Fairness-Aware Ride-sharing Assignment

Leshu Yuan[1,†], Ting Wang[1,†], Jianye Yang[1,2,✉], and Dian Ouyang[1]

[1]Cyberspace Institute of Advanced Technology, Guangzhou University
[2]Department of New Networks, PengCheng Laboratory
leshuyuan@e.gzhu.edu.cn, GIANTtina@163.com,
jyyangg@gzhu.edu.cn, dian.ouyang@gzhu.edu.cn

Abstract. With the rapid development of location based services, ride-sharing has become very popular in our daily life. Although the fairness has been considered in existing ride-sharing relevant tasks, they only focus on the overall revenue rather than the individual driver revenue. To fill in this research gap, in this paper, we propose a novel Fairness-aware Ride-sharing Assignment (FRA) model. In this model, we aim to maximize the lowest revenue among all drivers, and simultaneously minimize the total travel distance of the drivers. We theoretically show that it is NP-hard to find the optimal solution for our problem. To solve this problem, we first an approximation algorithm called Routing and Assignment (RA), which strikes a balance between efficiency and optimality. Observing that the efficient task assignment strategy of RA may not consider the drivers revenue adequately. We further propose an algorithm called Grouping, Assignment, and Routing (GAR). Our empirical results show the superior performance of our algorithms in comparison to existing methods focused on optimizing total distance.

Keywords: Ride-sharing · Task Assignment.

1 INTRODUCTION

With the rapid development of sharing economy, ride-sharing has become prevalent in recent years, which enables drivers to serve multiple passengers simultaneously [9,33]. Major taxi providers offer "carpooling" options to support this model [28], which can enhance transportation efficiency, reduce congestion and pollution [12], and lower costs for passengers while increasing provider profits.

Research on ride-sharing often focuses on algorithms for optimizing passenger-to-vehicle assignments to maximize benefits or meet specific objectives [14,18,22], typically aiming to minimize total distance or consider passenger preferences [8,10,21]. Salary is crucial for workers, influencing job-switching behavior when better pay arises [29]. Drivers expect monetary rewards for task fulfillment. However, assignments from the perspective of providers or passengers may overlook vehicles that are consistently unassigned [7].

† Equal contribution.
✉ Corresponding author.

The fairness of task assignment from the drivers' perspective is widely considered. Among the works which take drivers' revenue into consideration, [36] aims to maximize the overall utility which includes requester's utility, platform's utility and driver's utility. [21] formulates the matching utility based on social comfort and price revenue. [19] provides a solution to simultaneously weigh platform efficiency and driver fairness. [7] considers minimizing the maximum fairness cost while maximizing the total utility. The utility mentioned before can be any representation of performance such as task suitability and worker travel cost [8,30]. We note that there is no study to directly consider the issue of minimum revenue for drivers under the constraint of the total travel distance. In this work, we focus on optimizing the total vehicle distance traveled and the minimum driver revenue in ride-sharing platforms. Below is a concrete example of our proposal.

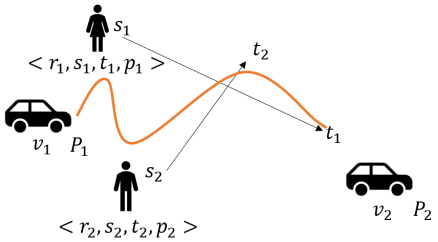

Fig. 1: A Motivation Example.

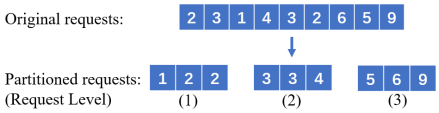

Fig. 2: Example of Level Partition.

Example 1. In Figure 1, v_1 and v_2 are located at P_1 and P_2 respectively, while passengers r_1 and r_2 request traveling from s_1 to t_1 and s_2 to t_2, respectively. With payments p_1 and p_2, v_1 is assigned to transport both passengers to minimize commute distance and waiting time, earning $p_1 + p_2$, while v_2 is left unassigned and receives no payment. If this situation continues for v_2, the driver may eventually leave the platform due to limited revenue, which is harmful for both drivers and the platform.

The above-mentioned unfairness motivates us to propose a new objective to ensure both the platform profits and driver equality. In specific, we define driver fairness by maximizing the minimum revenue among drivers across assignments. This objective also minimizes total vehicle travel distance. We formally define this bi-objective as the Fairness-aware Ride-sharing Assignment (FRA) problem, which aims to maximize drivers' minimum revenue and minimize total travel distance. Our contributions are summarized as follows:

- We propose a new problem called the Fairness-aware Ride-sharing Assignment (FRA) problem and offer a formal problem definition.
- We propose two heuristic algorithms, namely Routing and Assignment (RA) and Grouping, Assignment, and Routing (GAR) to solve FRA.

- We have conducted extensive experiments on synthetic and real datasets to show the efficiency and effectiveness of our proposals.

2 Preliminaries

In this section, we introduce the problem of fairness-aware ride-sharing assignment, which aims to assign passengers to the most suitable vehicles under two constraints, namely vehicle capacity and passengers' embarking and disembarking orders. Firstly, each vehicle can only take a limited number of passengers at the same time. Secondly, each passenger's embarking point must be served before their disembarking point to conform the common sense.

Let R be a set of n requests (i.e., passengers), where each request $r_i = (s_i, t_i)$ consists of a pickup location s_i and a drop-off location t_i, and each request has a predefined payment p for this trip. Let V be a set of m vehicles, where each vehicle $v \in V$ has a department location P_v and a capacity c. The goal is to find an assignment A from vehicles to requests. An assignment, which ensures that all requests are serviced, is a collection of walks, each of which departs from its current location to deliver a set of requests from their pickup locations to drop-off locations.

Definition 1 (Vehicle walk). *Given a set of requests R_w assigned to a vehicle $v \in V$, a vehicle walk is a sequence $W_v =< l_0 = P_v, l_1, l_2, \ldots, l_t >$, starting at the origin location of vehicle v, where $l_i \in s_r : r \in R_v \cup t_r : r \in R_v, 1 \leq i \leq t$. A vehicle walk W_v is feasible if*

(1) $\forall r \in R, s_r$ appears before t_r in W_v; and
(2) at any time point of the vehicle walk, the corresponding vehicle carries at most c passengers.

Further, the cost of a walk W_v is defined as $cost(W_v) = \sum_{i=0}^{t-1} d(l_i, l_{i+1})$, which is essentially the travel distance of each vehicle. A feasible assignment should deliver all requests, while ensuring that all vehicle walks are feasible.

Definition 2 (Fairness-aware Ride-sharing Assignment, FRA). *Given a set R of n requests, and a set V of m vehicles with a capacity c, the bi-objective is to minimize the total length of vehicle walks and maximize the lowest revenue of the vehicles under the constraint that each vehicle carries at most c passengers at any point along the walk. The bi-objective is defined formally as follows.*

$$OBJ(R, V) = \max((1-\alpha) \min_{v \in V} \varphi - \alpha \sum_{v} \frac{\mu}{|V|}) \tag{1}$$

where

$$\varphi = \frac{Revenue_v}{Revenue_{max} + 1}, \mu = \frac{Dist_v}{Dist_{max} + 1},$$

α is a weighting parameter, $Revenue_v$ is the service revenue of vehicle v, and $Dist_v$ is the distance traveled by vehicle v after moving each assigned requests

from its origin s_i to its destination t_i. Note that, $Revenue_{max}$ and $Dist_{max}$ are the maximum revenue and longest distance traveled among all vehicles.

A solution must satisfy the constraint that the passenger load of each vehicle does not exceed the capacity of the vehicle, that is, $\forall v \in V, |R_v| < c$ where R_v is the passenger load of vehicle v at a specific moment. Additionally, in the route, the starting point of each request must appear after its corresponding endpoint. Moreover, each request is exclusively handled by one vehicle.

Example 2. Consider the running example in Figure 1 with two vehicles originally located at P_1 and P_2, and two passengers r_1 and r_2. Suppose the vehicle capacity $c = 2$ and the distance metric is denoted as d. There are 2^2 possible assignments, and a vehicle can have many different orders to serve the assigned requests. For example, if r_1 and r_2 are assigned to vehicle v_1, there are 6 feasible walks: $\langle P_1, s_1, t_1, s_2, t_2\rangle$, $\langle P_1, s_1, s_2, t_1, t_2\rangle$, $\langle P_1, s_1, s_2, t_2, t_1\rangle$, $\langle P_1, s_2, t_2, s_1, t_1\rangle$, $\langle P_1, s_2, s_1, t_2, t_1\rangle$, and $\langle P_1, s_2, s_1, t_1, t_2\rangle$.

The cost of an assignment is the total length of vehicle walks. For example, the cost of walk $\langle P_1, s_1, t_1, s_2, t_2\rangle$ is $d(P_1, s_1) + d(s_1, t_1) + d(t_1, s_2) + d(s_2, t_2)$. The revenue of v_1 in this walk is $p_1 + p_2$, the revenue of v_2 is 0. Thus the largest minimum revenue of the two vehicles is 0. Then, the optimal solution is the minimum length walks with the largest minimum revenue of drivers that serve all requests. In this case, walk set $\{\langle P_1, s_1, t_1\rangle, \langle P_2, s_2, t_2\rangle\}$ can be a candidate solution which considers both the total distance and the lowest revenue of drivers.

Theorem 1. *The problem of FRA is NP-hard.*

Proof. Consider a special case of FRA where the source and destination of each passenger co-locate, and $\alpha = 1$ in Equation 1. Clearly, it is the Travelling Salesman Problem, which is a well-known NP-hard problem. Thus FRA is NP-hard.

3 Routing and Assignment, RA

3.1 Motivation

Inspired by a ride-sharing insertion approach [31], which uses dynamic programming to minimize additional route distance, we adapt this technique solve FRA to minimize total travel distance and maximize the minimum revenue of drivers. However, the insertion method alone does not take the revenue into consideration. To address this, we propose an approximation algorithm called Routing and Assignment (RA), which aims to strike a balance between route efficiency and fair revenue distribution. RA assigns payment requests to levels and ensures that each driver serves requests across these levels, trying to increase the minimum revenue among drivers. By inserting locations based on payment diversity rather than focusing solely on a single level, RA iteratively improves route planning and assignment. This method assigns each request to a specific vehicle, progressively optimizing routes to meet both distance and revenue objectives.

Algorithm 1 Level Partition

Input: Request set R of n requests, Vehicle number m and capacity c
Output: A requests level set L
Sort R
$L \leftarrow R$
$levels \leftarrow 0$
for $i = 1$ to n **do**
    if $i\%m = 0$ **then**
        $levels \leftarrow levels + 1$
    end if
    $L_i.level \leftarrow levels$
end for

3.2 Partitioning the level

To achieve a balanced assignment, requests are categorized into levels by first sorting them in ascending payment order, then constructing $\lceil \frac{n}{m} \rceil$ levels, and assigning levels via round-robin allocation (see Algorithm 1). This partitioning ensures that each vehicle serves requests from all levels, aiming to maximize the minimum revenue across vehicles.

Example 3. Consider the example in Figure 2 with 9 *requests, where the number denotes the payment of the corresponding request. Suppose there are* 3 *vehicles available. According to the Level Partition algorithm, the requests are initially sorted based on the payment amounts and then partition into* $\frac{9}{3} = 3$ *levels. As a result, in this example, requests with payment amounts of 1, 2, and 2 are labeled as level 1, requests with payment amounts of 3, 3, and 4 are labeled as level 2, the remaining requests are labeled as level 3.*

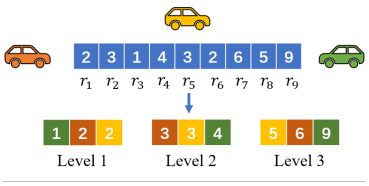

Fig. 3: Example of RA.

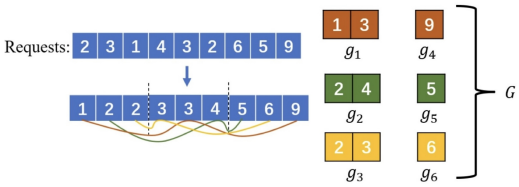

Fig. 4: Example of Grouping.

3.3 Routing and Assignment

We next discuss the details of RA algorithm. We begin by enumerating each vehicle, calculating for each one the distance between its current location and

the pickup and dropoff locations of each request. Subsequently, we sort these distances and select the top $\lfloor \frac{n}{m} \rfloor$ requests whose pickup location is nearest to the vehicle. According to the pickup service rules, a vehicle will only serve requests that have not been handled by other vehicles. Next, the vehicle moves to the nearest selected pickup location.

We use a queue S to determine if the level of a new request has already been served by the current vehicle. If not, the vehicle picks up the request. Otherwise, it proceeds to the next nearest unserved request. Dropoff locations are only served once the corresponding pickup for that request has been completed. Throughout this routing, capacity constraints are enforced to ensure the vehicle's total requests do not exceed its capacity. We also accumulate the distance traveled between each location visited for pickup or dropoff.

Algorithm 2 Routing and Assignment

Input: Request set R, Requests level set L, Vehicle set V, capacity c
Output: A feasible walk w
for each $v \in V$ **do**
    $w_0 \leftarrow \emptyset$
    $s \leftarrow$ initial location of v
    Append s to w_0
    $R_N \leftarrow$ the top $cnt = \lceil \frac{n}{m} \rceil$ nearest not served requests of v
    $Q \leftarrow$ priority queue of pickup and dropoff locations, sorted by distance from v
    $C \leftarrow$ empty
    while Q is not empty **do**
        $l \leftarrow Q.top()$
        Q.pop()
        if l is a pickup location and $|C| < c$ **then**
            Append l to C and w_0
        else if l is a dropoff location of r and $r \in C$ **then**
            Remove r from C and append l to w_0
        else
            Add l to S
        end if
        if Q is empty and S is not empty **then**
            move locations from S to Q
        end if
    end while
    Append w_0 to w
end for
return w

Example 4. Figure 3 illustrates the Level Partition approach with nine requests, r_1-r_9, represented by payment. Distances between vehicles and requests, as well as among requests themselves, are proportional to actual geographical distances. Three vehicles (orange, yellow, and green) are positioned nearby. After categoriz-

ing requests by payment levels, the orange vehicle starts with the nearest level 1 request, r_1, then moves to the next unserved level at r_2, bypassing r_3-r_6 as those levels are already served, and resumes at r_7. The yellow and green vehicles follow this pattern, each serving the nearest unserved requests at new levels, maximizing minimum revenue by ensuring varied level service. Figure 3 shows the requests each vehicle serves under Algorithm 2.

Complexity 1 *In the Routing and Assignment algorithm, a priority queue Q is used and updated dynamicly. During the process of priority queue, the operation of removing and appending uses constant time, the main time depends on the number of while recursions. In the worst case, the dropoff location always occurs before the corresponding pickup location in Q and the dropoff locations are saved to wait another recursion. Each vehicle is supposed to be assigned at most $\lceil \frac{n}{m} \rceil$ requests. Suppose the size of Q is $2 * \frac{n}{m}$, the total number of recursions is $2 * \frac{n}{m}/2c = \frac{n}{m}/c$. Traversing all the not processed locations in each recursion costs $O(\frac{n}{m})$. As there are m outlier recursion, the overall time complexity is $O(\frac{n^2}{m})$.*

4 Grouping, Assignment and Routing, GAR

4.1 Motivation

The RA Algorithm allows each vehicle to identify all necessary requests during route planning, thereby forming a distinct request group per vehicle. However, because it prioritizes proximity and payment levels, the fairness of driver revenue is still not guaranteed. To address this limitation, we propose a new algorithm that prioritizes the formation of request groups first, followed by allocation to vehicles and subsequent route planning.

In this section, we introduce the GAR algorithm, which also begins with forming request groups and allocates them to vehicles, but with a focus on maximizing the minimum revenue among drivers. First, requests are grouped to achieve revenue balance. Then, using the minimum spanning tree algorithm, each group is rooted by the nearest vehicle, which then serves as the root for that tree. GAR additionally diversifies requests by payment levels, ensuring each driver serves a range of payment levels, using a traveling salesman algorithm to optimize the route per group. This approach enables GAR to meet both route efficiency and minimum revenue objectives more effectively.

4.2 Grouping

We still sort the requests in ascending order based on their payment values and partition the requests into $\frac{n}{m}$ request sets. Within each set, the request intended for assignment to that set shifts to the next position on the right, and the last request among the m requests moves to the first position compared to the previous set of m requests. For example, as shown in Figure 4, in the first round, the first vehicle is assigned the first request; however, in the second round, the first vehicle is assigned the second request. The grouping algorithm is shown in Algorithm 3.

Algorithm 3 Grouping

Input: Request set R of n requests, Vehicle number m and capacity c
Output: A request group collection G
Sort R by the payment
$L \leftarrow R$
$G \leftarrow \emptyset$
$j \leftarrow 0$
for $i = 1$ to n **do**
    if $i\%m = 0$ **then**
        $j \leftarrow j + 1$
    end if
    $t \leftarrow \lfloor \frac{i}{m} \rfloor + (i + j)\%m$
    $s \leftarrow i\%m$
    while the size of the s-th group of G is c **do**
        $s \leftarrow s + m$
    end while
    Append t-th request to the s-th group of G
end for

4.3 Routing

In this section, we assign the grouped requests to vehicles. Firstly, we assign the nearest request groups which we aim to allocate to one specific vehicle and are not allocate to other vehicles to the vehicle v. After that, there are at least $\lfloor \frac{n}{mc} \rfloor$ request groups which are assigned. We build the minimum spanning tree (MST) of groups allocated to v, where v is the root of the tree. In the minimum spanning tree, a request or a vehicle is a node, and the distance between two requests are denoted as their weight. We plan to plan the route using TSP in a depth first algorithm based on the minimum spanning tree. The Routing algorithm is displayed in Algorithm 4.

Finding the minimum spanning tree. For each vehicle v, We will build the minimum spanning tree for the groups in group collection G which is output by the Grouping algorithm and assigned to v. These request groups that are allocated to the vehicle v are denoted as g_v. We construct a complete graph using nodes representing vehicle v and the request groups assigned to v to generate a minimum spanning tree, with the vehicle nodes serving as the root of the tree. Therefore, in the specific construction, the information required includes the initial position P_v of the vehicle v, as well as the pickup location s of the requests assigned to each vehicle. These details will be utilized to determine the edges and weights of the graph, ultimately leading to the generation of the minimum spanning tree. For the edge cost C, we define as follows: recall d is the underlying metric, and let C be

$$\left.\begin{aligned} C(P, P') &:= \min_{r_i \in P, r_j \in P'} d(s_i, s_j), \quad P, P' \in g_v \\ C(P_v, P) &:= \min_{r_i \in P} d(P_v, s_i), \qquad P \in g_v \end{aligned}\right\} \tag{2}$$

Algorithm 4 Routing

Input: Request group collection G of n requests, Vehicle set V of m vehicles and capacity c
Output: A feasible walk w
$G_v \leftarrow G$ with the nearest vehicle of each group
for each $g \in G_v$ **do**
    Let C be given as in Equation 2
    Find the minimum spanning tree on g with cost function d
    $w_0 \leftarrow \mathrm{DFS}(P, T)$
    Append w_0 to w
end for
return w

DFS on MST to serve the requests. The Depth-First Search (DFS) in Algorithm 4 is adapted from that in the reference [22]. DFS serves the requests on the tree following a depth first strategy along the edges of the tree. The edges of the tree are portals connecting two groups. The vehicle follows the portals to traverse the tree. The vehicle enters a group at the portal connecting to its parent. For each group, the walk is predetermined and the vehicle serves requests in the group accordingly. This process is recursive, as the vehicle traverses its children groups before returning to the parent group once it has served all requests within the current group. We omit the details of DFS due to limited space.

Complexity 2 *The complexity of GAR consists of two part, namely Grouping and Routing separately. In the Grouping part, the outer loop iterates n times, and within each iteration, there is a goal to find the group with size smaller than the vehicle capacity, where the worst-case time complexity is* $O(\frac{n}{c})$*. Consequently, the overall time complexity is* $O(n^2)$*. In the Routing part, there are m vehicles. For each vehicle, the computation runs in time* $O((\frac{n}{m})^2 c^2 logc)$*. Thus the total time complexity of Routing is* $O(\frac{n^2}{m} c^2 \log c)$*.*

5 Computational Experiments

5.1 Datasets and Baseline

We use the datasets for existing request assignment and route planning approach [22]. For datasets lacking payment information [23], we generate random payments for each request, setting a minimum payment of 3, with a mean of 15 and a standard deviation of 5.

Synthetic datasets. To assess the performance of RA and GAR algorithms, we benchmark them on two synthetic datasets, SY-U and SY-G, focusing on solution quality and runtime efficiency. In SY-U, locations for pickups, dropoffs, and drivers' initial positions are uniformly distributed across a $[0, 100]^2$ grid. For SY-G, locations are derived from a Gaussian mixed-model (GMM) with Z clusters, each centered on a $[0, 100]^2$ grid and characterized by a covariance

Table 1: Parameter settings for the datasets. Bold values indicate fixed parameter values for the sensitivity analyses.

Parameter	SY-U, NYC and SF	SY-G
n	2, 4, 6, **8**, 10 ($\times 10^3$)	2, 4, **6** ($\times 10^3$)
m	30, 60, **90**, 120, 150	30, 60, **90**
c	2, 4, 8, 16, **32**, 64	4, 8, **16**
Z		5, **10**, 20, 50, 100
σ		5, 10, 25, **50**, 100, 250

matrix $\sigma^2 I$, reflecting diverse demand clusters typical of sparse urban settings. The performance is evaluated across different parameter combinations of Z and σ for SY-G, as detailed in Table 1.

Realworld datasets. Transportation data from New York City (NYC) and San Francisco (SF) makes up of our realword datasets for the algorithms testing.

- NYC: The NYC dataset consists of 10,000 random trip records from NYC Taxi & Limousine Commission Trip Record Data [23] in May/2016.
- SF: The SF dataset consists of 10,000 random trip records from the origin dataset of Cab Spotting Data [26], which records approximately 500 taxis' trace data in a period of 30 days.

The location is specified by its latitude and longitude coordinates in these datasets. The distance between two locations is defined to be the graph (shortest-path) distance calculated via the actual road map of the two cities, which can be obtained from the OpenStreetMap database [25]. All parameter settings are detailed in Table 1. Note that we set the α in our objective function as 0.5.

Metrics and Baselines. We measure the performance of the algorithms with respect to two objectives. The first is OBJ, which is what RA and GAR are set to optimize. The second objective is the running time, which is crucial for the real execution. We adopt HGR [22] as the baseline in our experiments. It optimizes the total travel distance of vehicles and gives an $O(\sqrt{c}\log n)$ approximation guarantee and is also the state-of-the-art under this optimization.

5.2 Computational Results

Results on synthetic datasets are shown in Figure 5-Figure 9. Results on real-world datasets are shown in Figure 8 and Figure 9. Our algorithms exhibit clear superiority on both objectives in almost all parameter regimes and datasets. We discuss the effect of each parameter in more details as follows.

Synthetic datasets. Figure 5 and Figure 6 illustrate RA and GAR consistently surpass HGR by two orders of magnitude in OBJ under uniform/GMM distributions. RA excels in objective optimization via pre-partitioned profit intervals, while GAR reduces runtime with simplified path planning. RA balances

efficiency and optimization, whereas GAR trades extended runtime for computational speed.

Figure 7 evaluates algorithm performance under varying GMM parameters (cluster count Z, covariance σ). The first two columns show the effect of Z, while the latter two analyze σ. Objective values and time costs remain stable across Z and σ variations, likely because shortest-path optimization is only a partial objective component, limiting spatial distribution impacts. Nonetheless, our algorithms consistently outperform HGR in objectives. In time efficiency, RA surpasses GAR by one order of magnitude and HGR by four orders, demonstrating adaptability to dynamic real-world scenarios.

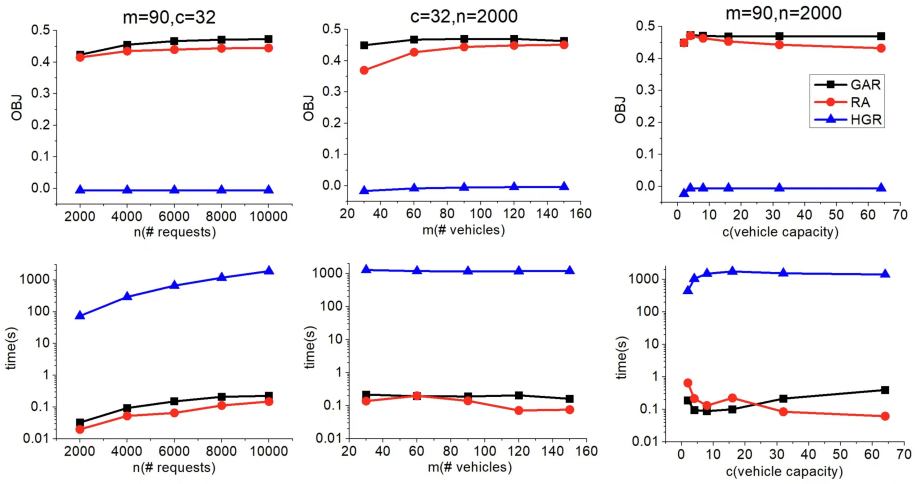

Fig. 5: Results on the Uniform synthetic dataset (SY-U): varying n, m, c.

Realworld data. Concerning the effect of n, m and c, results on the two realworld datasets as well as the synthetic datasets are quite similar, so we will take one of them as example. Figure 8 shows the result on the NYC dataset and Figure 9 shows the result on the SF dataset. The performance trends of the algorithm on two real-world datasets are highly consistent, and we specifically analyze the algorithm's performance on the NYC dataset. Both the OBJ and running time grow with n, and the gap between RA, GAR and HGR also grows with n (the first column of Figure 8). The performance of our algorithm on real-world datasets mirrors its performance on synthetic datasets, showing consistent excellence. In both types of datasets, our algorithms outperform HGR. With marginal differences in objective optimization performance compared to the GAR algorithm, RA consistently exhibits a time efficiency one order of magnitude higher than GAR. Thus, it appears that a straightforward path planning

approach and selecting all profit intervals are sufficient to address our objective problem.

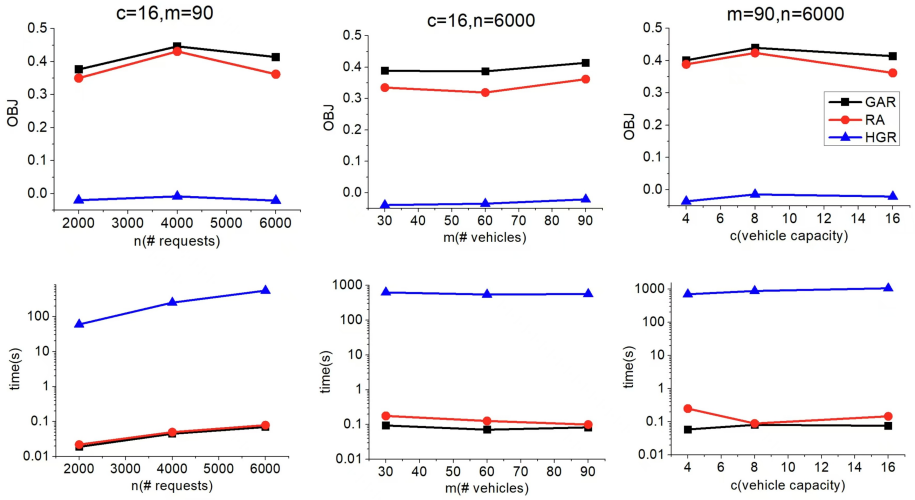

Fig. 6: Results on the GMM synthetic dataset (I): varying n, m, c.

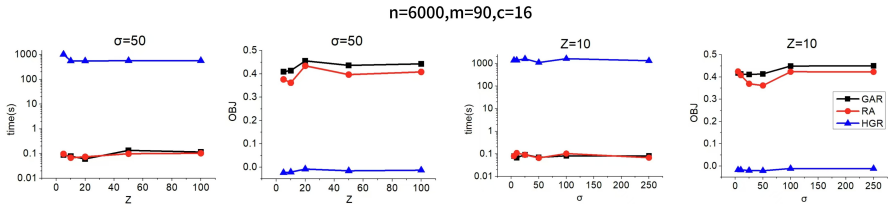

Fig. 7: Results on the GMM synthetic dataset (II): varying n, m, c.

6 Related work

6.1 The Ride sharing Problem

The dial-a-ride problem (DARP) [3,10,11,15,27] is aimed to optimize the routing of vehicles and passengers with specified origins and destinations. The goal is to minimize the total distance traveled by vehicles while ensuring that all passengers are successfully transported. DARP can be viewed as the traditional counterpart to the ridesharing problem, operating in an offline setting.

Existing works on ride sharing involve diverse optimization goals. Typically, the main objective is to find an allocation method that minimizes the total travel distance or overall travel expenses of the vehicles. Initially, vehicle allocation involved only a single vehicle [4,16], focusing on minimizing travel distance. Later, researchers expanded their focus to scenarios with multiple vehicles and requests based on real-world situations [17,22]. One optimization goal is income maximization. In [35], the objective is total platform revenue, while in [32], the focus is on driver compensation. [31,32] minimize total travel distance and penalties from rejecting requests using a dynamic programming (DP)-based insertion algorithm. [34] also uses DP insertion to optimize delivery time. [8] handnles the problem from the passenger perspective, aiming to meet their needs. [13] incorporates the option for users to reject service based on service quality.

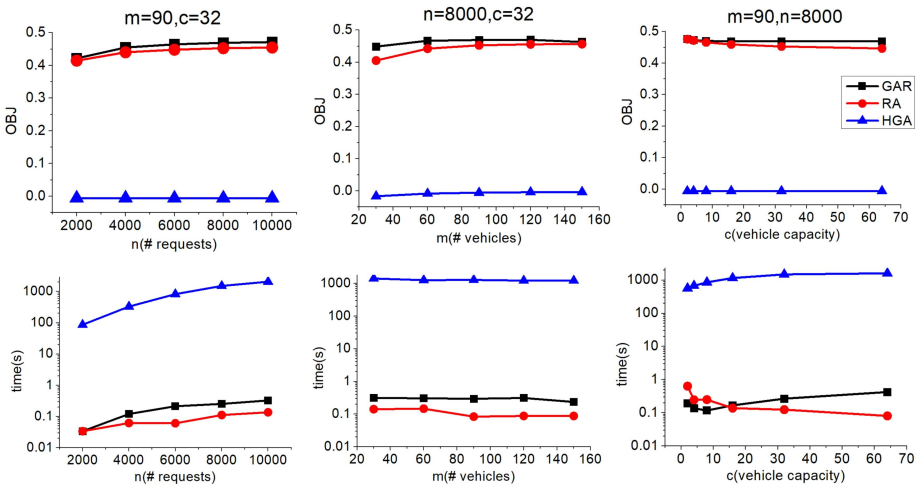

Fig. 8: Results on the NYC dataset: varying n, m, and c.

6.2 Fairness-related Ride Sharing

Among studies with respect to fairness,different research papers focus on different objects. We found research on platform income fairness in [20] while there are multiple platforms. Some works consider to split the cost of a sharable ride in a fair manner from the riders' perspective [5,6,37]. [2] introduces mechanisms such as reputation scores, time-locked deposits and smart contracts to ensure fairness when there exists malicious behavior like fare fraud and passengers not fulfilling their promises. In [24], the authors focus on the specific passengers' fairness on travel time, allowing them to vote for their expected route. [1] proposes a fair model to satisfy both the riders' and drivers' profiles. Two articles, [21] and [7], are particularly relevant to the fairness of driver income. While [21]

and [7] consider the issue of fairness in driver income, they do not address the problem of total travel distance. This paper, for the first time, discusses both the total travel distance of all vehicles and the fairness of driver income as joint objectives, aiming to provide effective solutions.

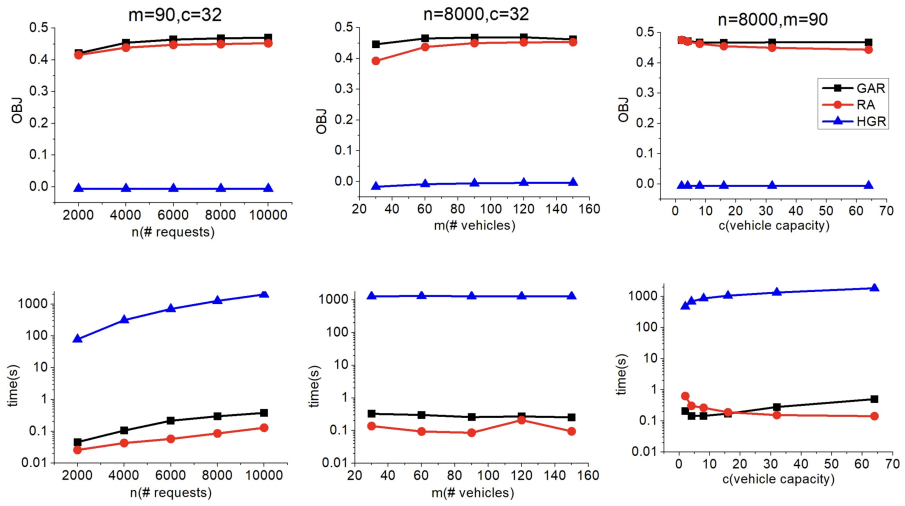

Fig. 9: Results on the SF dataset : varying n, m and c.

7 Conclusion

In this paper, we propose an objective in ride-sourcing, combining the fairness objective with shortest total distance. We provide two different algorithms with increasing runtime efficiency. We experimently demonstrate that both algorithms outperform on both synthetic and real world datasets.

Acknowledgments. This work was supported in part by the Major Key Project of PCL (PCL2024A05), in part by the National Key Research and Development Program of China (2022YFB3104100 and YS2022YFB2700093), in part by the Guangdong Basic and Applied Basic Research Foundation (2023A1515011655), and in part by the Guangzhou Education Department Foundation (2023KQNCX057).

References

1. Mohammad Asghari, Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, and Yaguang Li. 2016. Price-aware real-time ride-sharing at scale: an auction-based approach. In *Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems*. 1–10.

2. Mohamed Baza, Noureddine Lasla, Mohamed MEA Mahmoud, Gautam Srivastava, and Mohamed Abdallah. 2019. B-ride: Ride sharing with privacy preservation, trust and fair payment atop public blockchain. *IEEE Transactions on Network Science and Engineering* 8, 2 (2019), 1214–1229.
3. Claudia Bongiovanni, Nikolas Geroliminis, and Mor Kaspi. 2024. A ride timeoriented scheduling algorithm for dial-a-ride problems. *Computers & Operations Research* (2024), 106588.
4. M. Charikar and B. Raghavachari. 1998. The finite capacity dial-a-ride problem. In *Foundations of Computer Science, 1998. Proceedings.39th Annual Symposium on.*
5. Chi-Kin Chau and Khaled Elbassioni. 2017. Quantifying inefficiency of fair cost-sharing mechanisms for sharing economy. *IEEE Transactions on Control of Network Systems* 5, 4 (2017), 1809–1818.
6. Sid Chi-Kin Chau, Shuning Shen, and Yue Zhou. 2020. Decentralized ride-sharing and vehicle-pooling based on fair cost-sharing mechanisms. *IEEE Transactions on Intelligent Transportation Systems* 23, 3 (2020), 1936–1946.
7. Zhao Chen, Peng Cheng, Lei Chen, Xuemin Lin, and Cyrus Shahabi. 2020. Fair task assignment in spatial crowdsourcing. *Proceedings of the VLDB Endowment* 13, 12 (2020).
8. Peng Cheng, Hao Xin, and Lei Chen. 2017. Utility-aware ridesharing on road networks. In *Proceedings of the 2017 ACM International Conference on Management of Data*. 1197–1210.
9. Regina R Clewlow and Gouri S Mishra. 2017. Disruptive transportation: The adoption, utilization, and impacts of ride-hailing in the United States. (2017).
10. Jean-François Cordeau. 2006. A branch-and-cut algorithm for the dial-a-ride problem. *Operations research* 54, 3 (2006), 573–586.
11. Jean-François Cordeau and Gilbert Laporte. 2003. A tabu search heuristic for the static multi-vehicle dial-a-ride problem. *Transportation Research Part B: Methodological* 37, 6 (2003), 579–594.
12. Tatiana Bruce da Silva, Patrícia Baptista, Carlos A Santos Silva, and Luan Santos. 2022. Assessment of decarbonization alternatives for passenger transportation in Rio de Janeiro, Brazil. *Transportation Research Part D: Transport and Environment* 103 (2022), 103161.
13. Xiaotong Dong, David Rey, and S Travis Waller. 2020. Dial-a-ride problem with users' accept/reject decisions based on service utilities. *Transportation research record* 2674, 10 (2020), 55–67.
14. Inge Li Gørtz, Viswanath Nagarajan, and Ramamoorthi Ravi. 2015. Minimum makespan multi-vehicle dial-a-ride. *ACM Transactions on Algorithms (TALG)* 11, 3 (2015), 1–29.
15. Timo Gschwind and Michael Drexl. 2019. Adaptive large neighborhood search with a constant-time feasibility test for the dial-a-ride problem. *Transportation Science* 53, 2 (2019), 480–491.
16. Anupam Gupta, MohammadTaghi Hajiaghayi, Viswanath Nagarajan, and Ramamoorthi Ravi. 2010. Dial a ride from k-forest. *ACM Transactions on Algorithms (TALG)* 6, 2 (2010), 1–21.
17. Sin C Ho, Wai Yuen Szeto, Yong-Hong Kuo, Janny MY Leung, Matthew Petering, and Terence WH Tou. 2018. A survey of dial-a-ride problems: Literature review and recent developments. *Transportation Research Part B: Methodological* 111 (2018), 395–421.
18. Ehsan Jafari, Tarun Rambha, Alireza Khani, and Stephen D Boyles. 2016. The for-profit dial-a-ride problem on dynamic networks. In *the 95th Annual Meeting of the Transportation Research Board, Washington, DC.*

19. Nixie S Lesmana, Xuan Zhang, and Xiaohui Bei. 2019. Balancing efficiency and fairness in on-demand ridesourcing. *Advances in neural information processing systems* 32 (2019).
20. Boyang Li, Yurong Cheng, Ye Yuan, Yi Yang, QianQian Jin, and Guoren Wang. 2023. ACTA: Autonomy and Coordination Task Assignment in Spatial Crowdsourcing Platforms. *Proceedings of the VLDB Endowment* 16, 5 (2023), 1073–1085.
21. Yafei Li, Huiling Li, Xin Huang, Jianliang Xu, Yu Han, and Mingliang Xu. 2022. Utility-Aware Dynamic Ridesharing in Spatial Crowdsourcing. *IEEE Transactions on Mobile Computing* (2022).
22. Kelin Luo, Alexandre M. Florio, Syamantak Das, and Xiangyu Guo. 2023. A Hierarchical Grouping Algorithm for the Multi-Vehicle Dial-a-Ride Problem. *Proc. VLDB Endow.* 16, 5 (2023), 1195–1207. https://doi.org/10.14778/3579075.3579091
23. NYCTLC. 2020. New York City Taxi & Limousine Commission trip data. https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page.
24. Yi Cheng Ong, Nicos Protopapas, Vahid Yazdanpanah, Enrico H Gerding, and Sebastian Stein. 2024. Fair and efficient ride-scheduling: a preference-driven approach. *Journal of Simulation* (2024).
25. OpenStreetMap. 2021. San Francisco and New York City street map data. https://www.openstreetmap.org.
26. Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2009. CRAWDAD dataset EPFL/mobility. https://crawdad.org/epfl/mobility/20090224/cab.
27. Ulrike Ritzinger, Jakob Puchinger, and Richard F Hartl. 2016. Dynamic programming based metaheuristics for the dial-a-ride problem. *Annals of Operations Research* 236 (2016), 341–358.
28. Jennifer Saranow. [n.d.]. Carpooling for Grown-Ups —High Gas Prices, New Services Give Ride-Sharing a Boost; Rating Your Fellow Rider. ([n. d.]).
29. Barry Schwartz. 2023. Why we work. In *Rethinking Work*. Routledge, 31–35.
30. Hien To, Cyrus Shahabi, and Leyla Kazemi. 2015. A server-assigned spatial crowdsourcing framework. *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 1, 1 (2015), 1–28.
31. Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, and Ke Xu. 2022. Unified route planning for shared mobility: An insertion-based framework. *ACM Transactions on Database Systems (TODS)* 47, 1 (2022), 1–48.
32. Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. 2018. A unified approach to route planning for shared mobility. *Proceedings of the VLDB Endowment* 11, 11 (2018), 1633.
33. Hai Wang and Hai Yang. 2019. Ridesourcing systems: A framework and review. *Transportation Research Part B: Methodological* 129 (2019), 122–155.
34. Yi Xu, Yongxin Tong, Yexuan Shi, Qian Tao, Ke Xu, and Wei Li. 2020. An efficient insertion operator in dynamic ridesharing services. *IEEE Transactions on Knowledge and Data Engineering* 34, 8 (2020), 3583–3596.
35. Yuxiang Zeng, Yongxin Tong, Yuguang Song, and Lei Chen. 2020. The simpler the better: An indexing approach for shared-route planning queries. *Proceedings of the VLDB Endowment* 13, 13 (2020), 3517–3530.
36. Libin Zheng, Peng Cheng, and Lei Chen. 2019. Auction-based order dispatch and pricing in ridesharing. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*. IEEE, 1034–1045.
37. Ze Zhou, Claudio Roncoli, and Charalampos Sipetas. 2023. Optimal matching for coexisting ride-hailing and ridesharing services considering pricing fairness and user choices. *Transportation Research Part C: Emerging Technologies* 156 (2023),104326.