Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 5572200, 10 pages
https://doi.org/10.1155/2021/5572200

Research Article

Optimization Algorithm Design for the Taxi-Sharing Problem and Application

Yongjie Wang and Maolin Li

Yuncheng Vocational and Technical University, Yuncheng, Shanxi 044000, China

Correspondence should be addressed to Yongjie Wang; yjiewang67@outlook.com

Received 27 February 2021; Revised 17 June 2021; Accepted 28 July 2021; Published 3 August 2021

Academic Editor: Samuel Yousefi

With the development of mobility techniques, the transportation systems become smarter, pursuing higher goals, such as convenience for passengers and low cost. In this work, we investigate the taxi-sharing system, which is a promising system recently. The passengers can share the same taxis to different destinations to save cost. Considering the property of taxis' routes, the corresponding model is established and our aim is to design the trip for each taxi to reduce the total number of taxi trips in the whole system if one taxi can be shared by several passengers. Compared with the previous work, we do not have any constraint about the taxi stations. The taxi trips have more flexibility in reality. We analyze this problem and prove it is NP-Complete. There are two proposed algorithms to solve this problem, one is a heuristic algorithm and the other is an approximate algorithm. In the experiment, two real-world taxi data sets are tested, and our algorithm shows the superiority of our taxi-sharing system. Using the taxi-sharing system, the number of trips can be reduced by about 30%.

1. Introduction

In recent years, with the increment of traveling and commuting, the numbers of private cars and public transportation are increased. According to the Federal Highway Administration (FHWA)'s report, the total rural and urban vehicles travel 3,262 billion miles [1], which is a huge number. The road congestion problem [2, 3] and the limited parking spaces [4, 5] are becoming new problems for people.

With the increased price of gas line and limited parking space, an efficient car-sharing system has been introduced to a large number of people. This system is a method to share one car with more than one passenger who can follow a common route to similar or close destinations [6]. Although there are some issues about this system, such as reservation strategy, passengers having to walk to the nearest parking lot, it can also bring a lot of benefits. It makes the private transportation flexible, produce less pollution, and traffic congestion. Therefore, it is becoming more and more popular, and people are willing to engage in this new mode of transportation. In practice, the conventional car-sharing systems mainly combine the passengers who have close pick-up locations and destinations because the private cars have predetermined route and cannot stop at any location on the road. In addition, people using this system mainly focus on their commutation to the work location, which means that the routes are stable. Thus, there are a lot of restrictions for such private car-sharing systems. These restrictions make this system not practical in real life.

In this work, we mainly focus on the taxi-sharing system. For taxi drivers, their goal is to carry more passengers to destinations, obtaining more benefits. Given that taxi drivers travel all the time continuously, passengers do not need to have the close pick-up locations and destinations because they can be picked up along the way. Once passengers' trips have some continuous common routes and do not make a large detour, they can share taxis without the loss of much time. Thus, for the taxi-sharing system, it has fewer restrictions than the car-sharing system. In most of the time, passengers can provide their source and destination to the platform and according to the current trip of the taxi, the sharing trips can be dispatched to the corresponding taxi

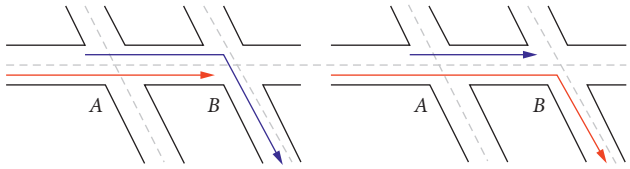

Figure 1: The red line and the blue line represent the trips of two passengers. Currently, the red passenger is picked up by the taxi driver, and the blue passenger is waiting for taxis at crossing A. In the left plot, two passengers can share one taxi because the red passenger can be dropped off at the crossing B and the taxi driver can continue to deliver the blue passenger. In the right plot, the taxi driver can drop off the blue passenger at the crossing B and continue to deliver the red passenger.

driver. In this way, the empty-load rate of a taxi can be reduced. Meanwhile, there can be fewer taxis on the road, which improves the environment.

In real-world application, this taxi-sharing system can still meet some challenges. First, passengers' requirements for taxi-sharing are different. Some passengers are sensitive to time. In the sharing process, drivers should deliver them to destinations on time. Thus, there is no detour in the process. Some passengers do not want to share with too many people, or frequently picking up other passengers. Thus, there should be some limitations on the number of passengers on taxis. How to guarantee the sharing process is very important in this system, meeting the different demands of passengers.

Second, if all the information about passengers is known, including pick-up locations, destinations, and pick-up time, it is important to find a schedule to dispatch taxi drivers to pick-up them to destinations. It is not easy to make such a schedule. In the conventional car-sharing system, passengers are connected according to their information, similar in time and space. In the taxi-sharing system, some passengers can be picked up in the middle, and do not need to be dropped off at the same destinations. Thus, for that, we wish to design the sharing modes for passengers, i.e., what kind of similarity between their trips can make them share the same taxi. There is an illustration figure shown in Figure 1. In these two cases, two passengers can be shared with one taxi if their time is matched.

Last, taxi demand is huge in city-scale. Given the pick-up location, time, and destinations of passengers, it is still hard to schedule taxi drivers and dispatch the delivery tasks. There are many traditional ways to do such schedules. At the same time, for online tasks, it becomes more difficult. We can use the greedy idea to pick-up the waiting passengers as more as possible. However, there is no guarantee for the results, which brings difficult for us to estimate the number of taxis needed. Thus, some better algorithms for the taxi-sharing scheduling problem can help us in city planning and traffic management.

Our Contribution. First, we design the taxi-sharing model based on real-world application. Given a large number of passengers in a region within a time duration, their pick-up locations, time, and destinations are known in advance or given sequentially. We assume that between any pair of pick-up location and destination, the shortest path is unique. All the passengers are sensitive to time, so the taxi drivers cannot make any detour to deliver passengers to destinations, as well as pick-up other passengers. At the same time, the sharing mode requirements are determined. For the trips of the current passengers, only when the other passengers can be satisfied to arrive at the destination on time, no matter they are dropped off earlier or later than the current passenger. This model is suitable for real-world applications. The taxi drivers just need to consider the current passengers and try to pick-up more passengers along the way. Of course, the number of passengers on the taxi is constrained, and the waiting time of passengers is also limited. If a passenger waits for a sharing taxi too long, an empty taxi will be dispatched to her.

Based on this model, our taxi-sharing problem is formally defined, which is to minimize the number of taxi-sharing trips. For the passengers who can share one taxi, they need only one taxi-sharing trip instead of the number of passengers. This problem is proven to be NP-complete, which does not have an optimal solution. Hence, we design two algorithms to solve it, one is a greedy algorithm and the other is an approximation algorithm. For the greedy algorithm, the taxi drivers are dispatched to each passenger if this passenger cannot share taxis with another passenger. Then, in the delivery process, taxi drivers pick-up all the satisfied passengers without exceeding the limitation of passengers on a taxi. In addition, this algorithm can also be applied into the online setting. For the approximation algorithm, we first try to find all the possible taxi-sharing trips, which is a time-consuming task in the raw data. To speed up this process, we convert these passengers into a graph network. Each directed edge represents whether these two passengers can share one taxi. Based on this graph, it becomes easier to obtain all possible taxi-sharing trips from the graph. Later the corresponding combinatorial algorithm is proposed, and the approximation ratio $O(\log m)$ is proved, where m is the number of passengers. However, the result of this algorithm performs well in the experiments.

In the end, we implement our model and algorithms into two real-world data set, which is taxi trajectories in San Francisco, America, and Porto, Portugal. There are more than 536 taxis in the city of San Francisco, over one month. Similarly, there are 442 taxis in the city of Porto, over one complete year. There are more than 170,000 taxi trajectories in the data set. All the trips of passengers are stored with their pick-up locations, time, destinations, and trajectories. Their trajectories are regarded as the shortest path from the pick-up location to the destinations. Thus, we check the number of trips that can be reduced if we count the taxi-sharing trip as one trip. When there are about 30,000 trips that happened in one day, in the original system, the same number of taxi trips needs to be dispatched. Now, with a taxi-sharing system, more than 10% trips can be shared with other passengers. With more trips given in the system, the probability of sharing is getting larger and the reduction is becoming more obvious, more than 20%. Similarly, if the requirement of taxi-sharing becomes loose, such as waiting time, waiting locations, and the maximum number of

passengers on a taxi, the reduction can also be enlarger, especially when there are a very large number of trips within a short period.

In the following, we start by reviewing the related work in Section 2. In Section 3, we introduce our taxi-sharing model and propose the problem definition. Then, two algorithms are provided in Section 4. The experiments are presented in Section 5, and Section 6 concludes this paper.

2. Related Work

Nowadays, there are many enterprises providing car-sharing service, e.g., ZipCar, EVCARD, Turo, Sixt, and so on. According to the rule of vehicle returning, the car-sharing system can be divided into two types [7], one is one-way and the other is round-trip [8, 9]. For the round-trip system, the car needs to be returned to the station where it is initially rented, while this limitation is removed in the one-way system. In our work, our system can be regarded as one-way system. At the same time, we also do not constrain the initial rental location. We consider the taxi that can pick up the passengers anywhere.

In recent years, many researchers began to investigate this area. The most important problem that they focus on is the vehicle dispatching problem. It is necessary for the car-sharing system. We need to consider how to deploy these cars to provide the service as soon as possible. In 2015, Nourinejad et al. [10] build an optimization model to investigate the trade-off between the vehicle fleet size and the human demand, to minimize the investigation cost. The vehicle allocation problem is also related to the capacity of the stop station [11]. For the real-time requests or online settings, some dynamic models are proposed to optimize the car dispatch problem with heuristic idea [12, 13]. With the development of the neural networks, many works [14, 15] are trying to predict the demand in real time, which can help the dispatch tasks get prepared in advance. There are some works to investigate the trajectories similarity, which can help saving the vehicles number [16]. For our work, although we do not have the limitation for the number of cars, our goal is to reduce the number of cars. According to the demand, we can dispatch the car to pick up the passengers within a comfortable waiting time.

With the development of electric cars, more and more car-sharing system began to consider the charging problems. The reason is that each electric car can only execute about ten hours, then it needs to get charged for about one hour. Thus, this charging time is a long time, which needs to be scheduled better [17]. Zhao et al. [18] present a mathematical model for the integrated electric car rebalancing and staff relocation for one-way station-based systems.

In our work, the taxi-sharing problem is different from the previous car-sharing problem. The taxi drivers do not have a fixed route and time restrictions. Thus, they can satisfy more passengers without the requirement for pick-up locations and destinations. There are many researchers studying the constraints of the taxi-sharing system. For example, the sharing agreement is based on social networks [19]. Passengers have permission to select other passengers, according to some features such as sex and age [20]. Some software systems were proposed to manage the schedules for sharing taxi and monitor whether the drivers make a detour [21].

3. Problem Definition

In this section, we introduce some preliminary concepts of our taxi-sharing system. The taxi drivers are allowed to carry more than one passenger to improve their benefit.

Given a region area Ω, many passengers are waiting for taxis to pick-up them and deliver them to their destinations. Suppose that there are m passengers in the region Ω within a long time duration, denoted $P = \{p_1, p_2, \ldots, p_m\}$. For each passenger p_i, she has her own pick-up location, pick-up time, and destination, represented by s_i, st_i, and d_i. In most of the time, it is hard to control the arrival time by taxi, but taxi drivers guarantee the passengers will be carried to destinations as soon as possible without any detour. Given a waiting time threshold δ, which is the maximum waiting for a sharing taxi, it means that it is possible for a passenger to wait for a taxi driver δ time with available spaces; otherwise, this passenger is not satisfied. For each passenger, she is willing to take a sharing taxi instead of an empty taxi because she can save some cost by sharing a trip with others.

Then let us focus on the taxi driver side. For each taxi-sharing trip, it should start from a pick-up location of one passenger and goes to the destination of another passenger. If there is only one passenger on this taxi-sharing trip, the pick-up location and the destination belong to the same passenger. If there are more passengers, the pick-up location and destination might belong to different passengers. In the taxi-sharing system, drivers are allowed to pick-up more passengers along the way, only if they can deliver passengers to their corresponding destinations on time. Thus, taxi drivers cannot make a detour for any passenger, which means that the taxis should be one of the shortest path to the destinations of all the passengers on the whole trip. In this work, we regard a continuous trip with at least one passenger as one taxi-sharing trip. If all the passengers are dropped off, this taxi-sharing trip is finished and the driver should wait for the next dispatch task. Now, we define the trip:

Definition 1 (trip). Each trip T, as a trajectory of the taxi driver within a short time, refers to a sequence of positional points that chronologically sampled during a time period, denoted as $T = \{(l_1, t_1), (l_2, t_2), \ldots, (l_r, t_r)\}$, where l_i is the location of the taxi driver at timestamp t_i and r is the length of this trip. This location l_i can be GPS location or other format of position.

Now, let us take a look at taxi-sharing trips. Given a parameter k, as the maximum number of passengers in one taxi, it is 3 or 4 for the general taxi cars. For a taxi-sharing trip, there is at least one passenger and at most k passengers on the taxi along all the way. Such trips are called taxi-sharing trips.

Fora taxi driver carrying the passenger p_i with trip T, there is $s_i = l_1$ and $st_i = t_1$, which means that the pick-up location and time of p_i are the start point of the trip T. Then,

along the way, the drive can meet another passenger p_j. If this driver wants to pick up the passenger p_j to her destination, there are two conditions, pick-up condition and destination condition.

The pick-up condition is easy to understand. When the taxi driver arrives at the pick-up location of the passenger p_j, the passenger p_j waits less than the maximum waiting time δ, i.e., $s_j = l_a$ and $t_a - st_j \le \delta$, where $1 \le a \le r$. If the passenger p_j is shared this taxi with p_i, their following trip should be same until one of them arrive the destination, which is the destination condition. There are two cases are shown in Figure 1:

(1)The passenger p_i arrives at the destination first, as shown in the left plot of Figure 1. Thus, the destination of p_i should be on the shortest path of passenger p_j.

(2) The passenger p_j arrives at the destination first, as shown in the right plot of Figure 1. Thus, the destination of p_j should be one the shortest path of passenger p_i.

When there is more than one passenger on the taxi, it also needs to satisfy the pick-up condition and destination condition to pick-up more passengers. The following trips of all the current passengers are the same at any timestamp. We hope to minimize the number of taxi-sharing trips satisfying all the passengers in the region. We define our problem under the offline setting, as shown below:

Definition 2 (taxi-sharing problem). Given a region Ω, for any two locations in this region, the shortest path between them are unique and known. With a set of m passengers $P = \{p_1, p_2, \ldots, p_m\}$ in the region, for any passenger p_i, her pick-up locations, pick-up time, and destinations are represented as s_i, st_i, and d_i. We need to design a set of taxi-sharing trips $\mathcal{T} = \{T_1, \ldots, T_n\}$, such that all the passengers are satisfied with the pick-up condition and destination condition. Our goal is to minimize the number of taxi-sharing trips, i.e., n, in the set.

$$\begin{aligned} &\min n = |\mathcal{T}| \\ &\text{s.t. } \forall p_i, \exists T_j, s_i = l_a, (l_a, t_a) \in T_j \\ &t_a - st_j \le \delta, (l_a, t_a) \in T_j. \end{aligned} \tag{1}$$

For the offline taxi-sharing problem, with the finite number of passengers, we can find all possible taxi-sharing trips through the enumerate or other better methods. Then, the optimal solution is one of the combinations of these taxi-sharing trips. We can show the difficulty of this problem below:

Theorem 1. *The offline taxi-sharing trip problem is NP-complete.*

Proof We can reduce this problem from a known NP-complete problem, named the set cover problem [22]. Given a set of element E and a collection of sets $\mathcal{S}$, the goal of set cover problem is to select the minimum number of sets from $\mathcal{S}$ such that each element can be covered by these sets.

For offline taxi-sharing problem, when we obtain all the possible taxi-sharing trips, we know which passengers can be satisfied by the special taxi-sharing trip. For any instance $\{\mathcal{S}, E\}$, each element can be regarded as a passenger, and the set can be a taxi-sharing trip. Suppose we have an algorithm $\mathcal{A}$ to compute offline taxi-sharing problem, the output can be seen as the solution to the set cover problem as well. Therefore, according to Cook's reduction, offline taxi-sharing problem is NP-complete.

4. Algorithm Design

In this section, we propose a heuristic algorithm for online taxi-sharing problem. Another approximation is proposed for offline taxi-sharing problem.

First, in the problem definition, it requires that each passenger needs to be delivered via the shortest path. It means that each taxi-sharing trip can be composed of multiple concatenated shortest path. To simplify this problem, we suppose that each taxi-sharing trip need to travel from a source point to the destination via the shortest paths. In this way, the taxi-sharing trip is a shortest path between two locations. The taxi drivers try to pick up as many passengers as possible along the way.

Each passenger has their own pick-up location and destination. We need to design the shortest path for taxi drivers to pick all the passengers up and take them to their destinations. We will introduce the concept of VC-dimension [23], used in our following analysis.

Definition 3 (VC-dimension). Given a set system $(X, \mathcal{R})$, let A be a subset of X. We say that A is shattered by $\mathcal{R}$ if $\forall Y \subseteq A, \exists R \in \mathcal{R}$ such that $R \cup A = Y$. The VC-dimension of $(X, \mathcal{R})$ is the cardinality of the largest set that can be shattered by $\mathcal{R}$.

Now, we consider the VC-dimension of our case.

Theorem 2. *For the passenger set system $(P, \mathcal{T})$, if each taxi-sharing trip $T_i \in \mathcal{T}$ is the shortest path, then the VC-dimension of this set system $(P, \mathcal{T})$ is at most 2.*

Proof We regard each passenger as an element in the set. Any set of passengers is tried to be shattered by the taxi-sharing trip. Now, each taxi-sharing trip is a shortest path between two locations. For any set of three passengers, a taxi-sharing trip cannot shatter it. In this case, the taxi-sharing trip is the shortest path between two locations. If these three passengers are all along this shortest path, we cannot shatter any two of them from the set. Thus, there is no taxi-sharing trip with the shortest path can shatter the set of three passengers.

Actually, our taxi-sharing trips are composed of multiple concatenated shortest paths. Suppose each taxi-sharing trips are at most k concatenated shortest paths. Then we can check its VC-dimension.

Theorem 3. *For the passenger system $(P, \mathcal{T})$, if each taxi-sharing trip $T_i \in T$ is k concatenated shortest paths, then the VC-dimension of this set system $(P, \mathcal{T})$ is at most $2k^2$.*

Proof Similarly, for any set of $2k^2 + 1$ passengers, there are at least $2k + 1$ passengers on the same shortest path, according to the Pigeonhole principle. Then we select the alternative passengers or passengers of the odd index. There are at least $k + 1$ passengers and they cannot be shattered by k shortest paths. The reason is that any two passengers cannot share one shortest path, so each one passenger needs one shortest path. Thus, the VC-dimension of this set system is at most $2k^2$.

4.1. Heuristic Algorithm. Basically, for each taxi driver, when she picks up the first passenger, she wants to find the next passenger who satisfies the pick-up condition and destination condition. Similarly, when one passenger is waiting for a taxi, the first option is a sharing taxi. Thus, our heuristic algorithm is based on this greedy idea: when a driver picks up one passenger, she will try to find another passenger along the way.

With the request arriving sequentially, there are many taxi-sharing trips on the road, if this request can be satisfied by one of the trips, the passenger can be picked up by one sharing taxi, otherwise, an empty taxi will go to the pick-up location. The pseudo-code is shown in Algorithm 1.

Here, the set of passengers are still given in advance. In this way, it can show that this algorithm can also solve the offline taxi-sharing problem. In Line 1, all the passengers are sorted by their pick-up time, which is suitable for the online setting. The set $f[i]$ represents the current sharing passengers on the taxi with the passenger p_i. Thus, we check the passengers according to their pick-up location in Line 4. First, all the existing taxi-sharing trips need to be checked in Line 5 and the current timestamp is st_i, i.e., the pick-up time of the passenger p_i. For each taxi-sharing trip, some passengers arrive at their destination before the time st_i, so we remove these passengers from the corresponding set $f[j]$. If there are still k passengers on the taxi, i.e., $\|f[j]\| = k$, it is impossible to accommodate the passenger p_i on this taxi. If there are still some available seats, we can check whether the pick-up condition and destination condition are satisfied in Line 9. If it is satisfied, the passenger p_j is included, and all the passengers are stored in the set $f[j]$. If it cannot be satisfied with all the taxi-sharing trips, an empty taxi will be dispatched to pick her up in Line 13–15. The number of trips is added in Line 15 and return the result in Line 16. For the offline taxi-sharing problem, with the finite number of passengers, we can find all possible taxi-sharing trips through the enumerate or other better methods. Then, the optimal solution is one of the combinations of these taxi-sharing trips. We can show the difficulty of this problem below:

We can see that this algorithm also works for the offline taxi-sharing problem. This idea is suitable for our daily life. Passengers are willing to wait for a taxi to share with someone, saving the cost. The taxi drivers do not make any detours and deliver every passenger to the destination on time.

4.2. Approximation Algorithm. In Section 3, we mention that we can enumerate all possible taxi-sharing trips for the given m passengers under the offline setting. It costs too much time to enumerate them. Thus, we first propose a better way to find all possible taxi-sharing trips, which is easy to understand.

Given that there are m passengers, all the information of them is known, including pick-up locations, pick-up time, destinations, and the shortest paths. It is not difficult to check whether any pair of passengers can share one taxi, i.e., checking the pick-up condition and destination condition. Thus, we can build a graph $G(P)$ for all the passengers to represent this relationship, which helps us to find all possible taxi-sharing trips.

For the passenger graph $G(P) = \{V, E\}$, where V is the set of vertices, represented by each passenger, and E is the set of directed edges. Owing to that the pick-up order of passengers needs to be considered, these edges have direction. If the passenger p_j can be satisfied by the taxi-sharing trip of the passenger p_i, there is an edge from v_i to v_j. Hence, our passenger graph is established. Each vertex has two attributes, pick-up time, and arrival time. This arrival time is computed by the speed of the taxi and the shortest path. In this way, we know when the passenger is dropped off. An illustration figure is shown in Figure 2.

Then, we can use depth-first search and topology sort to find all possible trips in the graph $G(P)$. As shown in Figure 2, we take a look at vertices v_1, v_2, and v_3. It shows that the trip with the passenger p_1 can pick up the passenger p_2, but cannot pick up the passenger p_3. Here, we need to check the arrival time of p_1 and the pick-up time of p_3. The below plot is much complex. There are edges from v_4 to v_5 and v_7, but there is no edge between v_5 and v_7. It means that the passengers v_5 and v_7 cannot share one taxi. Thus, the taxi-sharing trip of p_4 can only pick-up one of passengers p_5 and p_7. The case of the passenger p_6 is the same with the passenger p_3. Now, let uss look at the vertex v_8, which shows that the passengers p_4 and p_7 can share a taxi-sharing trip with p_8. Thus, we do not need to check the time of p_8.

In this search process, we need to use the technique of topology sort. If we want to include this vertex, we need to check the arrival time and pick-up time to remove some vertices in the set. Then, if all the vertices in the set have a directed edge to this vertex, this vertex can be included. At the same time, we also need to consider the maximum number of passengers on a taxi. Using this, we can find all possible taxi-sharing trips with a different combination of passengers, speeding up this process. Of course, some memory techniques can be applied in this search process to improve the time complexity.

Based on the set of all possible taxi-sharing trips Γ, we need to select the minimum number of taxi-sharing trips such that all the passengers are satisfied. For each taxi-sharing trip T, we denote $w(T)$ as the number of passengers can be satisfied by it. Then we design the approximation algorithm to select these taxi-sharing trips one by one. In each iteration, we select the taxi-sharing trip that can satisfy the most waiting passengers. Or we can imagine each taxi-sharing trip is a set. We need to select the minimum number

Input: The set of pick-up location s_i and destinations d_i from m passengers P
Output: Number of trips needed
(1) Sorted the passenger set P according to s_i
(2) $f[i] = \varnothing$, $1 \le i \le m$
(3) res = 0
(4) **for** $i = 1, \ldots, m-1$ **do**
(5) **for** $j = 1, \ldots, i-1$ **do**
(6) Remove the passengers who arrive destinations from $f[j]$;
(7) **if** $\|f[j]\| == k$ **then**
(8) Continue
(9) **if** p_j can share with passengers in $f[i]$ **then**
(10) $f[j] = f[i] \cup \{p_j\}$;
(11) $f[i] = \varnothing$;
(12) Continue;
(13) **if** $f[j] == \varnothing$ **then**
(14) $f[j] = \{p_j\}$
(15) res = res + 1
(16) **return** res

Algorithm 1: Greedy algorithm

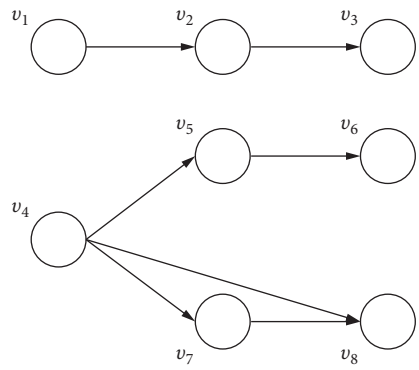

Figure 2: The passenger graph $P(G)$: here we omit the attributes on vertices. The vertices between one edge can share one taxi. If the passenger p_1 is dropped off before picking up the passenger p_3, then passengers p_1, p_2, and p_3 can share one taxi-sharing trip.

of sets to cover all the vertices in the passenger graph $G(P)$. In each iteration, we select a set that can cover the maximum number of uncovered vertices. We now show that this algorithm is an $O(\log m)$ approximation, where m is the number of passengers, which turns out to be the best approximation one can achieve (unless P = NP).

Theorem 4. *This approximation algorithm can achieve* $O(\log m)$ *approximation.*

Proof Consider a step of the algorithm. Let c be the number of waiting passengers before selecting a taxi-sharing trip on this step. Among the taxi-sharing trips that are not selected by the algorithm, the optimal solution uses some set of these taxi-sharing trips to satisfy the remaining waiting passengers. Let R_{OPT} be the optimal set of taxi-sharing trips based on the previous selection.

Let OPT be the number of taxi-sharing trips in the optimal solution. Clearly, we have that $|R_{\mathrm{OPT}}| \le \mathrm{OPT}$ since these taxi-sharing trips are a subset of the optimal solution, where $|R_{\mathrm{OPT}}|$ is the size of set R_{OPT}. In addition, we observe that there is $\sum_{T \in R_{\mathrm{OPT}}} w(T) \ge c$ since by definition these trips satisfy the remaining waiting passengers.

Therefore, we have:

$$\max_{T \in R_{\mathrm{OPT}}} w(T) \ge \frac{\sum T \in R_{\mathrm{OPT}} w(T)}{|R_{\mathrm{OPT}}|} \ge \frac{c}{\mathrm{OPT}}, \tag{2}$$

where the first inequality follows from an average argument, and the second inequality follows from our above observations. Thus, on this step where there are c remaining waiting passengers, there must be a taxi-sharing trip that can satisfy the average number which is at least c/OPT.

We try to divide up the cost of taxi-sharing trip to each passenger. The cost of each taxi-sharing trip is all the unit cost. Thus, each passengers gets a charge as $1/w(T)$. Now, we can claim that the jth passengers to be satisfied can receive a charge at most $\mathrm{OPT}/(m - j + 1)$. It follows equation (2) and the fact that when the jth waiting passenger is satisfied, there must be at least $(m - j + 1)$ waiting passengers on this step.

Thus, we can sum up the total charge of all the passengers. The total charge is

$$\sum_{j=1}^{m} \frac{\mathrm{OPT}}{m-j+1} = \mathrm{OPT} \cdot \sum_{j=1}^{m} \frac{1}{j} = \mathrm{OPT} \cdot O(\log m), \quad (3)$$

where the last equality follows the fact that the mth harmonic number is $O(\log m)$.

This approximation seems not good enough because the number of trips are very large. However, for the independent vertices, i.e., some passengers who cannot share with others are not counted into the approximation ratio. The most sharing cases are easy to find because there is no too many combinations. Thus, the performance of this algorithm in the real application is good enough to satisfy the requirement.

5. Numerical Experiments

In this section, we test our model and algorithms in two real-world data sets. First, we introduce our experimental setup, including data sets, hardware, and baseline algorithms as references.

Hardware. We implemented our algorithm in Python with version 3.8. We ran the experiments on the machine equipped with Intel(R) Core(TM) i7-8700 CPU @ 3.20 GHz-, and 32 GB of RAM.

Data set. We test our model and algorithms on two real-world data set. The first one is a set of taxi trajectories from San Francisco [24]. There are 536 taxis delivering passengers over one month in an area of $6,327 \times 6,827\mathrm{km}^2$. In each day, there are about 21,843 trips of these taxis, which is a huge number. The visualization map of San Francisco is shown in Figure 3. Each line represents a trip of taxi, and the color shows the frequency of traveling. We can see that the central part of the city is very dense and there is some high way around the city.

The second one is also a set of taxi trajectories in the center of Porto, Portugal [25], which is an area of size $8,116 \times 8,068\mathrm{km}^2$. There are 442 taxis running in the city over a complete year (from 01/07/2013 to 30/06/2014). More than 170,000 trajectories are included in this data set, which is visualized in Figure 4. We can see that these trajectories are radial shaped around the city. In the central part of the city, the trajectories are denser, which means that more passengers are traveling in a similar route. It has a large probability to take a sharing taxi to destinations.

For these two data sets, the passengers' demands are given, including their pick-up locations, pick-up time, and destinations. At the same time, their trips based on GPS locations and times are also provided. In our implementation, we can regard their trips from pick-up locations to destinations as the shortest paths. We also know the exact time of taxis arriving at each location, which helps us to determine whether another passenger can be picked up. The taxi drivers cannot make any detour in this process. The sharing modes should be based on their trips in the real-world data sets.

Tool chain. We implement our proposed algorithms based on Python, and the source code can be shared after publication. We provide a tool chain of our framework, including data processing, graph building, algorithm implementation, and comparison. The main processes are introduced below:

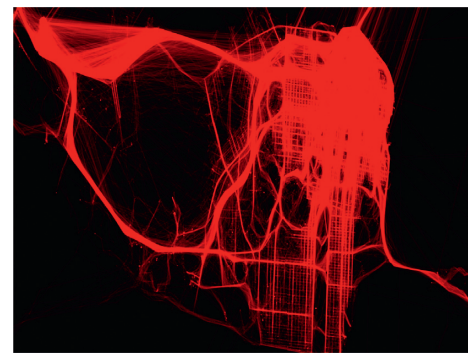

Figure 3: The visualization map of San Francisco, America.

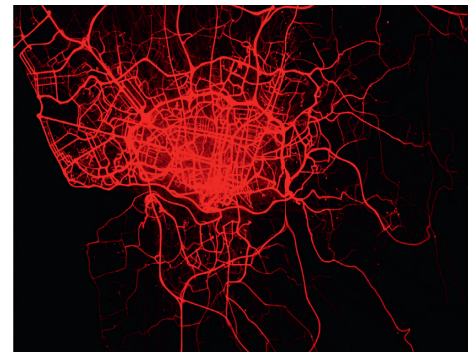

Figure 4: The visualization map of Porto, Portugal.

(1) Data set Processing. Given the initial real-world data sets, we first need to do some preprocessing jobs for data sets, including removing some outlier trips (with unrealistic speed or at an impossible location). For the San Francisco data set, the whole trajectories of taxis are given including empty states. Thus, we also need to extract the trips from the data sets. Then we have to sort all the trajectories according to the pick-up time of each passenger for these two data sets.

(2) Graph Building. For our approximation algorithms, we hope to embed all the passengers with their trip information into a graph $G(P)$. Each vertex represents passengers, and edges between vertices show whether two passengers can share one taxi. Similarly, our greedy algorithm can also do this step first and then try to include more passengers in one trip. Thus, in this step, we need to check whether one passenger can share the trip with the other passenger. Then, using this information to build our passenger graph. To speed up this process, we can use the hashing technique to store passengers, with the keys as the pick-up locations and destination separately. The lookup can be improved to constant time.

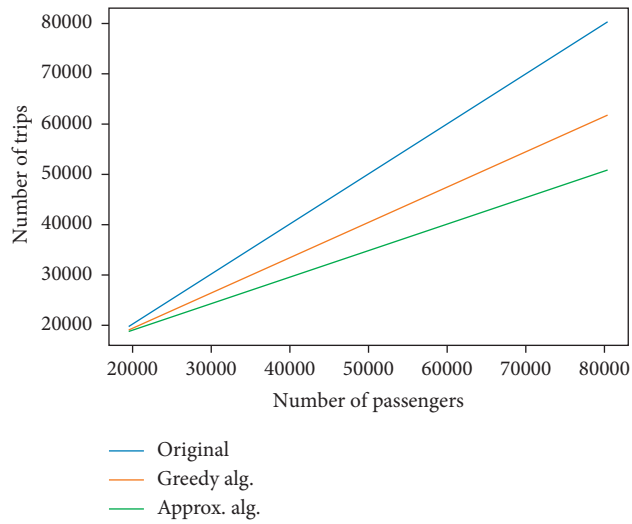

Figure 5: Number of taxi-sharing trips in San Francisco: the number of taxi-sharing trips is increased approximately linearly with the increased number of passengers. With the taxi-sharing system, the performance using approximation algorithm works best, which can reduce more than 40% trips compared with the case without taxi-sharing system.

Table 1: San Francisco: number of taxi-sharing trips (ten thousand).

# of Passengers	2	4	6	8
Original	2.03	4.09	5.97	8.11
Greedy alg.	1.92	3.25	4.65	6.17
Approximation alg.	1.84	3.02	3.93	5.04

(3) Algorithm Implementation. In this step, two proposed algorithms are implemented. For the greedy algorithm, each trip is selected based on its order of pick-up time. Then this driver will try to pick-up passengers in this continuous process as much as possible. For the approximation algorithm, we first need to find all possible taxi-sharing trips using the deep-first search process. In this process, the topology order needs to be considered. Based on all possible taxi-sharing trips, each trip with the maximum number of waiting passengers is selected in each iteration, until all the passengers are satisfied.

(4) Comparison. To investigate the benefit of our taxi-sharing system, we need to compare with the results without taxi sharing. Actually, the number of passengers is the number of trips needed without taxi-sharing system. It can be regarded as the baseline if we do not use the taxi-sharing system. Here, we denote the original number of trips as "Original" in the figure. The greedy algorithm and the approximation algorithm are denoted as "Greedy Alg." and "Approx. Alg." We can check how many trips can be reduced by the taxi-sharing system.

First, we look at the San Francisco data set. Owing to that there are only about 536 taxis in a large area, all the trips in one day are sparse in space and time. We can combine trajectories of multiple days to one day and check the relationship between the number of passengers and the number of trips needed. We start from all the passengers in one day and include more passengers gradually. The results are shown in Figure 5 and Table 1. We can see the numbers of taxi-sharing trips are increased linearly with the increased number of passengers, no matter whether we use the taxi-sharing system or not. We can also see that with taxi-sharing system, the number of trips is reduced compared with the conventional system, especially using the approximation algorithm. When there are more than 50, 000 passengers, the number of trips can be reduced by more than 20%. If the number of passengers exceeds 80, 000, the reduction is more than 40%. The reason is that when there are more passengers, their trips become denser in space and time. The probability that one passenger has someone to share one taxi is increased. At the same time, we can see that our approximation algorithm performs better than the greedy algorithm. Although the approximation ratio is $O(\log m)$, which is a large number, its performance is great in the real implementation.

Second, for the Porto data set, the trip of each passenger only has the pick-up time, without the timestamp in the delivery process. Thus, we assume that the GPS locations are sampled with about $30 - 50$ seconds according to the distance in the interval. Then we check whether one passenger can share a taxi with other passengers. The results are shown in Figure 6 and Table 2. The results are similar to the San Francisco data set, but the reduction is smaller than the one in San Francisco. The reason is that the size of Porto city is larger than San Francisco. In addition, we can also see from

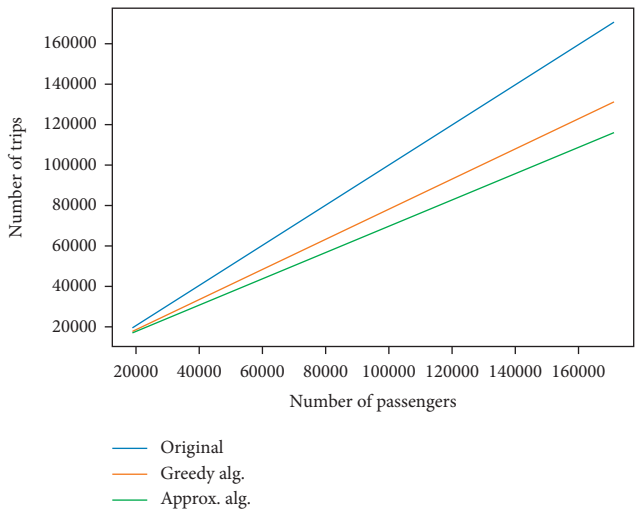

Figure 6: Number of taxi-sharing trips in Porto: the number of trips can be reduced about 30% compared with the conventional systems, where there are more than 170, 000 passengers in one day.

Table 2: Porto: number of taxi-sharing trips (ten thousand).

# of Passengers	2	6	10	14
Original	2.01	6.74	10.96	15.37
Greedy alg.	1.98	5.02	8.21	11.63
Approximation alg.	1.90	4.42	7.43	10.23

the visualization map, the trips in Porto are distributed in a larger area. In the central part, the trips are not denser than in San Francisco.

Some constraints for taxi-sharing trips also influence the results, such as the maximum number of passengers on one taxi and maximum waiting time. The results are similar to the above experiments. The number of taxi-sharing trips is increased with the number of passengers. When the constraints become looser, like more passengers can be shared in one taxi, and more waiting time, the reduction in the number of taxi-sharing trips becomes larger.

6. Conclusion and Future Work

In this work, we have studied the taxi-sharing problem, aiming at minimizing the number of trips needed. First, the formal taxi-sharing model is established. Then the problem is formulated, and we prove this problem is NP-complete. We designed two algorithms to solve this problem with some analysis and optimization for implementation. Extensive experiments on two real-world data sets show the superiority of our algorithms and the advantage of the taxi-sharing system.

In the future, we will continue to focus on this kind of problem. Some detours can be considered in the model. Some constraints can be flexible according to the time and passengers. Besides, we can use the machine learning method to predict the passengers' demand and dispatch taxis to the popular region. We hope to pay more attention to this topic to improve our traffic design and city planning.

Data Availability

The real-world trajectory data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

References

[1] U.S. Department of Transportation, "Highway statistical 2019," 2019, http://www.fhwa.dot.gov/policyinformation/statistics/2019/.

[2] J. Hardy and L. Liu, "Available forward road capacity detection algorithms to reduce urban traffic congestion," in *Proceedings of the 2017 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*, June 2017.

[3] M. R. Jabbarpour, H. Zarrabi, R. H. Khokhar, S. Shamshirband, and K.-K. R. Choo, "Applications of computational intelligence in vehicle traffic congestion problem: a survey," *Soft Computing*, vol. 22, no. 7, pp. 2299–2320, 2018.

[4] F. Shi, D. Wu, D. I. Arkhipov, Q. Liu, A. C. Regan, and J. A. McCann, "ParkCrowd: reliable crowdsensing for aggregation and dissemination of parking space information," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 11, pp. 4032–4044, 2018.

[5] G. Tasseron and K. Martens, "Urban parking space reservation through bottom-up information provision: an agent-based analysis," *Computers, Environment and Urban Systems*, vol. 64, pp. 30–41, 2017.

[6] C. Morency, M. Trépanier, B. Agard, B. Martin, and J. Quashie, "Car sharing system: what transaction datasets reveal on users' behaviors," in *Proceedings of the IEEE Intelligent Transportation Systems Conference*, pp. 284–289, IEEE, Bellevue, WA, USA, September 2007.

[7] X. Huo, X. Wu, M. Li, N. Zheng, and G. Yu, "The allocation problem of electric car-sharing system: a data-driven approach. Transportation Research Part D:," *Transportation Research Part D: Transport and Environment*, vol. 78, Article ID 102192, 2020.

[8] M. Balac, F. Ciari, and K. W. Axhausen, "Modeling the impact of parking price policy on free-floating carsharing: case study for Zurich," *Transportation Research Part C: Emerging Technologies*, vol. 77, pp. 207–225, 2017.

[9] Q. Li, F. Liao, H. J. P. Timmermans, H. Huang, and J. Zhou, "Incorporating free-floating car-sharing into an activity-based dynamic user equilibrium model: a demand-side model," *Transportation Research Part B: Methodological*, vol. 107, pp. 102–123, 2018.

[10] M. Nourinejad, S. Zhu, S. Bahrami, and M. J. Roorda, "Vehicle relocation and staff rebalancing in one-way carsharing systems," *Transportation Research Part E: Logistics and Transportation Review*, vol. 81, pp. 98–113, 2015.

[11] K. Huang, G. H. D. A. Correia, and K. An, "Solving the station-based one-way carsharing network planning problem with relocations and non-linear demand," *Transportation Research Part C: Emerging Technologies*, vol. 90, pp. 1–17, 2018.

[12] L. Caggiani, R. Camporeale, M. Ottomanelli, and W. Y. Szeto, "A modeling framework for the dynamic management of free-floating bike-sharing systems," *Transportation Research Part C: Emerging Technologies*, vol. 87, pp. 159–182, 2018.

[13] Y. Du, F. Deng, and F. Liao, "A model framework for discovering the spatio-temporal usage patterns of public free-floating bike-sharing system," *Transportation Research Part C: Emerging Technologies*, vol. 103, pp. 39–55, 2019.

[14] T. Liu, W. Wu, Y. Zhu, and W. Tong, "Predicting taxi demands via an attention-based convolutional recurrent neural network," *Knowledge-Based Systems*, vol. 206, Article ID 106294, 2020.

[15] K. Zhao, D. Khryashchev, and H. Vo, "Predicting taxi and uber demand in cities: approaching the limit of predictability," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, pp. 2723–2736, 2019.

[16] H. Wang and J. Gao, "Distributed human trajectory sensing and partial similarity queries," in *Proceedings of the 2020 19th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*, Sydney, Australia, April 2020.

[17] C. A. Folkestad, N. Hansen, K. Fagerholt, H. Andersson, and G. Pantuso, "Optimal charging and repositioning of electric vehicles in a free-floating carsharing system," *Computers Operations Research*, vol. 113, Article ID 104771, 2020.

[18] M. Zhao, X. Li, J. Yin, J. Cui, L. Yang, and S. An, "An integrated framework for electric vehicle rebalancing and staff relocation in one-way carsharing systems: model formulation and Lagrangian relaxation-based solution approach," *Transportation Research Part B: Methodological*, vol. 117, pp. 542–572, 2018.

[19] D. Santos and E. Xavier, "Dynamic taxi and ridesharing: a framework and heuristics for the optimization problem," in *Proceedings of the 23rd international joint conference on artificial intelligence*, pp. 2885–2891, Beijing, China, August 2013.

[20] C.-C. Tao, "Dynamic taxi-sharing service using intelligent transportation system technologies," in *Proceedings of the International conference on wireless communications, networking and mobile computing*, pp. 3209–3212, Honolulu Hawaii USA, August 2007.

[21] P. Lalos, A. Korres, C. Datsikas, G. Tombras, and K. Peppas, "A framework for dynamic car and taxi pools with the use of positioning systems," in *Proceedings of the Computation world: future computing, service computation, cognitive, adaptive, content, patterns*, pp. 385–391, IEEE, Athens, Greece, November 2009.

[22] D. S. Hochba, "Approximation algorithms for NP-hard problems," *ACM Sigact News*, vol. 28, no. 2, pp. 40–52, 1997.

[23] V. N. Vapnik and A. Y. Chervonenkis, "On the uniform convergence of relative frequencies of events to their probabilities," *Theory of Probability & Its Applications*, vol. 16, no. 2, pp. 264–280, 1971.

[24] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser, "CRAWDAD dataset epfl/mobility," 2009, https://crawdad.org/epfl/mobility/20090224,.

[25] KAGGLE data set ecml/pkdd 15, "Taxi trajectory prediction(1)," Downloaded from https://www.kaggle.com/c/pkdd-15-predict-taxiservice-trajectory-i/data, 2015.