FairFoody: Bringing in Fairness in Food Delivery

Anjali Gupta, Rahul Yadav, Ashish Nair,
Abhijnan Chakraborty, Sayan Ranu, Amitabha Bagchi
Department of Computer Science and Engineering
Indian Institute of Technology, Delhi
{anjali,Rahul.Yadav.cs517,Ashish.R.Nair.cs517,abhijnan,sayanranu,bagchi}@cse.iitd.ac.in

Abstract

Along with the rapid growth and rise to prominence of food delivery platforms, concerns have also risen about the terms of employment of the "gig workers" underpinning this growth. Our analysis on data derived from a real-world food delivery platform across three large cities from India show that there is significant inequality in the money delivery agents earn. In this paper, we formulate the problem of fair income distribution among agents while also ensuring timely food delivery. We establish that the problem is not only *NP-hard* but also *inapproximable in polynomial time*. We overcome this computational bottleneck through a novel matching algorithm called FAIRFOODY. Extensive experiments over real-world food delivery datasets show FAIRFOODY imparts up to 10 times improvement in equitable income distribution when compared to baseline strategies, while also ensuring minimal impact on customer experience.

Introduction

Food delivery platforms like DoorDash, Zomato, GrubHub, Swiggy and Lieferando have become popular means for people to order food online and get delivery at their doorsteps, increasingly so due to the covid-19 pandemic related restrictions (Pengonda 2021). Typically, when a customer orders food from a particular restaurant a delivery agent is assigned to pick up the food from the restaurant once ready and deliver it to the customer. Thus, apart from providing business opportunities to the restaurants, food delivery platforms also provide livelihood to thousands of delivery agents. However, recent media reports have highlighted a range of issues faced by these delivery agents: poor working conditions, long working hours, non-transparent job allocations, and meagre pay (D'Souza 2021; Anab 2021; Murthy 2021). Due to the 'gig' nature of delivery jobs, a delivery agent typically gets a small fixed commission per order (except occasional tips and other incentives), and few employment benefits. A recent survey by the non-profit Fairwork (Fairwork 2021) found that despite the gig labels associated with this work (denoting part time engagements in addition to more stable jobs), most delivery agents in developing countries actually work full time on these platforms, depending on them entirely for their livelihood (Fairwork 2021). Fairwork also found that none of the food delivery platforms in India guarantee local minimum wage to the delivery agents even if they work for more than 10 hours per day (Fairwork 2021).

Increasing the pay of delivery agents is a complicated proposition. If we decrease the number of agents so that the per-agent pay increases, there is a danger of increased customer wait time, which is anathema in the food delivery sector. Charging higher commission from restaurants or offering higher pay-per-delivery may disproportionately affect smaller restaurants and decrease their customer base. Recently such concerns led the city of Chicago to cap the delivery charge at 10% of the order value (City 2020). In some cities, restaurants are offering their own delivery services to cut back on the platform charges (Anand, Borah, and Majumdar 2021). In summary, it is not easy to increase the pool of money available to remunerate the delivery agents. However, apart from the issue of a small pool of money, the Fairwork report suggests that there is also high variability in pay – earnings on a platform can vary widely across agents (Fairwork 2021). To investigate this issue further, we perform an in-depth analysis of data obtained from a major food delivery platform for three Indian cities. Our analysis shows that there is significant inequality in the amount of money different agents earn from the platform. Interestingly, we see that the number of working hours of an agent, or her hours of operation cannot explain this inequality – it is rather the catchment area which makes a major difference.

In this paper, we present FAIRFOODY, the first algorithm to address the food delivery order allocation problem with fairness of pay distribution as a goal. We address the issue of catchment-based inequality by doing away with the restriction that a delivery agent must work within a single zone. Our algorithm, FAIRFOODY, does not, however, make any special effort to move agents across zones. In fact, in order to ensure timely delivery it limits the range from which a delivery agent can be picked to deliver a particular order. But, nonetheless, to ensure a more equitable pay distribution, FAIRFOODY ends up creating a more uniform geographical distribution of agent activity.

The challenge we faced in designing FAIRFOODY is that at any given time the number of orders may not be large enough to ensure that every idle agent is kept busy, and, hence, fairly remunerated. To deal with this issue we drew on Fairwork's finding that a number of delivery agents treat food delivery as a full-time job to relax this temporal constraint: since the orders arrive throughout the day, we *amortize fairness over a longer period of time than trying to be fair at each assignment*. In our scheme if an agent does not get her fair share of assignments in a given time period, she may still make up for it on subsequent periods, and get a fair income over a longer term. With this relaxation FAIRFOODY is able to fairly remunerate those delivery agents who spend a significant period of time on the platform, and, consequently, rely on it to provide them with a living wage.

In summary, our key contributions are as follows:

- **In-depth investigation of food-delivery data:** We perform the first in-depth study of real food delivery data from large metropolitan cities and establish that the income distribution, even when normalized by the active hours of a delivery agent, shows high levels of inequality.
- **Problem formulation and algorithm design:** We formulate the *multi-objective optimization* problem of fair income distribution in food delivery assignment, without compromising on customer experience. We show that the problem is not only *NP-hard*, but also *inapproximable* in polynomial time. To mitigate this computational bottleneck, we develop an algorithm called FAIRFOODY that uses *bipartite matching* on a data stream to perform real-time fair assignment of orders. ours is the first proposal to ensure fairness in food-delivery.
- **Empirical evaluation:** We evaluate FAIRFOODY extensively on real food delivery data across a range of metrics, and establish that it is successful in its dual objective of fair income distribution and positive customer experience.

Inequality in Delivery Agents' Income

Dataset Used

We use six days of food delivery data from three large Indian cities, provided to us by a major food delivery service provider in India. The six days span Wednesday to Monday and therefore includes both weekdays and weekends.Table 1 summarizes the dataset characteristics. The dataset consists of three components: trajectories of the delivery vehicles, the road network of each city (obtained from `OpenStreetMap.org`), and metadata describing various factors such as the vehicle IDs, information on each received order, locations of restaurants and customers, mean food preparation time in each restaurant, average speed in each road segment at different hours, etc. We match the vehicle GPS pings to the road network to obtain network-aligned trajectories (Newson and Krumm 2009).

Each of these three cities provides different flavors of customer behavior. While City B and City C are large metropolitan cities with more than 8 million residents, City A is a comparatively smaller city with 5 million inhabitants and lower order volume. Although City C has a larger number of restaurants, 41% more orders were fulfilled in City B in the time period under consideration. Furthermore, 27% more vehicles were employed in City B to cope with the higher order volume.

	City A	**City B**	**City C**
# Restaurants	2085	6777	8116
# Vehicles	2454	159160	10608
# Orders	23442	112745	112745
Food prep. time (avg.in min)	8.45	9.34	10.22
# Nodes	39k	116k	183k
# Edges	97k	299k	460K

Table 1: Summary of the dataset.

Inequality in Payment Distribution

We perform allocation using FOODMATCH (Joshi et al. 2022) on the described dataset and compute the payment earned by different delivery agents over the course of these 6 days. Fig. 1(a) shows the Lorenz curve of their income, where the y-axis represents the cumulative percentage of total income and the x-axis represents the cumulative percentage of all agents. The diagonal line in Fig. 1(a) is the equality line; the further the income distribution from this line, the higher is the inequality. We see in Fig. 1(a) that there is high inequality in the income earned by the agents; the top 10% earners get 50% of the total payment while the bottom 60% only get 10% of it. Such a high inequality can starve many agents from getting decent income and would force them to quit the platform.

What Drives Inequality?

Next, we try to uncover the sources of such high inequality.
I. Number of working hours: It is reasonable to assume that pay variability springs from variability in the number of hours worked. However this is not the case in our data. We normalized payment by the number of active hours worked: Fig. 1(b) shows the Lorenz curve of the hourly income. We can see in Fig. 1(b) that the high inequality persists even after accounting for the activity levels.
II. Hours of operation: We may conjecture that top earners are active during lunch (11AM-2PM) or dinner (7PM-11PM) times when order volumes are high, and the bottom earners are active during other periods of the day. Fig. 1(c,d,e) show the distribution of the operational periods (fraction of active times during lunch, dinner and all other time periods) for both the top 25% and bottom 25% earners. We can see that except City A, there is no noticeable difference in their activity patterns. It is not that the top earners were overwhelmingly more active during lunch and dinner times compared to the bottom earners. In fact, in City A, we can see that the bottom earners were more active during lunch times. Thus, difference in activity period is not a reasonable explanation for the inequality.
III. Geographical distribution: Next, we focus on the geographical spread of the order locations. Fig. 2 show heatmaps of restaurants' and customers' locations for the orders assigned to top earners and bottom earners, along with the location of the delivery agents when the orders were assigned to them. We can see a clear difference in the areas where top and bottom earners were active. For example, Fig. 2 shows that top earners deliver food mostly in the western part of City C, whereas the bottom earners are active

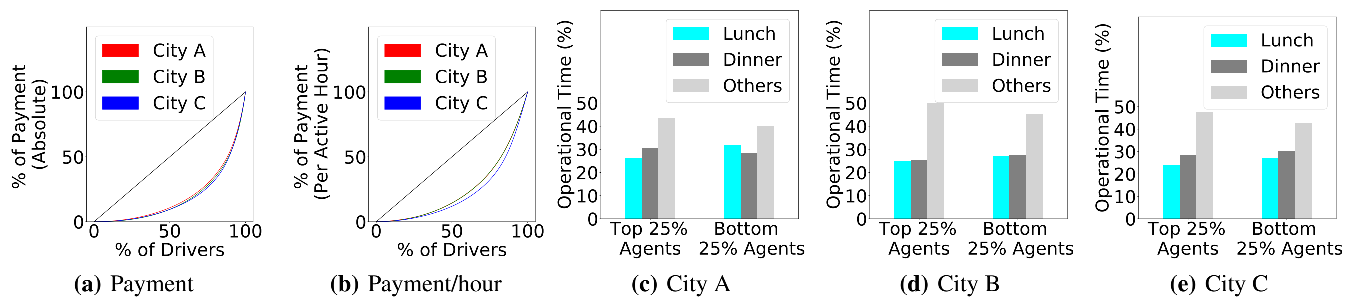

Figure 1: (a-b) Lorenz curves for the distribution of agent incomes. (c-e) Distribution of operational periods for top earners and bottom earners.

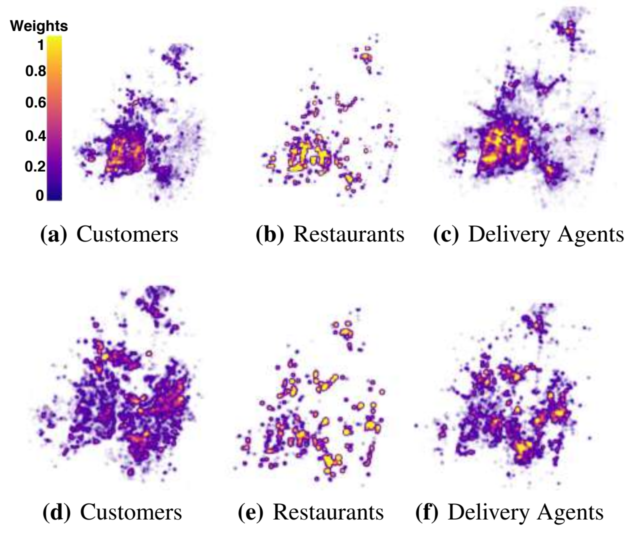

Figure 2: Heat map of order locations: the customers' locations, restaurants' locations and locations of the assigned delivery agents. In (a-c) we show this location data with respect to the orders serviced by the top 25% agents and in (d-f) those serviced by bottom 25% agents based on income.

more in the eastern part. It turns out that the order volume is higher in the western part, creating the inequality. We see similar trend in City A and City B as well.

Our finding here corroborates the delivery agents' experience as reported by Fairwork: agents often complained about not receiving orders in areas other than their chosen pick-up zone (Fairwork 2021). Even when an agent delivers food outside their zone, they are not allocated any orders on their way back, thus incurring fuel costs on their return journey, without getting any payment (Fairwork 2021). We suggest that the delivery platforms can distribute opportunities more fairly among agents by allowing them to deliver orders in different parts of the city, as long as it does not negatively impact the waiting time for customers.

Ensuring Fairness in Food Delivery

In this section, we define the concepts central to our work and formulate the problem of fair food delivery.

Fairness Notions

Since food delivery platforms essentially *distribute income opportunities* among the delivery agents, the key question is: *what would constitute a fair/just distribution?* Fairness of distribution have been studied for a long time in Moral Philosophy, particularly in Distributive Justice (Lamont 2017). Next, we discuss few key principles from distributive justice and interpret them in the context of fair food delivery.

Strict Egalitarianism: The underlying idea behind this fairness principle is that people are morally equal, and hence everyone should be treated equally (Arneson 2013). In food delivery context, this would mean that every delivery agent should earn the same income from the platform. To implement this in practice, the platform should pull together all delivery fees collected and then distribute them equally among the agents. However, such schemes are practically untenable; more so due to the gig nature of delivery jobs.

Difference Principle: In his seminal work on the theory of justice (Rawls 1971), John Rawls defined a system to be just if those affected by the system agree to be subjected to it. Rawls permit a departure from equality only if it provides *'greatest benefit to the least advantaged members of society'*. In our context, this would translate into allocating the agent with the lowest income to a new order. However, this scheme would not consider the number of hours different agents work for the platform.

Proportional Equality: Ronald Dworkin opposed the idea of complete equality and argued for eliminating inequality that happens by sheer luck, but allowing the impact of people's choice or hard work (known as 'Luck Egalitarianism') (Arneson 2018). In the food delivery context, agents' incomes should be proportional to their effort, i.e., the number of hours they are working. In this work, we consider this notion of *Proportional Equality*, and propose to ensure that an agent's income is proportional to the number of hours they work for the platform.

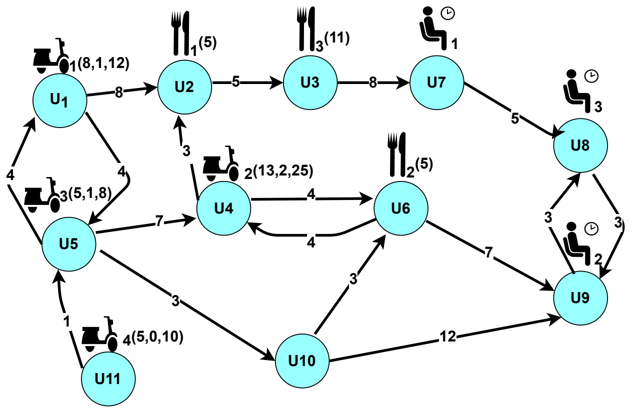

Figure 3: **A road network where the edge weights represent traversal times: vehicle icon represents delivery vehicle, cutlery icon represents restaurant and human icon represents customer location. Food preparation time at restaurant is in parentheses. Drive time, wait time and available time (dT, wT, aT) for the delivery agent is in parentheses next to the vehicle icons. The figure shows three order pick-up and drop-off points and three vehicles.**

Background: The Food Delivery Problem

Next, we formulate the problem of food delivery without any fairness consideration (Joshi et al. 2021). In general, the objective is to allocate orders to delivery agents such that the waiting time for customers is minimized.

Definition 1 (Road Network)**.** *A road network is a directed, edge-weighted graph $G = (V, E, \beta)$, where V is the set of nodes representing regions, $E = \{(u, v) : u, v \in V\}$ is the set of directed edges representing road segments connecting regions, and $\beta : (E, t) \mapsto \mathbb{R}^+$ maps each edge to a weight at time t. The edge weight at time t denotes the expected time required to traverse the corresponding road at time t.*

We use the notation $SP(u_i, u_{i+1}, t)$ to denote the length of the *shortest (quickest) path* from u_i to u_{i+1} at time t.

Definition 2 (Food Order)**.** *A food order $o = \langle o^r, o^c, o^t, o^p \rangle$ is characterized by four features. $o^r \in V$ denotes the restaurant location (pick-up node), $o^c \in V$ is the customer location (drop-off node), o^t is the time of request and o^p is the (expected) food preparation time.*

Let v be a food delivery vehicle. We use O^v_t to denote the orders assigned to v. Furthermore, $loc(v, t)$ denotes the node that is closest to v at time t. We assume all vehicles have a maximum carrying capacity of MaxO. Given a set of orders in O^v_t, a *route plan* is a *permutation* of $\{o^r_i, o^c_i : 1 \leq i \leq m\} \subset V$ such that for each i, o^r_i appears before o^c_i in the permutation. The *length* of a route plan $RP = \{u_1, \cdots, u_m\}$ is $\sum_{i=1}^{m-1} SP(u_i, u_{i+1}, t)$. The *quickest* route plan is, therefore, the one with the smallest length. We assume that all vehicles always follow the quickest route plan, and hence any reference to a route plan is implicitly assumed to be the quickest one.

Definition 3 (Order assignment)**.** *Given a set of orders O and vehicles $\mathcal{V}$, an order assignment function $A : O \to \mathcal{V}$ assigns each order $o \in O$ to a vehicle $v \in \mathcal{V}$. An order o may be assigned to v at time t only if $|O^v_t| <$ MaxO.*

Once order assignments are done, the *first-mile* distance, $firstMile(o, v)$, of order o is the distance from $v = A(o)$'s current location $loc(v, o^t)$ to the pick-up location o^r in the route followed by v. Similarly, $lastMile(o, v)$ is the *last-mile* distance from o^r to drop-off location o^c.

Example 1. *Let us consider Fig. 3. Suppose vehicle v_1, located at u_1, has been assigned to order o_1. o_1 needs to be picked up from the restaurant at u_2 and dropped-off at u_7. The quickest route for this task is $RP = \{u_1, u_2, u_3, u_7\}$. Thus, $firstMile(o_1, A) = 8$ and $lastMile(o_1, A) = 13$.*

Definition 4 (Expected Delivery Time)**.** *The expected delivery time of order o when assigned to vehicle $v = A(o)$:*

$$EDT(o, v) = \max\{time(A(o)) + firstMile(o, v), o^p\} + lastMile(o, v) \tag{1}$$

Here, $time(A(o, v))$ denotes the computation time taken by the assignment algorithm. To explain $EDT(o, v)$, the time to prepare food, and the time to assign a vehicle and reach the restaurant can progress in parallel. Thus, we take the maximum of these two components. If o^p is larger, then the driver waits at the restaurant, which is loss of productive man hours. On the other hand, if o^p is significantly smaller, then the food may get stale.

Example 2. *For simplicity, we assume $time(A(o))$ for all orders in Fig. 3. Continuing from Ex. 1, $EDT(o_1, v_1) = \max\{8, 5\} + 13 = 21$. On the other hand, if $A(o_2) = v_2$ the quickest route plan is $\{u_4, u_6, u_9\}$), and thus $EDT(o_2, v_2) = \max\{4, 5\} + 7 = 12$.*

Problem 1 (The food delivery problem (FDP))**.** *Given a set of orders O and vehicles $\mathcal{V}$, if $\mathcal{A}$ is the set of all possible assignments of O to $\mathcal{V}$, find the assignment A that minimizes the average expected delivery time.*

$$\arg\min_{A \in \mathcal{A}} \left\{ \frac{1}{|O|} \sum_{\forall o \in O} EDT(o, A(o)) \right\} \tag{2}$$

At this juncture, we highlight two practical constraints.
(1) In the real world, we work with a *data stream* of orders and vehicles instead of sets. The typical strategy to circumvent this issue is to accumulate orders over a time window Δ and assign this order set to available vehicles (Joshi et al. 2021; Reyes et al. 2018). This process is then repeated over the data stream.
(2) A food delivery service provider guarantees a *Service-level agreement (SLA)* of delivering the order within a stipulated time. This SLA is needed since food goes stale within a short time duration. Thus, it is desirable that the expected delivery time of *all* orders is within the SLA threshold Ω.

Problem Formulation: Fair Food Delivery

Problem 1 optimizes only the customer experience and does not incorporate the driver experience. As discussed earlier, a system would be fair if it equally distributes the *time-normalized income* among all delivery agents. We therefore formalize the notion of fairness for delivery vehicle.

Definition 5 (Time-Normalized Vehicle Income)**.** *Given any two time points $t_1 < t_2$ and a vehicle v, let $aT(v, t_1, t_2)$ be the time that v was available in time interval $[t_1, t_2]$, and let $dT(v, t_1, t_2)$ and $wT(v, t_1, t_2)$ be the total time spent driving and waiting at restaurant respectively by vehicle v in $[t_1, t_2]$. Then, if $aT(v, t_1, t_2) > 0$ we say that v's* time-normalized income *in $[t_1, t_2]$ is defined as:*

$$inc(v, t_1, t_2) = \frac{w_1 \cdot dT(v, t_1, t_2) + w_2 \cdot wT(v, t_1, t_2)}{aT(v, t_1, t_2)}, \quad (3)$$

where w_1, w_2 are payment parameters decided by the food delivery company.

Typically, $w_1 > w_2$. Note that $\mathrm{aT}(v, t_1, t_2) - (\mathrm{dT}(v, t_1, t_2) + \mathrm{wT}(v, t_1, t_2))$ is the time during this interval when v was either available, but had no orders assigned to it, i.e., it was idle.

Example 3. *For Vehicle v_2 in Fig. 3 ($dT = 13, wT = 2, aT = 25$) at $t = 100$ so $inc(v_2, 0, 100) = (w_1 \cdot 13 + w_2 \cdot 2)/25 = 0.584$ where $w_1 = 1$ and $w_2 = 0.8$.*

Problem 2 (Fair Income Distribution in Food Delivery)**.** *Given a set of orders O beginning at time 0 and ending at time T_m and a set of available vehicles $\mathcal{V}$, if $\mathcal{A}$ is the set of all possible assignments of O to $\mathcal{V}$, give an algorithm to find an assignment A to minimize the income gap.*

$$\arg\min_{A \in \mathcal{A}} \left\{ \max_{v \in \mathcal{V}} \{inc\,(v, 0, T_m)\} - \min_{v \in \mathcal{V}} \{inc\,(v, 0, T_m)\} \right\} \quad (4)$$

Optimizing Problem 2 may lead to an increase in delivery time, and hence hamper customer's experience. We therefore aim to minimize Problem 2 under *bounded* increase in delivery time. Formally,

Problem 3 (The Fair Food Delivery Problem)**.** *Find an assignment A minimizing Problem 2 under the constraint:*

$$Dist(A(o), o) \leq \Gamma \times nearDist_o, \forall o \in O \quad (5)$$

where $Dist(o, A(o)$ is the road network distance (shortest path) of order o from its assigned vehicle $v = A(o)$, $nearDist_o$ is the distance to the nearest vehicle from order o and $\Gamma > 1$ is a threshold.

Theoretical Characterization

Theorem 1. *There is no PTIME algorithm that can approximate Prob. 2 within any constant factor c, where $0 < c \leq 1$, unless $P = NP$.*

Proof. We prove it using a reduction from the *subset sum* problem, which is a known NP-hard problem. Here, we show that if there exists a c-approximation algorithm for Prob. 2, then we can separate the YES instance from the NO instances of subset sum in PTIME.

Definition 6 (Subset Sum Problem)**.** *Given a multi-set of items $I = \{a_1, \cdots, a_m\}$ and a target sum B, the subset sum is a decision problem that seeks to answer whether there exists a subset $I_1 \subseteq I$, such that $\sum_{\forall a \in I_1} a = B$. We denote an instance of a subset sum problem with the notation $\mathcal{S} = \langle I, B \rangle$. $\mathcal{S}$ is a YES instance if the constraint is satisfied.*

Let us consider the instance $\mathcal{S} = \langle I, B \rangle$ where $\forall a_i \in I,\ a_i \geq 1$ and $B = \sum_{\forall a_i \in I} \frac{a_i}{2}$. Clearly the problem is still NP-hard. Given $\mathcal{S}$, we first construct a road network as follows. We first create a source node S. For each $a_i \in I$, we create a node n_i. From S we have a directed edge to each n_i and vice versa. We assume that there is an order corresponding to each n_i, such that the restaurant is located at S with 0 waiting time for all orders, and customer's location is n_i. The income from serving this request is assumed to be a_i. Thus, the maximum income that can be earned is $2 \times B$.

We assume there are two vehicles and the maximum carrying capacity is $\mathrm{MaxO} = B$. Let m^* be the income gap of the optimal solution of Prob. 2 on the constructed instance. We first show the following claim holds.

Lemma 1. *If $\mathcal{S}$ is an YES instance of the subset sum problem, then $m^* = 0$.*

Proof. If $\mathcal{S}$ is a YES instance, then there exists a partition of the set I into I_1 and I_2 such that $\sum_{i \in I_1} a_i = \sum_{i \in I_2} a_i = B$. Thus, if we recommend the paths corresponding to elements of I_1 and I_2 to each of the two vehicles, then both earn an income of B, and hence, $m^* = 0$. ☐

Lemma 2. *If $\mathcal{S}$ is a NO instance of the subset sum problem, then $m^* > 0$.*

Proof. If $\mathcal{S}$ is a NO instance, then clearly one of the two vehicles will earn less than B and hence $m^* > 0$. ☐

This proves that Prob. 2 is NP-hard. Now, let us assume there exists a PTIME c-approximation., i.e., $m^* \leq m \leq cm^*$, where m is the output of the approximation algorithm and $c \geq 1$. We know $m^* > 0$ for NO instances. Since $m* \leq m$, $m > 0$. On the other hand, for YES instances, $m \leq cm^* \leq 0$ as $m^* = 0$. Thus, the boundaries of m for YES and NO instances will not overlap, and hence we will be able to separate those instances. This cannot be true unless $P = NP$. ☐

Corollary 1. *There exists no PTIME algorithm that can approximate Prob. 3 within any constant factor c, where $0 < c \leq 1$, unless $P = NP$.*

PROOF. Any instance of Prob. 2 reduces to an instance of Prob. 3 for $\Gamma > diameter \times min\_distance$, where $diameter$ indicates the longest shortest path in the road network and $min\_distance$ is the minimum distance between any two points in the network.

Our Proposal: FAIRFOODY

We propose a heuristic algorithm FAIRFOODY to solve Problem 3. It builds a *weighted bipartite* graph with vehicles in one partition and *clusters* of orders in the other. The weights of the edges are computed such that finding a *minimum weight matching* in this bipartite graph optimizes the criterion of Prob. 2, while also ensuring a good solution with respect to Prob. 1. We find the minimum weight matching by running the *Kuhn-Munkres* algorithm (Kuhn 1955; Munkres 1957) on the graph. We now elaborate on the key steps.

Definition 7 (Shortest Delivery Time (Joshi et al. 2021)). *The* shortest delivery time *for an order o is $SDT(o) = o^p + SP(o^r, o^c, o^t)$.*

$SDT(o)$ is a natural lower bound on EDT (Eq. 1).

Example 4. *For order o_1 in Fig. 3, food preparation time is* 5 *and travel time from $restaurant_1$ at u_3 to $customer_1$ at u_7 is $(5 + 8) = 13$, so $SDT(o_1) = 5 + 13 = 18$.*

Definition 8 (Augmented Order Delivery Time). *Consider a vehicle v at time t. Suppose at this time we add a cluster of orders O to v's route plan and v is able to deliver all orders in this augmented route plan by time $t' > t$, on the assumption that no new orders are added to v's route plan in time interval (t, t'). Then we define the Augmented Order Delivery Time $AODT(O, v, t)$ as $t' - t$.*

Example 5. *If order o_3 in Fig. 3 assigned to vehicle v_1 (at location u_1) at time $t = 100$ and v_1 was already assigned o_1, then v_1's route plan will now be $RP = \{u_1, u_2, u_3, u_7, u_8\}$. v_1 will take $(8 + 5 + 8 + 5) = 26$ time to deliver o_3 and it will reach u_8 at time $t' = 100 + 26 = 126$ so $AODT(o_3, v_1, 100) = t' - t = 126 - 100 = 26$.*

Definition 9 (Augmented Order Payment). *Consider a vehicle v at time t which may be currently idle or currently assigned some order. Suppose at this time we add a cluster of orders O to v's route plan (which may be empty if the vehicle is idle) and v is able to deliver all orders in this augmented route plan at time $t' > t$ such that the total driving time in the interval $[t, t']$ is t_1 and the total waiting time is t_2. Then, on the assumption that no new orders are added to v's route plan in time interval $[t, t']$, we define the* Augmented Order Payment *$AOP(O, v, t)$ as $w_1 \cdot t_1 + w_2 \cdot t_2$.*

Example 6. *Suppose at $t = 100$ order o_1 in Fig. 3 was already assigned to v_2 located at u_4. At this point, v_2's route plan is $\{u_4, u_2, u_3, u_7\}$. Suppose at this time, o_3 is assigned to v_2. Now, v_2 will follow route plan $RP = \{u_4, u_2, u_3, u_7, u_8\}$. v_2 will travel 3 time units to reach $restaurant_1$ located at u_2, then it will wait 2 time units till the order is ready. At $t = 105$, it will leave u_2 and travel 5 time units to u_3 where it will wait 1 time unit for o_3 to be prepared. Finally, v_2 will set out from u_3 at time $t = 111$ and, after delivering o_1 and u_7 at time $t = 119$, finish its augmented route plan by delivering o_3 at u_8 at time $t = 124$. The total time spent is therefore $124 - 100 = 24$, of which $2 + 1 = 3$ time units were spent waiting. So $AOP(o_3, v_2, 100) = 21 \cdot w_1 + 3 \cdot w_2 =$ 23.4 where $w_1 = 1$ and $w_2 = 0.8$. Recall, $AODT(o_3, v_2, 100) = 24$ (Ex. 5).*

As discussed in our problem formulation (Prob 1), we partition the data stream of orders into windows of length Δ, and allocate all orders that arrived within this window. This process is then repeated for each of the subsequent windows.

Definition 10 (Next-slot Normalized Income). *At time $\ell\Delta$, i.e. at the beginning of the $\ell + 1$st window, consider a vehicle v that is available. In addition, consider an unassigned order o. We define the next-slot normalized income of v if it is assigned o as*

$$\textit{ns-inc}(v, o, \ell) = \frac{\textit{inc}(v, \ell\Delta) \cdot aT(0, \ell\Delta) + AOP(o, v, \ell\Delta)}{\ell\Delta + AODT(o, v, \ell\Delta)}. \tag{6}$$

Example 7. *Continuing from Ex. 6 for Vehicle v_2 in Fig. 3 ($dT = 13, wT = 2, aT = 25$) at $t = 100$, $AODT(o_3, v_2, 100) = 24$, $AOP(o_3, v_2, 100) = 23.4$ and $\textit{inc}(v_2, 0, 100) = 0.584$ (see Ex. 3). So $\textit{ns-inc}(v_2, o_3, 100) = \frac{\textit{inc}(v_2,0,100)\cdot 25 + AOP(o_3,v_2,100)}{25 + AODT(o,v,100)} = (0.584 \cdot 25 + 23.4)/(25 + 24) = 0.76$.*

Creating a weighted bipartite graph: At time $\ell\Delta$, let us assume that $\mathcal{V}_\ell$ is the set of available vehicles and O_ℓ is the set of unallocated orders. We create a weight bipartite graph (U_1, U_2, E) where $U_1 = \mathcal{V}_\ell$, i.e., one side of the partition is the set of available vehicles and U_2 is a *cluster* of orders.

Clustering orders: If $|O_\ell| \leq f \cdot |\mathcal{V}_\ell|$ where f is a parameter chosen in $(0, 1)$, we set $U_2 = O_\ell$. Otherwise we perform a clustering on O_ℓ using Ward's method, i.e., we successively coalesce those two clusters whose being delivered by a single vehicle leads to the least increase in the extra delivery time. The cluster size is not allowed to cross MaxO. We stop either when the number of clusters falls to $f \cdot |\mathcal{V}_\ell|$ or when the increase in expected delivery time due to clustering crosses a threshold η. More simply, we stop when either further clustering significantly compromises customer experience or there are too few clusters to fairly allocate to all available vehicles. We denote the final clusters as O'_ℓ.

Edge weights in bipartite graph: Each edge of the bipartite graph is of the form (v, O) where $O \subset O_\ell$ is either a single order (a singleton set) or a cluster of orders. If vehicle v is at a distance not exceeding $\Gamma \times nearDist_O$ (Recall Prob. 3), we set the weight as follows:

$$w(v, O) = \textit{ns-inc}(v, O, \ell) - \min_{v \in \mathcal{V}} \textit{inc}(v, \ell\Delta). \tag{7}$$

To identify all vehicles whose distance not exceeding $\Gamma \times nearDist_O$, in an efficient manner, we perform *best-first search* on the road network graph (See Alg. 1). Specifically, we start from the restaurant locations of each order in O and visit all the nearby vehicles in the best first search order till the distance from source restaurant exceeds $\Gamma \times nearDist_O$. All vehicles beyond this boundary are assigned edge weight $\approx \infty$.

Finally, we run *Kuhn-Munkres* on the bipartite graph to obtain the allocation for the $\ell + 1$st window.

Algorithm

Algorithm 1 represents the pseudocode for bipartite graph construction using best first search approach. For a given

Algorithm 1: Bipartite graph construction

Input: Available vehicles $\mathcal{V}_\ell$, order batch O'_ℓ, parameter Γ, current time t, road network $G(V, E, \beta)$

Output: Bipartite Graph

```
1: Initialize bipartite graph $B$ with node sets $\mathcal{V}_\ell$ and $O'_\ell$, and empty edge set $E_b$.
2: for each $o' \in O'_\ell$ do
3:     $source \leftarrow loc(o', t)$
4:     $PQ \leftarrow$ Empty Priority Queue
5:     $PQ.insert(\langle source, 0 \rangle)$
6:     Initialize $\forall u \in V,\ visited(u) \leftarrow false$
7:     $\mathcal{V}_{o'} \leftarrow \emptyset$
8:     Set $found\_nearest\_v = false$
9:     Set $nearDist_{o'} = 0$
10:    while $PQ.empty() = false$ do
11:        $\langle u, \delta \rangle \leftarrow PQ.pop()$
12:        if ($found\_nearest\_v = true\ \ \&\ \ \delta >= \Gamma \times nearDist_{o'}$) then
13:            Break
14:        if $visited(u) = true$ then continue
15:        $visited(u) \leftarrow true$
16:        $I(u) \leftarrow \{v \in \mathcal{V}_\ell \mid loc(v, t) = u\}$
17:        if ($I(u) \neq NULL\ \ \&\ \ found\_nearest\_v = false$) then
18:            Set $nearDist_{o'} = \delta$
19:            Set $found\_nearest\_v = true$
20:        for each $v \in I(u)$ do
21:            Add edge from $v$ to $o'$ in $E_b$ with edge weight given by Eq. 7.
22:        end for
23:        $\mathcal{V}_{o'} \leftarrow \mathcal{V}_{o'} \cup I(u)$
24:        $N(u) \leftarrow (u' \mid (u, u') \in E,\ visited(u) = False,)$
25:        $\forall u' \in N(u),\ PQ.insert(\langle u', \delta + \beta(e = (u, u'), t) \rangle)$
26:    $\forall v \in \mathcal{V}_\ell \setminus \mathcal{V}_{o'}$, add edge from $v$ to $o'$ in $E_b$ with $\Omega$ edge weight.
27: end for
28: return bipartite graph $B(\mathcal{V}_\ell, O'_\ell, E_b)$
```

batch of orders $o' \in O'_\ell$ we extract its restaurant location of first unpicked order $source \leftarrow loc(o', t)$ (line 3) and from source we initiate best first search (lines 10-22). A priority queue PQ is used to store candidate nodes to be visited and is initialized with source (line 4-5). PQ stores a tuple $\langle u, \delta_u \rangle$ where δ_u is $SP(source, u, t)$, the shortest path from $source$ to u, and retrieves nodes in ascending of δ_u. Once top node from PQ is popped (line 11), we proceed further if it's distance does not exceed $\Gamma \times nearDist_{o'}$(line 12-13) and the node is not yet visited (line 14). Next, we mark u as visited (line 15). We take all vehicles that are at u (line 16). We set nearest vehicle distance $nearDist_{o'}$ to δ if nearest vehicle is not yet found (lines 17-19). We add an edge from o' to all vehicles available at u in edge set E_b with edge weight given by Eq. 7 (lines 20-22). After this, we insert all neighbors u' of u to PQ with $\delta' = \delta + \beta(e = (u, u'), t)$ where $\beta(e = (u, u'), t)$ is edge weight of $e = (u, u')$, i.e, average time taken in road network at time t (lines 24-25). Once PQ is empty, we add an edge from o' to all remaining vehicles with weight Ω ($\approx \infty$)(line 26). This whole process is repeated for all batches of orders $o' \in O'_\ell$ (line 2).

Theorem 2. *The time complexity of the proposed algorithm is $\mathcal{O}(m \cdot n(q + \max(m, n)))$, where $m = |\mathcal{V}_\ell|$, $n = |O'_\ell|$, $\mathcal{O}(q)$ is the time taken for shortest path computation.*

Proof. To create an optimal route plan, we need all permutations of at most 2MaxO locations with the constraint that the customer location for an order must come after the restaurant for that order. This is the same as the ways of arranging MaxO pairs of parentheses with the twist that in our cases the parentheses are distinguishable. So this number is $G(\text{MaxO}) = (2\text{MaxO})!/((\text{MaxO}+1)(\text{MaxO})!)$. For each permutation we need to find the shortest path between subsequent nodes in the order. Hence the time taken to find the route plan is $\mathcal{O}(G(\text{MaxO}) \cdot \text{MaxO} \cdot q)$ where $\mathcal{O}(q)$ is the time taken for shortest path computation. We compute $w(v, o')$ for each vehicle-order pair which takes a total of $\mathcal{O}\left((G(\text{MaxO}) \cdot \text{MaxO} \cdot q \cdot m \cdot n\right)$ time where $m = |\mathcal{V}_\ell|$ and $n = |O'_\ell|$. Let $k_\top = \max(n, m)$ and $k_\perp = \min(n, m)$. Kuhn-Munkres algorithm take $\mathcal{O}(k_\top^2 k_\perp)$ time to compute a maximum weighted matching. Hence an overall complexity is $\mathcal{O}((G(\text{MaxO}) \cdot \text{MaxO} \cdot q \cdot m \cdot n + k_\top^2 k_\perp) = \mathcal{O}(G(\text{MaxO}) \cdot \text{MaxO} \cdot q \cdot m \cdot n + m \cdot n \cdot \max(m, n))$.

Typically, MaxO is 2 or 3, and hence MaxO $\ll m, n, q$. Thus, we drop this term, which reduces the complexity to $\mathcal{O}(m \cdot n(q + \max(m, n)))$.

□

Experimental Evaluation

In this section, we benchmark FAIRFOODY and establish:

- **Fairness:** FAIRFOODY imparts more than $10X$ improvement in fairness over FOODMATCH and baseline approaches adopted from the cab service industry.
- **Cost of fairness:** FAIRFOODY maintains a comparable delivery time as that of FOODMATCH (Joshi et al. 2021).
- **Scalability:** FAIRFOODY is scalable to real-world workloads in large metropolitan cities.

Evaluation Framework

All implementations are in C++. Our experiments are performed on a machine with Intel(R) Xeon(R) CPU @ 2.10GHz with 252GB RAM on Ubuntu 18.04.3 LTS. Our codebase is available at https://github.com/idea-iitd/fairfoody.git.

Baselines:

• **FOODMATCH:** FOODMATCH squarely focuses on minimizing the delivery time. Hence, it is agnostic to driver incomes. A comparison with FOODMATCH reveals **(1)** the unfairness in the system when we optimize only delivery time, and **(2)** the impact on customer experience in terms of delivery time if fairness is included as an additional objective in the optimization function.

•**2SF:** 2SF is designed to ensure two-sided fairness in the cab-hailing industry. The two sides correspond to cab drivers and customers. Towards that end, 2SF optimizes a weighted combination of driver's income and the wait-time faced by customers (delivery time in the context of food delivery). We use λ to denote the weightage given to drivers income; $1 - \lambda$ corresponds to weightage of waiting time for customers. Although there are similarities between the cab and food delivery industry, there are several subtle differences that necessitates the need for a specialized algorithm for food delivery. For example, sending the nearest cab driver minimizes the waiting time of a customer, whereas that is not the case in

food delivery due to the intermediate operation of picking up food from the restaurant. This comparison allows us to precisely understand the impact of a food-delivery specific fairness algorithm.
We use *hierarchical hub labeling* (Delling et al. 2014) to index shortest paths queries in all benchmarked algorithms.

Metrics: The performance is quantified through:

• **Gini coefficient:** The Gini coefficient is the ratio of the area that lies between the line of equality and the Lorenz curve over the total area under the line of equality (Gastwirth 1972). Mathematically,

$$Gini = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n}|x_i - x_j|}{2n\sum_{j=1}^{n} x_j} \tag{8}$$

where x_i is the income per hour of driver i and n is the number of drivers. A lower Gini indicates fairer distribution.

• **DTPO:** DTPO measures the average delivery time per order. DTPO allows us to quantify the cost of fairness.

• **Percentage of SLA violations (SLA-V):** We measure the percentage of orders not delivered within the promised time limit (Prob. 1). The time limit is set to 45 minutes.

• **Spatial distribution distance (ψ):** we examine the spatial distribution of top-25% and bottom-25% drivers in terms of income in the form of a *heatmap*. We next formulate a metric to quantify the difference between these distributions. Specifically, we partition a city into a grid and compute three normalized distributions over the cells of the grid for the top-25% and bottom-25% drivers: **(1)** $\mathcal{L}$**:** their locations when an order was allocated to them, **(2)** $\mathcal{R}$**:** the restaurant location of allocated orders, and **(3)** $\mathcal{C}$**:** the customer drop-off locations. If there are n grid cells, then they give rise to an n-dimensional vector with non-negative entries that sum to 1 for each property. We denote by α_P and β_P the distributions of top-25% and bottom-25% drivers respectively on property P. Since α_P and β_P vectors have the property of probability distributions we compute the distance between them using the *Total Variation Distance* (c.f. (Levin, Peres, and Wilmer 2017)), which is guaranteed to be between 0 and 1, i.e.,

$$\psi_P = \frac{1}{2}\sum_{i=1}^{n}|\alpha_P[i] - \beta_P[i]| \tag{9}$$

City	Algorithm	Gini	Income Gap	DTPO	SLA-V (%)
A	FAIRFOODY	**0.035**	**20.5**	15.4	0.33
	2SF, $\lambda = 1$	0.32	58	17.3	0.37
	2SF, $\lambda = 0$	0.526	55.9	**15.2**	0.33
	FOODMATCH	0.518	59.2	**15.2**	**0.32**
B	FAIRFOODY	**0.047**	**30.4**	15.5	**0.22**
	2SF $\lambda = 1$	0.316	59.9	15.5	23.12
	2SF, $\lambda = 0$	0.471	59.4	**14.5**	19.62
	FOODMATCH	0.512	59.3	15.4	0.23
C	FAIRFOODY	**0.035**	**33.5**	16.2	0.33
	2SF, $\lambda = 1$	0.323	58.3	16.9	12.05
	2SF, $\lambda = 0$	0.513	56.1	**15.8**	6.63
	FOODMATCH	0.562	59.1	16.0	**0.32**

Table 2: Performance of FAIRFOODY, FOODMATCH, and 2SF across various metrics. The unit of DTPO is in minutes. The best performance in each metric is highlighted in bold.

City	Algorithm	25 ‰	50 ‰	75 ‰	95 ‰
A	FAIRFOODY	38	51.5	70	88
	FOODMATCH	13	29	65	205.3
	2SF $\lambda = 1$	20	44	80	142
B	FAIRFOODY	31	61	92	115
	FOODMATCH	12	33	75	264
	2SF $\lambda = 1$	23	53	90	156
C	FAIRFOODY	44	64	79	95
	FOODMATCH	13	31	74	250
	2SF $\lambda = 1$	28	52	87	154

Table 3: Number of orders across all six days per driver at various percentiles(‰).

Operational Constraints: Several constraints are required while operating food-delivery service. We use the same constraints adopted by(Joshi et al. 2021). An order is *rejected* if it remains unallocated for 45 minutes. The rejection penalty Ω (Recall Problem 1) is set to 7200 seconds since most orders are delivered in motorbikes.

Simulation Framework Our dataset contains the exact position of all orders, restaurant and delivery agents. The simulation environment to evaluate the impact of each allocation algorithm, therefore, only involves the order allocation mechanism. Order allocation is a deterministic procedure and hence we neither need to repeat the experiment multiple times, nor report the variance. Our reported results measure the performance on the various described metrics across all 6 days of data.

Estimation of road network speeds: We divide 24-hour period into 24 one hour slots and then for each slot we compute the weight of each road network edge as the average travel time across all of delivery vehicles in the corresponding road in that slot.

Parameters: The default size of accumulation window Δ is 3 minutes. Clustering parameter f, and payment weights w_1 and w_2 (Def. 5) are set to 0.8, 1.0 and 0.8 respectively as their default values.

Comparison with FOODMATCH and 2SF

Table 2 presents the performance of various algorithms across all three cities. In Fig. 1(b), we observed that FOODMATCH induces significant income disparity among drivers. This is reflected in the high Gini of FOODMATCH across all cities in Table 2. In contrast, FAIRFOODY reduces Gini more than 10 times across three cities. This reduction in Gini, however, does not come at the cost delivery time or SLA violations (SLA-V). Specifically, there is minimal increase in DTPO and SLA-V. Similar to Gini, FAIRFOODY is also significantly better in Income Gap. Overall, this shows that it is possible to ensure fairness without compromising on the customer experience.

We also compare with 2SF at $\lambda = 1$ and $\lambda = 0$. We choose these two λ values since they represent the two extremes; at $\lambda = 1$, 2SF optimizes only the driver income gap, whereas

$\lambda = 0$ minimizes only the delivery time. Thus, $\lambda = 1$ represents the best possible Gini by 2SF. We observe that even in this scenario, FAIRFOODY is 9 times better on average. At $\lambda = 0$, although 2SF achieves low delivery times, it fails to satisfy SLA across a large portion of orders.

In Table 3, we further examine the number of orders delivered by drivers across various percentiles based on normalized income (Def. 5). It is clear from the data that FAIRFOODY achieves the most equitable distribution.

Impact on Spatial Distribution

In Fig. 2, we showed that spatial distribution is a key driver of payment inequality. Now we will show that FAIRFOODY *equalizes the spatial distributions across the pay range*. Fig. 4 studies this question in City C. As visible, the heatmaps of the top-25% and bottom-25% are much more similar when compared to Fig. 2. This indicates that disparity in spatial distribution is correlated to income disparity.

To quantify this observation and examine whether the pattern holds across all cities, we compute ψ_P across all combinations of cities and properties (Recall Eq. 9). Table 4 presents the results. Two key observations emerge from this experiments. First, both for FAIRFOODY and 2SF, $\lambda = 1$, the ψ is lower across all cities and properties. This indicates that spatial distribution distance and Gini (as well as Income Gap) are indeed correlated. However, just minimizing spatial distribution distance is not enough in minimizing Gini. Specifically, although 2SF, $\lambda = 1$ has a lower spatial distance than FAIRFOODY in City B, its Gini is significantly higher. (Table 2).

Scalability

In food delivery, since we partition the data stream into windows of length Δ, it is imperative that the allocation time of all orders within this window is smaller than Δ. Otherwise, a queue will build up. We call a window "overflown" if the allocation time exceeds Δ. Table 5, presents the percentage of overflown windows as well as average allocation time across all windows. As visible, all algorithms have 0 overflown windows and hence are efficient enough for real-world workloads. Nonetheless, FAIRFOODY has the smallest allocation time among all algorithms.

City	Locations of the	FOOD-MATCH	2SF $\lambda=0$	2SF $\lambda=1$	FAIR-FOODY
A	Vehicle	0.543	0.500	0.293	**0.231**
	Customer	0.490	0.437	0.268	**0.235**
	Restaurant	0.469	0.442	**0.169**	0.278
B	Vehicle	0.394	0.394	**0.169**	0.191
	Customer	0.348	0.312	**0.152**	0.182
	Restaurant	0.354	0.342	**0.135**	0.174
C	Vehicle	0.386	0.403	0.243	**0.193**
	Customer	0.318	0.310	0.217	**0.185**
	Restaurant	0.318	0.382	**0.206**	0.239

Table 4: Comparison of spatial distribution ψ_P.

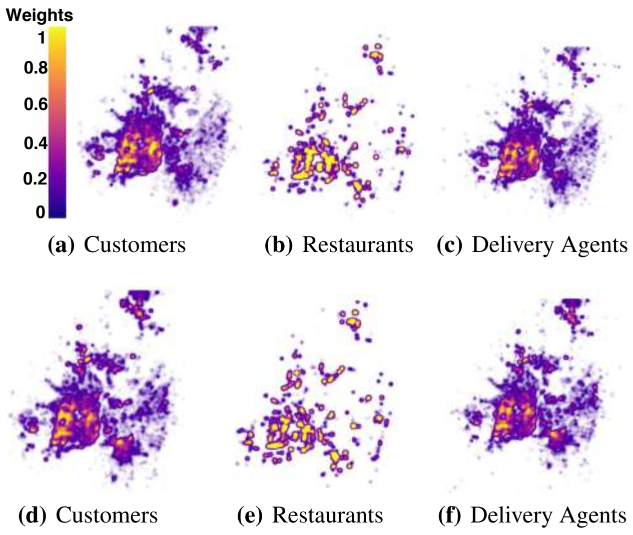

Figure 4: Heat map of order locations after applying FAIRFOODY. In (a-c) we show this location data with respect to the orders serviced by the top 25% agents and in (d-f) those serviced by bottom 25% agents based on income.

Impact of Parameters

Clustering parameter (f): Lowering of f leads to more clustering among orders. Fig. 5 analyzes variation in different performance metrics against f. Both Gini coefficient (Fig. 5(a)) and Average delivery time per order (Avg. DTPO) (Fig. 5(c)) increase as we decrease f. Higher clustering prevents equitable distribution of orders (and, therefore, income). Hence, Gini increases. DTPO increases since larger clusters lead to less simultaneous delivery of orders through multiple drivers. SLA violations remain unaffected Fig. 5(d)). There is no consistent trend across three cities in income gap Fig. 5(b)).

Impact of number of vehicles: Instead of considering all vehicles that were available in our real dataset, we randomly sample a subset of X vehicles. Next, we vary X in the x-axis and observe the impact of various metrics in y-axis. Fig. 6 presents the results. We observe that Gini increases with increase in vehicles (Fig. 6(a)). This is expected since decreasing the number of vehicles generates higher demand and more scope to fairly distribute orders. It is also natural that higher vehicle available leads to lower delivery time (Fig. 6(c)) and less SLA-violations (Fig. 6(d)).

Impact of traffic delays on fairness

To see the impact of traffic delays on fairness, we synthetically increase the edge weight (travel time) by 50% on 30% of the edges chosen uniformly at random causing unexpected delays on our expected delivery times. Table 6 presents the impact on Gini. While we observe an increase in Gini compared to FAIRFOODY, the Gini remains low and significantly better than FOODMATCH. The small increase in Gini happens due to our travel time estimations being

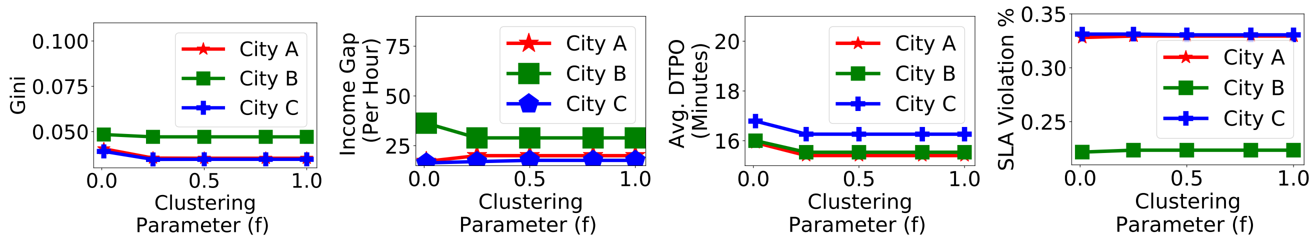

Figure 5: Impact of clustering parameter (f)

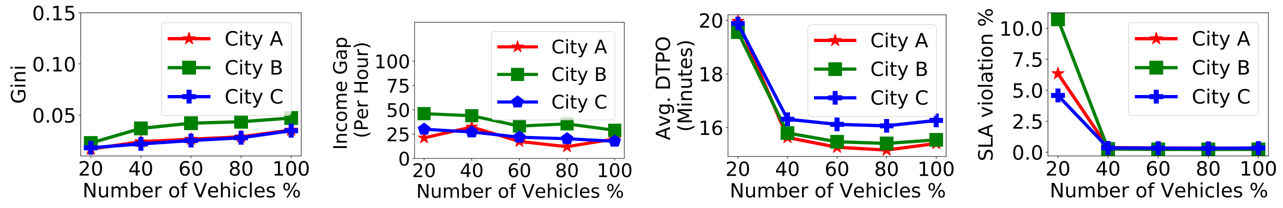

Figure 6: Impact of number of vehicles on various performance metrics.

violated.

Impact of fairness on drivers' retention

Table. 7 presents the percentage of drivers whose income increased under fair allocation through FAIRFOODY when compared to their original allocation. As visible, $\approx$ 75% drivers witness an increase in income. Hence, with fair allocation, retention gets easier.

Do the income-decreasing drivers have higher skill? Will they leave the system due to fair allocation? The skill level of a driver can be quantified through *extra delivery time (XDT)* = delivery time - shortest possible delivery time (Def. 7). In Table 8, we present the average difference in XDTs of income-increasing drivers under fair allocation with those whose income goes down ($\approx$ top-25%) from the real food-delivery data shared by the service provider. As visible, the difference is minimal indicating similar skill-levels. This results substantiates our earlier claim that higher income is more closely linked with driver location rather than skill level (Fig. 2 and Table 4).

Related Work

Food order assignment: On the problem of food-delivery, FOODMATCH (Joshi et al. 2021) is the only work to provide a realistic and scalable solution in food delivery domain. Other works on food delivery suffer from various unrealistic assumptions such as perfect information about arrival of orders (Yildiz and Savelsbergh 2019), ignoring the road network (Reyes et al. 2018), and ignoring food preparation time (Zeng, Tong, and Chen 2019).

Fairness in multi-sided platform algorithms: With the growing popularity of multi-sided platforms, a number of recent works have investigated the challenges of unfairness and bias in such platforms. For example, (Edelman, Luca, and Svirsky 2017) looked into the likelihood of racial bias in Airbnb hosts' acceptance of guests, while (Lambrecht and Tucker 2016) looked at gender discrimination in job advertisements. Few works have also looked at how producers and customers treat each other as a group. (Chakraborty et al. 2017) and (Sühr et al. 2019) proposed strategies for two-sided fairness in matching situations, whereas (Burke 2017) categorised distinct types of multi-stakeholder platforms and their required group fairness qualities. Individual fairness for both producers and customers is addressed by (Patro et al. 2020) in tailored suggestions in two-sided platforms. Despite these works on fairness in two-sided platforms, there has not been any studies on food delivery platforms. It is also worth noting that, as discussed in (Joshi et al. 2022), allocation algorithms for the cab service industry (Garg and Ranu 2018; Yuen et al. 2019; Ma, Zheng, and Wolfson 2013;

City	Algorithm	Running time (secs)	Overflow (%)	Peak Overflow (%)
A	FAIRFOODY	**0.92**	0	0
	FOODMATCH	4.77	0	0
	2SF $\lambda = 1$	2.46	0	0
	2SF $\lambda = 0$	2.63	0	0
B	FAIRFOODY	**14.00**	0	0
	FOODMATCH	41.93	0	0
	2SF $\lambda = 1$	21.00	0	0
	2SF $\lambda = 0$	20.67	0	0
C	FAIRFOODY	**10.76**	0	0
	FOODMATCH	50.66	0	0
	2SF $\lambda = 1$	28.21	0	0
	2SF $\lambda = 0$	26.12	0	0

Table 5: Comparison of running time and overflow windows. Peak Overflow considers only the lunch (11AM-2PM) and dinner (7PM-11PM) periods.

City	Gini		
	FAIRFOODY	FAIRFOODY + delay	FoodMatch
City A	0.035	0.08	0.518
City B	0.047	0.088	0.512
City C	0.035	0.067	0.562

Table 6: Impact of travel delays on Gini.

City	D1	D2	D3	D4	D5	D6
City A	74%	75%	73%	73%	71%	72%
City B	74%	75%	73%	74%	72%	72%
City C	78%	79%	77%	78%	77%	76%

Table 7: Percentage of drivers whose income increased under FAIRFOODY in each day (D1-D6).

Cheng, Xin, and Chen 2017) is not a natural fit food delivery.

Conclusion

In this work, we focused on the unfairness issues faced by delivery agents in food delivery platforms. Using data from a large real-world food platform, we showed that there exists high inequality in the income earned by the agents. To counter such inequality, we proposed an algorithm FAIRFOODY to assign delivery agents to orders ensuring that income opportunities are *fairly distributed* among the agents. By removing zone restriction, FAIRFOODY addresses the fact that the spatial spread of orders is a key driver of unequal pay. FAIRFOODY also achieves a more fair pay distribution by amortizing fairness over a reasonable period of time, thereby ensuring that agents who rely only on food delivery for their livelihood are fairly remunerated. Extensive experiments show that FAIRFOODY outperforms state-of-the-art baselines in lowering inequality while ensuring minimal increase in delivery time. Given the increasing adoptions of such platforms, it is the need of the hour and we hope that our work would lead to more followup works in this space.

Limitations and Future Work: Fairness can be studied from other angles as well. For example, do customers all over the city suffer equal extra delivery times? How does the presence of delivery vehicles in the neighborhood of a restaurant affect their order volumes? We plan to study these questions in the future.

City	XDT Diff (secs)
City A	11
City B	44
City C	68

Table 8: Performance of drivers in terms of delivery times.

References

Anab, M. 2021. Delivery staffers battle odds, health risks to be on job. https://bit.ly/3jXa3Gi.

Anand, S. N.; Borah, P. M.; and Majumdar, M. 2021. The great Indian food delivery tussle. https://www.thehindu.com/life-and-style/food/indian-restaurants-go-for-direct-delivery/article34770229.ece.

Arneson, R. 2013. Egalitarianism. In Zalta, E. N., ed., *Stanford Encyclopedia of Philosophy (Summer 2013 Ed)*. Stanford University.

Arneson, R. J. 2018. Dworkin and Luck Egalitarianism. In *The oxford handbook of distributive justice*.

Burke, R. 2017. Multisided Fairness for Recommendation. ArXiv:1707.00093 [cs.CY].

Chakraborty, A.; Hannák, A.; Biega, A. J.; and Gummadi, K. 2017. Fair Sharing for Sharing Economy Platforms. In *FATREC 2017*.

Cheng, P.; Xin, H.; and Chen, L. 2017. Utility-Aware Ridesharing on Road Networks. In *SIGMOD*, 1197–1210.

City, C. 2020. Notice of Fee Caps on Third-party Food Delivery Services. https://bit.ly/38U25aM.

Delling, D.; Goldberg, A.; Pajor, T.; and Werneck, R. 2014. Robust Exact Distance Queries on Massive Networks. Technical Report MSR-TR-2014-12, Microsoft.

D'Souza, P. M. 2021. Rising fuel prices, less earnings: Double whammy for food delivery agents amid lockdown. https://bit.ly/3zX0mNv.

Edelman, B.; Luca, M.; and Svirsky, D. 2017. Racial Discrimination in the Sharing Economy: Evidence from a Field Experiment. *American Economic Journal: Applied Economics*, 9: 1–22.

Fairwork. 2021. Fairwork India Ratings 2020: Labour Standards in the Platform Economy. https://bit.ly/3tsDIua.

Garg, N.; and Ranu, S. 2018. Route recommendations for idle taxi drivers: Find me the shortest route to a customer! In *ACM KDD*, 1425–1434.

Gastwirth, J. L. 1972. The estimation of the Lorenz curve and Gini index. *The review of economics and statistics*, 306–316.

Joshi, M.; Singh, A.; Ranu, S.; Bagchi, A.; Karia, P.; and Kala, P. 2021. Batching and Matching for Food Delivery in Dynamic Road Networks. In *Proc. ICDE*.

Joshi, M.; Singh, A.; Ranu, S.; Bagchi, A.; Karia, P.; and Kala, P. 2022. FoodMatch: Batching and Matching for Food Delivery in Dynamic Road Networks. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 8(1): 1–25.

Kuhn, H. W. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2): 83–97.

Lambrecht, A.; and Tucker, C. 2016. Algorithmic Bias? An Empirical Study into Apparent Gender-Based Discrimination in the Display of STEM Career Ads. *SSRN Electronic Journal*.

Lamont, J. 2017. *Distributive justice*. Routledge.

Levin, D. A.; Peres, Y.; and Wilmer, E. L. 2017. *Markov chains and mixing times*. American Mathematical Soc., 2e edition.

Ma, S.; Zheng, Y.; and Wolfson, O. 2013. T-share: A large-scale dynamic taxi ridesharing service. In *ICDE*, 410–421.

Munkres, J. 1957. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1): 32–38.

Murthy, D. 2021. The 'customer is king' motto has invisibilised the food delivery rider. https://www.thenewsminute.com/article/customer-king-motto-has-invisibilised-food-delivery-rider-153591.

Newson, P.; and Krumm, J. 2009. Hidden Markov Map Matching Through Noise and Sparseness. In *ACM SIGSPATIAL GIS*, 336–343.

Patro, G. K.; Biswas, A.; Ganguly, N.; Gummadi, K. P.; and Chakraborty, A. 2020. FairRec: Two-Sided Fairness for Personalized Recommendations in Two-Sided Platforms. In *WWW*. ACM.

Pengonda, P. 2021. Why covid-19 has been a shot in the arm for online food delivery firms. https://www.livemint.com/market/mark-to-market/why-covid-19-has-been-a-shot-in-the-arm-for-online-food-delivery-firms-11611558506486.html.

Rawls, J. 1971. *A theory of justice*. Harvard university press.

Reyes, D.; Erera, A. L.; Savelsbergh, M. W. P.; Sahasrabudhe, S.; and O'Neil, R. J. 2018. The Meal Delivery Routing Problem. Optimization Online.

Sühr, T.; Biega, A. J.; Zehlike, M.; Gummadi, K. P.; and Chakraborty, A. 2019. Two-sided fairness for repeated matchings in two-sided markets: A case study of a ride-hailing platform. In *ACM KDD*.

Yildiz, B.; and Savelsbergh, M. 2019. Provably High-Quality Solutions for the Meal Delivery Routing Problem. *Transportation Science*, 53.

Yuen, C. F.; Singh, A. P.; Goyal, S.; Ranu, S.; and Bagchi, A. 2019. Beyond Shortest Paths: Route Recommendations for Ride-sharing. In *Proc. World Wide Web Conference (WWW '19)*, 2258–2269.

Zeng, Y.; Tong, Y.; and Chen, L. 2019. Last-Mile Delivery Made Practical: An Efficient Route Planning Framework with Theoretical Guarantees. *Proc. VLDB Endow.*, 13(3): 320–333.