CARE-Share: A Cooperative and Adaptive Ride Strategy for Distributed Taxi Ride Sharing, forthcoming, IEEE-Transactions on Intelligent Transportation Systems

Preprint · March 2021

DOI: 10.13140/RG.2.2.20322.27846

CITATIONS
0

READS
280

5 authors, including:

Aishwarya Manjunath

8 PUBLICATIONS **7** CITATIONS

SEE PROFILE

Vaskar Raychoudhury
Miami University

68 PUBLICATIONS **786** CITATIONS

SEE PROFILE

Snehanshu Saha
BITS Pilani, K K Birla Goa

249 PUBLICATIONS **1,148** CITATIONS

SEE PROFILE

CARE-Share: A Cooperative and Adaptive Ride Strategy for Distributed Taxi Ride Sharing

Aishwarya Manjunath*, Vaskar Raychoudhury†, Snehanshu Saha‡, Dhananjai M. Rao†, Saibal Kar§
*PES University, Bangalore, India
†Department of Computer Science & Software Engineering Miami University, Oxford, OH, USA
‡Dept.of CS&IS and APPCAIR, BITS Pilani K.K. Birla Goa Campus, India
§Centre for Studies in Social Sciences, Kolkata, India & Institute of Labor Econonics, Bonn, Germany

***Abstract*—Given the fast growth of on-demand transportation services and ride-sharing platforms, the concept of private vehicle ownership is rapidly declining. Although there are multiple fully-grown ride-sharing systems, they are proprietary and centrally controlled. Facilitating ride-sharing using a localized distributed coordination between the riders and the drivers is in need. However, fully distributed systems deal with a large number of variables and objectives and are often sub-optimal. In this paper, we propose a distributed ride-sharing system with multiple objectives which are often conflicting to each other. Therefore, we model it as a multi-objective optimization problem and solve it using the Ant Colony optimization technique which sports a multi-agent behavior. We critically analyze the spatio-temporal challenges posed by the ride sharing problem and define novel performance metrics to capture the underlying subtlety of the distributed system performance. An in-depth experimentation with recent large-scale single-ride taxi trip data from Chicago shows that our solution can ensure up to 79.65% success rate of ride sharing. We have shown that ride sharing is more successful during non-peak traffic hours due to less contention and a healthy balance in passenger and taxi numbers. Further, it has been observed that ride-sharing always reduces the *total distance travelled* by all the taxis and the *total number of taxis* on-road; both of which positively impact road congestion and environment. The results obtained from the experiments are very much comparable to real time behaviour of taxi networks. Finally, a revenue framework is proposed to analyse nuances of the operating environment.**

I. Introduction

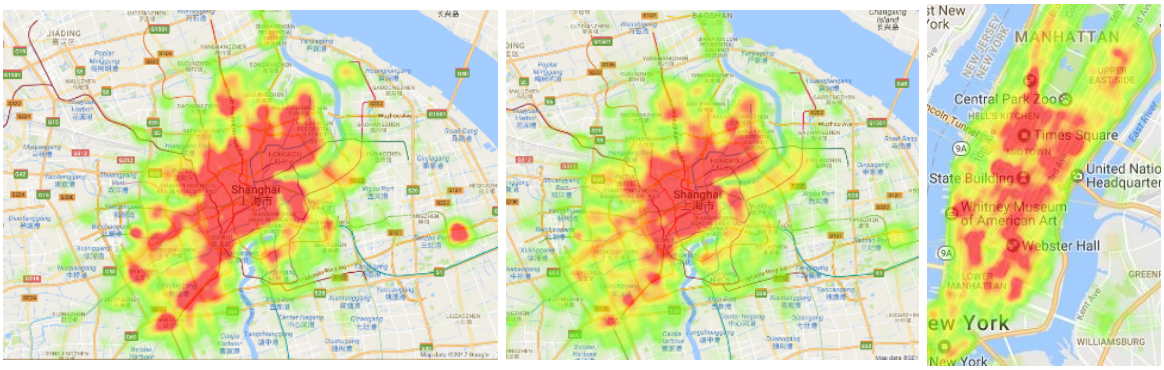
Taxis play a crucial part in urban transportation system and it is undergoing a paradigm shift in the mode of operation. A recent study by the Boston Consulting Group (BCG) predicts that in near future "25% of Miles Driven in US Could Be in Shared Self-Driving Electric Cars" [14]. This new direction of transportation is being credited to three major trends - Ride Sharing, Autonomous Driving, and Vehicle Electrification. Another report mentions that by 2030 personal cars in USA will largely be replaced with on-demand ride sharing using (autonomous electric) taxicabs [13].

Ride sharing has recently been very popular [23] as the shared rides are cheaper than individual ones. Moreover, it reduces the overall carbon footprint. In (Figure 12a), we have shown w.r.to Chicago city taxi ride data [22] that majority of the rides are spatio-temporally co-located, i.e, they originate and culminate at a similar time window and from/to some specific areas in the city. Similar spatial patterns can be observed for a large-scale taxi GPS traces in Shanghai (China) [2]. The heatmap of the pick up (Figure 2(a)) and drop off (Figure 2(b)) locations on a usual working day on Feb. 20, 2007 shows their close similarity. This surely shows that ride sharing can be a better option for travelling as it *reduces* (1) total number of vehicles on road by improving average taxi occupancy and (2) total distance travelled by all passengers by overlapping common shared paths. We have observed that (Figure 2(c)) the maximum taxi occupancy rate over a 22-day-period for 535 taxis in San Francisco [20] is just 53%, i.e, on average, only half the time a taxi has passenger. A (temporal) comparison of average hourly passenger counts in Shanghai, San Francisco and New York (NYC) [11] taxi ride data over an usual working day (Figure 2(d) shows distinctive peak and non-peak hours commensurate with usual office going/returning traffic. This shows that taxi ride sharing problem needs both spatial and temporal consideration for meaningful analysis.

However, ride sharing has its challenges while compared to carpooling. Carpooling is a static and pre-coordinated way of sharing a vehicle between a number of passengers. Ride sharing, on the other hand, is dynamic. In the most dynamic form, taxis must calculate and/or adjust their routes, in real time, to accommodate incoming ride requests. They also must make sure that the already confirmed as well as on-board passengers can also be picked and dropped according to their pre-arranged time. There are also other dynamic challenges, such as, last moment cancellation by passengers, road conditions, etc. One more critical challenge we faced is the absence of a dedicated publicly available taxi ride-sharing data.

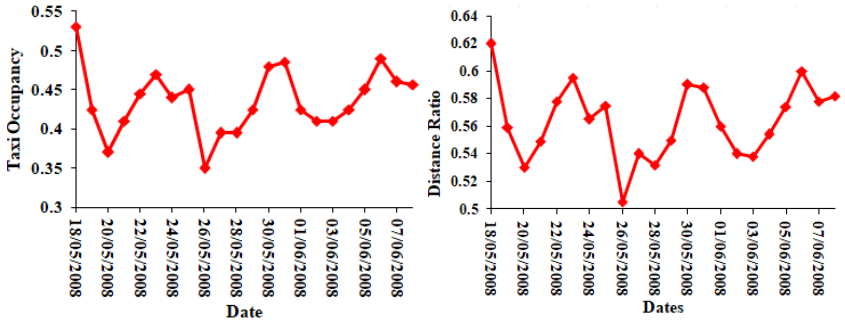
Existing taxi systems like Uber/Ola/Lyft are centralized systems [18]. Their centralized control allot taxis to passengers based on the shortest distance and other optimal factors. Apart from the scalability concerns associated with centralized systems, setting up such a large-scale system requires huge initial investment. Also, the taxi drivers receive only partial fare of the ride and that too on a weekly cycle [16] and they are solely dependant on the system to connect with the passenger.

An efficient distributed algorithm for ride sharing [8] has been introduced recently, which works by direct localized communication between nearby passengers and taxi drivers.



(a) Pick Up Location Hotspots (b) Drop-off Location Hotspots (c) Pick Up, NYC

Fig. 1: Pick and Drop Location Hotspots from Shanghai (Feb.20, 2007 - Weekday) and New York City (Morning Pick Up)



(a) Occupancy Rate of Taxis in SFO (b) Distance Ratio of Taxis in SFO

Fig. 2: Spatio-Temporal Analysis of Taxi Trip Data from Shanghai, San Francisco and New York City (Weekdays)

The results show the benefits of such a system including reduced road traffic, increased passenger savings and driver earnings. However, the authors did not consider optimizing the diverse conflicting goals. The paper did not mull over the complexity of finding a optimal solution under the multi-objective setting. Moreover, a critical research challenge in the ride sharing domain is the absence of any publicly available data dedicated to shared taxi trips.

In this paper, we model the dynamic and distributed taxi ride sharing problem and path estimation in a local cooperative taxi network as a multi-objective optimization problem. Passengers directly communicate with taxi drivers within a predefined wireless transmission range (T_x) and the requests are processed locally in the runtime by the taxi for making suitable ride decisions. Taxis forward the passenger requests to their peers in a hop-by-hop manner after calculating the optimality of each request and only if certain inherent constraints are not met. Every taxi in the network (has an associated pheromone matrix with it) executes Ant Colony Optimization Algorithm. We have proposed a novel framework with constraints and objective functions to find optimal match between taxi and a particular passenger request. This makes sure that a passenger is served by the best taxi, reducing his waiting time, travelling time, and fare. At the same time we also ensure that every driver gets a fair chance based on his/her location and the number of requests s/he has already served, so that a reasonable profit is made by all the taxi drivers during a given time frame.

In summary, this paper makes the following novel contributions:

- We propose the first ever Ant Colony based multi-objective optimization framework for solving the dynamic and distributed taxi ride sharing problem.
- We analyze large-scale single ride taxi trip data from Shanghai, San Francisco, NYC and Chicago to identify the similar spatio-temporal traffic patterns and trends. We have carried out a completely novel spatial analysis of the Chicago taxi trip data to distribute taxis to the most populated areas of the city, with precision and accuracy up to the building level. Finally, we decide to use the most recent data from the Chicago taxi trip dataset for the evaluation of our proposed algorithm.
- We have also provided a revenue model for the taxi drivers.
- We have defined benchmark performance metrics and our results of in-depth experimentation shows that the proposed distributed algorithm can achieve a ride sharing success rate up to 79.65% even in the presence of spatio-temporal coordination challenges.

II. Related works

Taxi ride sharing problems have been divided into centralized or distributed depending on the system architecture. Most of the taxi systems use a centralized approach. A centralized dynamic taxi ride-sharing algorithm has been proposed in [25] and it classifies an incoming request into one of nine classes and iteratively finds a solution. The nine classes are formed on the basis of number of passengers and male or female passengers only requests. A taxi that can satisfy the class constraint and accommodate more passengers within the same time interval and a small detour threshold is given more priority. This system works with a central database and a server that manages the requests and routing. The dynamic location of a taxi is periodically updated to the central server using GPRS. Among all the taxis that form the solution set the one that can reach the passenger first is chosen, it's travel path is rerouted and sent to the driver. The ride cost depends on total number of passengers, their preferences and distance travelled. For every five minute travel distance a standard fee increment is done. The main focus of this paper is to allow maximum shared rides for daily commuters with pick up and drop off to be "one-to-many" and "many-to-one".

A message-based distributed approach in taxi selection for a local cooperative taxi network has been proposed in [5]. The authors have also addressed the critical challenge in reducing blocking time, during when the unoccupied taxis are barred to participate in parallel selection processes. The algorithm starts when a passenger sends a request. All the neighboring taxis which have received it check if this the time to reach customer is less than the mentioned passenger reaching time in the request. If so, they replace it with their estimated passenger reaching time and forward to its neighbors and at the same time start their selection and rejection timers. The taxi at the farthest permissible end will not forward the request but send only Reply messages and start their acceptance timers. The intermediate nodes filter out the most minimum request to their parent and finally to the passenger. The passenger selects taxi with the most minimum passenger reaching time and sends back a confirmation. If the acceptance timer has expired than the taxi will unblock itself and hence blocking time is reduced. This paper considers only passenger waiting time as a decision factor in selecting the taxi and factors like minimizing passenger cost or detour distance of a taxi are not considered.

Depending on the nature of the solution, taxi ride sharing systems can be either static or dynamic. Static systems are predetermined group of riders travelling together as in carpooling. Some of the challenges in dynamic ride sharing problems has been highlighted in [4] and it includes dynamic arrival of riders and drivers, anticipation of future requests, deviations from the planned trip. Implementation of the dynamic ride sharing system in a large city can lead to large optimisation problem and hence a centralised system may not work well and decomposition approaches can be employed. The paper also highlights the need to understand user behaviour and preferences. On an overall outlook of the paper, the authors have highlighted the major optimisation challenges which arise when developing a dynamic ride sharing system.

Dynamic ride sharing has been addressed in [8] and in [19]. In [8], every passenger request is broadcast to a set of taxis in a local network (within 200 meters). The taxis that receive the request put the events into a temporary schedule and evaluate the eligibility of accounting them using a Triangulation method. The customer receives a set of responses with the cost, availability of seats and travel time which are sorted and the best solution is chosen. The main objective of this technique is to reduce the total distance travelled by the taxi and to decrease the delay in any passengers' travel due to the detour. The dynamic ride sharing has also been addressed in [19], where a fast taxi searching algorithm has been proposed with a lazy shortest path finding strategy which is claimed to satisfy 25% more shared ride requests in a simulation of GPS trajectory dataset. A central organization operates the whole service and maintains a spatio-temporal index for every taxi which is updated at regular interval whenever the taxi is a part of the service or when an event like pickup, drop off or re-routing occurs.

Modeling vehicle routing (both static and dynamic) as an optimization problem has also interested many researchers. In [12], [7], [21] the ant colony algorithm has been applied to Vehicle Routing Problems (VRP). Generally, there are multiple objective functions that need to be optimized. In [12], hierarchical ant colonies have been used, where one aims to minimize the number of vehicles and the other aims at minimizing total distance travelled with the former given more priority. The multi-objective optimization is adapted using two ant colonies, each trying to optimize one of the objective functions, that communicate using the pheromone updates. The main idea used is running two artificial ant colonies that parallelly try to optimize two objective functions over the previously computed shortest path, the ACS-VEI tries to find a solution that uses one vehicle less than the already computed solution and ACS-TIME tries to reduce the time take. If any of them are successful in finding a better solution then the shortest path is updated and the two processes are started again.

An extension of the MACS-VRPTW for a multi-objective context has been proposed in [7]. A single ant colony system is used to solve the three objective functions which minimizes the number of vehicles used, the total travelling time, and the total delivery time. The algorithm uses different visibilities for each of the objective functions. Various types of strategies that can be used in deciding pheromone trials for colonies, the paths which get pheromone update and defining the heuristic factors has been discussed in [6]. A generic framework which uses ACO to find the solutions to multi-objective problem is proposed and four different variants of the same are visited in detail. These variants are majorly based on the number of ant colonies and number of pheromone structures considered, where the approach to define pheromone factor, heuristic factor and pheromone update for each differ significantly.

A dynamic vehicle routing problem using the ant colony system is proposed in [21], where the entire day is divided into equal time slices. A request from a customer is assigned to a particular time slice. Each time slice runs a static VRP. The aim is to optimize the total travel time. The main elements are, the events manager, the ant colony system and the pheromone conservation procedure. The events manager receives the request from the customer, the events manager creates static problems for a time slice and runs the ant colony system and assign a commitment to a driver. The events manager also updates the pheromone matrix using the good solutions from a time slice. The paper formulates the solution considering the dynamic VRP to be similar to set of static VRP.

[17] has highlighted the use of online social networking sites, like Facebook, for dynamic ride sharing systems. The passenger will enter his pickup, dropoff locations along with his bid. Here, the drivers who are online and have less detour, among all of them, will be considered. The passenger can choose the driver based on the ratings the driver has received. The driver is free to choose the rides which have a smaller detour or a greater profit or a ride with a passenger who is connected to him on the social network graph. The paper also highlights the use of a multi-agent dynamic ride sharing system. Here, the main objectives include minimization of greenhouse emissions or the distance travelled and to maximize ride matches. One of the constraints include that a driver will accept the ride only if the dynamic request is compatible in terms of time with the other passengers destination reaching time. The optimal assignment between drivers and passengers is done by using a cost matrix and finding the best solution using a modified version Hungarian algorithm.

Ride sharing problem has been formulated and proved to be NP hard in [9], by reducing from the 3 Dimensional Matching problem. The authors have proposed a two-phase greedy approach to solve this problem. In the first phase, it matches the 2n requests into n pairs based on the shortest distance to serve any request pair but on the worse pickup choice. In the second phase, they have assigned drivers to the pairs formed in the previous phase, under the assumption that the distance from a driver k to a pair of requests is distance from k to the nearer pickup location of the two. This algorithm guarantees to output a solution with at most 2.5 times the optimal cost.

The paper [15], has used ant colony algorithm in a distributed approach for the travelling salesman problem. Use of multiple ant colonies has been explored in [17] to solve their own objective functions. The pheromone matrix maintained is unique for each of the ant colonies. It is claimed in the paper that use of separate ant colonies will be effective when the size of the problem increases and the number of vehicles increase. Although there is a limit on the communication, it is desired that different ant colonies will add to the concentrated efforts to simultaneously find paths by increasing the probability that a vehicle continues to use routes that it has found to be successful and not be distracted by the other routes. The paper also has explored the use of varying candidate list size to check if it will result in different solutions. From the experimentation that ant colony provides good results within 1% of the known optimum. Also, multiple ant colonies by assigning pheromone to each vehicle performed well only when the problem size increased. The method of determining the candidate list sizes becomes important when we want good solutions.

TABLE I: Variables Used

Variable	Meaning
P	Set of all passengers (in taxi), passenger $p_r \in$ P
T	Set of all taxis and $t_r \in$ T
T_x	Wireless Transmission Range/Radius
S_{loc}	Pick up (Source) Location Coordinates of p_r
S_{tm}	Preferred pick-up time of p_r
p_r^{wait}	Waiting time of p_r after CONFIRMATION and before pick-up
δt_1	threshold value for p_r^{wait}
τ_p	Timer at passenger end to receive REPLY
τ_t	Timer at taxi-end to receive CONFIRMATION
D_{loc}	Drop-off (Destination) Location Coordinates of p_r
D_{tm}	Preferred drop-off time of p_r
$p_r^{t_{sd}}$	Time to reach from S_{loc} to D_{loc} for p_r
δt_2	threshold $p_r^{t_{sd}}$ before dropoff/detour tolerance
t_r^{loc}	Current location of t_r
t_r^{cap}	Capacity of t_r
T_r^{on}	Set of onboard passengers in t_r (at current instant)
T_r^{con}	Set of passengers confirmed by t_r but yet to pick up
T_r^{un}	Set of passengers yet to be confirmed by t_r
$P_{t_r}^{P_{all}}$	Set of pick up locations of passengers in T_r^{on}, T_r^{con} and T_r^{un} for t_r
$P_{t_r}^{D_{all}}$	Set of drop-off locations of passengers in T_r^{on}, T_r^{con} and T_r^{un} for t_r
R	Route to plan through $[t_r^{loc} \cup P_{t_r}^{P_{all}} \cup P_{t_r}^{D_{all}}]$
$R_{i,j}$	$R_{i,j} \subseteq R, where i, j \in [t_r^{loc} \cup P_{t_r}^{P_{all}} \cup P_{t_r}^{D_{all}}]$
x_{ij}	$\begin{cases} 1, \text{if route i-j is taken,} \\ 0, \text{otherwise.} \end{cases}$
d_{ij}	distance between locations i and j
t_{ij}	time of travel between locations i and j
$prob_{ij}$	probability of choosing j after i in the ACO algorithm
p_{ij}	pheromone value between i and j
$Tot_{t_r}^{dist}$	Total distance travelled by t_r to serve $P_{t_r}^{P_{all}}$
$Tot_{t_r}^{time}$	Total time required by t_r to serve $P_{t_r}^{P_{all}}$
b	visibility influencing factor
β	factor to influence visibility w.r.to distance
M	(Ant) Memory (all the events already included in R)
a	Pheromone evaporation factor
C	A value for the combined detour and travel time of the optimal route R obtained according to Eqn 8

III. Problem Definition and Modeling

In this section, we formally define some key terms and concepts used in our algorithm. Prior to that, we list certain variables (Table I) and three different types of messages (Table II) used in our algorithm for communication between taxis and passengers.

TABLE II: Messages Used

Message	Meaning
$REQUEST(p_r, S_{loc}, S_{time}, D_{loc}, D_{time}, \delta t_1, \delta t_2)$	Request (single-hop broadcast) for ride sharing sent by p_r
REPLY $(t_r, t_{p_r}^{pick}, t_{p_r}^{drop}, C_{p_r}, C)$	Reply from t_r to p_r with estimated pick up time, drop off time, cost for ride and a value calculated from its route according to Eqn 8
CONFIRMATION (p_r)	Acceptance of Ride Sharing Service from p_r to t_r
ACKNOWLEDGEMENT (t_r)	Acknowledgment sent from t_r to p_r in response to *CONFIRMATION* (p_r)

A. Problem Definition

In this Section we define some of our key terminologies followed by the problem definition for ride sharing.

Passenger: Passenger is the prime entity in the ride sharing system and they send a REQUEST message for availing the service. Passengers distribution across the city is usually non-uniform which affects the performance of the ride sharing system.

Taxi: Taxis are independent entities which changes the location while travelling around the city. Taxi cabs are allowed to take a maximum of 4 ride sharing passengers at any time.

Events: A taxi ride for a passenger is composed of a pair of pick up and drop off events, each associated with a location (latitude and longitude) and a time stamp. Passengers in their REQUEST provide waiting tolerance time for pick up (δt_1) and drop off (δt_2). The δt_2 is also called *detour tolerance* as the detours are what responsible for the drop off delays.

Taxi Detour: Detours are deviations from the current 'path' of the taxi for picking up or dropping off of another passenger. Detours are only considered for new requests and calculated based on the existing passengers the taxi is transporting or has promised to transport (i.e., confirmed passenger request). Detour paths determine detour time and how it affects the existing passengers as every passenger has an associated *detour tolerance*.

Ride Sharing: Taxi ride sharing is a combinatorial problem. Given a set of passengers $P = \{p_1, p_2, ..., p_n\}$ and a set of taxis $T = \{t_1, t_2, ..., t_m\}$ each having a capacity of 4, the taxi ride sharing problem aims to find a set of passengers who can share the taxi ride for an entire journey or the part thereof. The requests for ride sharing can come in real time and continuously, which makes the ride sharing solutions to be sets of interleaved 'valid' events on a continuous time scale.

The 'validity' of an event is determined by whether a taxi can accept the REQUEST of which the event is a part. In order to decide whether an incoming REQUEST for ride sharing is possible to accept, a taxi considers its current occupancy as well as the detour tolerance of the existing passengers. Unless both the events associated with a REQUEST are valid, the REQUEST as a whole can not be accepted.

B. Problem Modeling

The ride sharing problem has multiple heterogeneous objectives to fulfill such as *pick up time optimization, detour distance optimization, total travel time optimization*, etc. Naturally, we model the ride sharing problem as an multi-objective optimization problem.

A typical multi-objective problem is bound to have a compromise solution since an optimal solution to all objectives simultaneously cannot be obtained. This "compromise" solution leads to an optimization framework where we try to reach a sub-optimal solution accommodating competing and collaborating objectives. This type of problems are hard to solve by classical optimization techniques where guaranteed convergence of the solution is one of the founding principles. Thus, we resort to Ant-Colony based meta-heuristic algorithm to find solutions to the constrained multi-objective problem.

An Ant Colony Optimization algorithm can build a sub-optimal solution from the strongest segments of the best solutions. ACO uses an iterative construction of solutions. The constructive behavior in the colony helps initiate transitions between each iteration. Essentially, this is a greedy approach but works well as a multi-agent system by leveraging the probability distributions between transitions for the iteration sequences.

However, the performance of ACO cannot be predetermined and depends on the tuning of parameters to solve the optimization problem. Therefore, it is difficult to comment on the superiority of ACO over its peers such as Genetic Algorithms, which uses stochastic operators or Simulated Annealing. However, we anticipate the flexibility of the ACO metaheuristic endowed with parallel search in computational grids and dynamic memory structures facilitated by pheromones to handle the challenges adeptly.

Our problem poses the challenge of accommodating diversity in objectives. The first objective function is defined to benefit the taxi driver while the second one aims to benefit the passenger. The constraints ensure that confirmed or on-board passengers are not unsatisfied with the taxi service when the taxi decides to serve another passenger in the same ride. The objective functions proposed are minimize detour and minimize total time.

Minimize detour: This is divided into non-peak and peak hours of the day in terms of number of passengers.

Minimize detour distance during non-peak hours: The objective function (Eqn 1) will be used during non-peak hours of the day.

$$min(\sum_{i,j} x_{ij}\Delta d) = min(\sum_{i,j} x_{ij} d_{ij} - Tot_{t_r}^{dist}) \quad (1)$$

Minimize detour time during peak hours of the day: The objective function (Eqn 2) will be used during peak hours of the day.

$$min(\sum_{i,j} x_{ij}\Delta t) = min(\sum_{i,j} x_{ij} t_{ij} - Tot_{t_r}^{time}) \quad (2)$$

Minimize total time (for both peak and non-peak hours): This aims to reduce the total travel time, t_{ij}

$$min(\sum_{i,j} x_{ij}t_{ij}) \tag{3}$$

The above objective functions are subject to the following constraints:

$$p_r^{wait} \leq \delta t_1 \tag{4}$$

$$t_r^{cap} \leq 4 \tag{5}$$

$$\forall p_r \in P, p_r^{t_{sd}} \leq time(S_{loc}, D_{loc}) + \delta t_2 \tag{6}$$

Each taxi which receives a REQUEST computes its route and sends REPLY comprising of detour and total travel time to the passenger. The passenger end has to make a decision to choose a single taxi solution. In order to determine this, a pareto front is obtained at the passenger end. When there are many solutions on the pareto front, then in order to select one taxi solution, a marginal utility function is used to determine the knee point. Marginal utility function is defined as in Equation 7.

$$f(U, w) = wf_1 + (1 - w)f_2 \tag{7}$$

In the above equation f_1 represents the detour and f_2 represents the total travel time. For every solution on the pareto front, the marginal utility equation is computed for 10 random values of w. The value of $w$is chosen in the range [0,1]. The count of pareto solution which leads to minimum value of the utility function is noted and the pareto solution which leads to maximum count over 10 computations is considered as knee point and the taxi corresponding to this solution is selected. For the taxi which is selected, a C value is computed according to Eqn 8. This value is used for updating pheromone matrix.

$$C = min(detour^2 + (Tot_{t_r}^{dist})^2)^{1/2} \tag{8}$$

where, detour during non-peak hours is defined as:

$$detour = \sum_{i,j} x_{ij}d_{ij} - Tot_{t_r}^{dist} \tag{9}$$

and detour during peak hours is defined as:

$$detour = \sum_{i,j} x_{ij}t_{ij} - Tot_{t_r}^{time} \tag{10}$$

IV. Dynamic Ride Sharing using Ant Colony Optimization

Figure 3 shows the overall architecture of our proposed ride sharing system which is distributed in nature. It has two main stake holders, passenger(s) and taxi(s) connected over the Internet. In this Section, we explain our pseudo-code followed by a complete walk-through of the actions performed by the system and the algorithms. The pseudo-code uses the variables and messages listed in Table I and Table II, respectively.

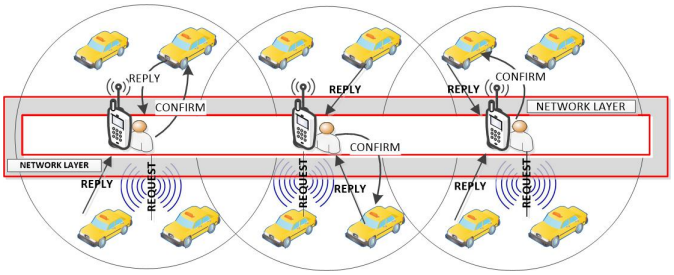


Fig. 3: Ride Sharing System Architecture

A. Proposed Algorithm

Our proposed distributed algorithm works in a synchronous message-driven manner. A city is divided into 1 x 1 KM grids and the pheromone matrix is divided into 3 x 3 KM grids (to make sure pheromones propagate through the city grids). The rows and columns of a pheromone matrix can represent the center of each grid. All locations within that grid will be represented by that center. However, these grids can also be replaced by clusters based on traffic intensity. Note, the passenger REQUEST triggers the ride-sharing process (Algorithm 1) and starts the timer τ_p to obtain REPLY back from the taxis until the timeout. The passenger REQUEST goes to a network layer entity which keeps track of the set of taxis in the same grid as the passenger and broadcasts the REQUEST to those taxis.

Each taxi has its own computing device and each computing device will have a pheromone matrix (adjacency matrix/list), initially assigned with random values. When a taxi receives a REQUEST, it executes Algorithm 2 which invokes the Ant Colony Optimization (ACO) algorithm (Algorithm 3) to obtain the new route which accommodates the new request in the taxi's existing path and the taxi sends back its detour and total travel time to the passenger (via REPLY message) and the timer τ_t is started to receive CONFIRMATION from passenger. This ensures that the taxis are not blocked forever. A passenger selects the taxi by finding the knee point solution which is calculated using marginal utility function 7 and sends CONFIRMATION to that. The taxi, in turn, sends an ACKNOWLEDGEMENT to the passenger. In case, the taxi-end timer(τ_t) expires before the CONFIRMATION from the passenger has been received, then ACKNOWLEDGEMENT in the form of a rejection would be sent to the passenger. The order and timing of the exchanged messages are shown in Figure 4.

Algorithm 1: Algorithm in Passenger-end

1 Receive set of nearby taxis T_{n} from Network Layer
2 Send REQUEST to taxis within T_{x}
3 Start timer τ_p
4 Collect REPLY from taxis before $\tau_p$expires
5 Find knee point solution among taxi solutions
6 Send CONFIRMATION to taxi with knee point solution

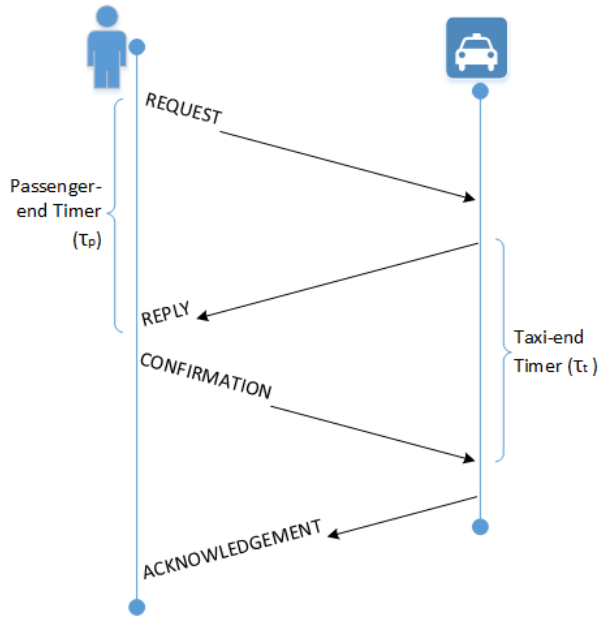


Fig. 4: Time Diagram for Message Exchange

Algorithm 2: Algorithm in Taxi-end

Taxi receives REQUEST
if $t_r^{cap} = 4$ **then**
    Broadcast to peers
    Stop computation // End Algorithm
Use ACO to calculate the new route //See Algo.3 below
Send REPLY
Start timer τ_p
if *taxi is confirmed from passenger before* τ_p *expires* **then**
    Broadcast its C value to other taxis
    Update its pheromone matrix according to Eqn 11
    Other taxis update their pheromone matrix using Eqn 11

If capacity constraint (Eqn. 5) is not satisfied while trying to include a new passenger request, then the taxi broadcasts the request to next hop peers. All taxis that received the REQUEST either send a response with the detour and total travel time or notify that it cannot accommodate the request to the passenger. One among all the taxis that sent REPLY to the passenger, receives a CONFIRMATION message from the passenger and updates its path to new route using the pre-computed route using an API [3]. The confirmed taxi then updates it's pheromone values according to the new path and broadcasts the same to all taxis in its grid. In this way, our distributed approach solves the overloading of a central server problem faced by centralized solutions.

Each taxi on receiving a REQUEST runs the ACO algorithm provided it satisfies the capacity constraint. The computation of the new path depends on the probability obtained from it's own pheromone and visibility factor values. The visibility is defined in terms of distance and time. The probability is given as:

$$prob_{ij} = \begin{cases} \frac{p_{ij} * d_{ij}^{b*\beta} * t_{ij}^{b*(1-\beta)}}{\sum_{u \notin M} p_{ij} * d_{ij}^{b} * t_{ij}^{b*(1-\beta)}} & if j \notin M, \\ 0 & \text{otherwise.} \end{cases}$$

Algorithm 3: Ant Colony Optimization Algorithm

All_Nodes = $P_{t_r}^{P_{all}} \cup P_{t_r}^{D_{all}}$
$b = 0.9$
$\beta = 0.5$
l = length(All_Nodes)
start=t_r^{loc}
Route = $Array(t_r^{loc})$
for $(i = 0; i < l; i++)$ **do**
    max_prob = 0;
    for $node \in All\_Nodes$ **do**
        pheromone_value = pheromone[start,node]
        visibility_dist =1/distance[start,node]$^{b*\beta}$
        visibility_time = 1/time[start,node]$^{b*(1-\beta)}$
        visibility =visibility_dist * visibility_time
        prob = pheromone*visibility //prob is $\equiv prob_{ij}$
        if *prob* > *max_prob* **then**
            max_prob = prob
            start = node
    Route.append(start)
    All_Nodes.remove(start)

The paper [10] explains local and global pheromone update, The local pheromone update is made when an ant chooses its next path so as to simulate the natural evaporation and to ensure that no path becomes too dominant whereas a global pheromone update is done after one complete iteration to incorporate deposition of pheromone on the path that gave the best cost as in [21]. This global update is given as:

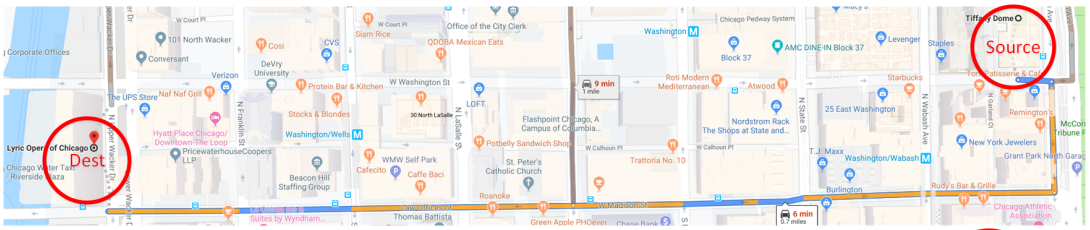
$$p_{ij} = (1 - a) * p_{ij} + a * C^{-1} \tag{11}$$

The order of i and j will be according to the optimal route. The updated values of the pheromone are utilized by the ants in the next iteration where the selection is dominated by the cumulative best path. After running the algorithm for several iterations it is proven that the ants follow the shortest path to their destination.

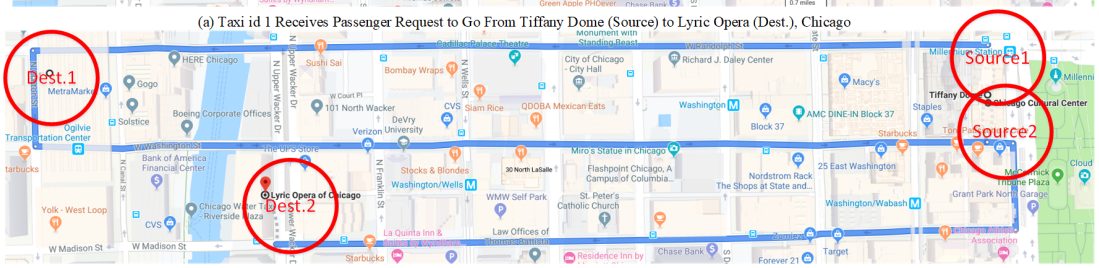
The taxi with knee point solution computes C value Eqn. 8. The route decided by this taxi will also be used to update the pheromone matrix of all other nearby taxis using Eqn 11.

B. Example Use Case

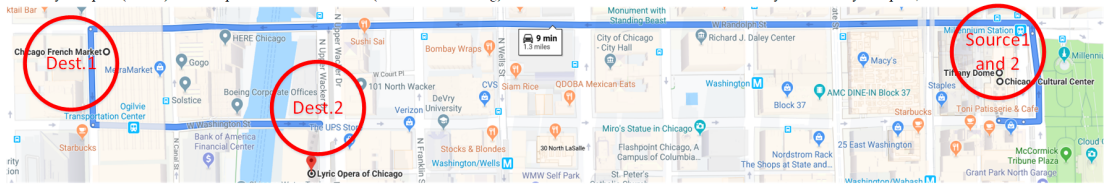
As an example (Figure 5), consider two taxis, with taxi Id as 1 and 2 respectively. Taxi with Id 1 has no passenger allotted to it and taxi with Id 2 has a passenger who was picked from Chicago Cultural Center (41.883764, -87.624953) and is on the way to his destination, Chicago French Market (41.884139, -87.640974). Suppose both the taxis (which are close by to the passenger request) receive a new REQUEST with pickup



(a) Taxi id 1 Receives Passenger Request to Go From Tiffany Dome (Source) to Lyric Opera (Dest.), Chicago



(b1) Taxi id 2 Carries one Passenger from Cultural Centre (Source 1) to French Market (Dest.1). It Receives Passenger Request to Go From Tiffany Dome (Source 2) to Lyric Opera (Dest.2). If the trips are done in this order (without ride-sharing): Cultural Centre → French Market → Tiffany Dome → Lyric Opera, Total distance: 2.5 miles



(b2) Taxi id 2 does the trips in this order (with ride-sharing): Cultural Centre → Tiffany Dome → French Market → Lyric Opera, Total distance: 1.3 miles

Fig. 5: Example Use Case with (b1) Sequential REQUEST processing and (b2) Ride Shared (Parallel) REQUEST Processing

location as Tiffany Dome (41.8839, -87.6250) and destination location as Lyric Opera of Chicago (41.8826, -87.6374).

Both the taxis run algorithm 3. According to this algorithm, All_Nodes array for taxi 1 will be Tiffany Dome, Lyric Opera of Chicago and for taxi 2 will be Tiffany Dome, Chicago French Market, Lyric Opera of Chicago. The taxis construct the route iteratively according to lines 7-19 where each start with their current location as first node and choose the next location as the node with highest probability. Suppose taxi 1 has its route to be current taxi location, Tiffany Dome, Lyric Opera of Chicago and taxi 2 route is current taxi location, Tiffany Dome, Chicago French Market, Lyric Opera of Chicago. Assuming the new route does not violate the capacity constraint Eqn (5) (Section III-B) for both the taxis, the detour distance for taxi 1 will be the distance from the current location to Tiffany Dome whereas taxi 2 has to undergo some detour from its previous route to accommodate the new passenger request. The taxi that best optimizes and is a knee point solution will be chosen and allotted to passenger. The pheromone value of the locations in the route of taxi that was allotted to passenger will be used to update the nearby taxis, according to Eqn (11). Considering a case where taxi 2 does not satisfy the capacity constraint Eqn (5) (Section III-B), then this taxi will broadcast the received REQUEST to nearby taxis. The taxis which receive the request, will run the ACO algorithm and compute the route. If any taxi cannot accommodate a REQUEST because the passenger waiting time is estimated to be more than 10 minutes then it sends a negative cost response to the passenger (without broadcasting to any nearby taxis as the other taxis will almost certainly be located further from the passenger).

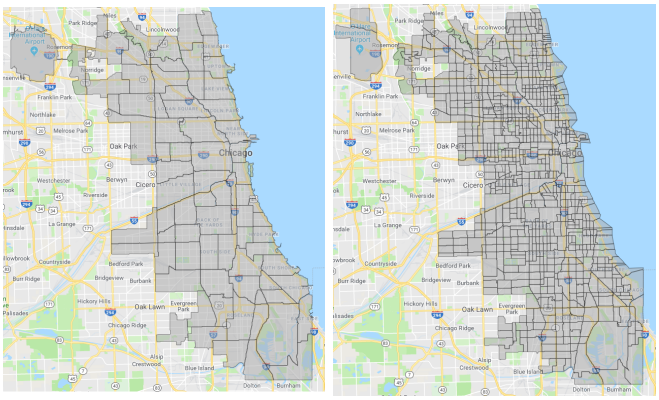
V. Dataset Description and Spatio-temporal Analysis

As already mentioned in Section 1, a critical challenge we faced is the absence of a dedicated publicly available taxi ride-sharing data. Hence, we have analysed single-user taxi trip datasets from four different cities and unearthed several common spatio-temporal patterns. In this section we describe our dataset and the data analysis processes undertaken.

A. Dataset Description

Our datasets include individual taxi trips from **Shanghai** *(Continuous GPS traces of 4000 taxis for 2 years ending in 2007)*, **San Francisco** (SFO) *(GPS traces of 535 taxis, 22 days in May, 2008)*, **New York City** (NYC) *(Single trip information worth 143,352,415 (≈143.35 million) trips for 1 year (2016))* and **Chicago** *(112,860,054 single-ride taxi trip data from 2013 till 2017)*. After careful consideration we have decided to evaluate our proposed algorithm over the Chicago taxi dataset which is both recent as well as voluminous and publicly available. The dataset has divided the Chicago Metropolitan region in 77 Community Areas (Figure 6(a)) and a even larger number of Census tracts (Figure 6(b)).

Over each dataset we carried out spatial analysis (Figure 2 and Figure 12a) to find out major (busy) pick up and drop off hot spots around the city. We also carried out temporal analysis to find out the peak and non-peak hours (by passenger count) during an average working day (24-hour-period). Intuitively



(a) Community Areas (b) Census Tracts

Fig. 6: Community Areas and Census Tracts in Chicago

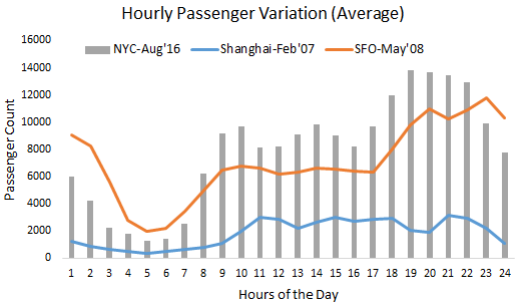


Fig. 7: Variation of Average Passenger Count

we inferred that ride sharing can only be successful iff both spatial and temporal patterns can be leveraged to recommend shared rides to passengers and drivers.

B. Temporal Analysis of Taxi Trip Data

Temporal analysis of taxi trip data from Shanghai, San Francisco and New York City has been plotted in Figure 2(d). Based on this variation of passenger count throughout an usual office day, we can distinctly identify certain non-peak (2-7 AM) and peak (8-11 AM and 5-9 PM) hours. Passenger count remains stable around 12 noon to 4 PM. We have further plotted (Figure 10) the passenger request variations for Chicago taxi-ride data over the weekdays and weekends of one whole week each during July 2017 summer. A similar trend was observed for another week in February 2017 (graph not shown due to space constraint). We have normalized number of requests in an hour with respect to the number of requests through the same week. The patterns during the weekdays are surely commensurate with those observed in Figure 2(d). Fridays and Saturdays have similar patterns while the Sundays are different. For Saturdays (party night), peak hours is 7 PM - 11:59 PM and non-peak hour is 4-10 AM. For Sundays, peak hour (much less passengers than Friday and Saturday) is 12 midnight - 3 AM and non-peak is 4-10 AM.

C. Spatial Analysis of Chicago Cityscape

In order to maintain the temporal privacy of the passengers and taxi drivers, Chicago Taxi trips are reported with a delay

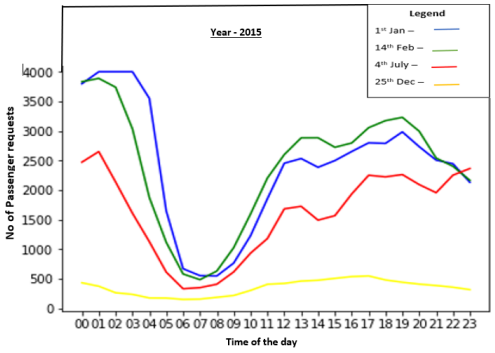


Fig. 8: Variation of Passenger Count on Holidays

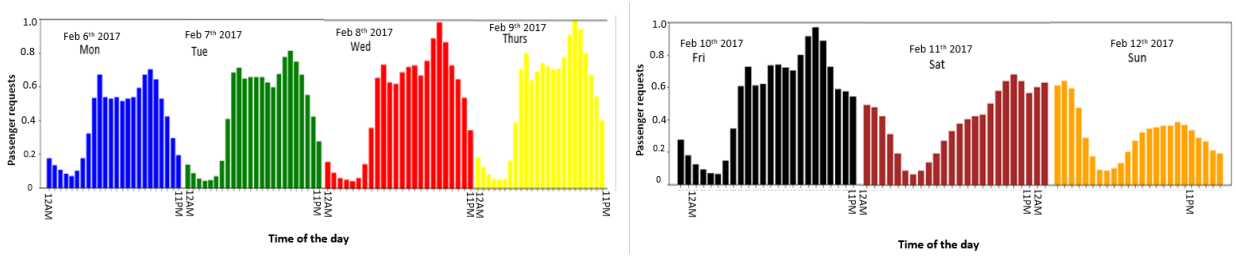
of a week to a month. Even then, for all the trips they have rounded start and end times to the nearest 15 minutes. Apart from temporal masking, they have also adopted spatial masking in which actual passenger pick up and drop off locations are replaced with the centroid location of the corresponding Census Tract (see Figure 11(a), (b)).

In order to tackle the coarse granularity of request origin and to distribute the requests more uniformly across the Census tracts, we carried out a spatial analysis of the Chicago metropolitan area.

In-depth analysis of the Chicago dataset has revealed that the the distribution of taxi-cab rides is heavily skewed, with a high concentration of rides originating/culminating from/into a few community areas (Figure 12a). The close correspondence between the pickups and drop-offs strongly suggests that commuters are traveling from their home to various locations and returning back home. Also, we can see that most cab rides are clustered around downtown Chicago with just the OHare airport being the outlier.

We have found that, the population density is strongly correlated with the number of ride sharing requests originating from an area. The Correlation coefficient is 0.87 with a p-value of 0.0 (very high confidence) (Figure 12b). Accordingly, in this study we have explored the use of a highly detailed model of the metropolitan area of Chicago to visualize population distribution up to the building level accuracy. The Chicago model has been developed by fusing data from three diverse data sources, namely: gridded human population data from ONRL LandScan [1], the Chicago metropolitan boundaries obtained from the City of Chicago [22], and building and road network data from Open Street Maps (OSM) [24]. Figure 13 presents an overview of the key steps involved in data fusion and model generation. These steps are incorporated into our model generation tools developed in C++. Below we describe in details the steps adopted in our spatial model generation.

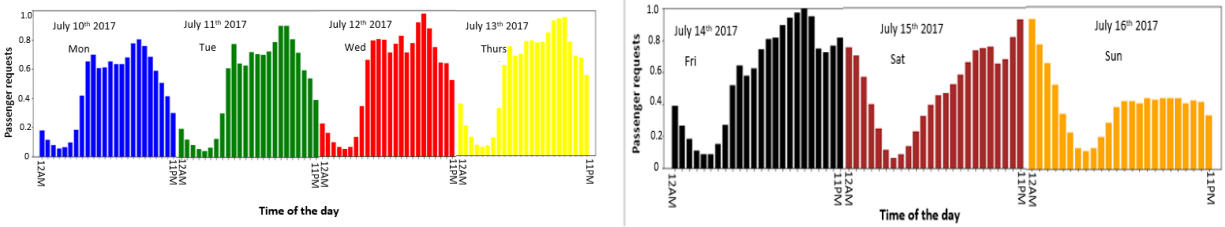
The initial phase of model generation extracts relevant population grids from ONRL LandScan dataset by intersecting it with the Chicago metropolitan area boundaries. The LandScan data is maintained by Oak Ridge National Lab (ORNL). It can be freely obtained from ORNL from https://landscan.ornl.gov/. LandScan has global population



(a) Feb 6-9, 2017

(b) Feb 10-12, 2017

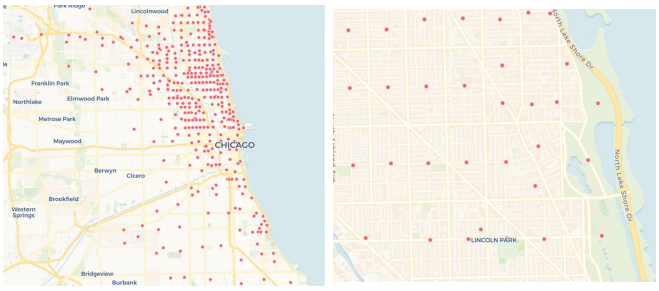
Fig. 9: Passenger Count Variations in Chicago over a week in Winter (February, 2017)



(a) July 10-13, 2017

(b) July 14-16, 2017

Fig. 10: Passenger Count Variations in Chicago over a week in Summer (July, 2017)

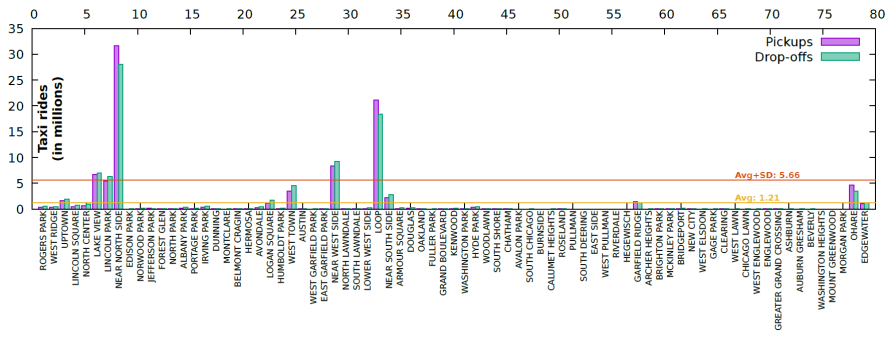


(a) Trip Locations

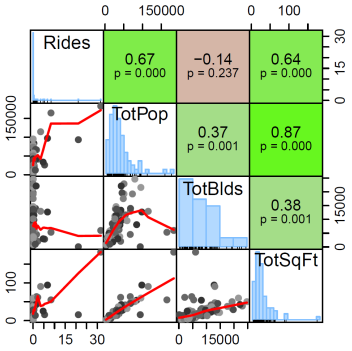
(b) Trip Locations (close-up)

Fig. 11: Census Tract Level Granularity of Pick up and Drop Offs in Actual Chicago Taxi Dataset

data that is approximately at 1 KM resolution (30" $\times$ 30") in a binary raster format with 20,880 rows and 43,200 columns. It is considered one of the most authoritative, unclassified (i.e., for non-military use) source for human population distribution. However, for this case study for this research we are interested only in the population areas of Metropolitan Chicago area. The boundaries of metropolitan Chicago was obtained as a vector GIS data from https://data.cityofchicago.org/Facilities-Geographic-Boundaries/Boundaries-Community-Areas-current-/cauq-8yn6. The city boundaries are defined by a collection of inclusion and exclusion GIS rings. In this context, a ring is a convex polygon with vertices indicated by latitude and longitude coordinates. More details on handling the boundary areas are omitted at this stage due to space constraints. The population grids for the Chicago has been extracted by intersecting the Chicago community boundaries with the LandScan grids. The intersection between the raster data and vector boundaries is accomplished by checking if any of the 4 corners of a grid lie within an inclusion ring. This intersection approach resulted in two types of edge cases. In the first type of edge case, 3 grids, at the edge of lake Michigan, the population grid spilled into the lake. These were manually corrected by redistributing the population to adjacent grids on land. In the second type of edge case, the grids around the edges of O'Hare airport were incorrectly included due to very small overlaps. These grids were manually tagged to be ignored. These edge case adjustments to the model are specified via a manual model adjustments text file as indicated in Figure 13. The resulting gridded population distribution for the Chicago metropolitan area is show in Figure 14. The population grids are then used to distribute people to homes based on building data from Open Street Maps as discussed below. At the second stage, Open Street Maps (OSM) data for Chicago Metropolitan area was used to extract nodes (i.e., vertices), roadways, and buildings. downloaded from http://extract.bbbike.org by specifying a rectangular region of interest. This approach was used because the size of the data set is sizable (910 MB) and most of the standard OSM WEB API's did not permit downloading such large regions. The downloaded data is in an XML format as specified by OSM. The key elements in the XML data are – 172 `nodes` that correspond to vertices in buildings and roadways, and 173 `ways` that correspond to road segments or buildings. A `way` consists of a collection of `nodes` along with additional metadata providing any additional information that may be available. The OSM XML data is processed using RapidXML C++ library. RapidXML has been used because of its memory efficient parsed representation of the XML via pointers and in-place changes to XML (by adding " string terminators). In our case, the 1 GB of XML was parsed into in-memory



(a) Pick-up and Drop-off Distribution in Chicago Community Areas



(b) Correlations between Population Density and Trip Requests in Chicago Community Areas

Fig. 12: Trip Distribution and Correlation in Chicago Taxi Trip Data

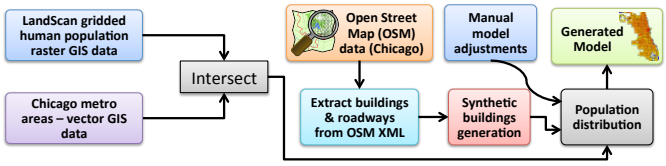


Fig. 13: Overview of the process of data fusion and model generation

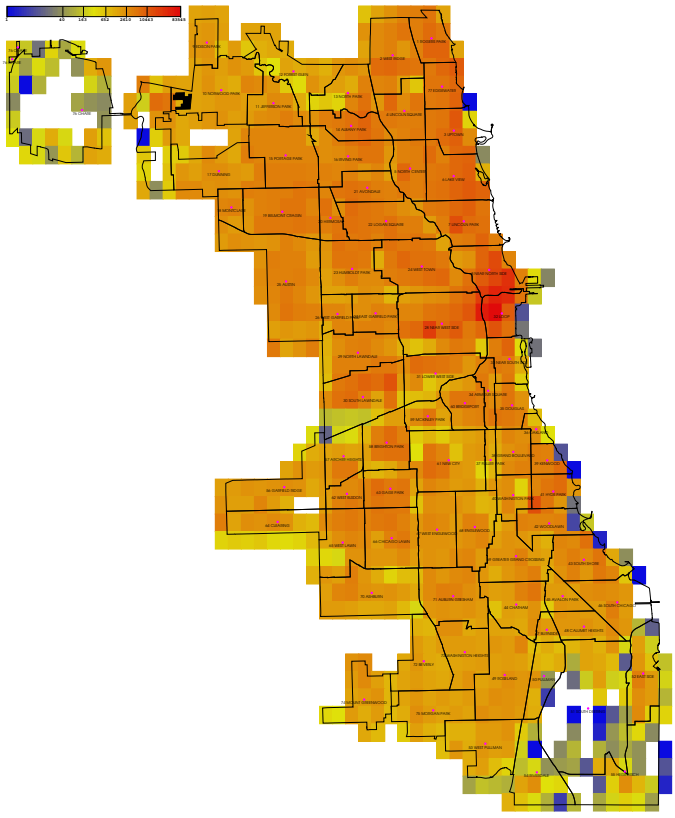


Fig. 14: Generated gridded population for Chicago metro area

representation that occupied about 5 GB of RAM. The parsed XML data is then

Synthetic homes generation. We observed that in certain "residential" areas of Chicago metropolitan areas, OSM data did not include residential buildings as shown in Figure 15(a). The corresponding region with buildings from Google maps is shown in Figure 15(b). The color of the buildings in Google maps has been darkened to improve readability. There were several large areas where residential buildings were missing and would result in an incomplete model. In order to address the issue of missing homes, we automatically generate synthetic homes in these regions. Synthetic home generation proceeds using the grids associated with population data. For each grid with human population, we identify roadways that are tagged as "residential" ways in OSM-XML but do not have any homes on them. The shapes of such empty residential-roads are then used to guide synthetic home generation. On each empty residential-road, homes of 1000 Ft^2 are generated generated as shown in Figure 15(c) and placed every 0.015 miles at a depth of 0.01 miles from the roadway. Note that synthetic homes generated in other grids is not shown for brevity. The homes are placed every 0.015 miles at a depth of 0.01 miles from the roadway. These values are averages estimated from the building data in OSM. As illustrated by a comparison between buildings in Google maps, our synthetic home generation is not perfect, but provides a good approximation of the homes in the area for the purposes of the analysis at hand. Out of 69503 ways in the Chicago metropolitan area, 3041 residential ways (about 4.38%) had 0 buildings on them. On these ways a total of 31238 synthetic homes have been generated adding about 4.92% of additional buildings to our generated model.

Entry coordinate computation. The next important step in building generation is to estimate the entry coordinates (i.e., latitude and longitude) for each building. The entry coordinates are essentially the coordinates on a street where a taxi ride would begin or end. The OSM-XML data does not include entry coordinates for buildings. Therefore, we have estimated

(a) View of OSM data (b) Buildings in Google maps (c) Generated Synthetic Homes

Fig. 15: Example of missing homes in OSM data but present in Google maps and the result of generating synthetic homes in a grid. Note that houses generated outside the grid are not shown for brevity.

entry coordinates for each building by finding the closest street intersection to the centroid of a building. The centroid computation uses the original building shapes from OSM-XML, when available. The thin blue lines in Figure 15(c) illustrates the entryway estimated for each building.

Performance improvements in model generation. The process of building generation and entryway estimation is time consuming due to the sheer size of the data and the mathematical intricacies of spherical geometry. In order to reduce model generation time, two different approaches have been used. From a spatial optimization perspective, we limit each operation to the bounds of rectangular grids shown in Figure 14. Specifically, for synthetic building generation and entry coordinate searches are only performed in ways that are present in a given ring. Restricting searches to a given ring significantly reduces the number of ways (to 1%) to be checked. In addition, We have parallelized each phase of model generation using Open-MP. Specifically, a given number of threads are used to perform the necessary operations for each ring in the model. Care has been taken to minimize synchronization necessary between threads for updating shared data structures. Using 8 threads on an Intel i7-3770K @ 3.5 GHz (*i.e.,* a conventional desktop CPU) the aforementioned optimizations enable model generation to complete in under 90 seconds, instead of over an hour without these optimizations.

1) Model persistence and visualization: The generated model is currently stored using a custom text file format. The persisted model includes information about population rings, nodes, roadways, and building information. The persisted model is used for further analyses. Furthermore, to aid visualization, two different types of diagrams in XFig file format are generated. The first type of diagram provides a summary overview as shown in Figure 14. The second type of diagram provides detailed information about a specified population ring as shown in Figure 15(c).

VI. Revenue Model

Consider a mass of taxi operators normalized to 1 who have to make a decision regarding picking up a passenger by making a detour, or follow a previously charted route for existing passengers in the car. The proposed structure applies to a shared taxi ride, where two or more passengers have to be accommodated. The choice of detour that has to be made by the driver involves the cost of fuel, the additional revenue to be earned and any unanticipated shock, like a traffic jam, that the non-detour (original) route is free of. The third condition may be considered as a strong assumption, and can be relaxed later. Nevertheless, it can be meaningful if we consider that both routes involve some risk, but the detour risk is greater. The risk involved in both direct and detour drives is measured by the coefficient of risk aversion 'r', according to the Arrow (1965) and Pratt (1964) measure of relative risk aversion, that follows a probability density function $g(r)$ supported by $r \in [0, 1]$. The taxi operator is risk averse and his preference follows a von Neumann-Morgenstern type expected utility specification.

The expected utility of profit for the driver choosing the straight (denoted by S) route is written as,

$$EU^S = -e^{\pi^S r} \tag{12}$$

where, $\pi^S = [p(x^S)x^S] - C(x^S)$ is the the profit from choosing the straight route. Note that, x^S is the number of miles travelled under route S, the price is a function of the number of miles travelled $(p'(xS) > 0, p'' < 0)$, the assumption of concavity is for allowing convergence and substitutability with other long-distance transport), while the variable cost (fixed cost does not offer additional inference to this model) is a strictly increasing function of the number of miles travelled. $C'(xS) > 0, C'' < 0$. We define $\delta > 0$ as a fixed return associated with the straight route. The fixed return is required to ensure interior solutions for the model, as we will show, and in terms of economic interpretation, it is an element to ensure that the taxi driver operates under risk and does not choose to accept a fixed reservation income of x that is available

from any other external sources, such as unemployment benefit. Substituting the above information, equation (12) can be written as $EU^S = -\int_0^1 e^{\{p(x^S)x^S+\delta]-C(x^S)\}r} g(r).dr$. The utility of profit for the driver choosing the *detour* (denoted by D) route is written as, $EU^D = -e^{\pi^D r}$ where, $\pi^D = [p(x^D)x^D + \delta + \varepsilon] - C(x^D)$ is the profit from choosing the detour route. Note that, $\varepsilon \in [-a, a]$ is the uniformly distributed support with a probability density function $f(\varepsilon)$ for the shock associated with the detour. x^D is the distance travelled under detour and remaining symbols are as before. Thus utility can be re-written as:$EU^D = -\int_0^1 \left\{\int_{-a}^{a} e^{\{p(x^D)x^D+\delta+\varepsilon]-C(x^D)\}r} f(\varepsilon).d\varepsilon\right\} g(r).dr$. The point where $EU^S = EU^D$, we determine indifference between the two options in terms of the fixed component of the model, namely, δ^*. Alternatively, one can also find critical values of any other parameter which would solve for this indifference. It is easy to show then that $\forall\delta > \delta^*$, the the taxi drivers would choose to take detour and conversely, $\forall\delta < \delta^*$ they would avoid the detour. One can find a distribution of drivers who would choose one or the other or do both from time to time. δ can technically represent many things, from EMI paid to banks for buying the car, insurance premium, cost of maintenance, fixed fee received etc.

VII. Performance Measurements

In this Section we describe our experimental setup and performance metrics followed by our result description and analysis.

A. Experimental Setup

We began simulation with a small corpus of taxis and passenger requests and scaled up to accommodate larger corpus (up to 800 taxis and passenger requests). These experiments were conducted on a high performance cluster with 2 compute nodes and 4 processors per node for running the experiments. Data from Chicago taxi dataset [22] was used. The results have been reported by dividing the day into peak and non-peak hours. The taxi system was setup on a local host. The taxis and the seeder (virtual network layer entity) were implemented as processes running on different ports. Flask module in Python was used for implementing this. Passenger requests were initiated as processes connecting to the various ports to make a request. Metrics mentioned in Section VII-B were measured. We have experimented with Chicago taxi trips data of December 11, 2016 (Sunday) after cleaning the data properly. Sunday traffic trends (from Saturday mid-night till Sunday mid-night) are explained in Section V-B.

The taxis (n) are initialized based on our spatial analysis but to a higher granularity than the building level (as the trip records are not available on that level). The city is divided into grids of size 1 $\times$ 1 km and the ratio of passenger (p_r) requests in a grid is x/p_r where x is the number of passenger requests coming from a grid. Then the number of taxis initialized in that grid is $x/p_r * n$. The passenger requests within each grid are then clustered using K-Means algorithm. The cluster centers are the initial positions which are allocated to the taxis.

For every confirmed passenger request a random 2 minute waiting time was considered at the pickup location of the passenger. The passenger would resend the request (randomly 2 or 3 times) within a random interval of 5 minutes if his request would be rejected. Ten iterations of the taxi system were simulated in order to check for consistency in the results across the 10 iterations.

B. Performance Metrics

Here we list our proposed performance metrics. Each metric is calculated for 24-hour-period from midnight to the next midnight and we observe how they varied over the entire day through peak and non-peak hours.

1) **Success Rate of Ride Sharing** ($SUCC_{rs}$)**:** Ratio of the total number of confirmed ride-sharing requests (successful) to the total number of REQUEST messages generated by all the passengers.
2) **Success Rate of n^{th} passenger (n<=4):**($SUCC_{rs}n^{th}PASS$)**:** Ratio of acceptance of n^{th} passenger, (given that there are already n-1 passengers in the taxi) to the total number of confirmed passenger requests. We take average of all taxis in the system.
3) **Distance Ratio of Ride Sharing (**$DIST_{rs}$**):** Ratio of the sum of total shortest-path distances between the pickup and drop-off locations of individual satisfied ride sharing requests to the total distance travelled by all the taxis with two or more passengers.
4) **Average Passenger Waiting Time (**WT_{rs}**):** Mean time elapsed between an instant a CONFIRMATION is sent to p_r and the instant at which p_r is picked up by a taxi. We take the average of all the passenger waiting times (p_r^{wait}).
5) **Average Number of Messages (per Shared Ride) (**$Messg_{rs}$**):** It is the ratio of the total number of messages exchanged in the system to the total number of successful shared rides.
6) **Request Processing Time (**RPT_{rs}**):** Mean time elapsed between between the instant a REQUEST is sent by P and the instant a corresponding REPLY comes back or the τ_p has expired, whichever occurs earlier.

C. Results

In this section we present our results with in-depth analysis followed by a description of the various simulation parameters and their values (see Table III). Same peak and non-peak hour trend (as in Figure 10 (b)) has been observed for a Sunday, Dec.11, 2016. Experiments were conducted separately for wireless transmission ranges (T_x), 100m, 250m and 500m, respectively during both peak and non-peak hours. It was observed that there were 1611 passenger requests during non-peak hours which were served by a total of 736 unique taxis. Given that all the trips were single-passenger rides, and a shared ride can accommodate a maximum of 4 passengers, we experimented with 1/4 (~200) and 1/2 (~400) of those 736 taxis. Similarly, during peak hours, there were 4022 passenger

TABLE III: Simulation Parameters

Parameters	Values
Data used for Date	December 11, 2016 (Saturday mid-night till Sunday midnight)
Peak hour time	12AM - 3AM
Non-peak hour time	4AM - 10AM
Transmission radius (T_x)	100 m, 250 m, 500 m
No of taxis (N_t) (Peak)	250, 500
No of taxis (N_t) (Non-peak)	200, 400
No. of passenger requests (Peak)	4022
No. of passenger requests (Non-peak)	1611
Passenger-end Timer (τ_p)	60 Sec, 90 Sec
Taxi-end Timer (τ_t)	60 Sec, 90 Sec

requests and a total of 973 unique taxis. So, we experimented with 1/4 (~250) and 1/2 (~500) of those 973 taxis.

Taxi Occupancy is the number of passengers being transported by a taxicab. The maximum possible occupancy of a particular taxi is assumed to be limited by the capacity of the vehicle and has been set to 4 in this particular research (Eqn 5). For a fixed number of taxis, occupancy increases with increase in the number of passengers and vice versa. Higher taxi occupancy results in higher rate of success for a ride-sharing system. However, occupancy might get affected by a number of factors, such as, time of the day, number of incoming passenger requests, wireless transmission range, etc.

We can observe from Figure 16 (a) that the overall $SUCC_{rs}$ increases with N_t for a fixed T_x and vice versa. Similarly, $SUCC_{rs}$ increases with T_x for a fixed N_t and vice versa. For both the cases, the reason is the availability of more taxis to handle the passenger requests (higher T_x implies that the passenger request reaches to more taxis). Usually we can notice that the $SUCC_{rs}$ is higher for non-peak hours compared to the peak hours. This can be reasoned as peak hours have more passenger requests than non peak hours and this leads to a tendency of higher taxi occupancy during peak hours. As taxi occupancy increases the chance of acceptance of an incoming passenger request is low as each incoming passenger request is bound to the constraints as in Eqn 5 and Eqn 6. The passenger waiting time constraint (Eqn. 4) also imposes that a taxi must reach the passenger within a certain time interval. This constraint is likely to get violated during peak hours due to traffic congestion, leading to lower $SUCC_{rs}$.

From Figure 16 (b) we can see that, generally, $SUCC_{rs}$ decreases (w.r.t Figure 16 (a)) as τ_p increases and the effect is more pronounced for peak hours with higher number of taxis N_t = 500 and higher value of T_x. As greater is the value of τ_p, the RPT_{rs} increases allowing the passenger to wait for a longer time before τ_p expires, however due to lower value of τ_t (60 seconds), the taxi end timer can timeout before getting the confirmation from the passenger resulting in lower $SUCC_{rs}$.

Figure 16(c) shows that the $SUCC_{rs}$ (marginally) decreases (w.r.to Figure 16(a)) at τ_t = 90 (during the peak hours) for higher T_x (=500), because as τ_t increases, the blocking time at taxi end also increases (which is the time that a taxi waits to get a confirmation from a passenger) leading to rejection of an incoming passenger request. This effect is more pronounced for T_x = 500m, as more no of taxis will be receiving passenger requests. For non-peak hours the values of $SUCC_{rs}$ remain mostly unaltered (compared to Figure 16(a)) as the number of passenger requests are more distributed during the non peak hours leading to lesser contention of the taxi queues.

In Figure 16(d) - $SUCC_{rs}$ decreases with increase in both τ_t and τ_p. But the effect is not as pronounced as observed in Figure 16(b). The effects are more for peak than the non-peak hours and can be explained similarly as done for Figure 16(a).

$DIST_{rs}$ (Figure 17) gives an intuition of the detour distance that a taxi takes and how it varies w.r.t taxi occupancy. For all the different categories considered, the values of $DIST_{rs}$ are all above 1 indicating that taxi ride sharing in fact saves total distances travelled compared to all the individual trips combined. Thus, it validates our assumption that ride sharing has environmental benefits as a whole. We can observe from Figure 17(a) that the $DIST_{rs}$ increases as T_x decreases and vice versa (for fixed N_t) as lower is the value of T_x, a taxi which is at a shorter distance will be chosen. This leads to lower detour or distance travelled by the taxi and according to the definition of $DIST_{rs}$, the value of denominator decreases and hence $DIST_{rs}$ increases. $DIST_{rs}$ increases (albeit marginally) with N_t (for fixed T_x). The effects are more pronounced for non-peak hours than the peak hours. This can be attributed to the fact that the $DIST_{rs}$ increases with the $SUCC_{rs}$ as higher taxi occupancy results in further shortening of total distance travelled, compared to the individual trips. Therefore, as N_t increases, $SUCC_{rs}$ increases and hence the $DIST_{rs}$ also increases. During peak hours the $DIST_{rs}$ decreases as the $SUCC_{rs}$ decreases.

Figure 17 (b) and (d) shows that for non-peak hours, $DIST_{rs}$ at T_x = 250 m is pretty high overall. The value of $DIST_{rs}$ would also depend on the set of passengers assigned to a taxi which impacts the detour distance travelled. Varying the timers (both τ_t and τ_p) does not have any significant effect on the $DIST_{rs}$ as $SUCC_{rs}$ is not significantly affected by varying the timers (already shown in Figure 16) and $SUCC_{rs}$ impacts the $DIST_{rs}$.

The results obtained for $Messg_{rs}$ are shown in Figure 18. Distributed algorithms aim to reduce the number of messages exchanged in order to reduce the demand on resources. Figure 18(a) shows the $Messg_{rs}$ exchanged between the stakeholders increases (approximately 2 times) with T_x and it is significantly higher for the peak than for the non-peak hours. Since the passenger requests are broadcast in nature, increased taxi availability increases the $Messg_{rs}$ exchanged. As T_x increases, more taxis are within the same T_x and during peak hours the number of passenger requests shoots up, and this increases the $Messg_{rs}$ exchanged. $Messg_{rs}$ marginally increases with higher N_t for peak hours and more than 2 times for non-peak hours, because with higher N_t probably more message drops as timer expires while handling contention during the peak hours, but it is not the same for the non-peak hours.

Since, $Messg_{rs}$ is defined as the ratio of the total number

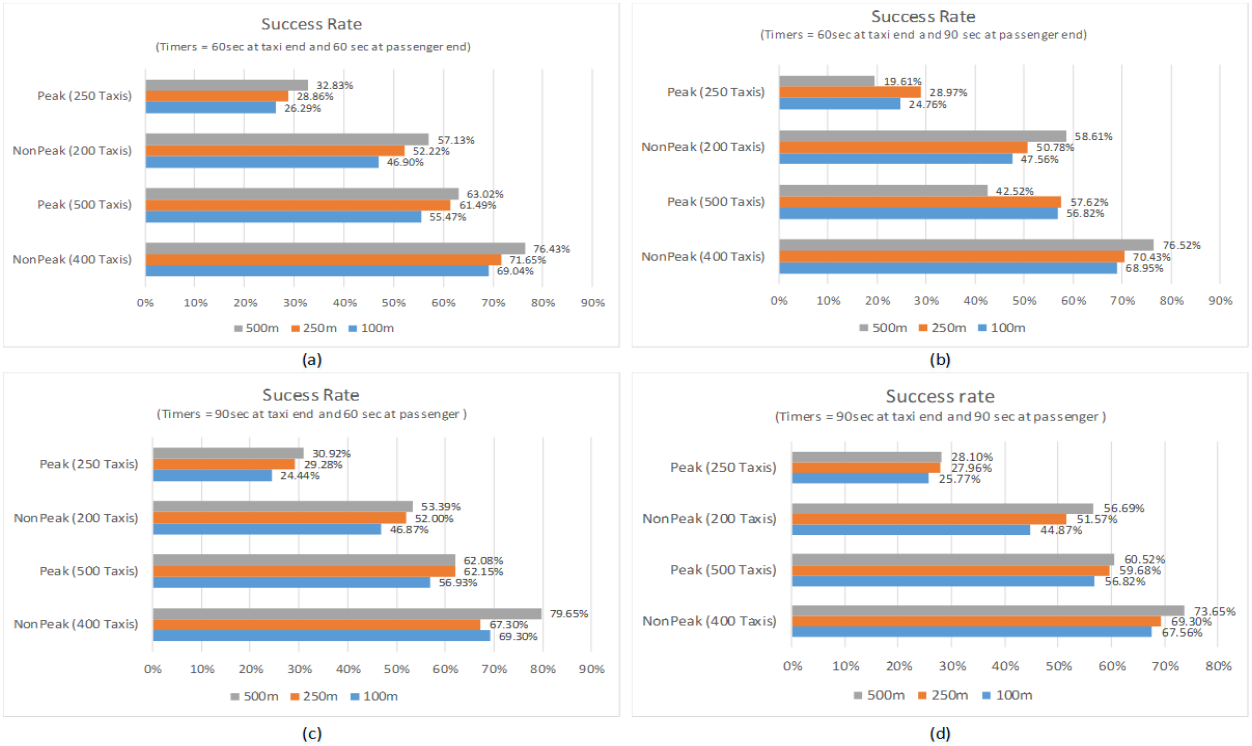


Fig. 16: $(SUCC_{rs})$ with varying τ_p and τ_t

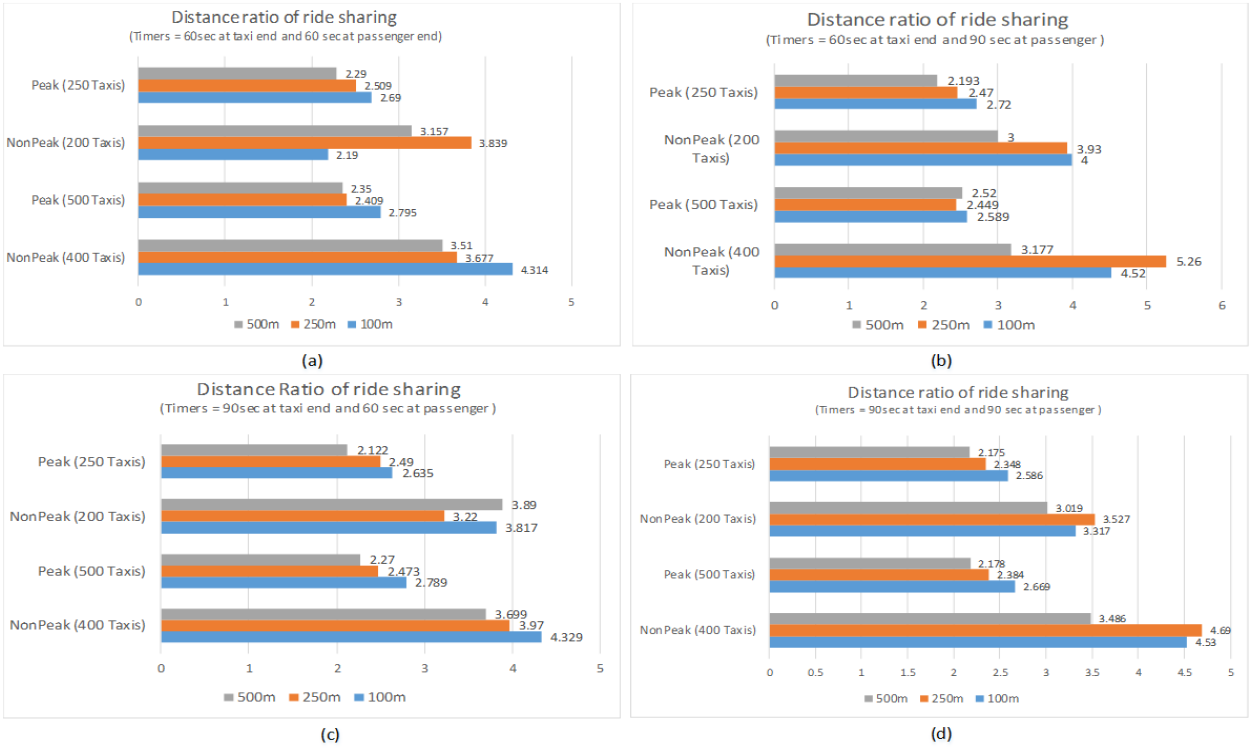


Fig. 17: $(DIST_{rs})$ with varying τ_p and τ_t

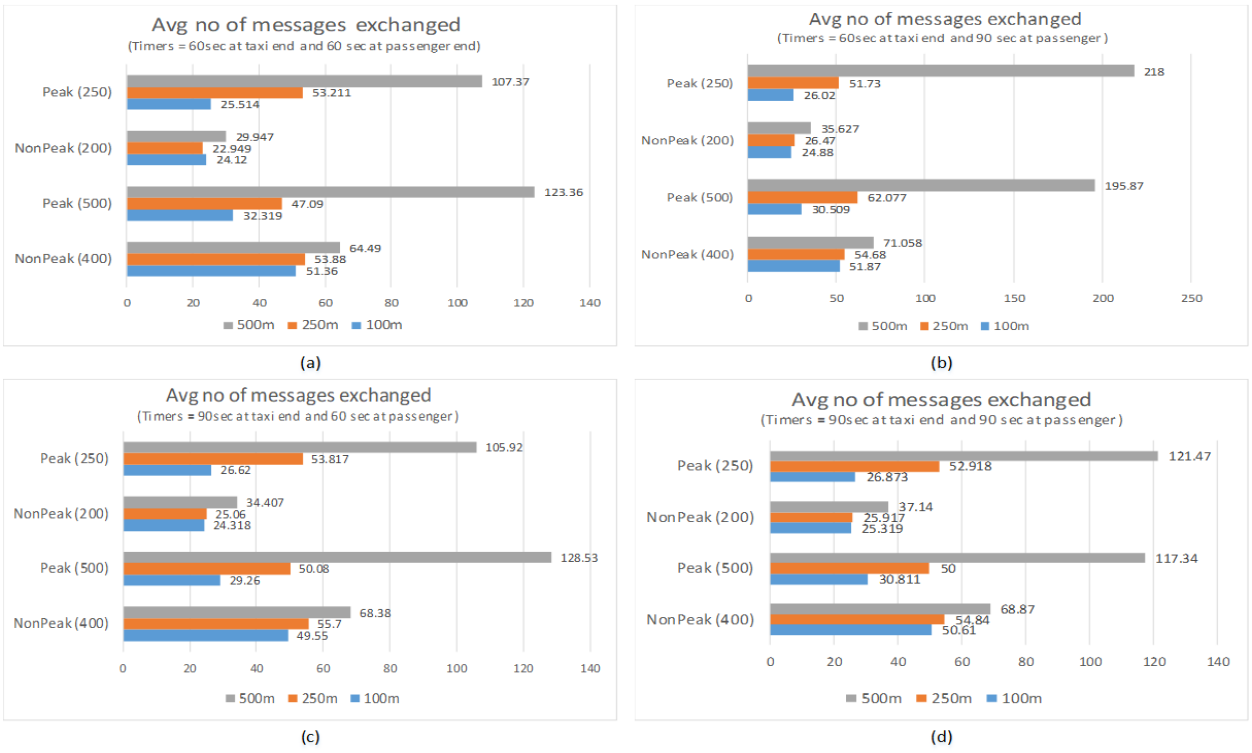


Fig. 18: ($Messg_{rs}$) with varying τ_p and τ_t

of messages exchanged in the system to the total number of successful shared rides. The high values of $Messg_{rs}$ observed in Figure 18(b) can be reasoned due to lower $SUCC_{rs}$ (as seen in Figure 16(b). $Messg_{rs}$ exchanged in the taxi network are not affected by the timers at the taxi end(τ_t) and passenger end(τ_p). Figure 18(c) shows that the increase in the value of τ_t has no effect as such on the $Messg_{rs}$ exchanged. Also, Figure 18(d) shows that increasing both the timers(τ_t and τ_p) have no different effects compared to the Figure 18(a).

WT_{rs} (Figure 19) is observed to be low across the system which is our objective as well. This is because the taxis were initialized based on distribution of passenger requests and this resulted in taxis to remain close to the areas with densely populated passenger requests. Figure 19(a) shows that the WT_{rs} increases with T_x (for fixed N_t) as greater is the T_x, the taxi will take a longer time to reach the passenger.

WT_{rs} decreases considerably with the N_t for non-peak hours considering fixed T_x . This is because, the greater is the availability of taxis within a fixed T_x, greater is the chance of selecting a taxi closer to passenger within the same T_x since the optimization function during non peak hours aims to reduce the detour distance. Lesser is the distance travelled, lesser will be the WT_{rs}. Due to congestion of traffic during peak hours, WT_{rs} marginally varies with N_t for peak hours considering fixed T_x. Both the effects are more pronounced with higher T_x values. WT_{rs} should not be affected with the variation of timers (τ_p and τ_t) as the WT_{rs} is calculated only after the request is confirmed by both the taxi and the passenger i.e., after the timers(τ_p and τ_t) have expired.

Lower RPT_{rs} is intended for handling the ride sharing requests. Figure 20(a) shows that the Request Processing Time (RPT) increases with T_x (for constant N_t) and it is much higher for peak than for non-peak hours. Also, RPT_{rs} increases with N_t (for constant T_x). This is because greater is the value of T_x or N_t, the $Messg_{rs}$ exchanged is higher as also seen in Figure 18, indicating that contention at the taxi-end (queue) is higher and therefore RPT_{rs} increases. Similarly, it can be reasoned for peak hours, as greater will be the number of incoming requests for a taxi, (as seen in Figure 23 , 24) compared to the non-peak hours (Figure 21, 22). During the peak hours, more number of requests get queued up at the taxi-end which increases the waiting time of a request in the queue before being processed and thereby increases the overall RPT_{rs}.

In Figure 20(b), we can find that for non-peak hours, RPT_{rs} increases with τ_p. As the value of τ_p increases, it allows more time for the request made by the passenger from timing out and which can increase the RPT_{rs}, in comparison to a condition in which if value of τ_p was lower then the RPT_{rs} would be less. RPT_{rs} is <= (τ_p + time to receive acknowledgment from taxi), so greater τ_p can lead to greater RPT_{rs}. Figure 20(c) shows that RPT_{rs} marginally varies with τ_t (compared to Figure 20(a)) for non-peak case compared to the peak case. The variation of τ_t does not impact the RPT_{rs} as the passenger confirmation request will be processed depending on the contention of the taxi end queue and this won't be affected by the timer. We can observe from Figure 20(d) that

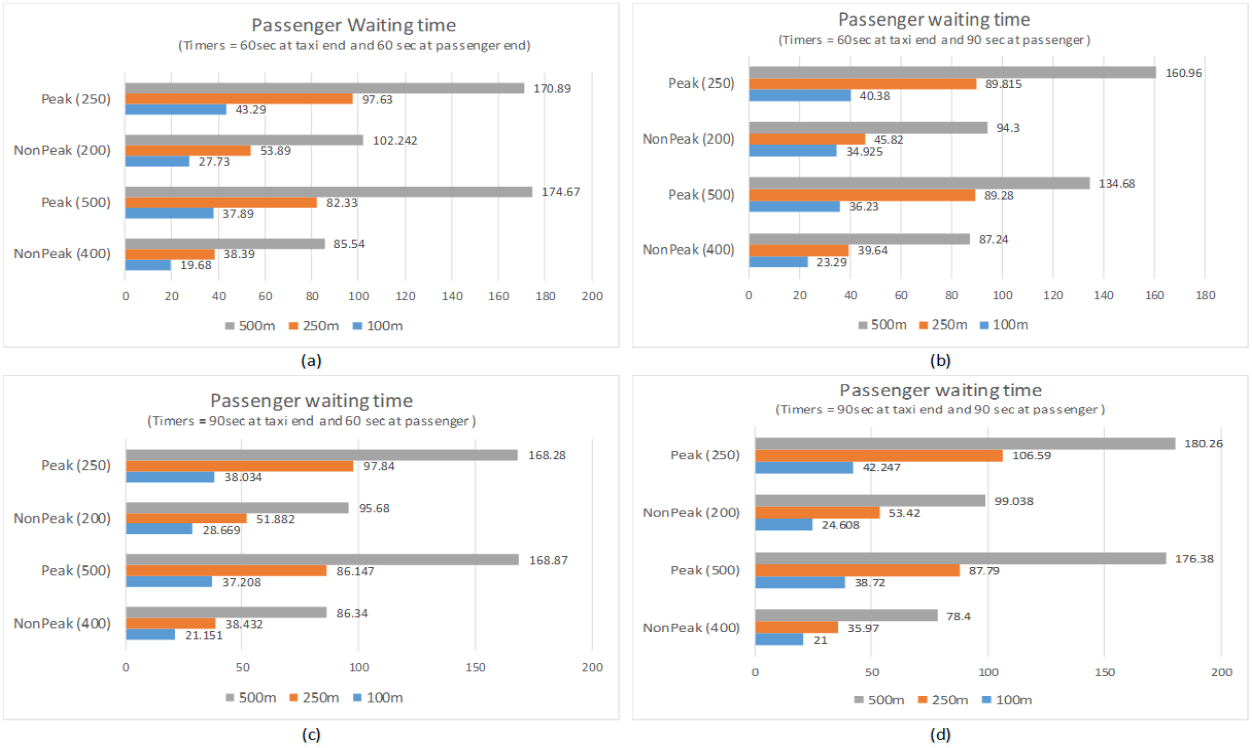


Fig. 19: (WT_{rs}) with varying τ_p and τ_t

higher values of both the timers (τ_p and τ_t), RPT_{rs} increases with N_t and also with T_x. The effects are more pronounced for the peak hours though. Since, more requests are to be processed during peak hour, RPT_{rs} increases as the timer values increase.

In order to understand how the queue length at the taxi end varies, the queue length was analyzed. In order to do so, whenever a request is received by a taxi these requests were collected in a list and after the simulation was done this would be visualized. The requests which are received by the taxi include the below:

1) Passenger request - This request is received by the taxi when a passenger requests for the taxi.
2) Passenger confirmation - This request is received by the taxi when the passenger back end chooses the taxi with minimum detour and total travel time.
3) Broadcast pheromone request - This request is received by the taxi when a taxi sends a broadcast pheromone request to other taxis.

The queue length was analyzed for both peak and non peak hours. The requests which are received by the taxi every 5 minutes is plotted in Figure 21 and Figure 22 during non peak hours for 2 taxis. During peak hours, the queue length is plotted for 2 taxis in Figure 23 and Figure 24.

It can be observed that during non peak hours the count of requests received by the taxi every 5 minutes is less. The plots are very sparsely distributed, indicating that the requests which are received by the taxi are also sparsely distributed. During peak hours, it can be observed that the count of requests received by taxi the taxi every 5 minutes is comparatively high. The plots for the incoming requests are densely distributed, indicating that the requests which are received by the taxi are very frequent.

The requests which are received by the taxi are dependent on the location of the taxi as well as the time that the request originates. As a taxi with passenger keeps moving towards his destination the taxi will be able to receive the requests on its way only. Hence the taxi's destination will be a factor in determining the requests which it will receive.

When there are multiple solutions on the pareto front , one solution needs to be selected. This is because when a passenger receives multiple solutions from the taxis to which it sent a request, then a passenger needs to select one solution in order to confirm the request. In a truly multi-objective context, a pareto front needs to be determined and a knee point in the pareto front is determined. This knee point solution corresponding to the respective taxi will be selected. The pareto front has been plotted for few passenger requests in the Figure 25. The graphs indicate all the taxi solutions (both detour as well as total travel time) obtained for a particular passenger request, the non-dominated (pareto) front and the knee point solutions as well.

The knee point was determined using the linear marginal utility function for passenger Id 1 and 2 in Figure 25. It can be observed for passenger Id 1, there were three taxis that returned their solution to the passenger and the passenger's backend had to choose one among the three solutions. All the

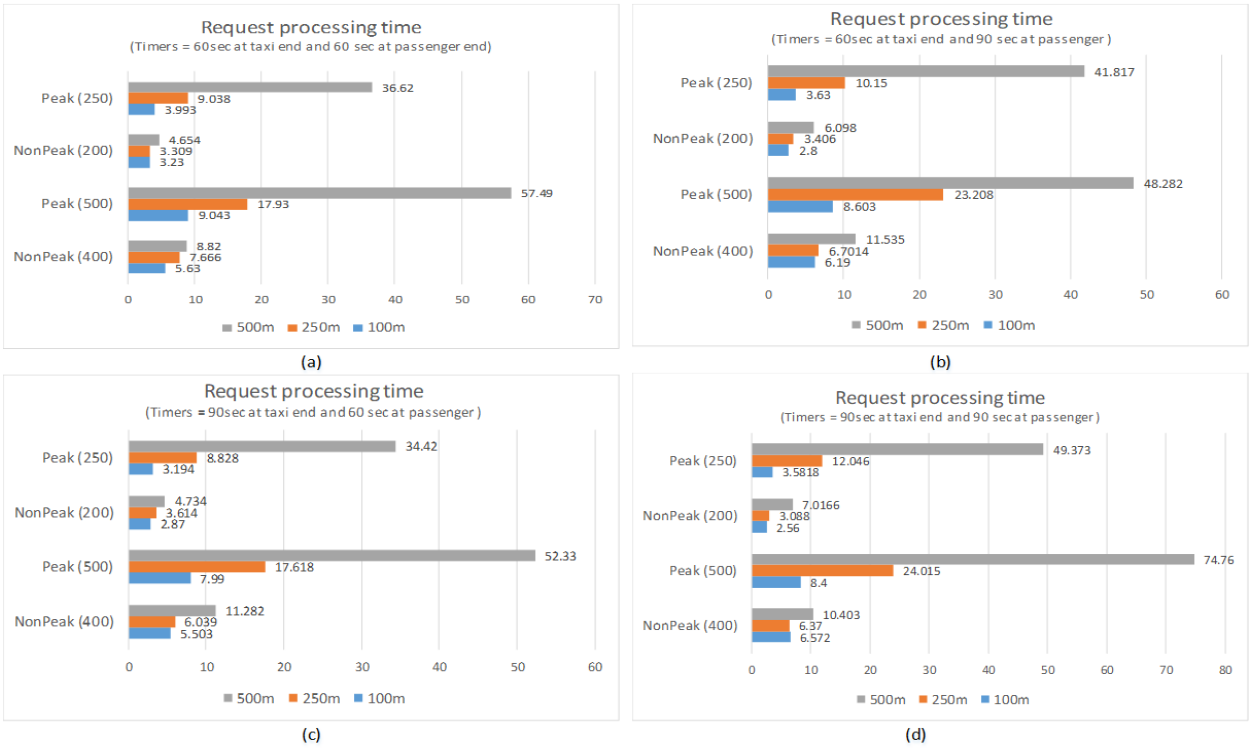
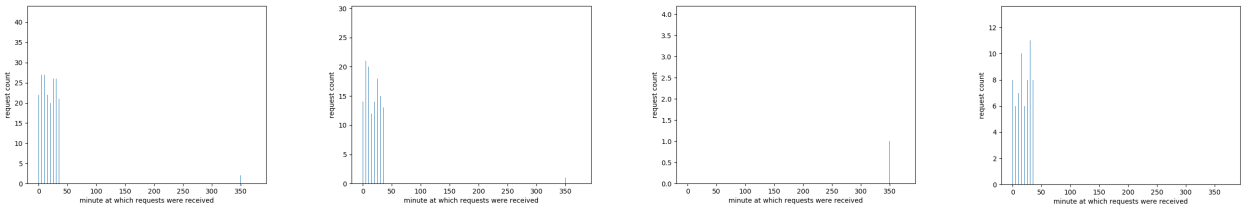
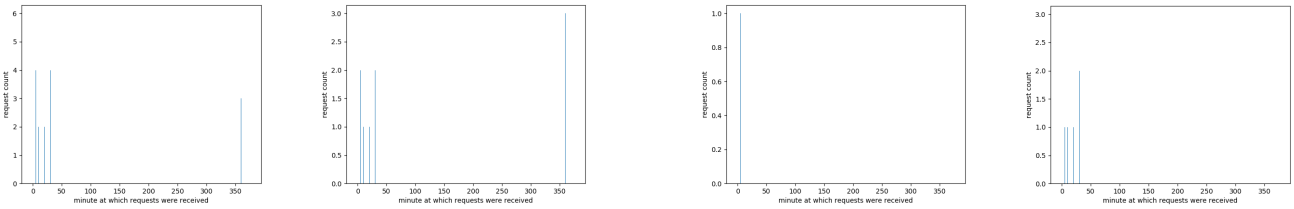


Fig. 20: (RPT_{rs}) with varying τ_p and τ_t



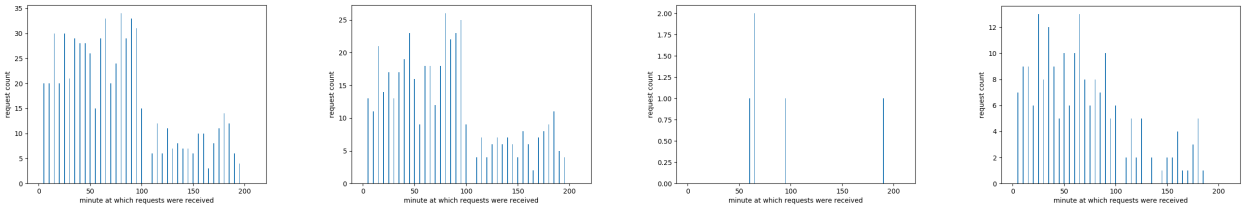
(a) All requests (b) Passenger requests (c) Passenger confirmation (d) Pheromone broadcast

Fig. 21: Queue length distribution for taxi Id 1 during non peak hours



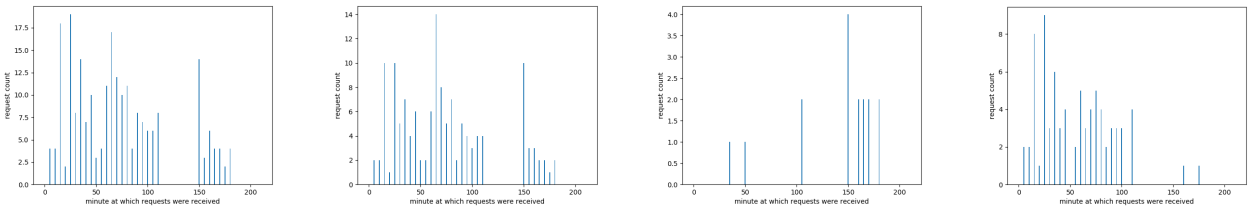
(a) All requests (b) Passenger requests (c) Passenger confirmation (d) Pheromone broadcast

Fig. 22: Queue length distribution for taxi Id 2 during non peak hours



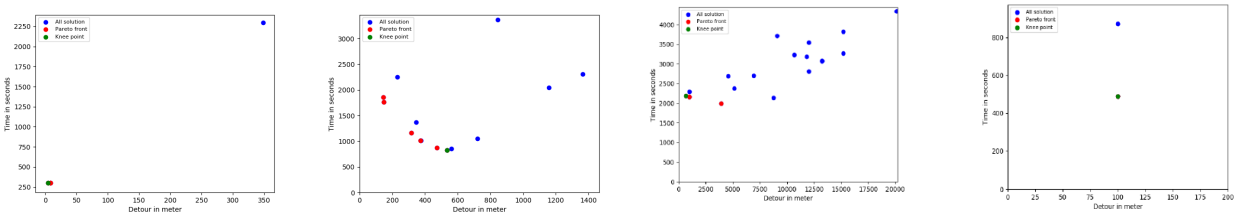
(a) All requests (b) Passenger requests (c) Passenger confirmation (d) Pheromone broadcast

Fig. 23: Queue length distribution for taxi Id 11 during peak hours



(a) All requests (b) Passenger requests (c) Passenger confirmation (d) Pheromone broadcast

Fig. 24: Queue length distribution for taxi Id 22 during peak hours



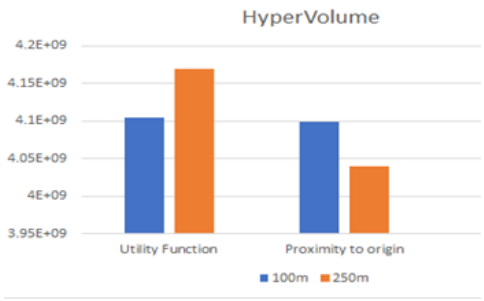
(a) Passenger Id 1 (b) Passenger Id 2 (c) Passenger Id 3 (d) Passenger Id 4

Fig. 25: Solution space for some passenger requests

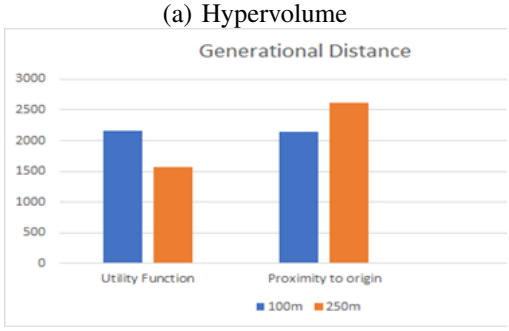
solutions returned by taxis are marked in blue. The red points indicate non dominated solutions and green point indicates knee point. By using the marginal utility function, the knee point was determined. It can be observed that the knee point obtained is much closer to the origin than the other solutions on the pareto front. For passenger Id 2, many taxis responded to the passenger request with their solutions comprising of detour and travel time. These solutions were filtered and the non-dominated solutions were obtained. Among the non-dominated solutions, the knee point was determined and the taxi which had obtained that solution was selected and a confirmation from the passenger end was sent to the taxi. Using the utility function is similar to doing a scalarization of the multi objective problems. Hence another approach of identifying the pareto front was done. Here, the proximity to line y =x was considered and the point (0,0) was chosen as the reference point. The reason for choosing y = x line is that it ensures to provide an equal trade-off between both the objectives and does not favour either of the objectives. The point (0,0) is an ideal point with detour 0 and travel time 0. Euclidean distance between the pareto front and origin was chosen to select the taxi to send response back to the passenger.

For passenger Id 3 and 4 in Figure 25, the knee point was determined by finding the point which was closer to origin. For passenger Id 3, many taxis sent back their solutions but only 3 solutions were on the non dominated front and in this one solution was selected for confirmation to the passenger. The solution selected was closer to the origin compared to other non-dominated solutions. For passenger Id 4, two taxis returned their solutions and between the two solution one of the solution was on the pareto front and this was considered the knee point and this was returned to the taxi as a confirmation.

Hypervolume and generational distance were measured and the results are plotted in Figure 26. In 2D, hypervolume measures the area between the pareto front and the reference point. Generational distance calculates how far an approximation set is from the Pareto optimal front PF (or reference set R). In general, greater the value of hypervolume indicator and lower the value of generational distance, better is the performance of the algorithm. From the Figure 26 , it can be seen that hypervolume and generational distance showed the best performance when knee point was obtained using marginal utility function for T_x = 100m.



(a) Hypervolume



(b) Generational Distance

Fig. 26: Hypervolume and Generational distance

VIII. Limitations and Future Scope

Some of the limitations of our strategy include Tx restriction i.e. if no taxi is present within the transmission range of a requesting passenger, the search will not return any taxi. As it is a distributed taxi system, if the timer at the passenger end expires before receiving any REPLY then also the ride sharing fails. In the current implementation of the taxi system, a TCP port is attached to each taxi. An alternate implementation is to consider each taxi as a thread for simulation or by using MPI programming model. Also, the proposed revenue model needs to be implemented to get a sense of commercial viability of the system.

IX. Acknowledgments

This work is partially supported by the Alexander von Humboldt Foundation postdoctoral fellowship and Miami University CFR Summer 2019 Research Award to Dr. Vaskar Raychoudhury. We thank Mr. Haoxiang Yu for his feedback and suggestions on the manuscript.

References

[1] Landscan geographic information science & technology. https://landscan.ornl.gov/.

[2] Sjtu wireless and sensor network lab - taxi trace data. http://wirelesslab.sjtu.edu.cn/taxi_trace_data.html.

[3] Build apps with here maps api and sdk platform access - here developer 2019. https://developer.here.com/, 2019.

[4] Niels Agatz, Alan Erera, Martin Savelsbergh, and Xing Wang. Optimization for dynamic ride-sharing: A review. *European Journal of Operational Research*, 223(2):295–303, 2012.

[5] Anmol Agrawal, Vaskar Raychoudhury, Divya Saxena, and Ajay D Kshemkalyani. Efficient taxi and passenger searching in smart city using distributed coordination. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1920–1927. IEEE, 2018.

[6] Ines Alaya, Christine Solnon, and Khaled Ghedira. Ant colony optimization for multi-objective optimization problems. In *Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on*, volume 1, pages 450–457. IEEE, 2007.

[7] Benjamín Barán and Matilde Schaerer. A multiobjective ant colony system for vehicle routing problem with time windows. In *Applied informatics*, pages 97–102, 2003.

[8] Kanika Bathla, Vaskar Raychoudhury, Divya Saxena, and Ajay D Kshemkalyani. Real-time distributed taxi ride sharing. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 2044–2051. IEEE, 2018.

[9] Xiaohui Bei and Shengyu Zhang. Algorithms for trip-vehicle assignment in ride-sharing. In *Proc. AAAI*, 2018.

[10] Alberto Colorni, Marco Dorigo, Vittorio Maniezzo, et al. Distributed optimization by ant colonies. In *Proceedings of the first European conference on artificial life*, volume 142, pages 134–142. Paris, France, 1991.

[11] NYC Open Data. Open data for all new yorkers. https://opendata.cityofnewyork.us/, 2017.

[12] LME Gambardella and G Taillard. A multiple ant colony system for vehicle routing problems with time windows.[in:] corne d., dorigo mf: New ideas in optimization, 1999.

[13] Leanna Garfield. Only 20% of americans will own a car in 15 years, new study finds. https://www.businessinsider.in/only-20-of-americans-will-own-a-car-in-15-years-new-study-finds/articleshow/58519992.cms, MAY 4, 2017.

[14] Boston Consulting Group. Press release : By 2030, 25% of miles driven in us could be in shared self-driving electric cars. https://www.bcg.com/d/press/10april2017-future-autonomous-electric-vehicles-151076, 2017.

[15] Sorin Ilie and Costin Bădică. Multi-agent approach to distributed ant colony optimization. *Science of Computer Programming*, 78(6):762–774, 2013.

[16] Uber Technologies Inc. Payments and earnings. https://www.uber.com/en-GH/drive/resources/payments/, 2019.

[17] Alexander Kleiner, Bernhard Nebel, and Vittorio A Ziparo. A mechanism for dynamic ride sharing based on parallel auctions. In *IJCAI*, volume 11, pages 266–272, 2011.

[18] Rachel Linnewiel. Uber dismantled by blockchain: Decentralized ride-hailing is coming. https://medium.com/davnetwork/uber-dismantled-by-blockchain-decentralized-ride-hailing-is-coming\-9c85d2bba6fa, September, 2018.

[19] Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 410–421. IEEE, 2013.

[20] Matthias Grossglauser Michal Piorkowski, Natasa Sarafijanovic-Djukic. Crawdad dataset epfl/mobility (v. 2009-02-24), traceset: cab. https://crawdad.org//epfl/mobility/20090224/cab/, February, 2009.

[21] Roberto Montemanni, Luca Maria Gambardella, Andrea Emilio Rizzoli, and Alberto V Donati. Ant colony system for a dynamic vehicle routing problem. *Journal of Combinatorial Optimization*, 10(4):327–343, 2005.

[22] Chicago Data Portal. Taxi trips. https://data.cityofchicago.org/Transportation/Taxi-Trips/wrvz-psew, August, 2017.

[23] Statista: The Statistics Portal. Monthly number of uber's active users worldwide from 2016 to 2018 (in millions). https://www.statista.com/statistics/833743/us-users-ride-sharing-services/, 2019.

[24] Wolfram Schneider. Bbbike extracts. https://extract.bbbike.org/, 2019.

[25] Chi-Chung Tao. Dynamic taxi-sharing service using intelligent transportation system technologies. In *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, pages 3209–3212. IEEE, 2007.