A Dynamically Carpooling Dispatching Algorithm for Improving Efficiency of Self-Driving Taxis in the Connected Vehicles Environment

Hsu-Cheng Chung, Yu-Jung Chang, Kuo-Feng Ssu
Institute of Computer and Communication Engineering, National Cheng Kung University, Tainan, Taiwan
Email: q36054099@gmail.com, yjc@dcl.ee.ncku.edu.tw, ssu@ee.ncku.edu.tw

***Abstract*—Since there is a great number of ride demands during the rush hour in major cities, passengers typically spend a lot of time waiting for the available taxis among the limited number of taxis. To address the issue, carpooling is a good way to reduce the waiting time of passengers. Most of the current taxi-sharing approaches have been proposed to deal with the taxi-sharing problem by utilizing traditional vehicles. In this paper, a dynamic taxi carpooling dispatching algorithm is developed in the connected vehicles environment to provide real-time taxi-sharing services. The approach schedules the proper taxis to provide the services for passengers based on the locations of taxis, the destinations of passengers, and the arranged destinations of taxis. The algorithm has been implemented and simulated by using the Simulation of Urban MObility (SUMO) simulator. The results show that the algorithm improves the service rate of taxis, the empty car rate of taxis, the average waiting time of passengers, and the driving distance when taxis are serving passengers.**

Keywords–Intelligent transportation systems; Dynamic ridesharing systems; Taxi dispatch schedule; Cooperative dispatch mechanism; Connected vehicles.

I. INTRODUCTION

With the rapid economic development of modern society, taxi is one of the important public transportation, which plays a vital role in the daily commuting for millions of passengers in urban areas. However, there is a large number of empty seating capacity of vehicles which are not fully utilized during the rush hour of major cities. For example, the survey conducted by the Federal Highway Administration (FHWA) shows that average vehicle occupancy in the US remains unchanged at 1.67 from 2009 to 2017 [1]. This result indicates the solution to under-utilized available transportation resources is still a challenging issue.

To tackle the issue, carpooling is a good way to make more efficient use of the seating capacity of vehicles to reduce the waiting time of passengers. Drivers share their trips with one or more passengers who have similar travel paths. Compared to the non-sharing scheme, the ridesharing scheme utilizes fewer transportation resources to satisfy the same quantity of ride demands. Therefore, the vehicle occupancy rate can be significantly increased by reducing the number of empty seats by using carpooling via the ridesharing scheme.

Several taxi-sharing approaches have been proposed for providing taxi-sharing services. These approaches [2] [3] can be broadly classified into two categories: static taxi-sharing scheme and dynamic taxi-sharing scheme. First, the static taxi-sharing schemes need prior knowledge of the information of taxis and passenger requests for scheduling proper taxis to satisfy passenger requests. However, the static taxi-sharing scheme cannot provide the satisfied service for the passengers that ask their rides at the varying locations and/or at different time. Second, the dynamic taxi-sharing schemes provide real-time taxi-sharing services without prior knowledge of the information of taxis and ride requests. In the most of the dynamic taxi-sharing approaches, the transport is supplied by traditional vehicles driven by human drivers to deliver passengers. Self-driving taxis will be one of the most important transportation in the future. Waymo has launched the commercial self-driving taxi service in Arizona in the Unite States [4]. Self-driving taxis can cooperate with each other to complete the tasks required from the cloud. Consequently, the dynamic taxi-sharing scheme using the self-driving taxi to transport the passengers could be a better choice to improve the whole system efficiency.

In this paper, a Dynamic Taxi Ridesharing Dispatching Algorithm is developed to provide real-time taxi-sharing services in the connected vehicles environment. With the proposed scheme, when passengers need rides, the method will dispatch the proper taxis for these passengers who need taxi-sharing services based on the location of taxis, the arranged destination of taxis, and the destination of passengers. The algorithm has been implemented and simulated by using the SUMO simulator. The results show that the proposed algorithm improves the service rate of the taxis, decreases the empty car rate of taxis, saves 30.93% of the average waiting time of passengers, and reduces 11.81% of the driving distance when taxis are serving passengers.

The remainder of the paper is organized as follows. Related work is described in Section II. The system model is presented in Section III. The dynamically carpooling dispatching algorithm is presented in Section IV. The performance of the proposed scheme is evaluated in Section V. Finally, Section VI concludes this paper.

II. RELATED WORK

Various approaches aim to deal with taxi-sharing problems. These approaches can be broadly classified into two categories. This section describes a summary of the static taxi-sharing scheme and the dynamic taxi-sharing scheme.

A. Static Taxi-sharing Scheme

The static taxi-sharing schemes need prior knowledge of the information of taxis and passenger requests for scheduling the matches between taxis and passengers [3]. The static taxi-sharing problem can be viewed as one variant of the static Dial-a-Ride problem (DARP) [5]. In the static DARP, all transportation requests are known in advance.

Users specify pick-up and delivery requests between the origins and destinations of the vehicle services. Transport is supplied by a fleet of vehicles that provide shared services. The solution is to search for a set of minimum cost vehicle routes that serve as many user requests as possible under a set of constraints. Cordeau [6] proposed a branch-and-cut algorithm for DARP and reduced both the CPU time and the number of nodes explored in the branch-and-bound tree. This proposed method cannot be used to solve large-scale cases containing more than hundreds of users. Attanasio et al. [7] introduced a number of parallel implementations for the dynamic multi-vehicle dial-a-ride problem, based on a Tabu search for the static DARP. The proposed algorithms can meet a high percentage of user requests. Furthermore, most of the static taxi-sharing approaches focus on using either the ride reservation or the fixed-point taxi station to meet the ridesharing services [8]. Therefore, the static taxi-sharing scheme cannot meet the passenger demands in different locations and/or at different times.

B. Dynamic Taxi-sharing Scheme

The dynamic taxi-sharing schemes provide taxi-sharing services without prior knowledge of the information of taxis and ride requests. There are several dynamic taxi-sharing schemes have been studied in several previous works [9]–[16]. Most of these approaches utilize traditional vehicles to transport passengers. The approaches also consider the profit of human drivers or interpersonal trust among the drivers and the passengers. Ma et. al [5] developed a mobile-cloud based real-time taxi-sharing system which considered that the monetary constraints in ridesharing to provide incentives for passengers and taxi drivers. Although monetary constraints make the proposed model more realistic, some useful schedules are discarded even if they can significantly reduce the waiting time of passengers. Huang et. al [17] proposed the intelligent carpool system for drivers and passengers to find carpool matches at any time and in any place. The proposed system utilized a genetic algorithm-based algorithm to solve carpool service problems. The solutions typically require a greater amount of computation time.

III. System Architecture

This section describes the proposed system architecture. First, the system overview and the assumptions are presented in Section III-A. Then, the proposed scenario is illustrated in Section III-B.

A. Assumptions

Figure 1 illustrates the proposed system architecture, including a cloud, self-driving taxis, and passengers. There are three main assumptions in the system. First, each of the taxi participating in the system is the self-driving and connected vehicle, which is equipped with Internet access for mobile communication, and GPS receivers for obtaining its current location. In addition, each taxi has sufficient power and the capability of computation to perform all of the required operations for taxi-sharing services without the assistance of human drivers. Each taxi automatically reports

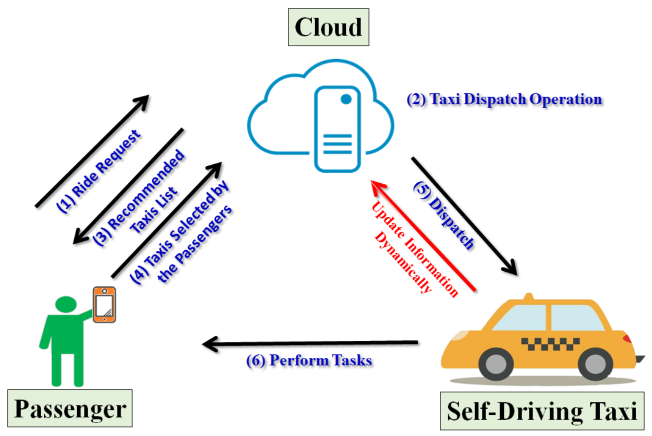

Cloud
(2) Taxi Dispatch Operation
(1) Ride Request
(3) Recommended Taxis List
(4) Taxis Selected by the Passengers
(5) Dispatch
Update Information Dynamically
(6) Perform Tasks
Passenger
Self-Driving Taxi

Figure 1. System overview.

it current location and the information of its status to the cloud. Second, passengers utilize the mobile application built on a handheld device to interact with the cloud system. The handheld devices with the GPS receivers provide current locations of passengers and have the capability for mobile communications. Third, the cloud has the capacity of computation, storage, and communication to conduct all of the operations for taxi-sharing services. The cloud continuously collects and updates the information of taxis and passengers. Real-time traffic conditions are provided by the connected vehicles environment.

B. Scenario

The cloud has the real-time location and status of taxis and passengers. The real-time road conditions are also available. When a passenger asks for a taxi, the passenger submits a ride request to the cloud. Each request consists of the origin and destination, the waiting time limit, the travel time limit, and the magnification rate of detour distance of the passenger's trip. The passenger can specify the origin location to be picked up, or the default location provided by his/her handheld device. The waiting time limit denotes the maximum time of the time a passenger is willing to wait. The detour distance ratio to the original driving distance cannot exceed the magnification rate. After the cloud receives the new ride request, the proper taxis are assigned by the algorithm based on the locations of taxis, the origin and destination of the passenger. The passenger thereby can check the waiting time for the taxis recommended by the cloud. After the passenger selects the desired taxi, the reply request will be sent back to the cloud. Once the cloud receives the reply request, the dispatch command is sent to the selected taxi. When the selected taxi receives the command, the taxi will pick up the passenger according to the schedule and the route given by the cloud.

IV. Dynamically Carpooling Dispatching Algorithm

This section describes the Dynamic Carpooling Dispatching Algorithm. As shown in Figure 2, the algorithm consists of three stages: 1) Preprocess Phase; 2) Inference Phase; 3) Dispatch Phase.

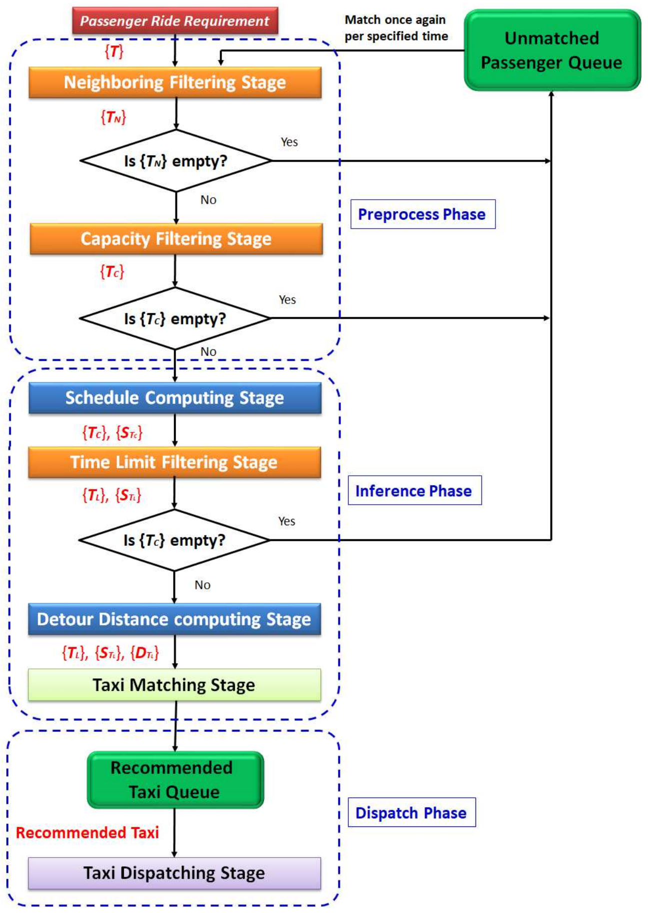


Figure 2. The flow chart of the proposed algorithm.

A. Preprocess Phase

The cloud takes the real-time information of taxis, passengers, and the traffic conditions as the input data. When the cloud receives the taxi-sharing requests, the detailed procedures of the algorithm will be conducted as follows.

Preprocess phase in the algorithm is a two-step procedure, which consists of Neighbor Filtering Stage and Capacity Filtering Stage. According to the waiting time limit of the passengers, Neighboring Filtering Stage filters out the taxis among all taxis T, which cannot arrive at the start locations of the requests in time. The eligible taxis which can satisfy the request in this step are stored in the list T_N. If there is no eligible taxi in the T_N to serve the requests, these requests will be stored into the Unmatched Passenger Queue for waiting for the next match. The match is performed every two minutes in this paper.

Capacity Filtering Stage filters out the taxis from the T_N, whose available capacity limit is less than the requested capacity. The remaining eligible taxis are stored in the list T_C. The seating capacity limit of each vehicle is set as four.

B. Inference Phase

Inference phase is a three-step procedure, which consists of Schedule Computing Stage, Time Limit Filtering Stage, and Detour Distance Computing Stage. Schedule Computing Stage schedules each legal route for each taxi in the T_C to satisfy the requirements of the requests by searching for all possible combinations of the taxis to serve all of the passengers in this match.

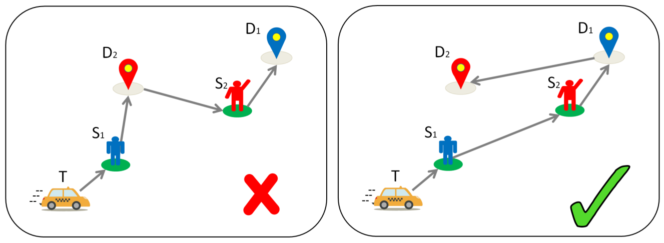


Figure 3. The examples of illegal and legal route schedules.

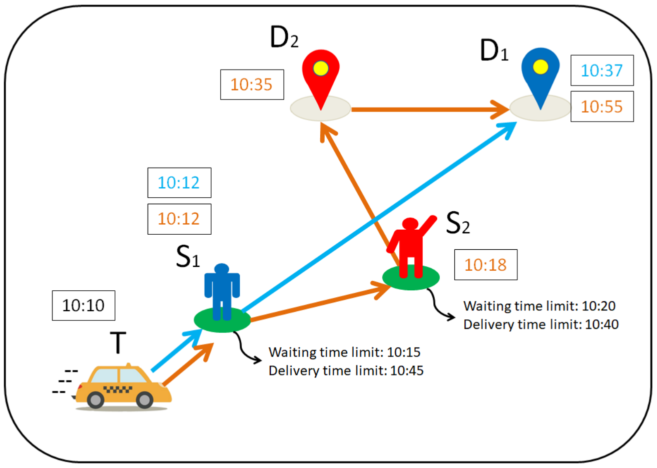

D₂
D₁
10:35
10:37
10:55
10:12
10:12
S₂
S₁
10:18
10:10
T
Waiting time limit: 10:20
Delivery time limit: 10:40
Waiting time limit: 10:15
Delivery time limit: 10:45

Figure 4. An example of the detour distance of ridesharing.

The examples of illeal and legal route schedules are shown in Figure 3. A taxi T is dispatched to serve two Passengers, P_1 and P_2, where S_1 and S_2 are the start locations of P_1 and P_2, respectively; D_1 and D_2 are the destination locations of P_1 and P_2, respectively. In order to satisfy the two ride demands, the taxi has to visit S_1, D_2, D_1, and D_2. Since not all of the routes are reasonable, some illegal cases are eliminated to reduce the computation in this phase. For example, the scheule $(T \to S_1 \to D_2 \to S_2 \to D_1)$ is illegal because each passenger should get on the taxi before getting off the taxi. On the other hand, the scheule $(T \to S_1 \to S_2 \to D_1 \to D_2)$ is a legal case. As a result, the number of legal route schedules can be computed as

$$\sum_{i=1}^{n} \frac{(2P_{R_i} + P_{T_i})!}{2! \times P_{R_i}}, \tag{1}$$

where n is the number of taxis in the T_C; P_{R_i} is the number of passengers who are picked up by the taxi T_i; P_{T_i} is the number of passengers who ride in the taxi T_i.

After considering each possible route schedules for each taxis T_i in the T_C to serve each combination of the passengers, the shortest distance of each legal route schedule is computed and stored in the S_{T_C}.

When a new passenger submits a taxi-sharing request for the taxi, Time Limit Filtering Stage will examine the legality of the waiting time limit and the travel time limit of those passengers who have already matched with the taxi. Some schedules passed the Schedule Computing Stage cannot satisfy the requirement of the requests. For example, the ridesharing schedule with a new passenger could increase the travel time of the original passenger who has sat in the taxi, resulting in exceeding the travel time limit of this passenger. Therefore, when a new passenger submits a taxi-sharing request, the waiting time limit and the travel time limit of the original passengers should be examined again.

Figure 4 shows an example of examining the legality of the waiting time limit and the travel time limit. P_1 is the original passenger who is served by the taxi T, and P_2 is a new passenger who wants to be served by the taxi T. The blue line is the original route of the taxi T, and the orange line is the presumed best route schedule for the taxi T to serve both P_1 and P_2. After P_2 asks for the ride request, the waiting time limit and the travel time limit of P_1 and P_2 will be calculated. The waiting time limit is expressed as

$$t_{w_i} = t_{r_i} + t_l, \tag{2}$$

where t_{r_i} is the time when the passenger submits a ride request and t_l is the waiting time limit of a passenger (the default value is 10 minutes).

The travel time is related to the travel distance d_i and the average velocity v_i of the trip. The travel time can be roughly predicted by dividing d_i by v_i. Assume that the average velocity of v_i for each taxi is set as a fixed constant. Then, the travel time limit can be expressed as

$$t_{d_i} = t_{w_i} + \alpha\frac{d_i}{v_i}, \tag{3}$$

where the detour rate of the passenger α is set as 1.5 (the maximum detour distance of the passenger trip is 1.5 times as long as the original detour distance). Algorithm 1 illustrates the pseudocode of eliminating the matches which are not suitable for the ridesharing schedule.

After examining the legality of the waiting time limit and the travel limit of each passenger in each legal route schedule in the S_{T_C}, the eligible taxis are stored in the T_L and the eligible schedules are stored in the S_{T_C}. If there is no eligible taxi in T_L to serve the requests in this stage, these requests will be stored into the Unmatched Passenger Queue, waiting for the next match.

Detour distance computing stage computes the total detour distance of the taxi, which serves all passengers in the route schedule. Specifically, Detour distance computing stage adds up each detour distance of each passenger, who is beening served by the taxi in the route schedule. The total detour distance of the taxi t denotes as d_t, which is stored in the set of D_{T_L}. The total detour distance D_T of all passengers D_p who are served by the taxi T can be expressed as

$$D_T = \sum D_p, \quad \forall p \in P_T, \tag{4}$$

Algorithm 1 : Eliminating the improper match

```
Definition:
  $V$: The set of taxi $t$, $\forall t \in T_C$.
  $S$: The set of schedule, $s_v$ $\forall v \in V$.
  $p_v$: The set of passenger $p$ in $v$, $\forall v \in V$.
Algorithm:
  for all $s_v$ in $S$ do
    repeat
      pop mission $m$ from $s_v$
      if $m$ is start location of $p \in p_v$ then
        if $m$ does not meet the waiting time limit of $p$ then
          remove $s_v$ from $S$
          remove $v$ from $V$
          break
        end if
      end if
      if $m$ is destination of $p \in p_v$ then
        if $m$ does not meet the travel time limit of $p$ then
          remove $s_v$ from $S$
          remove $v$ from $V$
          break
        end if
      end if
    until $p$ is empty
  end for
```

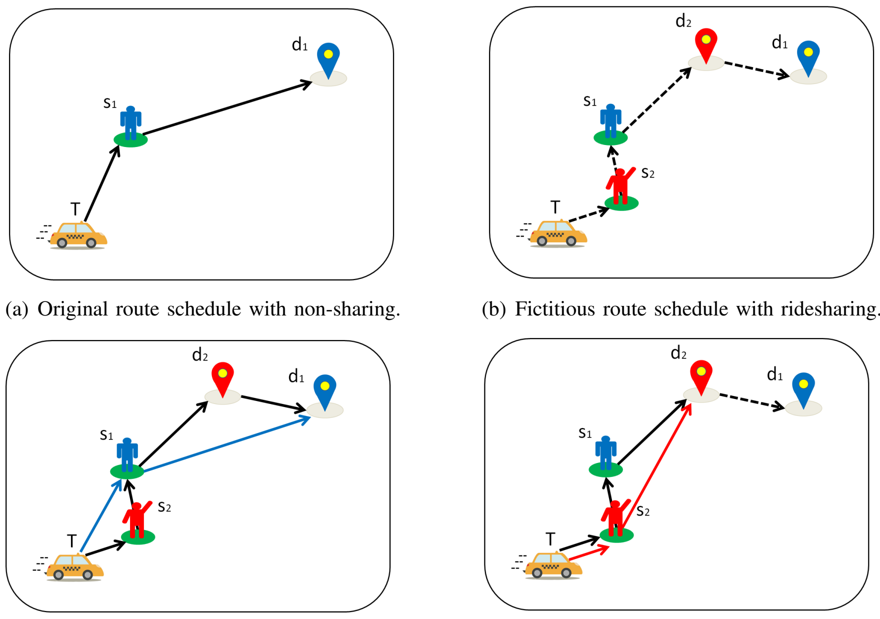
where the P_T consists of all original passengers who have been served by the taxi and the new passenger who selects the taxi for ridesharing.

$$D_p = D(T \to d_p) - D(T, s_p, d_p), \tag{5}$$

where D_p is the detour distance for a passenger p who is one of the passengers served by the taxi T. s_p and d_p are the start point and the destination of the passenger p, respectively. $D(T \to d_p)$ denotes the travel distance of the passenger p with the ridesharing scheme. $D(T, s_p, d_p)$ is the travel distance of the passenger p with the non-ridesharing scheme.

An example of computing the detour distance D_T is shown in Figure 5. In the beginning, the taxi T picks up the passenger p_1 at S_1 and delivers p_1 to his/her destination in Figure 5(a). A few moments later, another passenger p_2 asks for a ride. Figure 5(b) shows the shortest travel distance for serving both p_1 and p_2. Figure 5(c) shows the difference between p_1's driving schedules with ridesharing strategy and non-sharing strategy.

The detour distance of p_1 can be computed by subtracting the blue schedule distance from the black schedule distance, which is $D(T, s_2, s_1, d_2, d_1) - D(T, s_1, d_1)$. Similarly, the detour distance of p_2 presented in Figure 5(d) is $D(T, s_2, s_1, d_2) - D(T, s_2, d_2)$. Finally, the total detour distance D_T of the taxi T is calculated by adding up each D_p who is served by the T. Note that there might not only one passenger to be matched with the T when a new passenger asks for a ride. Algorithm 2 illustrates the pseudocode of the



(a) Original route schedule with non-sharing.

(b) Fictitious route schedule with ridesharing.

(c) p_1 detour distance illustration.

(d) p_2 detour distance illustration.

Figure 5. An example of computing the detour distance D_T.

Algorithm 2 : Detour Distance Computing Stage

```
Definition:
  $T$: The set of taxi $t$, $\forall t \in T_L$.
  $S$: The set of schedule $s_t$, $\forall t \in T$.
  $R$: The set of sorted route $r_{s_t}$, $\forall s_t \in S$.
  $M$: The set of sorted mission $m_{s_t}$, $\forall s_t \in S$.
  $D$: The set of total detour distance $d_t$, $\forall t \in T$.
Algorithm:
  for all $d_t$ in $D$ do
    $d_t = 0$
  end for
  for all $t$ in $T$ do
    for all $m_{s_t}$ in $M$ do
      for all $m$ in $m_{s_t}$ do
        if $m$ is the destination of a passenger $p$ then
          origin_dis = $Dijkstra(v, p\_start) + Dijkstra(p\_start, p\_destination)$
          $carpool\_dis = \sum Dijkstra(r)\ \forall r \in r_{s_t}$ end when $m$ is arrived
        end if
      end for
    end for
    $d_t = carpool\_dis - origin\_dis$
    store $d_t$ to $D_{T_L}$
  end for
```

Detour Distance Computing Stage. Note that the algorithm calculated the distance between the locations by using the Dijkstra algorithm. To reduce the computation load, the calculated distance information is stored in the database for the next utilization.

Taxi Matching Stage generates the Recommended Taxi Queue Q_{RecT}. In the Q_{RecT}, the taxis are sorted based on the following principles. First, the taxi with the smaller d_{T_L} has the higher priority. Second, when there are more than one taxis have the same d_{T_L}, the taxi closer to the passenger will have the higher priority.

C. Dispatch Phase

Dispatch Phase sends the recommended taxi requests to the passengers who need the taxi-sharing services. The recommended taxi which is pushed from the Recommended Taxi Queue definitely is the closest taxi with the shortest detour distance to the passenger. If the passenger agrees to the match, the selected taxi will be dispatched to the passenger immediately. On the other hand, if the passenger disagrees to the match and he/she sends an override reply back to the cloud, then the lower priority taxi in the Recommended Taxi Queue will be transferred to the passenger, reciprocally. The procedure will be iteratively executed until there is no taxi to be recommended in the Recommended Taxi Queue. When the Recommended Taxi Queue is empty, the requests will be pushed to the Unmatched Passenger Queue for the next match.

V. Performance Evaluation

In this section, the proposed algorithm is implemented and simulated to evaluate its performance. First, the simulation setting and evaluation metrics are introduced. Then the simulation results are presented to illustrate the effectiveness of the proposed algorithm.

A. Simulation Setting

The proposed algorithm is implemented and simulated using the SUMO simulator [18]. As shown in Figure 6, the

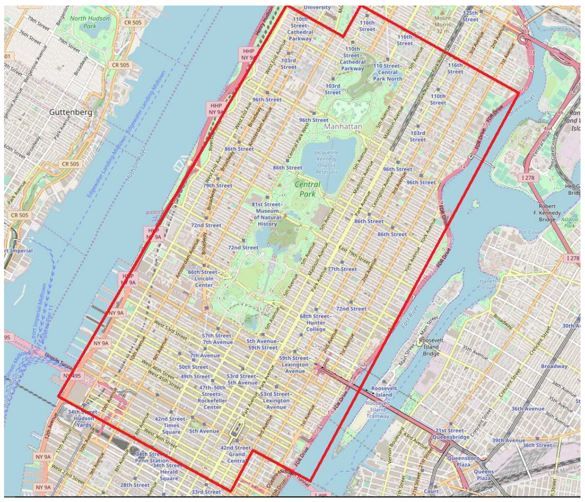


Figure 6. Manhattan on OpenStreetMap.

road network data is obtained through Openstreetmap [19], which is part of the Manhattan in New York City. The environment with a size of 30 square kilometers contains 4370 road segments. The time for each simulation is 7200 seconds. There are 250 taxis to serve 2,200 ride requests in the environment. In the simulation, it is assumed that the taxis will arbitrarily roam in the map if the taxis do not serve any ride request. Each taxi provides the taxi-sharing service for up to 4 passengers at the same time. The taxis must obey the maximum speed limit of the roads (60 km per hour). The trip information of all passengers, including request times, start positions, destination positions, are randomly generated. The trip distance of each passenger ranges from 1.5 km to 12 km. The maximum waiting time for each passenger is set to 10 minutes. The detour magnification in the simulation is set to 0.5.

B. Performance Results

The performance result of each metric is the average of 10 simulations. The proposed taxi-sharing algorithm was benchmarked against the non-sharing scheme.

TABLE I. OVERALL AVERAGE TRAVEL TIME OF PASSENGERS

Time Type	Non-sharing Scheme	Taxi-Sharing Scheme
Waiting Time	760.07 (675.49-819.84) (s)	239.38 (233.66-250.83) (s)
Riding Time	888.2 (884.86-895.91) (s)	899.06 (895.36-903.46) (s)
Travel Time	1648.34 (1597.91-1708.81) (s)	1138.44 (1130.26-1152.19) (s)

1) Overall Average Travel Time of Passengers: Table I shows the average waiting time, riding time, and travel time of the passengers for both the non-sharing strategy and the taxi-sharing strategy. The waiting time of the passenger is defined as the time between the passenger submits a ride request and the passenger gets on the selected taxi. The waiting time of the taxi-sharing strategy outperforms the non-sharing strategy. Since the passengers can utilize the taxi-sharing service with other passengers rather than spend time waiting for a vacant taxi. The riding time of the passenger is defined as the time between the passenger gets on the selected taxi and the passenger arrives at his/ her destination. Due to the additional detour distance for serving the carpooling passengers, the taxi-sharing strategy has a slightly longer average riding time than the non-sharing strategy. The travel time of the passengers is defined as the sum of the waiting time and the riding time. Based on the simulation results, the taxi-sharing method has the better travel time on average.

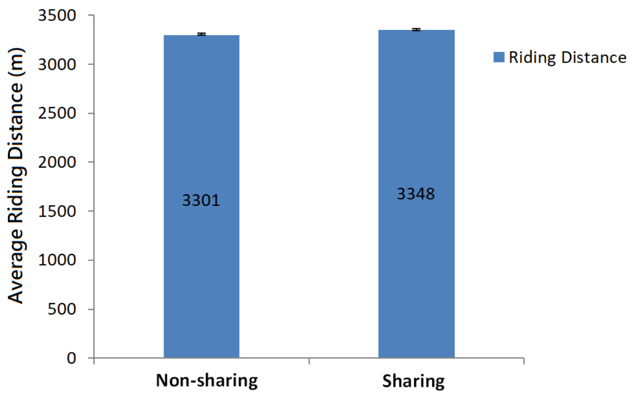


Figure 7. Average riding distance of passengers.

2) Average Riding Distance of Passengers: Figure 7 depicts that the average riding distance of the taxi-sharing approach is slightly longer than the non-sharing strategy. The main reason is that the passengers with the non-sharing approach are delivered to their destination with the shortest routes. The result also explains why the non-sharing algorithm has the better riding time.

TABLE II. COMPARISON RESULTS FOR THE SHARING AND THE NON-SHARING

Distance Type	Non-sharing	Ridesharing
Total Driving Distance	6616.07 (km)	6602.47 (km)
Driving Distance when Serving Passengers	6544.01 (km)	5771.01 (km)
Driving Distance while Carrying Passengers	4435.22 (km)	4929.27 (km)

3) Driving Distance Comparison: Table II shows that the comparison of the driving distances for both strategies. The driving distance can be divided into three types. First, the total driving distances of the two strategies are roughly the same because the taxis never take a break (except waiting for traffic lights and passengers to get on or get off). Second, the ridesharing strategy needs the shorter driving distance to serve all passengers due to its better efficiency. Third, the driving distance with carrying passengers for the ridesharing strategy is larger. The results can indicate that the taxis need the shorter distances to pick up the passengers.

TABLE III. COMPARISION OF CARPOOL RATE, EMPTY CAR RATE AND IDLE CAR RATE

Rate Type	Non-sharing	Ridesharing
Carpool Rate	0.00%	26.55%
Empty Car Rate	32.96%	25.34%
Idle Car Rate	1.09%	12.59%

4) Comparison of Carpool Rate, Empty Car Rate, and Idle Car Rate: The carpool rate, empty car rate and idle car rate are showed in Table III.The carpool rate is the proportion of the carpool participant among all passengers. The average carpool rate is 26.55% in the proposed taxi-

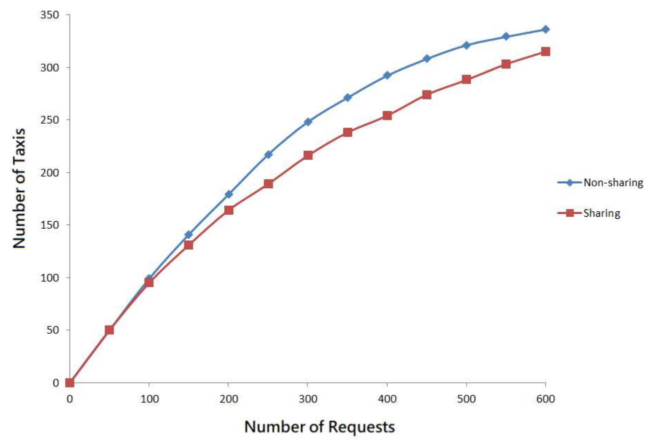


Figure 8. Number of taxis with respect to varying numbers of requests

sharing strategy, which denotes that approximately one-fourth of passengers share rides with others. An empty car is defined as if there is no passenger in the car; an idle car is defined as the car has no ride task to perform. The non-sharing strategy has 1.09% idle car rate and 32.96% empty car rate, so the strategy is not efficient. On the other hand, with the taxi-sharing strategy, the results show that 12.59% idle car rate and 25.34% empty car rate, which demonstrate that the ridesharing system can achieve the better ride performance.

5) Number of Taxis with respect to varying numbers of requests: As shown in the Figure 8, when there are 400 ride requests, the taxi-sharing strategy can reduce 11% of the number of the needed taxis to satisfy all the requests compared to the non-sharing approach.

6) Service Rate of Taxis: Service Rate is defined as the average number of passengers who are served by a taxi per hour. Figure 9 displays the performance of the service rate with varying numbers of passenger requests. The service rate of the proposed taxi-sharing approach is always higher than the non-sharing strategy. In addition, the service rate of the proposed method continues to grow as the number of the request is increased from 1050 to 1250. The main reason is that the proposed approach can alleviate the higher ride demands during rush hour.

VI. CONCLUSION

This paper develops a dynamic carpooling dispatching algorithm to provide the ridesharing service in real time for improving efficiency of self-driving taxis in the connected vehicle environment. The algorithm has been implemented and simulated by using the SUMO simulator. Compared to the non-sharing strategy, the results demonstrate that the algorithm enhances the service rate, reduces 30.93% of the average waiting time of the passengers, and shortens 11.81% of the driving distance during service.

ACKNOWLEDGMENT

This research was supported in part by the Ministry of Science and Technology of Taiwan under Contracts 107-2221-E-006-091.

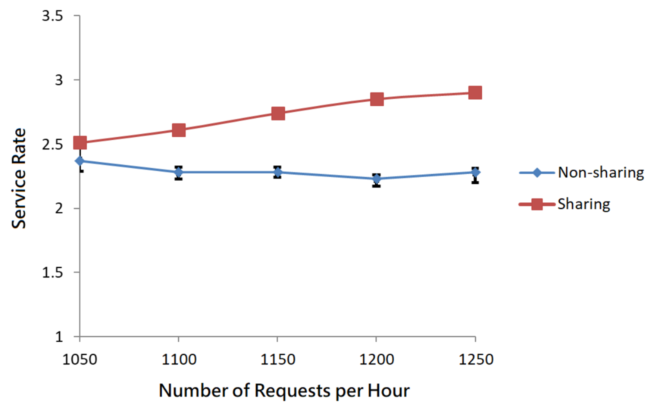


Figure 9. Service rate with the different number of passenger requests.

REFERENCES

[1] "National Household Travel Survey," 2019, URL: https://nhts.ornl.gov/ [accessed: May 17, 2019].

[2] M. S. N. Agatz, A. Erera and X. Wang, "Optimization for dynamic ride-sharing: A review," *European Journal of Operational Research*, vol. 223, no. 2, pp. 295–303, Dec. 2012.

[3] M. Furuhata, M. Dessouky, F. Ordóñez, M. Brunet, X. Wang, and S. Koenig, "Ridesharing: The state-of-the-art and future directions," *Transportation Research Part B: Methodological*, vol. 57, pp. 28–46, Nov. 2013.

[4] "Waymo," 2019, URL: https://waymo.com/ [accessed: May 17, 2019].

[5] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 7, pp. 1782–1795, July 2015.

[6] J.-F. Cordeau, "A branch-and-cut algorithm for the dial-a-ride problem," *Operations Research*, vol. 54, no. 3, pp. 573–586, May 2006.

[7] A. Attanasio, J.-F. Cordeau, G. Ghiani, and G. Laporte, "Parallel tabu search heuristics for the dynamic multi-vehicle dial-a-ride problem," *Parallel Computing*, vol. 30, no. 3, pp. 377–387, Mar. 2004.

[8] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to find my next passenger," in *Proceedings of ACM International Conference on Ubiquitous Computing (UbiComp)*, Spet. 2011, pp. 109–118.

[9] P. Chen, J. Liu, and W. Chen, "A fuel-saving and pollution-reducing dynamic taxi-sharing protocol in vanets," in *IEEE Vehicular Technology Conference - Fall*, Sept. 2010, pp. 1–5.

[10] S. Cheng, J. Li, and G. Horng, "Game theory based recommendation mechanism for taxi-sharing," in *Proceedings of International Conference on Advanced Information Networking and Applications Workshops*, May 2014, pp. 645–650.

[11] H. Zheng and J. Wu, "Online to offline business: Urban taxi dispatching with passenger-driver matching stability," in *Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 816–825.

[12] J. Hargrave, S. Yeung, and S. Madria, "Integration of dynamic road condition updates for real-time ridesharing systems," in *Proceedings of IEEE International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, Oct. 2017, pp. 585–589.

[13] S. Yeung, E. Miller, and S. Madria, "A flexible real-time ridesharing system considering current road conditions," in *Proceedings of IEEE International Conference on Mobile Data Management (MDM)*, June 2016, pp. 186–191.

[14] D. Pelzer, J. Xiao, D. Zehe, M. H. Lees, A. C. Knoll, and H. Aydt, "A partition-based match making algorithm for dynamic ridesharing," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 5, pp. 2587–2598, Apr. 2015.

[15] J. P. Hanna, M. Albert, D. Chen, and P. Stone, “Minimum cost matching for autonomous carsharing,” *IFAC-PapersOnLine*, vol. 49, no. 15, pp. 254–259, July 2016.

[16] D. Zhang, Y. Li, and F. Zhang, “Carpooling Service for Large-Scale Taxicab Networks,” *ACM Transactions on Sensor Networks (TOSN)*, vol. 12, no. 3, pp. 18:1–18:35, Aug. 2016.

[17] S. Huang, M. Jiau, and C. Lin, “A genetic-algorithm-based approach to solve carpool service problems in cloud computing,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 352–364, Feb. 2015.

[18] “Simulation of Urban MObility,” 2019, URL: http://sumo.sourceforge.net/ [accessed: May 17, 2019].

[19] “Openstreetmap,” 2019, URL: https://www.openstreetmap.org [accessed: May 17, 2019].