Dynamic Taxi Ride-Sharing Through Adaptive Request Propagation Using Regional Taxi Demand and Supply

Haoxiang Yu[1(✉)], Vaskar Raychoudhury[2], and Snehanshu Saha[3]

[1] Department of Electrical and Computer Engineering, The University of Texas at Austin, Austin, TX, USA
hxyu@utexas.edu
[2] Departmen of CSE, Miami University, Oxford, OH, USA
raychov@miamioh.edu
[3] Department of CS&IS and APPCAIR, BITS Pilani Goa Campus, Goa, India

Abstract. Taxi ride-sharing is an emerging public transportation model that provides several benefits in terms of cost, environmental impact, and road congestion. It is further popularized through market available app-based systems, such as Uber, Lyft, Didi, etc. However, those systems are limited due to their centralized architecture, high cost sharing with the driver, and proprietary business model. Distributed ride-sharing, on the other hand, involves only passengers and drivers and operates in a peer-to-peer manner. But, distributed ride-sharing systems often suffer due to Spatio-temporal constraints associated with taxi demand and supply as well as broadcast message storms. While we have observed dynamic and distributed ride-sharing systems which address the Spatio-temporal issues, there is hardly any effort to reduce their message overhead. In this paper, we present a hybrid model of ride-sharing where a central server adaptively calculates transmission range for passenger request propagation using Spatio-temporal information of ride-sharing success rate for the past 30-minute. Passengers use the adaptive transmission range to find the best shared-ride using a distributed manner. Our extensive empirical evaluation shows that our proposed approach increases the overall ride-sharing success rate and taxi utilization while significantly reducing the communication overhead, request processing time, and passenger waiting time.

Keywords: Dynamic ride sharing · Adaptive transmission range · Distributed algorithm · Spatio-temporal constraints

1 Introduction

United Nations has recently forecasted that by 2050, 68% of the world's population is going to live in cities in order to avail a decent livelihood and better

T. Hara and H. Yamaguchi (Eds.): MobiQuitous 2021, LNICST 419, pp. 40–56, 2022.
https://doi.org/10.1007/978-3-030-94822-1_3

infrastructures [12]. This "could add another 2.5 billion people to urban areas by 2050". Needless to say that this will stretch the infrastructure and public facilities available in the cities as they hardly grow at the same level as the population. Gradual expansion of urban boundaries force people to take up residences farther away from the centre and commute to work on a daily basis. Multitude of individual vehicles lead to traffic congestion and the resulting revenue loss and environmental impact. In order to address the problem ride-sharing is coming up as an efficient solution which can transport multiple commuters sharing part or whole of their commute. Taxicabs and other online app-based taxi services, such as *UberPool*[1], *Lyft*[2] and *DiDi*[3] can do miracles in this domain. We have seen huge growth of such services in recent year. However, most of the market available ride-sharing service are proprietary and have a centralized system model which is non-scalable and failure-prone.

In order to address the challenges associated with the centralized ride-sharing systems, researchers have proposed various alternate distributed ride-sharing systems which work by peer-to-peer coordination of passengers and taxi drivers [6,15]. However, distributed ride-sharing problem is non-trivial as it heavily depends on the various user and application related parameters. The spatio-temporal co-location of passengers are crucial to combining individual rides and assigning them into a shared taxicab. While this is feasible for static ride-sharing environment where a group of passengers share the same start and end points or take a cab operating on a fixed route (e.g., from Airport to the downtown), it is not the same for dynamic ride-sharing situations. In dynamic ride-sharing, a passenger request can arrive at anytime and a taxi might need to calculate in real-time, the feasibility of a detour from the current route (while carrying one or more passengers) in order to accommodate the new request. Research shows that dynamic ride matching is a NP-hard problem [10] in itself. Moreover, distributed ride-sharing aims to propagate passenger requests to a larger area in order to reach more taxis, which increases the chance of successfully finding a matching ride. However, this approach is sure to increase the number of messages and the overall processing and response time of the system even though the success rate improves.

An in-depth analysis of large-scale taxi demand and supply information for the Chicago metropolitan region [1,2] shows that passenger and taxi distribution over an urban area is unbalanced. Figure 1 shows the spatial distribution of passenger requests per hour across the community areas of Chicago. We can notice three distinct zones with different levels of passenger density - significantly high (dense zone), significantly low (sparse zone), and the areas which are in between the two extremes (target zone). We can create an intervention in the target zone in terms of adaptive transmission range control and can achieve the desired impact. Adaptive transmission control strategy will reduce transmission range in areas with a high success rate of ride-sharing, so that, fewer taxis receive

[1] Uber: https://www.uber.com/.
[2] Lyft: https://www.lyft.com/.
[3] DiDi: https://www.didiglobal.com/.

new passenger requests while maintaining the current success rate. Alternately, the transmission range will be increased for areas with a lower success rate of ride-sharing, in order to reach more taxis and to improve the success rate.

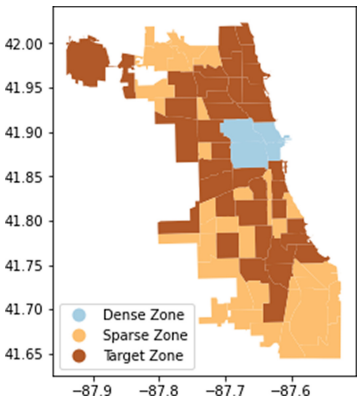

Fig. 1. Spatial Distribution of Ride-share Requests across Chicago Metropolitan Region

In this paper, we have proposed a hybrid ride-sharing system that uses peer-to-peer communication between passengers and taxi drivers to book shared rides. A server is deployed to analyze real-time passenger and taxi distribution in Chicago metropolitan area (see Fig. 1) based on a historical trend analysis (of the past 30-minute) of ride-sharing success rate. The server then provides a suitable transmission range to a querying passenger based on their spatio-temporal requirements. The server also prepares itself to serve better by using predictive analysis of passenger demand across the Chicago metropolitan region using the Prophet forecasting model [11]. Extensive empirical evaluation shows that our proposed ride-sharing system with an adaptive transmission range can improve the ride-sharing success rate for different community areas with various traffic demands while reducing the message overhead and passenger waiting time. Our system also enhances taxi occupancy and utilization by efficiently implementing ride-sharing algorithms through adaptive transmission range control.

In summary, this paper makes the following novel and unique contributions.

- We analyze the ride requests to understand the spatio-temporal distribution of passenger demand and taxi supply throughout the Chicago metropolitan area. To adapt to the constantly changing passenger demand and taxi supply, we propose a novel adaptive transmission range control system for request propagation.
- We develop a hybrid taxi ride-sharing system that includes passenger feedback to dynamically fine-tune transmission range to achieve a higher success rate for ride-sharing while reducing computation and communication overhead. Our taxi ride-sharing algorithm works in a purely distributed manner.

– Simulation studies using the Chicago taxi dataset show that adaptive transmission range can successfully control message propagation and corresponding processing overhead of taxi ride-sharing. Also, our system reduces passenger waiting time while improving taxi utilization.

2 Related Work

Taxi ride-sharing is coming up at a fast pace as a transportation solution for the next generation. It originated as a Dial-a-Ride Problem (DARP) [14] which is basically a static ride-sharing problem much like Carpooling (trips pre-coordinated between a set of passengers for a fixed set of destinations, like home to office) [4]. Psaraftis, et al. [10] have shown that the DARP is NP-hard with possible multi-objective extensions. Given the intricacies of the problem, taxi ride-sharing has attracted many researchers throughout the last decade [5,8,9,13]. Agatz, et al. [3] have provided a thorough analysis of the various aspects of both static and dynamic ride-sharing problems. However, the Spatio-temporal challenges posed by the recent on-demand dynamic and distributed taxi ride-sharing systems have not been paid enough attention. The first known distributed taxi-ride-sharing algorithm was proposed by D'Orey, et al. [7] but without significant empirical evaluation. Bathla, et al. [6] proposed the first properly evaluated and significantly efficient distributed Taxi Ride Sharing (TRS) algorithm was tested using large-scale continuous GPS traces of 4,000 taxis in Shanghai, China. An extension of this work with better results was presented by Yu, et al. [15]. However, their results are simulated using single trip taxi data in Chicago metropolitan region [1]. We, on the other hand, proposed a distributed and dynamic taxi ride-sharing system with a partly hybrid architecture to control the request propagation range. We are using a server which has a global view of the taxi demand and ride sharing success rate only through passenger communicated messages. We have also used a voluminous set of actual ride-sharing data [2] made available by Chicago Transport Authority.

3 System Model and Assumptions

Figure 2 shows the overall architecture of our proposed ride-sharing system which works in a hybrid manner. Passengers query the T_x-server to find out the best transmission range for propagating their requests. The operation of the T_x-server is further explained in Sect. 3.1. Once the adaptive transmission range is obtained from the T_x-server, the rest of the operation (finding shared rides) is performed in a purely distributed fashion involving the passengers and the taxi drivers. Request messages for ride-sharing are sent to all the taxis within the transmission range (T_x) in a single or multi-hop manner.

Our algorithm uses a variety of variables and data structures (see Table 1) and a set of seven (7) different types of messages (see Table 2) required to facilitate the adaptive ride-sharing operations.

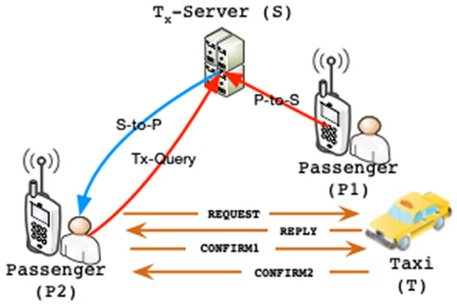

Fig. 2. System architecture

Table 1. Variables used

Variable	Significance
P_{id}	Unique integer identifier of a Passenger P
S_p	Pick-up location (latitude, longitude) of P
D_p	Drop off location (latitude, longitude) of P
t	Time at which P made a ride-sharing request
$time_d$	Time (in units) estimated by T to reach D_p from S_p (considering current state of Q_t and real road network topology)
Δ	Maximum Time (in mins.) up to which the time d can be exceeded from the actual time of trip (aka. detour tolerance)
T_{id}	Unique integer identifier of a Taxi T
T_{loc}	Current location (latitude, longitude) of T
P_{nbr}	All taxi within neighborhood of P_{id}
N_{vac}	Number of available seats (in integer) in T
e	Pick-up (e_{pick}) or drop-off (e_{drop}) event
Q_e	Temporary queue of e at T_{id}
Q_t	Temporary queue of $REQUEST$ at T_{id}
L_{con}	Permanent list of confirmed passengers at T_{id}
L_e	List of e associated with L_{con}
$time(T_{loc}, D_p)$	Time (in units) estimated by at T_{id} to reach D_p from T_{loc}
L_{reply}	Sorted list of $REPLY$ messages stored at P_{id}
T_{nbr}	All taxi within neighborhood of T_{id}
τ_p	Timer at at P_{id}
S_f	Success Flag which records if the trip request is successful or not
L	Spatial region (community areas of Chicago)
N	Number of available Taxicabs for the whole city (across all the L)
$T_x^{C_L}$	The adaptive transmission range for area L at current time instant

Table 2. Messages used in algorithm

Messages	Significance
P-to-S (L, S_f)	P reports to T_x-Server current values of L and S_f.
$T_x Query$ (P_{id}, L)	T_x value query message from P to T_x-Server
S-to-P (T_x^{Now})	Reply from T_x-Server to P with the new T_x
REQUEST $(P_{id}, S_p, D_p, t, \Delta)$	Request for ride-sharing sent by P
REPLY $(T_{id}, time_p, time_d)$	Reply from T to P with estimated pick-up time, drop off time and cost for ride
CONFIRM 1 (P_{id})	Acceptance / Rejection status sent by P to T
CONFIRM 2 (T_{id})	Acceptance / Rejection status sent by T to P

3.1 Generating Adaptive Transmission Range

Figure 2 shows the presence of the T_x-server for generating an adaptive transmission range for passenger request propagation. T_x is adaptively determined based on historical data about the ride-sharing success rate of all the passengers from a particular area and for the previous 30-minute-time span. Passengers query the T_x-server to find out the best transmission range and the server provides the same calculation using the historical data of ride-sharing success rate based on Spatio-temporal parameters.

The Eq. 1 shows the formula for calculating the current transmission range $(T_x^{C_L})$ for area L based on historical values of localized success rates.

$$T_x^{C_L} = \begin{cases} T_x^{C_L - t} * \kappa^+, \; if \; (Succ_{rs}^{(L)} < \Gamma_{low}) \\ T_x^{C_L - t} * \kappa^-, \; if \; (Succ_{rs}^{(L)} > \Gamma_{high}) \end{cases} \tag{1}$$

Where, $t = 30$ minutes, so $C_L - t$ signifies time which is 30 min prior to the current time (C). κ $(\kappa^+ >= 1, \kappa^- <= 1)$ is a parameter depending on the localized success rate of ride-sharing $(Succ_{rs}^{(L)})$ (see Sect. 6.2) which is again upper bounded by Γ_{high} and lower-bounded by Γ_{low}. We have varied κ and Γ to empirically find best values for these parameters.

3.2 Practical Assumptions

We make the following assumptions for our proposed ride-sharing system:

- For any passenger, drop-off time should be after the pickup time.
- Wireless communication between taxis and passengers may incur message loss and transmission delays. So, we use timers at passenger and taxi-end to handle lost messages.
- Taxis are mobile entities and they can change position significantly during the time a passenger sends out a ride-sharing request and by the time the request is confirmed. We have incorporated measures in our algorithms to handle the overall system dynamics.
- Each ride-sharing request is considered separately and for a single passenger.
- Taxi drivers must address the detour tolerance (Δ) specified by a passenger.

4 Algorithm for Taxi Ride-Sharing Using Adaptive Transmission Range

In this Section, we describe the functioning of our ride-sharing algorithm which contains a passenger to T_x-server communication and a separate passenger to Taxi communication algorithm. So, we have divided the algorithm into passenger-end (Algorithm 1), T_x-Server-end (Algorithm 2) and taxi-end (Algorithm 3).

4.1 Algorithm at Passenger-End

Ride-share scheduling starts by a passenger P_{id} when s/he broadcasts a *REQUEST* message and initializes a timer (τ_p). In order to prevent any broadcast storm, a transmission range (T_x) is provided for first hop message distribution. E.g., if $T_x = 100$ then the REQUEST message will be received by all the taxi drivers within 100 m from the passenger. If none of the taxis are available to serve the passenger, then they distribute the REQUEST to their 1-hop neighborhood and propagate it in this way until a maximum request distribution range is reached. The passenger receives zero (0) or more *REPLY* messages from taxi drivers until the timer (τ_p) expires. S/he then chooses the taxi which can complete the trip in minimum amount of time (or we can select some other parameter, like cost), acknowledges with a *CONFIRM 1* message and re-starts the timer (τ_p). The passenger must wait for a *CONFIRM 2* message from the same taxi before the trip can be finally confirmed. The *CONFIRM 2* message has been introduced to account for the highly dynamic nature of the environment as a taxi can move or confirm some other passenger before the current passenger receives a confirmation. So, we can say that our system is resilient to system dynamics resulting from node mobility, message loss and delay. In case no *CONFIRM 2* message is received by the time τ_p expires, the passenger can resend the REQUEST message up to two more times. Once a ride-sharing request is confirmed or not after a total of three attempts, the passenger sends a *P-to-S* message to the T_x-server carrying information about his local community area and the status of finding the shared-ride (successful or not).

4.2 Algorithm at T_x-Server-end

The T_x-Server receives *P-to-S* messages from passengers across the complete region. The messages are then stored in a queue in the temporal order of arrival. Every 30 min, messages are de-queued and used to calculate success rate of ride sharing for a particular community area L. The server then calculates the adaptive transmission range for a particular area L following Eq. (1). When a passenger sends out the T_xQuery for their current location, the server sends the $T_x^{C_L}$ through a *S-to-P* message.

Algorithm 1: Algorithm at Passenger P_{id} Side

```
Broadcast REQUEST to all T_id ∈ P_nbr
//Collect incoming REPLY messages until τ_p expires
L.append (REPLY message)
Sort L based on time_d
Send CONFIRM 1 to the taxi with minimum time_d and start τ_p
Wait for CONFIRM 2 from T until τ_p expires
if CONFIRM 2 received then
    Send P-to-S to the T_x-Server with L and S_f = True
else if τ_p expired then
    Send P-to-S to the T_x-Server with L and S_f = False
```

4.3 Algorithm at Taxi-End

Passenger requests for ride-sharing are received by taxi and stored locally in a temporally ordered message queue. As long as a taxi has vacant seats (we assume that a taxi can carry at most 4 passengers at a time) and pending requests, it will execute the *RouteValidation()* function. This function actually generates all possible route permutations considering the confirmed events and the temporary events in the queues (Q_e and Q_{con}) and finds the best route for the passenger in terms of the minimum detour delay. The route selected must abide by the practical assumptions listed for our system in Section III. B.

After a taxi determines a passenger whom it can pickup, a *REPLY* message is sent. Taxis sequentially process requests and can send multiple REPLY for one vacant seat. As discussed in Section IV.A, passengers can choose the best offer and send a *CONFIRM 1* message. When a taxi receives the first *CONFIRM 1* message from a passenger for an ongoing booking negotiation, it re-calculates the *RouteValidation()* function to make sure that the passenger can still be accepted given its current status. This step helps us to cope with the extremely dynamic nature of this distributed system. If the route re-validation is successful, a *CONFIRM 2* message is sent to the passenger confirming the booking. The passenger record is then added to L_{con} and the corresponding pickup and dropoff events (e_{pick} and e_{drop}) are added to the taxi schedule (L_e). The taxi will keep checking all of the incoming *CONFIRM 1* messages until there is no remaining vacant seat.

5 Spatio-Temporal Passenger Demand and Taxi Supply Analysis

Our T_x-Server calculates the adaptive transmission range based on the historical data and also predicts the future passenger demand in an area based on the historical trend. It takes into account periodic and non-periodic trends into consideration following the Prophet model that we discuss in this Section. Prophet model [11] is a time series data forecasting model to capture non-linear trends

Algorithm 2: Algorithm at T_x-Server Side

//On receiving a P-to-S message
Enqueue P-to-S(L, S_f) in L_s and add the current time stamp - T_c
//Keep processing the messages in L_s in FIFO order
Dequeue P-to-S(L, S_f) from L_s
$M_s \leftarrow (L, S_f, T_c)$
//For all the M_s received at time T_c, where $T_c = T_{curr} - 30$ (minutes)
Calculate ride-sharing success rate for area L
Calculate $T_x^{C_L}$
//On receiving a T_xQuery message from P_{id} in area L
Send $T_x^{C_L}$ to P_{id} via S-to-P message

along with seasonal effects which varies in a daily, weekly, and yearly manner as well as during holidays. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet can robustly handle seasonal variation time series data and can efficiently deal with outliers. Prophet forecasting model is composed of several linear and non-linear time series functions and can be represented using the Eq. (2):

$$y(t) = g(t) + s(t) + h(t) + \epsilon(t) \tag{2}$$

where, y(t) signifies the predicted value; g(t) captures the non-periodic changes through a linear or a logistic function; s(t) represents the period or seasonal changes in terms of yearly, monthly, weekly, daily or any other patterns; h(t) shows the irregular or holiday patterns for 1 or more days and finally, $\epsilon(t)$ will capture any other types of changes not captured by the rest of the parameters. So, in summary, the model can be represented as a combination of various regular and irregular trends:

Forecasted Value = Non-periodic segment-wise time series trend + seasonal time series trend + holiday / festival impacts + errors / noise

Prophet model can be declared either as a linear or as a logistic model. While in linear models, there is no pre-specified maximum or minimum limit; for logistic models, the highest and lowest values are pre-set. If the data contains parameters which can attain saturation, like processing power, etc., then we should use a logistic function for that saturating growth model. Otherwise, the default linear model with constant growth rate can be used. In this study, we used the linear model in which the various outliers in the dataset are normalized to have a more homogeneous data pool. The linear trend model is shown in Eq. (3):

$$g(t) = (k + a(t)^T\delta)t + (m + a(t)^T\gamma) \tag{3}$$

Here k represents the growth rate, δ captures the rate adjustments, and m is the offset parameter. In order to make the function a continuous one, the parameter γ_j is set to: $-s_j\delta_j$.

Algorithm 3: Algorithm at Taxi T_{id}

```
   //On receiving a REQUEST message
 1 Enqueue REQUEST in temporally ordered Q_t
   //Keep processing REQUEST messages from (Q_t) in FIFO order
 2 r = (P_id, S_p, D_p, t) ← Dequeue(Q_t)
 3 Q_e ← Enqueue(r.e_drop)
 4 Q_e ← Enqueue(r.e_pick)
 5 if Q_t ≠ ∅ AND N_vac ≠ 0 then
 6 |   if Route Validation(Q_e, L_e) ≠ ∅ then
 7 |   |   Send REPLY to P_id
 8 |   else
 9 |   |   Forward r to all T_id ∈ T_nbr
10 else if N_vac = 0 then
11 |   Forward r to all T_id ∈ T_nbr
   // On receiving a CONFIRM 1 message
12 if N_vac ≠ 0 then
13 |   if Route Validation(Q_e, L_e) ≠ ∅ then
14 |   |   Send CONFIRMATION 2 to P_id (for r)
15 |   |   L_con ← r
16 |   |   L_e ← r.e_pick
17 |   |   L_e ← r.e_drop
   Route Validation(Q_e, L_e)
18 for Q_e.length do
19 |   for L_e.length do
20 |   |   Routes[k] = Q_e × L_e
   |   |   //Generate all possible permutations of events
21 foreach route ∈ Routes[k] do
22 |   foreach P_id ∈ L_con do
23 |   |   if ((time(T_loc, D_p) > ((time_d + Δ)− time already travelled)) then
24 |   |   |   The route is NOT valid  drop it
25 if Routes[k] ≠ ∅ then
26 |   We choose the route with the minimum detour time for P_id
27 |   return route
```

Rate adjustments are parameters associated with trend changes that can occur in the growth model and are represented as *changepoints*. If there are S changepoints at times $S_j = 1, 2, 3, ..., S$, then rate adjustment parameter δ can be defined as

$$\delta \in \mathbb{R}^{\mathbb{S}} \tag{4}$$

where, δ_j denotes the rate change occurring at time s_j. $a(t)$ in Eq. (3) can be expressed as a vector

$$a(t) \in \{0, 1\}^S \tag{5}$$

whose values can vary as follows:

$$a_j(t) = \begin{cases} 1, & \text{if } t \geq s_j, \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

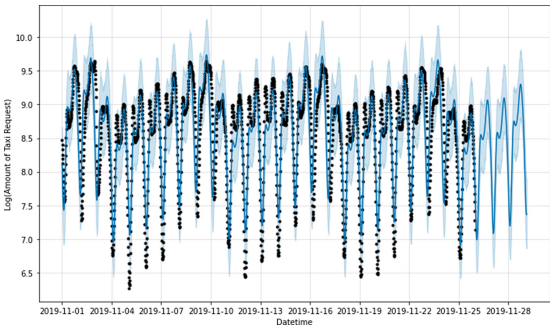

Fig. 3. Result of Prophet Forecast Model for Chicago Community Area 8

So, the growth rate at time t will be $k + a(t)^T\delta$, where k is the base rate and the remaining terms signify required adjustments up to the time t. At the time of the adjustment of base rate k, the offset parameter m must be adjusted as well, in order to connect the endpoints of the segments. In Fig. 3, we present an example graph showing prediction results of the number of passenger requests in Chicago community area 8, depending on the pre-trained model, date-time, and historical 3-hour-data. While the black dots in the figure show real data points from the dataset, the blue lines show the predicted result.

The demand prediction model described above, helps the T_x server to calculate the adaptive transmission range more accurately depending on the spatio-temporal variation.

6 Performance Evaluation

We have resorted to simulation-based experimentation and testing for evaluating our proposed algorithms as it was not possible to consider a dynamic deployment using a number of real taxis due to various practical and possibly legal restrictions. In this section, we discuss in detail the platform, metrics, and results of our simulation experiments.

6.1 Simulation Setup

In the simulation of our algorithm, we use the Chicago taxi ride-share dataset [2] which is open, recent (2018–2021), and voluminous. We have tested with a

subset of the total records and our data contains 259351 records from 06–19–2019 covering all the 77 community areas in the Chicago metropolitan area. The dataset obfuscates the actual pickup and drop-off location and time of passengers in order to maintain their privacy. We have also simulated using data for longer periods (up to 1 month) and the trends of the results are observed to be similar. More details of the dataset and other simulation parameters are presented in Table 3. We would like to mention that after experimenting with different values of κ and Γ, we settled for the values shown in the table. We found that the current values can achieve stability of adaptive transmission range and not create race conditions.

Table 3. Simulation Parameters

Parameters	Value
Initial transmission range $T_x(init)$	100, 800 (in mtr.)
Max. transmission range (T_{max})	2 KMs
Simulation period	06–19–2019 00:00 - 23:59
Number of taxis (N)	200, 400, 2000
Capacity of each Taxi (N)	4
Message delay (δ)	1 - 5 Millisecond(s)
Detour tolerance / Δ	5 min
τ_p	3 min
κ^+ (See Eq. 1)	1.2
κ^- (See Eq. 1)	0.8
Γ_{high} (See Eq. 1)	0.9
Γ_{low} (See Eq. 1)	0.5

Our simulation framework is built over Java and makes use of an Intel(R) Xeon (TM) Gold 6126 (2.6 GHz 12 Cores/24 Threads) processor with 96 GB memory for the experiment.

6.2 Performance Metrics

We evaluate our algorithms with respect to the following four performance metrics.

1. **Localized Success Rate** ($SUCC_{rs}^{(L)}$): For a particular community area L in Chicago, where $L \in \{1, 2, ..., 77\}$, *localized success rate* is measured as the ratio of the number of confirmed shared-ride requests to the total number of requests originating from area L.

2. **Taxi Activation Ratio** (Act_r): What percentage of taxicabs are really being used (carried one or more passengers) during the simulation period.
3. **Average Occupancy Period (per Taxi)** ($Occu_{avg}$): Ratio of the time a taxi had two or more passenger(s) to the total time the taxi travelled on road. We take the average of all the taxis that were actually used.
4. **Average Waiting Time to Pickup** ($Waiting_p^{avg}$): It is the average waiting time between the instant at which a passenger sent out the first *REQUEST* message and the instant s/he got picked up by a taxi (for all the successful ride-sharing requests).

6.3 Area Types

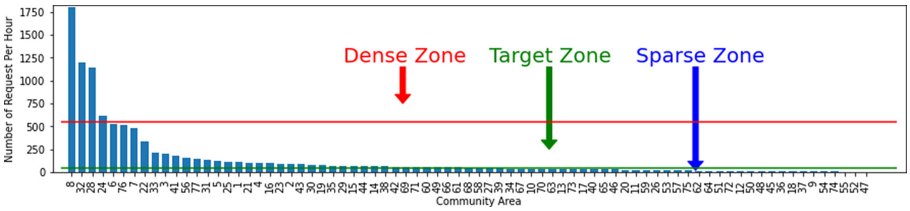

Fig. 4. Sorted list of community areas in Chicago in the order of passenger density

Based on the average number passenger requests originating from different community areas (77) in Chicago, we separate Chicago metropolitan region into three different types of zones as discussed below (see Fig. 4). Those three different zones are:

1. **Dense Zone:** Community areas with significantly high number of passenger requests ($\geq$ 545 requests per hour) are considered as dense. For the dense zone, our goal is to reduce the M_{avg}^L and R_{avg}^L while maintaining high $SUCC_{rs}^{(L)}$ by auto adjusting the T_x.
2. **Sparse Zone:** Sparse zone are those which generates significantly low number of passenger requests ($\leq$ 46 requests per hour). Our experiments show that sparse zones does not show any noticeable improvement $SUCC_{rs}^{(L)}$ by adaptively changing the T_x.
3. **Target Zone:** Except the dense and sparse zones, the remaining community areas are our target zones which generates considerable number of passenger requests ($\geq$ 46 and $\leq$ 545). We call this as our target zone, since we noticed maximum improvement in $SUCC_{rs}^{(L)}$ takes place by auto adjusting the T_x.

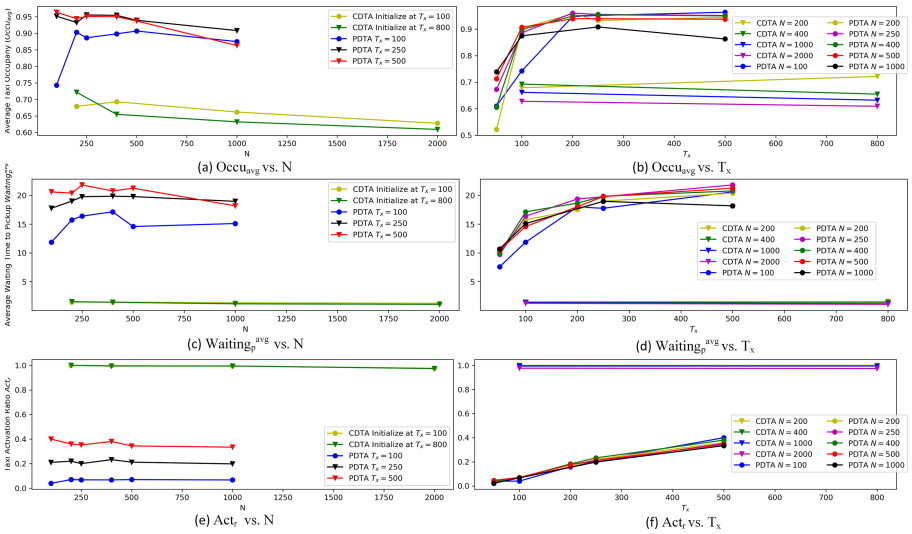

Fig. 5. Performance results

6.4 Evaluation Results and Analysis

In this section we present our results along with in-depth analysis and explanation. Our empirical results have been compared with a previous distributed ride-sharing system [15] which we call 'PDTA' for the sake of convenience. Similarly, we call the currently proposed ride-sharing solution as 'CDTA'.

Figure 5 (a) and (b) shows the variation of average taxi occupancy $Occu_{avg}$ of our system with respect to changing values of total number of taxis (N) and the variable transmission range (T_x). Similarly, Fig. (c) and (d) are plotting the average passenger waiting time to pick up ($Waiting_p^{avg}$) and the Fig. (e) and (f) are depicting how much of the available taxis in the system are being used for ride-sharing (Act_r).

Figure 5 (e) and (f) shows that the adaptive T_x significantly increase the Act_r over PDTA and reaches as high as 100%. For the CDTA, T_x values of 100 and 800 hardly has any effect on the final result. Similar result can be seen in Fig. 5 (b) and (d) for $Occu_{avg}$ and $Waiting_p^{avg}$. As a result of achieving high Act_r, our system (CDTA) improves average taxi utilization $Occu_{avg}$ (see Fig. 5 (a)) compared to PDTA. Since, the adaptive T_x increases overall ride-sharing success rate, the passenger waiting time is reduced as shown in Fig. 5 (c). We can notice that $Waiting_p^{avg}$ for CDTA is reduced by at least 10 times while compared to the PDTA.

While analyzing physical significance of the results, we notice the following trends. Act_r in Fig. 5 (e) is very high because our adaptive T_x control strategy can improve T_x in sparse zones and thereby achieve better taxi utilization and high rate of ride-sharing success. Also, we can see in Fig. 5 (c), the overall passenger waiting time before pickup is reduced. We believe that our adaptive T_x method

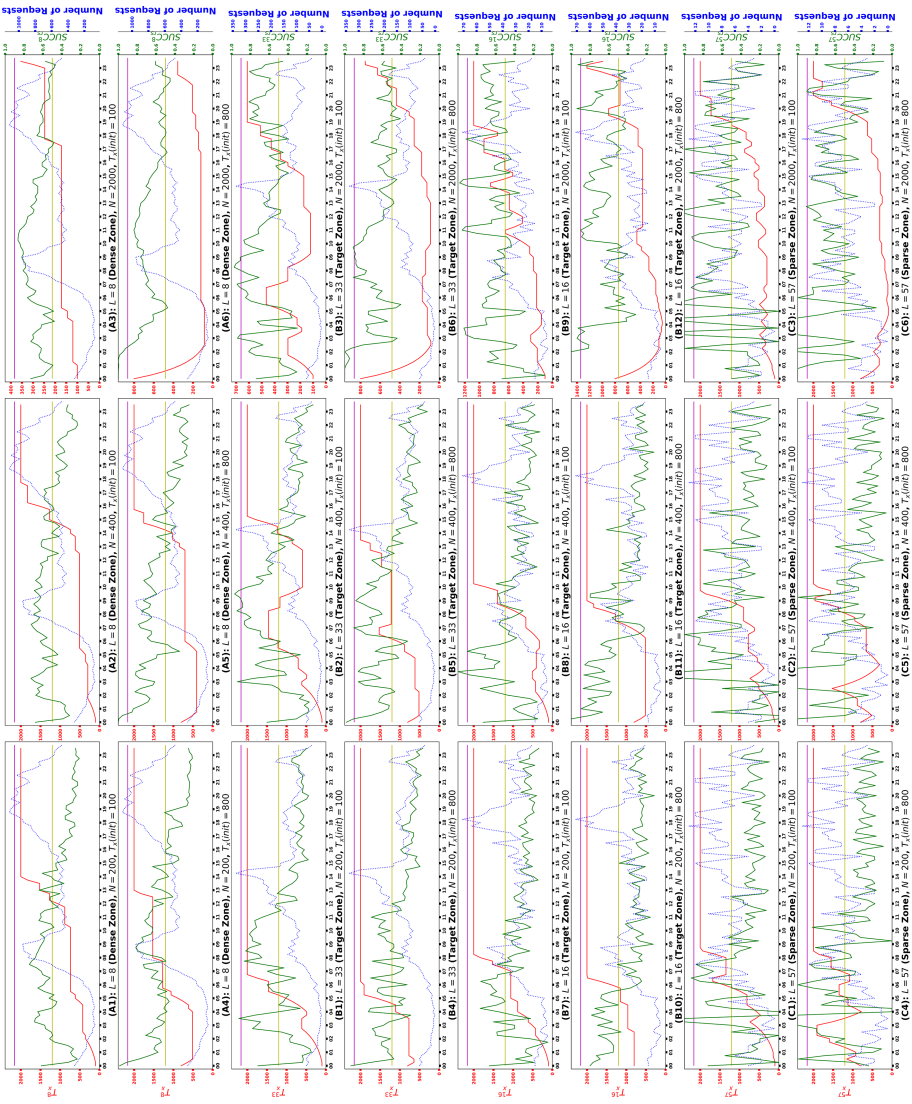

Fig. 6. A1 to A6: For Sample Dense Zone 8, B1 to B12: For Two Sample Target Zones 33 and 16, C1 to C6: For Sparse Zone 57

reduce request propagation in dense zones which reduces average response time and leads to faster taxi response. Since, the bulk of the request originate from the dense zone, reducing processing overhead for those brings down the overall processing time and endures less waiting time before passenger pickup.

Figure 6 represents variation of adaptive T_x with respect to changing values of taxi numbers (N) and initial transmission range ($T_x(init)$) for the three types of zones - *dense*, *sparse* and *target*. We observe that for same community area (L)

and same N, adaptive T_x always stabilizes to the similar range even the $T_x(init)$ varies from anywhere between 100 to 800 m (Fig. 6 (A1)). On the other hand, if we vary N, while keeping $T_x(init)$ constant, the adaptive T_x will change until it reaches the maximum request propagation range (T_{max}). We must clarify that with less value of N, T_x keeps increasing until it reaches T_{max} in order to find matching rides for the user. Clearly, with higher value of N, ride sharing success rate can stay high even at a lower value of T_x which reduces communication and processing overhead (Fig. 6 (B4, B5)). Evidently, with fixed N and $T_x(init)$, the dense zones require lower value of T_x than the target and sparse zones (Fig. 6 (A1, B1, and C1)).

7 Conclusion and Future Work

In this paper, we have proposed a novel ride-sharing system and algorithm which works by peer-to-peer interaction between the major participants - passengers and taxi drivers. In order to ensure a high success rate of ride-sharing and to reduce communication and processing overhead, we have introduced a server that takes passenger feedback on the success or failure of their ride-sharing attempt. This information is used to generate historical analysis of the Spatio-temporal distribution of ride-sharing success rates across the Chicago metropolitan area. Based on that, the server recommends different transmission range for ride-share request propagation to different users depending on their current location and time. Our approach has been evaluated to improve the ride-sharing success rate while reducing the message overhead and overall response time. In the future, we would like to apply our technique to other metropolitan regions of the world where traffic data is made available. We also plan to experiment with parameters such as κ and Γ to study their effects on the system performance in more detail.

Acknowledgments. This work was supported by Dr. Jens Mueller, Director, High-Performance Computing Services at Miami University through the use of the Redhawk cluster.

References

1. Chicago data: taxi trips. data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew/. Accessed Jan 2020
2. Chicago data: transportation network providers. data.cityofchicago.org/Transportation-Network-Providers-Trips/m6dm-c72p/. Accessed Jan 2020
3. Agatz, N., Erera, A., Savelsbergh, M., Wang, X.: Optimization for dynamic ride-sharing: a review. European J. Oper. Res. **223**(2), 295–303 (2012)
4. Baldacci, R., Maniezzo, V., Mingozzi, A.: An exact method for the car pooling problem based on lagrangean column generation. Oper. Res. **52**(3), 422–439 (2004)
5. Barann, B., Beverungen, D., Müller, O.: An open-data approach for quantifying the potential of taxi ridesharing. Decision Support Syst. **99**, 86–95 (2017)
6. Bathla, K., Raychoudhury, V., Saxena, D., Kshemkalyani, A.D.: Real-time distributed taxi ride sharing. In: 2018 21st International Conference on Intelligent Transportation Systems (ITSC), pp. 2044–2051. IEEE (2018)

7. d'Orey, P.M., Fernandes, R., Ferreira, M.: Empirical evaluation of a dynamic and distributed taxi-sharing system. In: 2012 15th International IEEE Conference on Intelligent Transportation Systems, pp. 140–146. IEEE (2012)
8. Ma, S., Zheng, Y., Wolfson, O.: T-share: a large-scale dynamic taxi ridesharing service. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 410–421 (2013). https://doi.org/10.1109/ICDE.2013.6544843
9. Ma, S., Zheng, Y., Wolfson, O.: Real-time city-scale taxi ridesharing. IEEE Trans. Knowl. Data Eng. **27**(7), 1782–1795 (2014)
10. Psaraftis, H.N.: A dynamic programming solution to the single vehicle many-to-many immediate request dial-a-ride problem. Transp. Sci. **14**(2), 130–154 (1980)
11. team, F.C.D.S.: Prophet: Automatic forecasting procedure (2021). github.com/facebook/prophet
12. UN: UN Report (2018). www.un.org/development/desa/en/news/population/2018-revision-of-world-urbanization-prospects.html. Accessed 29 Apr 2021
13. Wang, Y., Zheng, B., Lim, E.P.: Understanding the effects of taxi ride-sharing-a case study of singapore. Comput. Environ. Urban Syst. **69**, 124–132 (2018)
14. Xiang, Z., Chu, C., Chen, H.: A fast heuristic for solving a large-scale static dial-a-ride problem under complex constraints. European J. Oper. Res. **174**(2), 1117–1139 (2006)
15. Yu, H., Raychoudhury, V., Silwal, S.: Dynamic taxi ride sharing using localized communication. In: Proceedings of the 21st International Conference on Distributed Computing and Networking, pp. 1–10 (2020)