 otherwise. We will reffer to a matching in a given round t as $M \in \{0,1\}^{I \times J}$, where the entry at position i, j is the binary variable $M_{i,j}^t$ for I the number of drivers and J the number of customers. Every match brings some *utility* to the matched pair. For instance, matching customer C_j to driver D_i in round t brings utility $U_D^t(i,j)$ to the driver and utility $U_C^t(i,j)$ to the customer. We aggregate these gained utilities over the considered matching rounds for every user. For example, until round t, driver D_i's accumulated utility can be expressed as $U_D^t(i)$, while customer C_j has accumulated utility $U_C^t(j)$.

In traditional matching literature, matching is done based on user preferences - defining, for each user of one group, their priority list (using relative or absolute values) over the users from the other group. Note that in our setup, such preferences can be derived from the matching utility values $U_D^t(i,j)$ and $U_C^t(i,j)$.

3.2 Modeling Utility for Both Sides

We assume that the customers want to minimize their waiting time. Assuming for simplicity a constant driving speed, we define *customer utility* using a measurable waiting time proxy, which is the reverse of the distance d to a driver:

$$U_C(i,j) = -d(D_i, C_j)$$

Similarly, we assume that the drivers want to maximize financial gains from serving a request. The drivers earn proportionately to the distance between a customer j and ride destination in round t, $\text{dest}(C_j^t)$ (assuming for simplicity a constant rate per kilometer), but lose proportionately to the distance between their location and the time of the request and the customer location (assuming for simplicity that loss happens at the same rate as the gain from performing the job.) We thus define driver utility as effective kilometers of the request:

$$U_D(i,j) = d(C_j, \text{dest}(C_j^t)) - d(D_i, C_j)$$

Note that under these utility assumptions, optimizing the utility of the customers partly optimizes the utility of the drivers.

3.3 Other Assumptions

Note that in this work we make no assumptions about request loads and distributions, thus do not use information about expected incoming requests to inform a current match. Such a setup naturally leads to algorithms that amortize heuristically.

3.4 Parity Fairness

A simple notion of fairness would be to maintain equality between the drivers, i.e., ***over time, the sum of received utilities of all drivers should be equal***:

$$U_D^T(i) = U_D^T(j) \quad \forall i, j$$

While operationalizing, such notion of fairness requires that all drivers accumulate similar utilities over the long term.

$$\sum_i \sum_j |U_D^T(i) - U_D^T(j)| < \epsilon$$

where ϵ is close to 0. Simultaneously, we can think of similar notions for the customers as well, which requires that over time, the utility gained by every customer should be similar. An important argument for equality as a notion of fairness, is the assumption of diminishing returns of income utility [2].

3.5 Proportional Fairness

While requiring driver utilities to be equal, we have not considered the amount of time different drivers are active on the platform. It may be intuitive to require higher utility for drivers who drive for more hours than other drivers. Thus, we propose another notion of fairness in ride-hailing platforms: ***over time, the accumulated utility, proportional to the active time for every driver, shold be equal***:

$$\frac{U_D^T(i)}{\Lambda_D^T(i)} = \frac{U_D^T(j)}{\Lambda_D^T(j)} \quad \forall i, j$$

where $\Lambda_D^t(i)$ is the total amount of time driver i has been active on the platform until round t.

To operationalize this definition, the matching platform should ensure that utility normalized by active time for different drivers should be similar:

$$\sum_i \sum_j |\frac{U_D^T(i)}{\Lambda_D^T(i)} - \frac{U_D^T(j)}{\Lambda_D^T(j)}| < \epsilon$$

4 INEQUALITY OF DRIVER INCOME

Having established the fairness notions we wish to study, in this section, we analyze data from a real world taxi platform to see whether the drivers earn utility equally in practice. To evaluate different matching mechanisms, we use the **Generalized Entropy Index** [26], a measure of income inequality (zero means total equality), adopted from information theory. For a constant $\alpha \neq 0$, Generalized Entropy Index is defined as

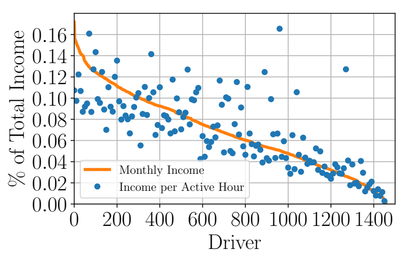
$$GE_\alpha(b_1, b_2, ..., b_n) = \frac{1}{n\alpha(\alpha-1)} \sum_{i=1}^{n} ((\frac{b_i}{\mu})^\alpha - 1) \qquad (1)$$

where $b_1, \ldots, b_n$ denote benefits/utilities and $\mu = \frac{1}{n} \sum_i^n b_i$.

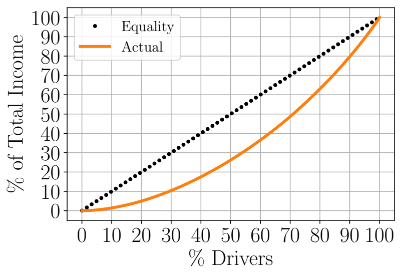
4.1 Dataset Gathered

We gathered the ride-hailing data from a mobile app-based platform operating in a major Asian city[1]. The dataset consists of a random

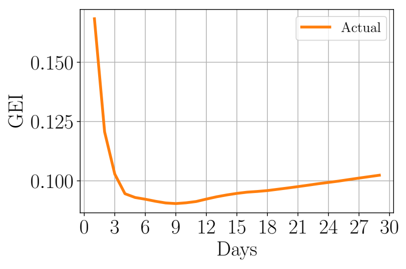
[1]Under our terms of agreement, we can not reveal the name of the company or other business details.



(a) Monthly and Hourly Income



(b) Lorenz Curve



(c) Generalized Entropy Index

Figure 1: (a) Income per Active Hour per Driver (blue), and Monthly Income per Driver (orange). (b) The Lorenz-curve shows that 50% least successful drivers only earn 28% of the available income. (c) reveals the Generalized Entropy Index – income inequality increases the longer the system runs.

sample of the jobs and positions of different drivers servicing the jobs, collected over a period of one month in 2016. The dataset consists of 1462 registered drivers and an unknown number of passengers, since every request is seen as a unique job and the passenger information is not part of the data. In total, there are 231,268 unique jobs.

4.2 Reconstructing Platform’s Algorithm

We hypothesized that the platform matches requesting customers to the nearest available taxis. We confirmed the hypothesis by comparing the positions of available drivers to the requests at a given timestamp. In all of the cases, there was no taxi that was closer to the location of the requester at request time than the taxi that was actually assigned by the platform.

4.3 Supply Exceeds Demand

In the data, the number of available drivers oscillated between 200 and 650 in different time periods. In these oscillations, we observe a day-night pattern, where less drivers are available at night, as well as a weekly pattern where less drivers are available on every seventh day. At the same time, the number of available jobs oscillated between zero and 75 at any point in time, which marks a large discrepancy between the number of available drivers and customers who want to use the service. The average capacity utilization of supply lies at about 20%. While this is a desired situation for the ride sharing platform, because customers will experience short waiting times, at the same time the available driver income constitutes a limited resource.

4.4 Resultant Inequality in Driver Utility

Figure 1b shows that after the entire period, 50% of the drivers only earned a total share of 28% of available income, while the most successful 20% of the drivers earned 40% of the income. The least successful 10% of drivers almost made no money at all, which in turn leads to a negative revenue if one takes costs of gas and depreciation into account [28].

Figure 1c shows that the longer the system runs, the more unequal income is distributed. Note that the decrease of inequality in the beginning (day 1 to day 7) is to be interpreted rather as a system warm up, than a real decline of inequality. The lowest value of the GEI in figure 1c at day 8 marks the point, where many drivers have a gain of 1 job and a few drivers still have not earned anything, and probably never will as indicated by figure 1b. This means that for our dataset and the “nearest-driver-first” policy, even at the fairest point there are drivers without any income and the situation worsens from that point on. Our experiments later reveal that this strategy is originally adopted by the platform. This situation has several problematic aspects: First, it is hard to justify the disparate amounts of revenue with differences in driving quality, as it can be (mostly) assumed that the drivers’ abilities resemble one another. Additionally a recent study [32] shows that on the biggest ride sharing platforms the least successful 30% of drivers actually loose money once their costs for gas and depreciation are included.

Second, if a driver does not earn enough they have several options: they can try to find a better placement or switch to a different ride sharing platform, either just for a while or entirely. Platforms that operate on surge prices argue that this mechanism regulates the number of drivers in a certain area as well as the number of ride requests, meaning that badly placed drivers will go to those areas in which high demands lead to high fares and hence higher income for them. Yet case studies by [15] and [10] show that surge pricing only influences passenger demand and has little effect on driver supply and hence on driver income. We did not find a connection between driver location and job supply, meaning that a driver is not likely to be able to place herself at a more lucrative location.

It is therefore likely that a driver instead of trying to find a better spot just start platform hopping or leave the system entirely for a different ride sharing company. A high supply however is crucial to customer waiting time and hence to economic success of the platform. A matching function that tries to equalize jobs across all available drivers can therefore help to keep them loyal to one particular provider and avoid platform hopping.

5 METHODS TO MATCH DRIVERS AND CUSTOMERS

In this section, we consider different strategies to match drivers and customers in ride-hailing platforms.

5.1 Nearest Driver First (NDF)

Under this strategy, each customer, making a request in a given round, chooses a driver available in the round that is closest to the customer. Ties are broken randomly. The pseudocode for this algorithm is provided in the Reproducibility Supplement A.

It is worth making the following observation in this section: Any divergence from the Nearest Driver strategy will necessarily decrease the *average* utility of the drivers in a given round under the assumed utility functions if ties are broken perfectly. The average utility of the drivers in round t is proportional to:

$$\sum_i \sum_j M_{i,j} \cdot U_D^t(i,j) = \sum_i \sum_j M_{i,j} \cdot \left(d(C_j, \text{dest}(C_j^t)) - d(D_i, C_j) \right)$$

Assuming that variables $M_{i,j}$ encode a proper matching where all requesting customers get matched to a driver, the value of the expression $\sum_i \sum_j M_{i,j} \cdot d(C_j, \text{dest}(C_j^t))$ is independent of the applied matching strategy. Its value depends only on the set of requests made by customers in a given round. Only the expression $\sum_i \sum_j -M_{i,j} \cdot d(D_i, C_j)$ depends on the matching strategy. The average utility will thus be highest when $\sum_i \sum_j -M_{i,j} \cdot d(D_i, C_j)$ is smallest, and this happens when customers are matched to their closest drivers. Note that this observation holds for a single round, but is not necessarily true when considering a multiple matching rounds. Different matchings influence the future positions of the drivers and thus future utility. It might be possible that other strategies position the drivers for very high utilities in subsequent rounds. Since we do not assume any knowledge about future request loads, we only study the accumulated utilities empirically.

5.2 Worst-Off Driver First (WDF)

While the Nearest Driver Fist strategy advantages the customers by letting them choose the matches that bring them the best utility, we also experiment with achieving equity for the drivers using a heuristic that lets the drivers who are worst-off in terms of the accumulated utility choose their preferred customers. More specifically, we define a priority list of available drivers determined by the increasing accumulated utility. Every driver chooses the customer to be matched with in the order determined by the priority list. We additionally assume each driver chooses a request that brings them the highest utility.

Interestingly, implementing this heuristic in a naive matter may lead to a number of undesirable effects. First of all, if deserving drivers are forced to choose a match in a given round, they might end up worse-off than before. It is namely possible that all the matches available in a given round bring a driver a negative utility, which is possible when the driver’s location is very far away from any of the requesting customers and and all the requested rides are very short-distance. To overcome this caveat, we match a deserving driver only if the utility of the match is positive. The pseudocode for this algorithm is provided in the Reproducibility Supplement A.

5.3 Two-Sided Optimization

We consider the distribution of driver income and passenger waiting time using an optimization mechanism where inequality levels for drivers and passengers are controlled with a single parameter. The abstract form of the formulation is as follows:

$$\begin{array}{ll} \underset{M}{\text{minimize}} & \lambda \cdot \text{inequality}_D(M) + (1-\lambda) \cdot \text{inequality}_C(M) \\ \text{subject to} & \quad\text{constraints ensuring a correct matching.} \end{array}$$

The real-valued parameter $\lambda \in [0,1]$ controls how much penalty we add to the objective for inequality of each side. As a boundary case, with $\lambda = 1$, we only explicitly minimize inequality for the drivers, while ignoring the level of inequality among the customers. With $\lambda = 0$, we only explicitly minimize inequality for the customers, while ignoring the inequality among the drivers.

Note an interesting correspondence between this formulation and the optimization problems commonly found in algorithmic fairness literature. The problem of fair matching is explicitly two-sided. Fairness in ranking [3] and recommendation [8] is often optimized under constraints on utility. While utility is often measured per task without taking into account returning consumers, utility control can also be seen as providing fairness for ranking consumers. In the following subsections we instantiate the objective and the constraints of the optimization problem.

5.3.1 *Optimization objective.*

While we might want to quantify inequality using the Generalized Entropy index, directly optimizing for this objective is, however, practically infeasible. Recall from Eq. 1 that computing GE requires computing the mean over the population. GE in a given round is thus not decomposable to factors dependent just on a single match (or pairs of matches) between a customer and a driver, which rules out integer linear or quadratic programming formulations.

Instead, we propose to approximate equality using the following optimization objective: minimize the difference of the utilities of drivers (customers) as compared to the maximum utility gained by any driver (customer) up until the previous matching round:

$$\begin{aligned} \sum_{i=1}^{I} \sum_{j=1}^{J} \lambda \cdot & \left| \max_{i'} U_D^{t-1}(i') - \left(U_D^{t-1}(i) + M_{i,j}^t \cdot U_D^t(i,j) \right) \right| \\ & + (1-\lambda) \cdot \left| \max_{j'} U_C^{t-1}(j') - \left(U_C^{t-1}(j) + M_{i,j}^t \cdot U_C^t(i,j) \right) \right| \end{aligned} \tag{2}$$

Proportional equality of the drivers can be approximated in a similar way, replacing absolute utilities by utilities normalized by the number of active rounds. Let

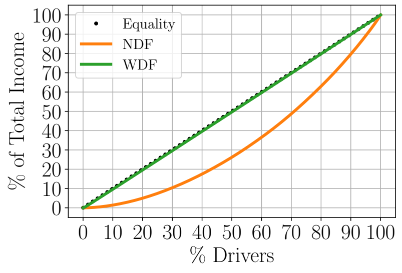
$$T_t(i) := \sum_{k=1}^{t} A_i^k$$

be the number of rounds driver D_i was active in the system up until round t. The modified optimization objective tries to equalize the normalized utilities of the drivers:

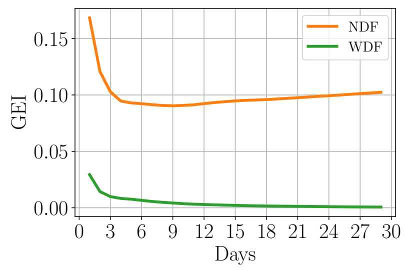
$$\begin{aligned} \sum_{i=1}^{I} \sum_{j=1}^{J} \lambda \cdot & \left| \max_{i'} \frac{U_D^{t-1}(i')}{T_{t-1}(i')} - \left(\frac{U_D^{t-1}(i)}{T_t(i)} + M_{i,j}^t \cdot \frac{U_D^t(i,j)}{T_t(i)} \right) \right| \\ & + (1-\lambda) \cdot \left| \max_{j'} U_C^{t-1}(j') - \left(U_C^{t-1}(j) + M_{i,j}^t \cdot U_C^t(i,j) \right) \right| \end{aligned} \tag{3}$$

5.3.2 *Constraints.*

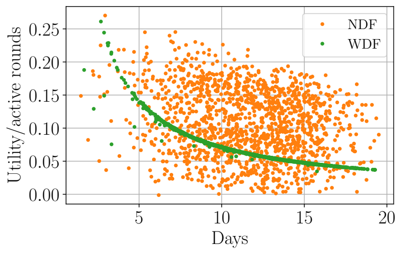
We need to introduce the following constraints to ensure the construction of a syntactically correct and feasible matching:



(a) Lorenz curve



(b) Generalized Entropy Index



(c) Driver's utility per active days.

Figure 2: Inequality distributions of total driver utility under nearest-driver-first and worst-off-driver-first policies. Each dot in (c) represents a driver with her total active time after 30 days.

Matching variables are binary: $$M_{i,j}^t \in \{0,1\},\ \forall_{i,j} \tag{4}$$

Match only and all requesting customers: $$\sum_i M_{i,j}^t = R_j^t,\ \forall_j \tag{5}$$

Match only available drivers: $$\sum_j M_{i,j}^t \leq A_i^t,\ \forall_i \tag{6}$$

Constraint 4 simply ensures that the values of the matrix M are indicators for matchings. Not all matchings are feasible, however, under the limitations imposed by the driver availability and the customer requests.

Constraint 5 enforces the matching of only and all requesting customers, and the following argument proves the correctness of this proposition. If a customer C_j makes a request in round t, $R_j^t = 1$. In this case, we require that $\sum_i M_{i,j}^t = 1$, and since $M_{i,j}^t \in \{0,1\}$, there will be exactly one index i such that $M_{i,j}^t = 1$, denoting a match between C_j and D_i. Reversely, if C_j does not make a request in round t, $R_j^t = 0$, and by a similar argument there will be no i such that $M_{i,j}^t = 1$, denoting the lack of a match between C_j and any of the drivers.

Constraint 6 ensures that only the drivers who are available in a given round are matched, and the following argument proves the correctness of this proposition. If a driver D_i is active in round t, $A_i^t = 1$. For an active driver, we require that they are matched to at most one customer in a given round, and thus there should exist at most one j, for which $M_{i,j}^t = 1$. For drivers who get matched $\sum_j M_{i,j}^t = 1$, and for drivers who do not get matched $\sum_j M_{i,j}^t = 0$, both of which can be jointly captured by $\sum_j M_{i,j}^t \leq 1 = A_i^t$. If a driver D_i is not active in round t, $A_i^t = 0$. There should exist no index j for which $M_{i,j}^t = 1$, in which case also $\sum_j M_{i,j}^t = 0 \leq A_i^t$.

5.3.3 *Casting into an Integer Linear Program.*

Under the specific conditions of the model and data we have at our disposal, it is possible to simplify this optimization problem casting it into an Integer Linear Program (ILP), and make it more efficient by reducing the number of variables M.

The first condition we leverage to simplify the formulation in practice is that, for privacy reasons, the data does not contain passenger IDs (deanonymization of such data would result in leaking potentially sensitive location information; studies in this space of similar datasets also do not have access to passenger information [19].) We thus assume each customer is a unique individual and do not amortize over customer history (all accumulated utilities $U_C^t(j) = 0$ for any customer C_j). Thus, the inequality minimization becomes: $\left|M_{i,j}^t \cdot U_C^t(i,j)\right| = \left|U_C^t(i,j)\right| \cdot M_{i,j}^t$. In each optimization round, we also only consider the customers who make requests.

On the driver side, we also restrict the optimization only to the drivers active in a given round. We can cast the optimization problem as an ILP, if we sum over the matching variables multiplied by a constant weight. We make the following observation: If we introduce dummy requests of no utility ($U_D^t(i,j) = 0$) such that in each round each driver gets matched to exactly one request (real or dummy), we can rewrite the objective for the drivers as:

$$\sum_{i=1}^{I}\sum_{j=1}^{J} \lambda \cdot \left| \max_{i'} \frac{U_D^{t-1}(i')}{T_{t-1}(i')} - \left(\frac{U_D^{t-1}(i)}{T_t(i)} + \frac{U_D^t(i,j)}{T_t(i)} \right) \right| \cdot M_{i,j}^t$$

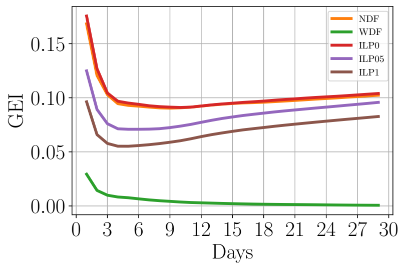
This way, combined with the previously defined constraint, we are minimizing the equality contributions from all the active drivers (contributions from inactive drivers could not be changed in a given round anyway), by having $M_{i,j} = 1$ exactly once for each driver and exactly once for each customer (real or dummy).

6 EXPERIMENTAL RESULTS

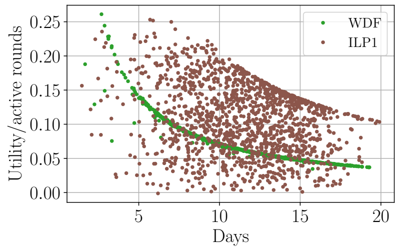
In this section we analyze the performance of the ILP with Objectives 2 and 3, using $\lambda = \{0, 0.5, 1.0\}$. We denote the methods optimizing for Objective 2 by ILP0, ILP05 and ILP1, and the methods optimizing for Objective 3 by ILPNorm0, ILPNorm05 and ILPNorm1, respectively. Results are shown in table 1.

6.1 Data

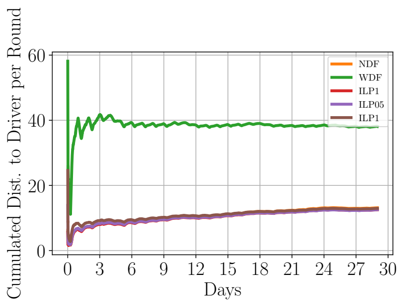
We describe the dataset and the preprocessing methods in the Reproducibility Appendix A.1. The filtered dataset consists of 28k matching rounds with a total of 1462 drivers and 231k job requests. The average duration of a job is 14.39 rounds, which corresponds to 22 minutes. The average distance from pickup location to the destination of the customer is 8.89 km. The number of active rounds



(a) Generalized Entropy for drivers over time.



(b) Driver's utility per active rounds ILP1 vs WDF.



(c) Average waiting time of passengers over time.

Figure 3: Inequality development over time of driver utility and passenger waiting time.

for the drivers is 10420.39 which translates to 11 days of work, or 29 days of 8.9 working hours.

6.2 The Worst-off Driver First Strategy

We observed in Section 5.2 that naively implementing the WDF strategy might lead to utility loss for drivers who are already worst-off. To overcome this problem, we modify this strategy to assign a job to a worst-off driver only if it leads to an increase in the driver's total utility. Figures 2a and 2b show that, with the described restriction, the WDF strategy results in equality of amortized total utility after one month. While at first glance this might look promising, the details reveal a more complex situation when utility is normalized by time drivers are active on the platform: figure 2c shows the effective hourly rate of drivers at different amounts of active periods. Under the WDF strategy longer working drivers are punished by reduced rates over time, meaning that the longer a driver stays active in the system, the less utility they earn per active round. We therefore conclude that a fair matching strategy should optimize for the driver hourly rate instead of the total amount of income, as well as take the driver's distance to the pick-up location (customer utility) into account.

6.3 ILP: Optimizing Equality of Total Income

General performance. We observe that ILP achieves higher utility for drivers and passengers simultaneously in all settings compared to NDF. Even when $\lambda = 0$, that is when we optimize only for equality of passenger utility, ILP0 outperforms NDF in terms of driver utility and passenger waiting time. This effect can be explained by the fact that the driver and passenger utilities are partially correlated. We do, however, see a slight decline in driver equality.

Interestingly, ILP05 and ILP1 outperform NDF not only in terms of driver equality, but also in terms of utility for customers. This is merely an artifact of the data in which drivers happen to become positioned geographically closer to future customers. This effect might not occur in general, and especially would not occur if the starting positions of the drivers were fixed at each request.

ILP1 achieves higher equality of driver utility (13.1% increase in the GE measure) and higher equality of passenger distance to drivers (9.545% increase in the GE measure). This result suggests that certain sequences of requests as well as driver and customer positions might not necessarily incur a trade-off between passenger utility and driver income equality.

Inequality over time. Figure 3a reveals another interesting behavior of ILP. ILP05 and ILP1 achieve the minimum inequality faster than ILP0 and NDF because, additionally the to passenger utility, the driver equality is taken into account and maximized simultaneously. Note that the high inequality measure in the first three days is due to the fact that many drivers have not been active yet and therefore received a utility of zero while other drivers were already assigned a job. After reaching a minimum, the inequality slightly increases over time for ILP0 and NDF, because the number of drivers who leave the system and are hence unavailable for the rest of the matching sequence monotonically increases, while less and less new drivers appear with zero received utility. Hence, the inequality increases because passengers are matched with a smaller set of drivers. For ILP05 and ILP1 we observe the same effects, but with a steeper increase of the inequality measure over time.

There are a number of other factors possibly contributing to this increasing trend in inequality. Figure 3c shows that both ILP05 and ILP1 distribute the increased distances to drivers in the first rounds, yielding a lower average utility for the drivers (compare this result with Figure 3a). If a driver goes offline in those initial rounds, her received utility will be low compared to ILP1 and NDF. We can, however, observe that ILP05 and ILP1 catch up to the other mechanisms in utility per round which leads to a faster increase of difference between offline and online drivers, which explains the steeper increase of inequality. The increase of inequality might however be an artifact of our dataset and its incorporated time limit. If the algorithms ran unlimited, and drivers came online again, we expect the inequality to lower again for ILP under all settings with $\lambda \neq 0$. If drivers were to come online again during a second month, the GEI would consequentially quickly converge to a lower value for ILP05 and ILP1. We denote that ILP1 and ILP05 achieve a lower total value of inequality than ILP0 or NDF.

Decreasing effective hourly rates under WDF strategy. Figure 3b shows that the variability of driver utility for WDF and ILP1 follows a chronological trend, that is, drivers who stay active for longer will receive less utility per active round. ILP1, however, shows a higher

-	NDF	WDF	WDFNorm	ILP 1.0	ILPNorm 1.0	ILP 0.5	ILPNorm 0.5	ILP 0.0
Driver Utility	1689738.428	993584.214	981251.588	1695494.19	1643690.085	1706300.168	1706726.529	1707337.862
Driver Utility vs. NDF	—	-41.199 %	-41.929 %	+0.341 %	-2.725 %	+0.980 %	+1.005 %	+1.042 %
Median Util. Driver	1650.591	1022.584	1023.937	1712.095	1162.488	1655.433	1688.046	1655.424
Median Util. Driver vs. NDF	—	-38.047 %	-37.965 %	+3.726 %	-29.571 %	+0.293 %	+2.269 %	+0.293 %
GE(2) Util. Drivers	0.137	0.001	0.042	0.119	0.369	0.134	0.133	0.139
GE(2) Util. Drivers vs. NDF	—	-99.592 %	-69.096 %	-13.137 %	+135.369 %	-2.493 %	-3.066 %	+1.611 %
GE(2) Util./Active Time Drivers	0.083	0.066	0.003	0.078	0.453	0.079	0.083	0.082
GE(2) Util./Active Time Drivers vs. NDF	—	-21.298 %	-96.878 %	-6.619 %	+444.090 %	-4.715 %	-0.784 %	-1.197 %
Passenger Dist. to Driver	365634.867	1060838.726	1073203.906	360751.646	412555.751	349941.933	349519.307	348907.974
Passenger Dist. to Driver vs. NDF	—	+190.136 %	+193.518 %	-1.336 %	+12.833 %	-4.292 %	-4.408 %	-4.575 %
Median Dist. to Driver Passengers	1.022	3.930	3.943	1.062	1.327	0.994	0.981	0.980
Median Dist. to Driver Passengers vs. NDF	—	+284.540 %	+285.812 %	+3.914 %	+29.843 %	-2.740 %	-4.012%	-4.110 %
GE(2) Dist. to Driver Passengers	0.580	0.236	0.243	0.525	0.423	0.565	0.577	0.580
GE(2) Dist. to Driver Passengers vs. NDF	—	-59.350 %	-58.106 %	-9.545 %	-27.079 %	-2.620 %	-0.479 %	-0.092 %

Table 1: Performance of all Matching-Mechanisms after 30 Days; Green values indicate a positive change vs. NDF, Red values indicate a negative change vs. NDF

variance than WDF and the values seem to be upper-bounded. These upper bounds can be explained as follows: if a driver is close to reaching the current maximum utility, she will only be assigned an additional job, if no other driver reduces the overall difference to the current maximum more. Once most of the active drivers have received high utility, ILP1 and ILP05 will match more distant passengers to them in order to keep their utility close to the current maximum.

While the WDF strategy does not show an increase of inequality within our experiments, it will always match the worst-off driver even if their resulting utility is very low. This can result in a severe decline of driver utility per active round. Figure 3b and figure 3c show these effects. We observe a strongly decreasing utility per active round for the drivers as well as very high waiting time for the passengers. On the upside, the inequality does not rise over time.

6.4 ILP: Optimizing Equality of Hourly Income

In this section we analyze the mechanisms that aim for proportional equality (Equation 3). Note that ILP0 is equal to ILPNorm0. We also analyze a modified version of the WDF mechanism in which the worst-off driver is the one with *the lowest utility per active round*. We denote this heuristic by WDFNorm.

General performance. WDFNorm equalizes the utility per active time of the drivers nearly perfectly as shown in Figure 4, even though this comes with a high loss in the overall utility volume. Table 1 shows that ILPNorm05 outperforms NDF in every computed measure. Interestingly, even in comparison to ILP0, ILPNorm05 achieves almost equally good results for the passengers. Furthermore, ILPNorm05 decreases the inequality of the total utility among drivers by 3%. However, ILPNorm1 underperforms in every measure.

Caveats of Optimizing for Normalized Utility. Figure 4 reveals that drivers with an active time of nine days or less achieve a high utility per active round through ILPNorm1, while drivers with higher active times receive close to zero utility per active round. This uncovers a challenge in our objective function which is a consequence of a too high value of $maxU_D^{t-1}$. Once a driver with

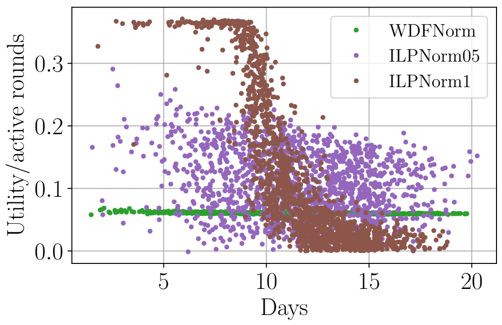


Figure 4: Driver’s utility per active rounds ILPNorm vs WDFNorm after 27840 iterations

high utility per active round goes offline, $maxU_D^{t-1}$ will stay this high for the rest of the matching sequence. If a driver i_1 with a high active time competes with a driver i_2 with low active time for a customer j, ILPNorm1 will prefer the new driver most of the times since $\frac{U_D^t(i_1,j)}{T_t(i_1)} < \frac{U_D^t(i_2,j)}{T_t(i_2)}$. In consequence the objective can be minimized easier by matching driver i_2. This effect grows over time because it is increasingly advantageous for ILPNorm1 to match customers with even more distant drivers. We observed that a possible way to bypass this problem is to modify the objective to use the maximum received utility per active round of a driver who is active and free in the current matching round (instead of the global maximum). This prevents $\max U_D^{t-1}$ to get stuck on a high and unreachable level so that drivers with longer active times have a better chance to catch up in utility.

ILPNorm05 is, however, not affected by this problem because of the passenger side of the objective. The positive results come from the longer active drivers *catching up* through matches which are decided by the passenger side of the objective. Fortunately, the driver side of the objective lowers the variance of utility per active round. Even if $\max U_D^{t-1}$ is still very high, drivers with longer active time receive higher priority in the matches compared to ILPNorm1. Figure 4 reveals the positive results of both effects: Most of the drivers receive a higher utility per active round compared to

WDFNorm and ILPNorm1 while the drivers with low active time are not advantaged anymore.

6.5 Limitations

We would like to explicitly acknowledge several limitations of the study performed in this paper. First of all, there are a number of simplifying assumptions that our model makes. We simulated driver positions after reassigning the matchings, and we assumed that each ride takes a fixed amount of time when determining driver availability. Further limitations involve the data in which, for privacy reasons, we did not have access to passenger IDs and assumed each request comes from a unique individual. In terms of efficiency, the high runtime of the optimization models would be a bottleneck when implementing a fair matching strategy in practice. Finally, it is important to acknowledge that our work does not look into behavioral aspects of fair matching, and in practice it is possible that passengers and drivers would decline assignments that violate their individual time or distance constraints.

7 CONCLUSION

This paper explored the problem of two-sided fairness for producers and consumers in a sharing economy platform. Our case study specifically focused on ride-hailing platforms and the equitable income distributions as fairness criteria for the drivers, and an equal distribution of the resulting increased waiting times for the customers. We have explored the space of various mechanisms for achieving these fairness goals. We showed in particular that an intuitive heuristic letting the worst-off drivers choose their matches first, while helping achieve equality, can lead to a decrease in the average and median utility gained by the drivers. We experimented with an optimization problem balancing equality of both consumers and producers, which led to an increase in equality while preserving the utility volumes. We also showed that implementing an optimization objective which is a direct translation of equality goals in practice still requires a very careful thought about initialization of utility values.

Our hope is that this study will add to the discussions about redesigning on-demand platforms to take the well-being of the providers into account. While such transformation would be a long-term process because of the complexity of the underlying problems (ranging from exploitation of the providers in unfair wage schemes [14] all the way to customer biases creeping in the provider ratings and influencing their job assignments [16, 20]), we believe that the question of equitable income distributions is pertinent in the constantly growing sharing economy.

The goal of implementing this new paradigm in practice opens up a number of fascinating research problems. First, to be able to perform matchings real-time, we need scalable algorithms for matching with fairness goals. Second, research into platform users’ perceptions and needs is necessary to adopt a new paradigm without major disruptions to platforms’ operations. Last but not least, more research is needed to understand the complex social impacts of the on-demand platforms.

Acknowledgments: This research was supported in part by a European Research Council (ERC) Advanced Grant for the project “Foundations for Fair Social Computing", funded under the European Union’s Horizon 2020 Framework Programme (grant agreement no. 789373). We sincerely thank Aniko Hannak for the discussions at the initial stage of this project and Sophia Rötschke, Junaid Ali and Radu Marginean for inspiering discussions and proofreading.

REFERENCES

[1] Gad Allon, Maxime Cohen, and Park Sinchaisri. 2018. The Impact of Behavioral and Economic Drivers on Gig Economy Workers. *SRN 3274628* (2018).
[2] Anthony B Atkinson. 1975. The economics of inequality. (1975).
[3] Asia J Biega, Krishna P Gummadi, and Gerhard Weikum. 2018. Equity of Attention: Amortizing Individual Fairness in Rankings. In *ACM SIGIR.*
[4] Eszter Bokanyi and Aniko Hannak. 2019. *Evaluating Algorithm Fairness in an Agent-Based Taxi Simulation.* Technical Report. Working Paper.
[5] Felix Brandt, Vincent Conitzer, Ulle Endriss, Jérôme Lang, and Ariel D Procaccia. 2016. *Handbook of computational social choice.* Cambridge University Press.
[6] Ryan Calo and Alex Rosenblat. 2017. The taking economy: Uber, information, and power. *Columbia Law Review* (2017).
[7] Abhijnan Chakraborty, Aniko Hannak, Asia J Biega, and Krishna P Gummadi. 2017. Fair Sharing for Sharing Economy Platforms. (2017).
[8] Abhijnan Chakraborty, Gourab K Patro, Niloy Ganguly, Krishna P Gummadi, and Patrick Loiseau. 2019. Equality of voice: Towards fair representation in crowdsourced top-k recommendations. In *ACM FAT**. 129–138.
[9] Harshal A Chaudhari, John W Byers, and Evimaria Terzi. 2018. Putting Data in the Driver’s Seat: Optimizing Earnings for On-Demand Ride-Hailing. In *WSDM.*
[10] Le Chen, Alan Mislove, and Christo Wilson. 2015. Peeking Beneath the Hood of Uber. In *ACM IMC.*
[11] Ettore Damiano and Ricky Lam. 2005. Stability in dynamic matching markets. *Games and Economic Behavior* 52, 1 (2005).
[12] Patricia Everaere, Maxime Morge, and Gauthier Picard. 2013. Minimal concession strategy for reaching fair, optimal and stable marriages. In *AAMAS.*
[13] David Gale and Lloyd S Shapley. 1962. College admissions and the stability of marriage. *The American Mathematical Monthly* 69, 1 (1962).
[14] Carla Green and Sam Levin. 2017. Homeless, assaulted, broke: drivers left behind as Uber promises change at the top. https://theguardian.com/us-news/2017/jun/17/uber-drivers-homeless-assault-travis-kalanick.
[15] Jonathan Hall, Cory Kendrick, and Chris Nosko. 2015. The effects of Uber’s surge pricing: A case study. *Univ. of Chicago Booth School of Business* (2015).
[16] Anikó Hannák, Claudia Wagner, David Garcia, Alan Mislove, Markus Strohmaier, and Christo Wilson. 2017. Bias in Online Freelance Marketplaces: Evidence from TaskRabbit and Fiverr. In *ACM CSCW.*
[17] Ernan Haruvy and M Utku Ünver. 2007. Equilibrium selection and the role of information in repeated matching markets. *Economics Letters* 94, 2 (2007).
[18] Kazuo Iwama, Shuichi Miyazaki, and Hiroki Yanagisawa. 2010. Approximation algorithms for the sex-equal stable marriage problem. *ACM Trans. on Algorithms* 7, 1 (2010), 2.
[19] Yongzheng Jia, Wei Xu, and Xue Liu. 2017. An optimization framework for online ride-sharing markets. In *IEEE ICDCS.*
[20] Michael Luca. 2016. Reviews, reputation, and revenue: The case of Yelp. com.
[21] F Masarani and Sadik S Gokturk. 1989. On the existence of fair matching algorithms. *Theory and Decision* 26, 3 (1989).
[22] Hughston McBain. 1944. Are customers always right. *The Rotarian* (1944).
[23] Erhun Ozkan and Amy Ward. 2017. Dynamic matching for real-time ridesharing.
[24] Alvin E Roth. 1986. On the allocation of residents to rural hospitals: a general property of two-sided matching markets. *Econometrica: Journal of the Econometric Society* (1986).
[25] Juliet Schor. 2016. DEBATING THE SHARING ECONOMY. *Journal of Self-Governance & Management Economics* 4, 3 (2016).
[26] Anthony F Shorrocks. 1980. The class of additively decomposable inequality measures. *Econometrica: Journal of the Econometric Society* (1980).
[27] Ashudeep Singh and Thorsten Joachims. 2018. Fairness of Exposure in Rankings. In *ACM SIGKDD.*
[28] Arun Sundararajan. 2014. Trusting the ‘sharing economy’to regulate itself. *The New York Times* 3 (2014).
[29] Rudy Telles Jr. 2016. Digital matching firms: A new definition in the “sharing economy” space. *ESA issue brief* 01-16 (2016).
[30] Siddharth Tiwari. 2017. ‘Underhand tactics’ boost sales in e-commerce. sundayguardianlive.com/investigation/9881-underhand-tactics-boost-sales-e-commerce.
[31] Georgios Zervas, Davide Proserpio, and John W Byers. 2017. The rise of the sharing economy: Estimating the impact of Airbnb on the hotel industry. *Journal of Marketing Research* 54, 5 (2017).
[32] Stephen M Zoepf, Stella Chen, Paa Adu, and Gonzalo Pozo. 2018. The Economics of Ride-Hailing: Driver Revenue, Expenses, and Taxes.

A SUPPLEMENT ON REPRODUCIBILITY

A.1 Data Preparation and Cleaning

A.1.1 Data Description.
We perform our experiments on a dataset from a taxi company in a large Asian city. The dataset consists of a daily *logfile* and *jobfile* over the course of twenty nine consecutive days. A daily jobfile consists of the following dimensions: *Timestamp, Taxi Id, Driver Id, Latitude of the Taxi, Longitude of the Taxi, Taxi Status.* The *Taxi Status* values *BUSY, BREAK, OFFLINE* and *POWEROFF* indicate that a taxi is currently not available and not serving a customer. A taxi with the status *FREE* is available to receive a job. Having received a job where the passenger is not at the exact same position as the taxi, is indicated by the status *ONCALL*, meaning that the taxi is on its way to the pickup-location of the passenger. The status *ARRIVED*, indicates arrival at the requested pickup-location, remaining such as long as the taxi waiting for the passenger. The status *POB* (Person on Board) means that the taxi is currently carrying a passenger. Arriving at the passenger destination will usually change the taxi status to *PAYMENT*. The taxi company creates one log every 30 seconds for every driver who is currently not assigned to a job (*ONCALL, ARRIVED, POB, PAYMENT*). For every driver who is currently assigned to job, logs are created every 90 seconds.

A.1.2 Data Cleaning.
The ride process can be described by the changing values of the *Taxi Status* field. However, in some cases the logs are incomplete or inconsistent. We observed that one of the most common status sequences related to jobs is *FREE, POB,..., FREE*, suggesting that the drivers do not report intermediate changes of their status. In order to retrieve the actual jobs from the logfile, we applied a filter to remove the entries with missing or defective data, then retrieved the first appearance of each taxi id combined with its initial position. Because of the different time intervals of the logs we chose a 90 second *matching round interval.* Every taxi that is online during at least one log within the interval is considered *active* in the corresponding matching round. In conclusion one day has 960 matching rounds.

A.1.3 Retrieving Jobs.
Even though we had the jobfile of the matchings made by the taxi company, we had to retrieve the job requests from the jobfile because there was no information about the destination of the passengers, position of the taxi during the time of the request and no information on the drop-off time in the jobfile. Mapping the jobs of the jobfile to the logs in the logfile proved difficult since the request timestamp and the time when the matching was made were not exactly the time when a status of a taxi in the logfile changed. Nevertheless we could use the jobfile to retrieve the matching algorithm of the company by comparing taxi positions at request time. Since we could only map very few jobs of the jobfile to logs in the logfile, we reconstructed the taxi jobs through the taxi status field of the logfile. Therefore we considered status sequences of taxis as valid if they represent a possible job process. We had to apply an additional filter at this point because some of the resulting jobs have a very short duration, or the necessary speed of the car is unrealistic. We filtered the jobs for a minimum ride length of 10.5 minutes and a maximum ride length of two hours and a minimum speed of the taxi of at least 20 kilometers (air-line distance) per hour but not faster than 30 kilometers per hour. We chose these values for the filter, since the resulting number of jobs are about the same as in the original jobfile.

A.2 Algorithm: Nearest Driver First

We implemented the Nearest Driver First algorithm as seen in the pseudocode 1. The methods *getFreeDriversOfRound()* returns all drivers that are free and active in a round, where all expiring matches of the rounds before are considered too and analogous *getCustomersOfRound()*. The list of customers also has to be shuffled especially if customers occur more than one time. However one has to pay attention to the following details for implementing the algorithm with another dataset. The location data of the drivers in our dataset was very precise so that for each passenger the nearest driver was unambiguous. If there exist multiple nearest drivers for one passenger, we suggest including a random choice of the driver in the method *getNearestDriver()*.

Algorithm 1: Nearest Driver First

```
INPUT: Drivers $D$, Customers $C$, Rounds $k$;
OUTPUT: List of sets of tuples $(d, c)$, $M$;
$M \leftarrow \{\}$;
$activeDrivers \leftarrow \{\}$;
$activeCustomers \leftarrow \{\}$;
for $i := 1$ to $k$ do
    $activeDrivers \leftarrow getFreeDriversOfRound(i, D, M)$;
    $activeCustomers \leftarrow getCustomersOfRound(i, C)$;
    $matching \leftarrow \{\}$;
    for $c \in activeCustomers$ do
        $d \leftarrow getNearestDriver(c, activeDrivers)$;
        $d.position \leftarrow c.destination$;
        $matching \leftarrow matching \cup (d, c)$;
    end
    $M \leftarrow M \cup matching$;
end
return $M$;
```

A.3 Algorithm: Worst-Off Driver First

As in the implementation of the Nearest Driver First algorithm, the exact implementation of Worst-Off Driver First is dependent on the accuracy of the location, more accurately, on the precision of the utility function for the drivers in order to get a distinct ranking of the drivers. Especially during the first rounds of the matching sequence where all drivers have a received utility of 0. The function *getWorstOffDriver()* has to be implemented with a random choice for all drivers with the same utility. If the utility function has possible negative values, one has to give the drivers *the choice to drive.* If not, the algorithm can be caught in a downward spiral for some drivers. This can happen if even the first preference of a driver provides negative utility and the matching would make the worst-off driver even more worse off.

A.4 Algorithm: Linear Program

Since $\max U_D^{t-1}$ and $\max U_C^{t-1}$ are not defined for $t = 0$, one has to simulate them for the very first matching. Additionaly we observed that ILP1 does not perform well if we start with $\max U_D^{t-1} = 0$ because the mechanism tries to keep the initial equality. One has

Algorithm 2: Worst-Off Driver First

```
INPUT: Drivers $D$, Customers $C$, Rounds $k$;
OUTPUT: List of sets of tuples $(d, c)$, $M$;
$M \leftarrow \{\}$;
$activeDrivers \leftarrow \{\}$;
$activeCustomers \leftarrow \{\}$;
$utilities \leftarrow \{\}$;
for $d \in D$ do
    $utilities.put(d, 0)$;
end
for $i := 1$ to $k$ do
    $activeDrivers \leftarrow getFreeDriversOfRound(i, D, M)$;
    $activeCustomers \leftarrow getCustomersOfRound(i, C)$;
    $matching \leftarrow \{\}$;
    for $j := 1$ to $activeDrivers.length$ do
        $d \leftarrow getWorstOffDriver(utilities, activeDrivers)$;

        $p \leftarrow d.getFirstPreference(activeCustomers)$;
        $d.position \leftarrow c.destination$;
        $matching \leftarrow matching \cup (d, c)$;
        $utilities.put(d, utilities.get(d) + d.getUtilityOf(c))$;
    end
    $M \leftarrow M \cup matching$;
end
return $M$;
```

to create an initial level of entropy to start the dynamics of the mechanisms. The best way to do so is to set U_D^{-1} and U_C^{-1} high for the first round only. The initial positions of the drivers match exactly the positions of the first appearing passengers because we retrieved the jobs from the logfile of the drivers. However, since we did not track the movement of the drivers between the jobs, this effect lasts only for the first matches. Additionally setting a high starting value will simulate an ongoing taxi business. The drivers who started the file on a job will be matched with them. Resulting distance to the driver of zero creates desirable optimization goals for both sides.

A.5 Technical Details

We performed our experiments on a machine with the following technical data:

- **CPU:** 2x Intel Xeon E5-2667 v2
- **Memory:** 256GB DDR3, 1866 MHz, ECC
- **OS:** 64bit Linux distribution based on Debian

We implemented all data processing infrastructure and algorithms in Java with SDK version 1.8 using the Java interface of the Gurobi Software [2] version 7.0.1 for solving the ILP.

[2]http://www.gurobi.com