ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230507

Balancing Fairness and Efficiency in 3D Repeated Matching in Ridesharing

Garima Shakya[a;*] **and Makoto Yokoo**[a]

[a]Kyushu University, Fukuoka, Japan

Abstract. Ride-hailing services' main feature is mediating the assignment and transactions between drivers and passengers. Essentially, they decide on the quality of passengers' experience and the drivers' workload balancing. To boost the company's profit, these services try to maximize the utility for the passengers by optimizing the matching, resulting in shorter waiting times and better service availability. Often, in the process of maximizing revenue, drivers' interests get sidelined. We focus on two objectives: efficiency (minimizing total distance traveled by drivers) and fairness (minimizing the maximum traveled distance by any driver) for shared-mode rides, where the vehicles' capacity is two passengers. We theoretically show the relation between the optimal solutions of both objectives and as the problem is computationally intractable, we propose a heuristic algorithm to achieve an approximately optimal solution. We also propose a re-assignment-based algorithm when the aim is to achieve maximum matching with fairness up to a given threshold, if that is feasible. The experimental analysis for the proposed algorithms on real-world data from Chicago city shows that our approach can significantly improve fairness for drivers without losing much efficiency.

1 Introduction

Ride-hailing and food-delivery services such as Uber, Lyft, Ola, and Foodora have become essential components in increasing urban transportation's sustainability. These services are quickly changing the urban transportation ecosystem [11]. Due to the flexible working hours, ride-hailing services are a popular alternative for people looking for a side job or a new career.

Ridesharing is a ride-sourcing mode in which a vehicle can simultaneously service more than one request. We investigate the ridesharing problem in the following setting: at a time instance of a day, there are a set of available drivers in a city and a set of riding requests made; each request has a pick-up location and drop-off location; between any two locations, there is a weight function representing the length or cost of the shortest route between them. The problem is to assign maximum requests to the drivers such that each driver serves at most two requests simultaneously while *maximizing efficiency* (minimizing the sum of the cost of travel over all the drivers) and/or while *maximizing fairness* (minimizing the maximum traveled distance by any driver). For efficiency, we adopt the 'minimizing total cost' criterion, defined as the sum of traveled distance by the drivers. For fairness, we adopt the 'maximize the utility of least advantaged driver' criterion based on well-recognized Rawlsian egalitarian justice [17].

Our ultimate goal is to optimize combined efficiency and fairness. However, optimizing daily criteria before the end of the day is impossible since we need to know future requests in advance. This is because the assignment at a time instance affects the drivers' locations in the next time instance and so on. Thus, we aim for a compromised goal. We apply the greedy approximate criterion to minimize the total cost for each time period separately and balance the workload distribution by minimizing the travel distance of the most traveled driver until the current time period.

1.1 Motivation

Shared-mode rides reduce the number of automobiles needed by travelers, which leads to long-term social and economic benefits such as 1) reductions in greenhouse gas emissions and energy consumption, 2) congestion mitigation, 3) reduced parking infrastructure demand, and others. Here are some case studies showing the environmental benefits of ridesharing in various cities [3, 4, 19, 20].

Ride-hailing services' main feature is mediating the assignment and transactions between drivers and passengers. Essentially, they decide on the quality of passengers' experience and the drivers' workloads. To boost the company's profit, these matching platforms try to maximize the utility for the passengers by optimizing the matching that results in shorter waiting time and better service availability [13]. The main reason might be that the passengers contribute more directly to the platform's revenue. In the process of maximizing revenue, drivers' interests get sidelined [12]. The solutions may result in undesirable social outcomes. Some drivers may be assigned to undesired trips, resulting in an unbalanced workload; therefore, a loss of fairness from the drivers' perspective [14].

Recent studies on two-sided matching platforms raise concerns about the exploitation of employees, including unfair pay, working conditions and safety [7]. The distribution of drivers' workload has yet to get much attention in the algorithm design domain. An investigation by Bokányi and Hannák [2] shows the effect of algorithm design decisions on wage inequality in ride-hailing markets and how small changes in the system parameters can cause large deviations in the income distributions of identically performing drivers. Our concern is that the short-term differences may result in long-term welfare gaps. Nonetheless, ensuring *fair workload distribution* for drivers might also prove beneficial in sustaining the business in the long run. Otherwise, unsatisfied drivers may leave or remain inactive on the platform. Therefore, fair workload distribution on the drivers' side should receive more attention.

* Corresponding Author. Email: garima@inf.kyushu-u.ac.jp

The ridesharing platforms make request-to-driver assignments with a repeated set of drivers and customers where the assignment for each individual is often for some limited duration in the day. Once that duration is over, the driver can be assigned to another available request in the following duration. In line with the current matching scenarios, we are specifically interested in investigating the repeated matching between drivers and requests that lead to a fair distribution of drivers' daily workload while maintaining efficiency.

1.2 Related work

Ridesharing systems are widely studied in the literature. Finding an efficient ridesharing allocation is based on the 2-1 assignment problem reviewed by Goossens et al. [10]. The 2-1 assignment problem is as follows: given a set W of m white balls and a set B of $2m$ black balls; there is a cost function for every triple combination containing two balls from B and one ball from W; the objective is to find a collection of triple combinations such that the sum of costs of triple combinations is minimum, while each ball is in precisely one combination. In terms of the ridesharing problem, W and B can be considered as the set of drivers and passengers' requests, respectively. Goossens et al. [10] provide a $4/3$ approximation ratio algorithm for 2-1 matching. However, the expression for the cost function in [10] differs from that in ridesharing settings.

Bei and Zhang [1] studied the *maximizing efficiency* problem in ridesharing and proved that finding the most efficient 2-1 matching is NP-hard, and proposed a polynomial time 2.5 approximation ratio algorithm. Luo and Spieksma [15] investigated the problem of *minimizing total latency* along with efficiency and provided two algorithms with approximation ratios 2 and $5/3$ for efficiency and total latency, respectively.

The idea of loss in efficiency in achieving a fair solution is well explored in many resource allocation settings. However, to the best of our knowledge, the problem needs to be explored more in the request-to-driver assignment domain. Mainly due to unique constraints (e.g., on pick-up distance) the existing literature for other resource allocation problems can not be directly applied to the ridesharing settings. This is due to additional constraints unique to the ridesharing problem, such as the constraint of pick-up distances.

In line with fairness on ride-hailing platforms, Nanda et al. [16] examined the rider's fairness due to drivers' discriminative cancellations. Xu and Xu [18] construct a bi-objective linear program focused on profit and fairness on the platform and propose two LP-based parameterized online algorithms. Our objectives also differ from the literature mentioned above and traditional resource allocation settings as we study the problem of achieving a balance of fairness and efficiency in *3-dimensional* request-to-driver matching where two requests can be serviced by a driver simultaneously. The addition of one more dimension brings more challenges.

An exciting and closely related work by Lesmana et al. [14] provides a reassignment algorithm to find a balance between the two objectives while finding one-to-one (2-D) request-to-driver matching. The algorithm in [14] can also be applied to shared ride settings by assigning one passenger in each repetition of the algorithm. Repeating the algorithm provides a greedy solution by looking for the best solution in each repetition. We are designing an algorithm that finds a match for the combined input of two repeats for the algorithm in [14]. This allows the solution set concerning the capacity of the vehicles and opens up the opportunity to find a better solution. Along with the importance of finding the optimal solution for ridesharing, we intend to look for the algorithmic question of finding the balanced solution for multiple objectives in 3-D matching.

1.3 Our contributions

In summary, this paper focuses on the following research questions: 1) How are efficient and fair solutions for 2-1 matching in ridesharing related? 2) How can we get a balance between efficiency and fairness in ridesharing? 3) What is the price, in terms of efficiency, for fair distribution of work and income among drivers?

To answer these questions, this paper contains the following:

1. We provide a theoretical bound on the 'loss in efficiency' required to achieve a fair solution (Theorems 1 and 2).
2. We propose a two-phase algorithm that accounts for the natural tension between these two objectives (Algorithm 1). We also propose a re-assignment based algorithm to achieve fairness with respect to a given fairness threshold (Algorithm 2).
3. We experiment on a real world dataset and attempt to answer question (3) and analyze the performance of the proposed algorithms (Figure 4).

2 Model and problem formulation

Consider the city as a weighted connected graph, $GC = (\mathcal{L}, E, w)$, where $\mathcal{L}$ is a finite set of locations, E is the edge set s.t. $E = \{(l_1, l_2) \mid l_1, l_2 \in \mathcal{L}\}$ and $w : E \to \mathbb{R}^+$ is the edge weight function. The weight function w can be any metric satisfying (1) non-negativity $w(l_x, l_y) \geq 0$, (2) symmetry $w(l_x, l_y) = w(l_y, l_x)$. Typically, w can be considered as the distance functions ℓ_2, ℓ_1, or distance on a road graph. The weight notation is extended for the path as well. With slight abuse of notations, the weight of a path $(l_1, l_2, \ldots, l_x)$ is defined as $w(l_1, l_2, \ldots, l_x) = \sum_{i=1}^{x-1} w(l_i, l_{i+1})$.

Denote the daily working hours of the ridesharing platform as T. The platform accumulates the requests entered during an accumulating time window λ and performs batch assignment of the accumulated requests to the available drivers. Let $(t_1, t_2, t_3, \ldots, t_i, t_{i+1}, \ldots, t_{|T|})$ be the sequence of discrete time instants, such that $t_{i+1} - t_i = \lambda$. The set of accumulated requests at $t \in T$ are represented by $\mathcal{R}^t = \{r_1, r_2, r_3, \ldots\}$. Each request r consists of two elements, $r = (s, d)$, where, $s, d \in \mathcal{L}$ denoting the source and destination of the request r, respectively.

Let $\mathcal{D}$ denote the set of all drivers.[1] The set of drivers available at t are represented as $\mathcal{D}^t = \{v_1, v_2, v_3, \ldots v_n\}$. The attributes of each driver v is represented as a 2-tuple (CL_v, DT_v), where, $CL_v \in \mathcal{L}$ is the current location of v, $DT_v^t \in \mathbb{R}_{\geq 0}$ is total distance travelled by v since the beginning of the day. At a time instant t, DT_v^t denotes the total travelled distance by v till the end of t.

In this paper, we assume the capacity of each vehicle is 2. Let $\mathcal{M}_t$ denotes the set of all possible 2-to-1 matching for sets $\mathcal{R}^t$ and $\mathcal{D}^t$. A matching $M \in \mathcal{M}_t$ is defined as $M = \{(v_k, \mathcal{R}_k) \mid v_k \in \mathcal{D}^t, \mathcal{R}_k \subseteq \mathcal{R}^t, |\mathcal{R}_k| \leq 2\}$, where, $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_n$ are mutually disjoint subset of $\mathcal{R}^t$. Each element $(v_k, \mathcal{R}_k)$ of M denotes that the driver v_k is assigned to a set of requests $\mathcal{R}_k$. Let $M(v)$ denote the requests assigned to driver v. And, $M(\mathcal{R}_k)$ denote the driver v_k assigned to requests in $\mathcal{R}_k$.

We define a function $cost : M \to \mathbb{R}^+$ s.t., $cost(v, M(v))$ is the minimum distance that needs to be covered by v to service $M(v)$. If $M(v) = \{(s_1, d_1), (s_2, d_2)\}$, i.e., two requests are assigned to v,

[1] We use the term 'driver' to denote a 'vehicle.' And, therefore, use both terms interchangeably.

there can be six possible routes, depending on the order of the pick-up and drop-off of the passengers in $M(v)$. More formally,

$$\begin{aligned} cost(v, M(v)) = \min(&w(CL_v, s_1, d_1, s_2, d_2),\\ &w(CL_v, s_1, s_2, d_1, d_2),\\ &w(CL_v, s_1, s_2, d_2, d_1),\\ &w(CL_v, s_2, s_1, d_2, d_1),\\ &w(CL_v, s_2, s_1, d_1, d_2),\\ &w(CL_v, s_2, d_2, s_1, d_1)) \end{aligned}$$

If $M(v) = \{(s, d)\}$, i.e., only one request is assigned to v, then $cost(v, M(v)) = w(CL_v, s) + w(s, d)$. The cost of a matching M is defined as,

$$cost(M) = \sum_{v \in \mathcal{D}^t} cost(v, M(v))$$

We also define cost of servicing a request pair (r_i, r_j) starting from source of r_i as follows,

$$\begin{aligned} c(s_i, (r_i, r_j)) = \min(&w(s_i, d_i, s_j, d_j),\\ &w(s_i, s_j, d_i, d_j), w(s_i, s_j, d_j, d_i)) \end{aligned}$$

Similarly, we define $c(s_j, (r_i, r_j))$ as the length of shortest route to service (r_i, r_j) starting from s_j. The minimum of both costs is denoted as,

$$\mathfrak{c}(r_i, r_j) = \min(c(s_i, (r_i, r_j)), c(s_j, (r_i, r_j)))$$

For $r_i = r_j$, $\mathfrak{c}(r_i, r_i) = w(s_i, d_i)$.

Driver's per day income: The payment scheme to the driver might differ between service providers and cities. Sometimes, even the same service provider might rapidly change the pricing scheme within a city [6]. We omit those calculations and assume that the drivers get a fixed pre-decided daily wage. As there is some cost of travel between any two locations, such as fuel cost, the daily utility of a driver depends on the distance traveled on that day. In other words, the utility of a driver depends on the shortest distance covered by the driver to serve the assigned ride. The utility is more if the cost of the ride is less.

We represent the matching M among the rides and drivers as a 3 dimensional matrix $A := [a_{i,j,k}]_{\forall (r_i, r_j) \in \mathcal{R}^t \times \mathcal{R}^t, \forall k \in \mathcal{D}^t}$ such that, $a_{i,j,k} = 1$ if $i \neq j$, $M(v_k) = \mathcal{R}_k$ and $\mathcal{R}_k = \{r_i, r_j\}$, otherwise, $a_{i,j,k} = 0$, and $a_{i,i,k} = 1$ if $M(v_k) = \mathcal{R}_k$ and $\mathcal{R}_k = \{r_i\}$, otherwise, $a_{i,i,k} = 0$.

2.1 Desired properties

We focus on multiple objectives defined as follows:

DEFINITION **1 (Maximum matching)** *Given $(\mathcal{R}^t, \mathcal{D}^t)$, a matching $M \in \mathcal{M}_t$ is a maximum matching if,*

$$M = \arg\max \sum_{v_k \in \mathcal{D}^t} \sum_{(r_i, r_j) \in \mathcal{R}^t \times \mathcal{R}^t} a_{i,j,k}$$

If r_i and r_j are matched with driver v_k then $a_{ijk} = 1$, otherwise $a_{ijk} = 0$. And, for $r_i = r_j$, $a_{iik} = 1$ if r_i is assigned for a solo ride to v_k, else $a_{iik} = 0$. In Definition 1, we maximize the sum of a_{ijk} over all (r_i, r_j) ride pairs and all drivers v_k. Hence the shared ride is counted as 2 (once of (r_i, r_j) and then again for (r_j, r_i)) and a solo ride as 1 (for (r_i, r_i)) which incentivizes to assign more shared mode rides but keeps the option for solo rides if shared mode matching is not feasible.

In other words, given $(\mathcal{R}^t, \mathcal{D}^t)$, maximum matching M consists of a maximum number of assignments among $\mathcal{R}^t$ and $\mathcal{D}^t$. This objective ensures that we assign a maximum number of requests to the drivers for each $t \in T$, which reduces the overall waiting time of the passengers. Let $\mathcal{M}_t^*$ denote a set of all maximum matchings in $\mathcal{M}_t$.

The matching applied at time t affects the problem solved at time $t+1$, and so on. Ideally, for efficiency, we want to minimize the total cost, we want to find $\boldsymbol{M} = (M_1, ..., M_{|T|})$ such that it minimizes $\sum_{t \in T} cost(M_t)$. However, optimizing this criterion is impossible since we do not know future requests in advance. Thus, we apply a greedy approach and desire to minimize the cost for each time period separately.

DEFINITION **2 (Efficient Maximum Matching)** *Given $(\mathcal{R}^t, \mathcal{D}^t)$, a maximum matching $M \in \mathcal{M}_t^*$ is an efficient (maximum) matching if there does not exist a maximum matching $M' \in \mathcal{M}_t^*$ such that,*

$$\sum_{v \in \mathcal{D}^t} cost(v, M'(v)) < \sum_{v \in \mathcal{D}^t} cost(v, M(v))$$

We desire to reduce the total traveled distance by drivers that reduce the total emission of greenhouse gases and overall traffic congestion. Our objective is to get an efficient matching and hence to find an $M \in \mathcal{M}_t^*$ that has a minimum cost.

Based on the difference principle in the seminal work on the theory of justice by John Rawls in [17], we defined the *unfairness* (UF) of a matching M for given $(\mathcal{R}^t, \mathcal{D}^t)$ as,

$$UF(M) = \max_{v \in \mathcal{D}^t} (cost(v, M(v)) + DT_v^{t-1})$$

Ideally, we want to minimize the total travel distance within a day for the most traveled driver. Since this is also impossible as we do not know future requests in advance, we consider an approximate criterion, in which we try to minimize the travel distance of the most traveled driver until the current time period.

DEFINITION **3 (Fair Maximum Matching)** *Given $(\mathcal{R}^t, \mathcal{D}^t)$, a matching $M \in \mathcal{M}_t^*$ is fair if there does not exist a maximum matching $M' \in \mathcal{M}_t^*$ such that,*

$$UF(M') < UF(M).$$

We aim to maximize fairness and hence to find an M that minimizes the distance traveled by the driver who has traveled the most. In other words, we aim to *maximize the utility of the least advantaged driver*.

We summarize our objectives as follows:

DEFINITION **4 (OBJ1: Efficient Maximum Matching)** *Given an instance $(\mathcal{R}^t, \mathcal{D}^t)$, find maximum matching $M \in \mathcal{M}_t^*$, s.t. $\sum_{v \in \mathcal{D}^t} cost(v, M(v))$ is minimum.*

DEFINITION **5 (OBJ2: Fair Maximum Matching)** *Given an instance $(\mathcal{R}^t, \mathcal{D}^t)$, find maximum matching $M \in \mathcal{M}_t^*$, s.t., $\max_{v \in \mathcal{D}^t} DT_v^t$ is minimum. Mathematically,*

$$\arg\min_{M \in \mathcal{M}_t^*} UF(M)$$

3 Relation between efficient and fair solutions

In this section, we analyze the loss in efficiency and the gain in fairness in terms of each other. Due to uncertainty in future requests

and driver locations, we limit the analysis for a single accumulated window ($|T| = 1$) where, initially, $DT_v^0 = 0\ \forall v \in \mathcal{D}$. Unless mentioned otherwise, all the results in this section consider this assumption. Therefore, we omit the superscript t to denote the single window notations, e.g., DT_v^t is denoted as DT_v.

There exists a case where we cannot optimize both objectives simultaneously, as shown in Example 1.

EXAMPLE **1** *Consider two drivers $\mathcal{D} = \{v_1, v_2\}$ with capacity 2, $DT_{v_1} = DT_{v_2}$ and located at $A \in \mathcal{L}$. There are four requests $\mathcal{R} = \{r_1, r_2, r_3, r_4\}$ such that, $\forall r_i \in \mathcal{R}$, $s_i = A$. And the destination locations for r_1, r_2, r_3, and r_4 are B, C, D and E, respectively (Figure 1). The edges and weights in Figure 1 show the connectivity and the cost of travel between any two locations. One of the optimal solution for $OBJ1$ is $M_1^* = \{(v_1, (r_1, r_2)), (v_2, (r_3, r_4))\}$ with $OBJ1(M_1^*) = 3.5{+}8.5 = 12$. However, M_1^* is not an optimal solution for $OBJ2$ as $OBJ2(M_1^*) = 8.5$ and there exist another matching $M_2^* = \{(v_1, (r_1, r_3)), (v_2, (r_2, r_4))\}$ with increased fairness; $OBJ2(M_2^*) = 6.5$, but also increased cost, $OBJ1(M_2^*) = 13$.*

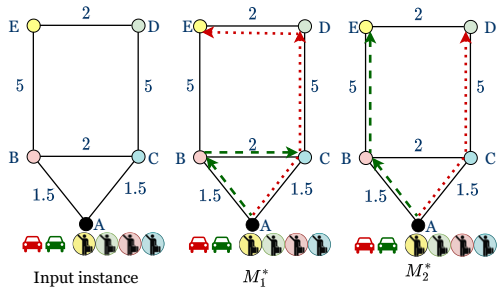


Figure 1: Example 1 with $|\mathcal{D}| = 2$ and $|\mathcal{R}| = 4$.

Let M_1^* and M_2^* denote optimal matchings according to $OBJ1$ and $OBJ2$, respectively. Let Δ denote the loss in efficiency in achieving a fair solution, i.e.,

$$\Delta = OBJ1(M_2^*) - OBJ1(M_1^*).$$

Similarly, let Γ denote the loss in fairness in achieving an efficient solution, i.e.,

$$\Gamma = OBJ2(M_1^*) - OBJ2(M_2^*).$$

The lower bound for Δ and Γ is 0, as there exists a case where the efficient matching is also fair, as shown in the example below.

EXAMPLE **2** *Consider the example 1 with a change in the edge weight as shown in Figure 2. One of the optimal solution for $OBJ1$ is $M_1^* = \{(v_1, (r_1, r_3)), (v_2, (r_2, r_4))\}$ with $OBJ1(M_1^*) = 11$. Nevertheless, M_1^* is an optimal solution for $OBJ2$ as well with $OBJ2(M_1^*) = 5.5$ and there exist no another maximum matching $M_2^{'}$ with increased fairness.*

Next, we show that the loss in efficiency to get a fair matching can not be more than n times the fairness in the efficient matching.

Theorem 1 *If M_1^* and M_2^* are optimal solutions for $OBJ1$ and $OBJ2$, respectively, then $\Delta \leq n \cdot OBJ2(M_1^*)$.*

PROOF **1** *Suppose the above statement is not true and there exists a scenario where $(OBJ1(M_2^*) - OBJ1(M_1^*)) > n \cdot OBJ2(M_1^*)$. As $cost(.)$ is always non-negative, we get, $OBJ1(M_2^*) > n \cdot OBJ2(M_1^*)$. This implies, $n{\cdot}OBJ2(M_2^*) > n{\cdot}OBJ2(M_1^*)$, which leads to contradiction that M_2^* is an optimal solution for $OBJ2$.*

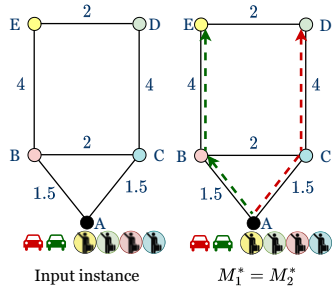


Figure 2: Example 2 with $|\mathcal{D}| = 2$ and $|\mathcal{R}| = 4$.

Next, we show a lower upper bound for Δ (loss in efficiency to get a fair matching) in terms of fairness in the fair matching, $OBJ2(M_1^*)$.

Theorem 2 *If M_1^* and M_2^* are optimal solutions for $OBJ1$ and $OBJ2$, respectively, then there exists an input instance, where $\Delta = \Omega((n-1)OBJ2(M_1^*))$ for $|\mathcal{R}| = 2n$ and $|\mathcal{D}| = n$.*

PROOF **2** *We construct an input instance to prove the above statement. Consider n drivers $v_1, v_2, \ldots, v_n$ with $CL_v = X, DT_v = 0$, and $2n$ ride requests each with source location X. The destination location of half of the rides, say r_1 to r_n are $A_1, A_2, \ldots, A_n$ and for rides $r_{n+1}, r_{n+2}, \ldots, r_{2n}$ are $B_1, B_2, \ldots, B_n$, respectively. All these locations and their cost of travel between them are shown in Figure 3. The notations in edge weights are as follows: $x, \epsilon, \delta > 0$, $\epsilon < \delta/2$ and $\epsilon + \delta < x$. The most expensive edge is (X, A_1) with cost $x - \epsilon$.*

One of the efficient maximum matching M_1^* is $\{(v_i, (r_i, r_{n+i}))$ for $1 < i < n\}$ with $OBJ1(M_1^*) = x{+}2\epsilon(n{-}1)$. The matching M_1^* has $OBJ2(M_1^*) = x$. There exists a different matching $M_2^* = \{(v_i, (r_{i+1}, r_{n+i}))$ for $1 \leq i < n, (v_n, (r_n, r_1))\}$ with $OBJ2(M_2^*) = x - \delta + \epsilon$. However, the efficiency is decreased in M_2^* as the $OBJ1(M_2^*) = n(x - \delta + \epsilon) + \epsilon$.

The difference in the efficiency in the two matching Δ

$$= n(x - \delta + \epsilon) + 2\epsilon - (x + 2\epsilon(n-1))$$

with $\epsilon, \delta \ll x$,

$$= \Omega((n-1)x)$$
$$= \Omega((n-1)OBJ2(M_1^*))$$

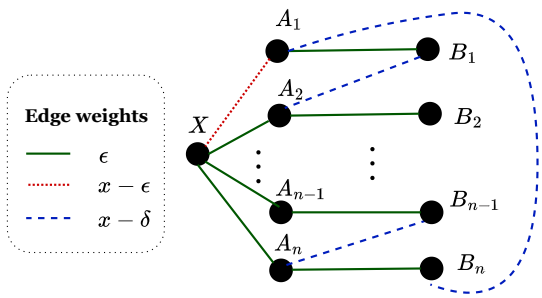


Figure 3: Problem instance for Theorem 2.

4 Proposed algorithms

Finding the efficient maximum matching is equivalent to finding a 3-dimensional perfect matching (3DM), known to be NP-hard [1, 9].

Algorithm 1 `Efficient-match Fair-assign Algorithm` (`EMFAA`)$(\mathcal{R}^t, \mathcal{D}^t, t)$

1: Construct a weighted graph $G_1 := (\mathcal{R}^t, E_1, W_1)$ such that, $E_1 := \{(r_i, r_j) \mid \forall r_i, r_j \in \mathcal{R}^t, r_i \neq r_j\}$, $W_1(r_i, r_j) := \mathfrak{c}(r_i, r_j)\ \forall (r_i, r_j) \in E_1$.
2: Find minimum weight matching M_1 for G_1.
3: $SM_1 :=$ Sorted elements (ride pairs and unmatched rides) in M_1 in increasing order of their cost $\mathfrak{c}(\cdot)$.
4: $SD^t :=$ Sorted $\mathcal{D}^t$ in decreasing order of $DT_{v_k}^{t-1}$.
5: Construct an unweighted bipartite graph $G_2 := (M_1, D^t, E_2)$ such that, $E_2 := \{((r_i, r_j), \forall v_k) \mid \forall (r_i, r_j) \in M_1, v_k \in D^t\}$.
6: Find a matching M_2 on G_2 by assigning the unallocated elements in SM_1 with the maximum cost $\mathfrak{c}(.)$ to the unmatched driver v_k in SD^t with the minimum value of $DT_{v_k}^{t-1}$, and so on.
7: **return** M_2 as the batch assignment for t.

We believe that finding an optimal solution for $OBJ2$ is also computationally intractable as the objective is similar to finding the optimal solution of minimum makespan scheduling, which is known to be NP-hard [9], with an additional constraint. The additional constraint is over the maximum number of jobs that can be allocated to a machine, where the processing time of a job on a machine depends also on the other jobs assigned to that machine.

4.1 Two-phase algorithm

We are looking for a computationally inexpensive algorithm that can provide close-to-optimal solutions. There are two major decisions to be made to achieve a 2-to-1 matching. First, to find the request pairs; second, to assign a driver to each request pair. To find a balance between the two objectives, we propose a two-phase heuristic algorithm (Algorithm 1) that focuses on the objectives individually. More specifically, Algorithm 1 finds the assignment at the end of each accumulated window. At each time interval t, the algorithm has two phases. In the first phase, the algorithm matches two requests (without considering drivers) based on their combined travel time. Then, in the second phase, it assigns the matched request pairs with the available drivers considering fairness. Thus, the algorithm focuses on efficiency in the first phase, while it focuses on fairness in the second phase, in the hope that the obtained result strikes a good balance between efficiency and fairness. The algorithm resembles the bi-level optimization where we partially optimize $OBJ1$ and $OBJ2$ at the lower and upper levels, respectively.

Remarks: (a) A minimum matching in a weighted graph of m vertices can be computed in time $O(m^3)$ [8]. Therefore, Algorithm 1 runs in $O(n^3)$ time.
(b) The step (6) results in the maximum possible assignments by considering the number of available drivers and the number of request pairs found in step (2). Hence, Algorithm 1 finds a *maximum* matching from the set of feasible matching. **(c)** In Step (2), if $|\mathcal{R}^t|$ is odd, one request cannot be matched. If r_i is not matched, we assume r_i is matched to itself, and the cost of this matching is $\mathfrak{c}(r_i, r_i)$.

4.2 Match and re-assign algorithm

Next, we describe an alternative algorithm (Algorithm 2), which first obtains a matching considering $OBJ1$ only, then gradually improves its fairness by re-assigning requests. More specifically, within this algorithm, we repeatedly solve a problem defined as follows: given a fairness factor κ, find an allocation M^*, such that $UF(M^*) \leq \kappa$. We add the following notation in this section,

$$\mathcal{X}(v, M) = cost(v_k, M(v)) + DT_{v_k}^{t-1}$$

The algorithm begins with a 2-1 approximately efficient matching given by a slight variant of a state-of-the-art approximate algorithm proposed by [1] (steps (1-4) in Algorithm 2). We begin re-assignment greedily by picking edges $(v_k, (r_i, r_j))$ such that $\mathcal{X}(v, M) > \kappa$, and check whether the re-assignment with $(v_k', (r_i', r_j'))$ minimizes $UF(.)$, where (r_i', r_j') has the minimum $\mathfrak{c}(\cdot)$. If so, we re-assign, else we check for another possible re-assignment. The algorithm results in the matching, which cannot be improved any further by re-assignment.

Algorithm 2 `Match and Re-assign Algorithm` (`MRAA`) $(\mathcal{R}^t, \mathcal{D}^t, \kappa, t)$

1: Construct a weighted graph $G_1 := (\mathcal{R}^t, E_1, W_1)$ such that, $E_1 := \{(r_i, r_j) \mid \forall r_i, r_j \in \mathcal{R}^t, r_i \neq r_j\}$, $W_1(r_i, r_j) := \mathfrak{c}(r_i, r_j)\ \forall (r_i, r_j) \in E_1$.
2: Find minimum weight matching M_1 for G_1.
3: Construct a weighted bipartite graph $G_2 := (M_1, \mathcal{D}^t, E_2, W_2)$ such that, $E_2 := \{((r_i, r_j), v_k) \mid \forall (r_i, r_j) \in M_1, v_k \in \mathcal{D}^t\}$, $W_2((r_i, r_j), v_k) := cost(v_k, (r_i, r_j))\ \forall (r_i, r_j) \in M_1, v_k \in \mathcal{D}^t$.
4: $M_2 :=$ minimum weighted bipartite matching for G_2.
5: $M_{new} = M_2$. $\ell = 1$. $\mathfrak{z} = 1$.
6: **while** $\exists (v_k, (r_i, r_j)) \in M_{new}$ s.t. $\mathcal{X}(v, M_{new}) > \kappa$ **do**
7: Pick $(v_k, (r_i, r_j))$ which has $\mathfrak{z}^{th}$ maximum $\mathcal{X}(v, M_{new})$ with $\mathcal{X}(v, M_{new}) > \kappa$.
8: Pick (r_i', r_j') with ℓ^{th} minimum $\mathfrak{c}(.)$ among $\{M_{new}(v) \mid v \in \mathcal{D}^t\}$.
9: $v_k' := M_{new}(r_i', r_j')$.
10: $m_2 =$ Fair maximum 2-1 matching for sets $\{r_i, r_j, r_i', r_j'\}, \{v_k', v_k'\}$. {Find best among the six possible solutions.}
11: **if** $m_2 = \{(v_k', (r_i', r_j')), (v_k, (r_i, r_j))\}$ **then**
12: **if** $\ell < |M_{new}|$ **then**
13: $\ell = \ell + 1$. Go to step 8. {Check re-assignment for v_k with next least $\mathfrak{c}(.)$ ride pair.}
14: **else**
15: **if** $\mathfrak{z} < |M_{new}|$ **then**
16: $\mathfrak{z} = \mathfrak{z} + 1$. $\ell = 1$. Goto step (6).
17: **else**
18: There does not exist any 2-1 maximum matching M^* s.t. $UF(M^*) < \kappa$. Goto step (22).
19: **else**
20: $M_{new} = M_{new} \cup m_2 - \{(v_k', (r_i', r_j')), (v_k, (r_i, r_j))\}$.
21: $\mathfrak{z} = 1$. $\ell = 1$.
22: **return** M_{new} as the assignment for t.

Remarks: The Algorithm 2 finds a *maximum* matching from the set of feasible matching. At first, the Algorithm 2 finds maximum 2-1 matching then, the re-assignments step (20) allocates rides up vehicles' maximum capacity.

5 Experimental analysis

We experiment on a real-world dataset to analyze the behavior of Algorithm 1 concerning fairness and efficiency.

5.1 Framework

The details of the dataset and values of the parameters used for the experiments are as follows.

Trip Dataset: We use the publicly available dataset of taxi trips (solo rides) in Chicago city [5]. The dataset contains all the taxi trips, starting January 2013, reported to the City of Chicago. We extracted the dataset during the busy three hours in the morning $(8-11AM)$ on 'April 4, 2022' (Monday). The dataset includes a unique identifier for each taxi and some essential attributes about trips, e.g., the source and destination locations and time of the trip. We choose this day arbitrarily, and we strongly believe the results will have a similar pattern on other parts of the dataset.

Although our experiments were conducted using only solo rides, the Chicago dataset satisfied all other requirements of our model, including trip source, destination, and timings. Due to the impact of Covid-19, the occurrence of shared mode rides decreased significantly, making it difficult to gather enough data for our experiments. By focusing on solo rides, we were able to observe the actual demand and supply situation without any concerns about the algorithms used by taxi service providers to match ride requests for shared rides in the dataset.

Accumulating window (λ): In the dataset, the trip start and end time are rounded to the nearest 15-minute interval to preserve the passengers' privacy. We consider λ=15 minutes and divide all the trips into 15-minute intervals depending on their start time. We randomly pick 30 requests in each 15-minute interval from the dataset.

Drivers: The dataset does not contain the location of drivers before the trips are assigned to them. Therefore, the data points for drivers are generated synthetically. We fix $|\mathcal{D}| = 50$. Randomly chosen 50 locations on the Chicago city map are fixed as CL_v. Initially, $DT_v = 0\ \forall v \in \mathcal{D}$.

Fairness threshold (κ) for Algorithm 2: In each iteration, we fix $\kappa = 0.6 \times \max_{v\in\mathcal{D}^t}(\mathcal{X}(v, M_2))$. We try to reduce the existing unfairness by 0.6 times.

Procedure in every iteration: We assume each drivers' average speed is 27 miles/hr. For each iteration, we pick a 15-minutes chunk of trips to be allocated as $\mathcal{R}^t$. Each driver $v_k \in \mathcal{D}$ is checked whether v_k is already busy in providing service to the ride assigned to it in previous iterations or the driver is free. This we do by comparing the time passed since the ride was given to v in previous iterations, assuming the speed is 27 miles/hr. If $v \in \mathcal{D}$ has already completed the service or is free, it is added to $\mathcal{D}^t$. Hence, we get $\mathcal{D}^t$ consisting of all the available drivers for t. We apply algorithms with input $(\mathcal{R}^t, \mathcal{D}^t)$. The procedure in every iteration is summarized in Algorithm 3.

Algorithm 3 Procedure in each iteration

1: Pick a 15-minutes chunk of trips to be allocated as $\mathcal{R}^t$.
2: Add the unmatched rides from $\mathcal{R}^{t-1}$ to $\mathcal{R}^t$.
3: **for** $\forall v \in \mathcal{D}$ **do**
4: Check whether v is busy in providing service.
5: **if** already completed the service **then**
6: Add v to $\mathcal{D}^t$.
7: Apply Algorithm 1(or, Algorithm 2) for $(\mathcal{R}^t, \mathcal{D}^t)$.
8: Find the unmatched rides in $\mathcal{R}^t$.
9: Update $v = (CL_v, DT_v), \forall v \in \mathcal{D}$.

State-of-the-art algorithm for comparison: We apply the procedure on the algorithm (we call, **BZ**) proposed by Bei and Zhang [1] for $OBJ1$. **BZ** guarantees to achieve a 2.5 approximate solution for $OBJ1$ in polynomial time.

5.2 Comparison metrics

We compare **BZ**, **EMFAA**, and **MRAA** based on the following metrics:

Cumulative cost: Comparing cumulative costs by the algorithms over the iterations reveals the behavior of the algorithms for efficiency over the long run.

Increase in fairness: We compare $\max_{v\in\mathcal{D}} DT_v^t$ as *unfairness* after every iteration, corresponding to the allocations by **EMFAA**, **MRAA**, and **BZ**.

Price of fairness: We call the loss in efficiency for aiming for fairness the 'price for fairness' achieved. Finding the most efficient 3-D assignments is computationally expensive, therefore, for comparison with **EMFAA** and **MRAA**, we consider *lower bound (LB)* for $OBJ1$ in each iteration. *LB* is computed as the sum of the cost of minimum weighted matching among the nodes $\mathcal{R}^t$ and that of minimum weighted perfect bipartite matching between $\mathcal{R}^t$ and $\mathcal{D}^t$. For **EMFAA** and **MRAA**, the lower bound of efficiency might be different (denoted by **LB-E** and **LB-R**, respectively) for every $t > 1$. This is because CL_v for $v \in \mathcal{D}^t$ for **EMFAA** and **MRAA** depends on the outcome by these algorithms in previous iterations $(1, \ldots, t-1)$.

5.3 Results

The results after comparing **EMFAA**, **MRAA**, and **BZ** are summarized in Figure 4. Figure 4a shows that the cumulative costs by **EMFAA** and **MRAA** follow a similar pattern as that by **BZ**. Moreover, at the end of three hours duration, **BZ** results in less total cost than others. Although **BZ** only aims for OBJ1, sometimes (9-10 AM) **BZ** is worse than **MRAA**. This is because **BZ** is optimizing only for each time step, which can be suboptimal for the cumulative costs.

Figure 4b shows the 'gain in fairness' for the matching by **EMFAA** and **MRAA** in comparison with that by **BZ**. In initial iterations, none of the three algorithms is consistently better than the others. However, over the long run, we see **EMFAA** and **MRAA** perform better than **BZ**. This is because they consider the unfairness in past allocations as well. The allocation in iteration t by an algorithm is dependent on the allocation in previous iterations. Therefore, comparing $\max_{v\in\mathcal{D}^t} DT_v^t$ for a particular iteration by any two algorithms may not be a better way to analyze their performance. However, over time increase in $\max_{v\in\mathcal{D}^t} DT_v^t$ hints us the increase in unfairness. Figure 4b shows that the increase in $\max_{v\in\mathcal{D}^t} DT_v^t$ is slower by **EMFAA** and **MRAA** than **BZ** which leads to improved fairness at the end of the three-hour interval.

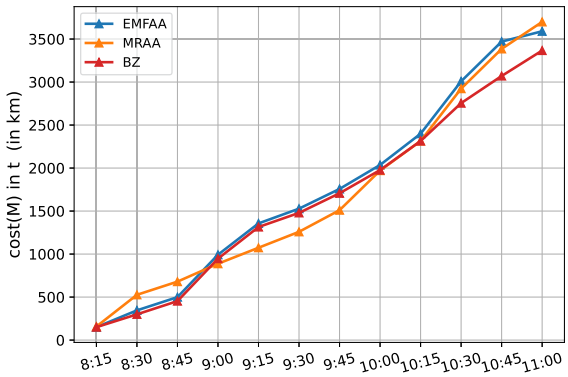
Figure 4c shows 'loss in efficiency' in matching given by **EMFAA** and **MRAA** in each iteration, comparing with their respective *LB* of the efficiency. The approximation factor concerning the *LB* is at most 1.2, which means the efficiency by **EMFAA** and **MRAA** is at most 2.2 times their *LB*. This hints that the matching provided by **EMFAA** and **MRAA** does not lose much efficiency. Notice that **BZ** provides at most 2.5 approximation factor for efficiency.

Comparing the running time, **EMFAA** is faster than **BZ**, and **MRAA** is most expensive. For instance, in a particular iteration for $|\mathcal{D}^t| = 50, |\mathcal{R}^t| = 30$, **EMFAA**, **BZ**, and **MRAA** took around $57, 89, 168$ minutes, respectively.

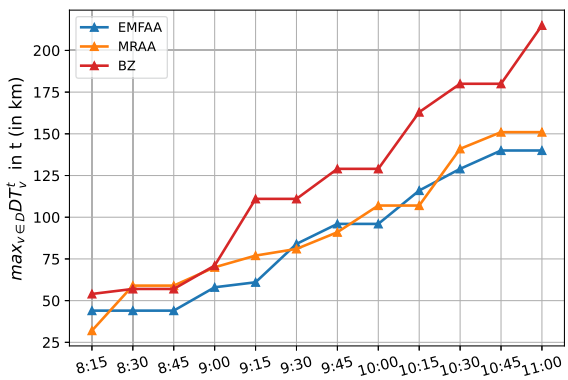
In summary, Figure 4 shows that a little attention to fairness can bring significantly fair allocation for drivers while not losing much efficiency.

6 Conclusion

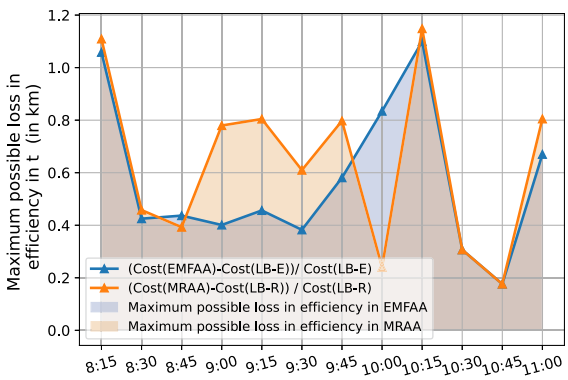
From the theoretical analysis in the paper, we conclude that simultaneously achieving efficiency and fairness is infeasible or challenging.



(a) Cumulative cost by **EMFAA**, **MRAA**, and **BZ**.



(b) Gain in fairness by **EMFAA** and **MRAA**.



(c) Loss in efficiency by **EMFAA** and **MRAA**.

Figure 4: Results on dataset for trips between 8-11 AM on April 4, 2022 in Chicago city.

However, from the experimental analysis on proposed algorithms, we conclude that a bit of consideration towards fairness while aiming for efficiency can result in a fine balance between the two objectives. Theoretical analysis of the approximation factor given by the proposed algorithms is an exciting direction to follow. In the future, we plan to run experiments on new, and variants of the proposed algorithms on a more extensive data set. We expect the experiment results for a more extensive data set will have a similar pattern.

Acknowledgements

We extend our gratitude to the multi-agent laboratory members at Kyushu University for their insightful discussions and comments. Shakya is supported by a project funded by Grants-in-Aid for Scientific Research, Japan Society for the Promotion of Science. We also appreciate Rahul Jain's valuable feedback and comments.

References

[1] Xiaohui Bei and Shengyu Zhang, 'Algorithms for trip-vehicle assignment in ride-sharing', *Proceedings of the AAAI Conference on Artificial Intelligence*, **32**(1), (Apr. 2018).

[2] Eszter Bokányi and Anikó Hannák, 'Understanding inequalities in ride-hailing services through simulations', *Scientific reports*, **10**(1), 1–11, (2020).

[3] Hua Cai, Xi Wang, Peter Adriaens, and Ming Xu, 'Environmental benefits of taxi ride sharing in beijing', *Energy*, **174**, 503–508, (2019).

[4] Brian Caulfield, 'Estimating the environmental benefits of ride-sharing: A case study of dublin', *Transportation Research Part D: Transport and Environment*, **14**(7), 527–531, (2009).

[5] Chicago Data Portal. Taxi trips. https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew, 2022. Accessed: 2022-09-20.

[6] Nicholas Diakopoulos. How uber surge pricing really works. https://www.washingtonpost.com/news/wonk/wp/2015/04/17/how-uber-surge-pricing-really-works/?utm{_}term=.b16b8c1e885d, 2015. Accessed: 2022-10-15.

[7] Fairwork. Fairwork india ratings 2020: Labour standards in the platform economy. https://fair.work/en/fw/publications/fairwork-2020-annual-report/, 2020. Accessed: 2022-10-13.

[8] Harold N Gabow, 'Data structures for weighted matching and nearest common ancestors with linking', in *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pp. 434–443, (1990).

[9] Michael R. Garey and David S. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W. H. Freeman & Co., USA, 1990.

[10] Dries Goossens, Sergey Polyakovskiy, Frits CR Spieksma, and Gerhard J Woeginger, 'Between a rock and a hard place: the two-to-one assignment problem', *Mathematical methods of operations research*, **76**(2), 223–237, (2012).

[11] Jonathan V. Hall and Alan B. Krueger, 'An analysis of the labor market for uber's driver-partners in the united states', *ILR Review*, **71**(3), 705–732, (2018).

[12] Yongzheng Jia, Wei Xu, and Xue Liu, 'An optimization framework for online ride-sharing markets', in *2017 IEEE 37th international conference on distributed computing systems (ICDCS)*, pp. 826–835. IEEE, (2017).

[13] Dan Kedmey. This is how uber's 'surge pricing' works. https://time.com/3633469/uber-surge-pricing/, 2014. Accessed: 2022-10-13.

[14] Nixie S Lesmana, Xuan Zhang, and Xiaohui Bei, 'Balancing efficiency and fairness in on-demand ridesourcing', in *Advances in Neural Information Processing Systems*, eds., H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, volume 32. Curran Associates, Inc., (2019).

[15] Kelin Luo and Frits C. R. Spieksma, 'Approximation algorithms for car-sharing problems', in *Computing and Combinatorics*, eds., Donghyun Kim, R. N. Uma, Zhipeng Cai, and Dong Hoon Lee, pp. 262–273, Cham, (2020). Springer International Publishing.

[16] Vedant Nanda, Pan Xu, Karthik Abhinav Sankararaman, John Dickerson, and Aravind Srinivasan, 'Balancing the tradeoff between profit and fairness in rideshare platforms during high-

demand hours', in *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 2210–2217, (2020).

[17] John Rawls, *A Theory of Justice*, Belknap Press of Harvard University Press, Cambridge, Massachussets, 1 edn., 1971.

[18] Yifan Xu and Pan Xu, 'Trade the system efficiency for the income equality of drivers in rideshare', in *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI-20*, ed., Christian Bessiere, pp. 4199–4205. International Joint Conferences on Artificial Intelligence Organization, (7 2020). Main track.

[19] Longxu Yan, Xiao Luo, Rui Zhu, Paolo Santi, Huizi Wang, De Wang, Shangwu Zhang, and Carlo Ratti, 'Quantifying and analyzing traffic emission reductions from ridesharing: A case study of shanghai', *Transportation Research Part D: Transport and Environment*, **89**, 102629, (2020).

[20] Biao Yin, Liu Liu, Nicolas Coulombel, and Vincent Viguié, 'Appraising the environmental benefits of ride-sharing: The paris region case study', *Journal of Cleaner Production*, **177**, 888–898, (2018).