IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received September 5, 2018, accepted October 3, 2018, date of publication October 17, 2018, date of current version November 14, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2876595

An Optimal Ride Sharing Recommendation Framework for Carpooling Services

HAJRA QADIR[1], OSMAN KHALID[1], MUHAMMAD U. S. KHAN[1], (Member, IEEE), ATTA UR REHMAN KHAN[2], (Senior Member, IEEE), AND RAHEEL NAWAZ[3]

[1]Department of Computer Science, COMSATS University Islamabad, Abbottabad Campus, Abbottabad 22010, Pakistan
[2]Faculty of Computing and Information Technology, Sohar University, Sohar 311, Oman
[3]Department of Operations, Technology, Events and Hospitality Management, Business School, Manchester Metropolitan University, Manchester M15 6BH, U.K.

Corresponding author: Osman Khalid (osman@ciit.net.pk)

This work was supported by the Centre for Advanced Computational Science, Manchester Metropolitan University.

ABSTRACT Carpooling services allow drivers to share rides with other passengers. This helps in reducing the passengers' fares and time, as well as traffic congestion and increases the income for drivers. In recent years, several carpooling-based recommendation systems have been proposed. However, most of the existing systems do no effectively balance the conflicting objectives of drivers and passengers. We propose a highest aggregated score vehicular recommendation (HASVR) framework that recommends a vehicle with highest aggregated score to the requesting passenger. The aggregated score is based on parameters, namely: a) average time delay; b) vehicle's capacity; c) fare reduction; d) driving distance; and e) profit increment. We propose a heuristic that balances the incentives of both drivers and passengers keeping in consideration their constraints and the real-time traffic conditions. We evaluated HASVR with a real-world data set that contains GPS trace data of 61,136 taxicabs. Evaluation results confirm the effectiveness of HASVR compared with existing scheme in reducing the total mileage used to deliver all passengers, reducing the passengers' fare, increasing the profit of drivers, and increasing the percentage of satisfied ride requests.

INDEX TERMS Transportation system, vehicular recommendation, information filtering.

I. INTRODUCTION

With the continuous manufacturing of vehicles, the traffic congestion and air pollution have become two of the major challenges in urban cities of the world. According to a survey conducted in New York City, approximately 13000 taxis consume around 32 million gallons of gas per year with an average rate of 25 miles per gallon (MPG) [1]. This results in excessive amount of gas consumption every year, more than the annual gas utilization in smaller countries e.g., Central African Republic. Usually in urban cities passengers have to wait for a long time for taxicabs to become available. Passengers have to wait more than 30 minutes on average for a taxicab and average fare is more than 6 times of a public transport fare in urban cities [2].

The above mentioned issues have gained much research interest in the recent years leading to an emerging solution of carpooling services. Carpooling refers to a service where multiple passengers with similar schedules and itineraries share ride. The shared use of a vehicle by more than one passenger reduces the number of vehicles on roads, resulting in reduced fuel consumption, traffic congestion, and pollution. Carpooling improves vehicles' availability during rush hours or bad weather conditions, resulting in reduced waiting time for passengers.

The total fare is shared among all the passengers participating in a carpool. This reduces the fare for each individual passenger. The combined carpool fare is higher than the regular taxicab fare, resulting in increased profit for drivers [3]. Recently, the companies like Uber and Careem have also launched carpooling in recent years [4] [5]. The Uber's carpooling service is "Uber Pool" [4] whereas the Careem has launched a service named "Careem Sawa" [5] to allow passengers to share their rides. Carpooling has become a popular vehicular service that represents 10% of all commute trips in United States in 2009 [6].

Although carpooling service has solved the problems of passengers and drivers, yet at the same time it has also slightly increased the travel time and inconvenience of passengers. Currently, there are many important challenges in the implementation and adoption of carpooling services. There are three primary stakeholders involved in a dynamic carpooling system, namely: (a) existing passengers, (b) requesting (new)

passengers, and (c) drivers. Considering objectives of existing passengers while recommending a carpooling vehicle to a new passenger is an important issue that needs to be addressed by the existing carpooling services. Moreover, striking a balance between multiple incentives of all parties involved in carpooling while recommending a vehicle is a major challenge faced by most of the existing dynamic carpooling systems. This is because, the attempt to reduce the fare of passengers also decreases the profit of the drivers. Alternatively, if the objective is to increase profit of drivers, then passengers will have to pay more, as well as may get time penalty. The techniques to schedule the ride requests at real-time need to be improved in a way that can minimize the travel time of each individual passenger due to carpooling.

A significant amount of research work is being carried out to minimize the inconvenience of an individual passenger due to carpooling [7]. Call Cab [8] recommended taxicab on the basis of single passenger's objective; taxicab with minimum passenger's detour ratio without considering travel time and taxicab capacity constraints. Moreover, the aforementioned system did not consider the objectives of existing passengers. Real-time City-Scale Taxi Ridesharing [9] is able to handle real-time ride requests but it performs simple scheduling. The aforementioned system served each new ride request by dispatching a taxicab that satisfies it with minimum increase in taxi's scheduled travel distance. It did not consider Quality of Service (QoS) i.e., travel time and inconvenience of passengers in the chosen taxi while scheduling. Although the authors in [9] considered the objectives of existing passengers e.g., time and fare, yet at the time of selecting the taxi that can satisfy new ride request, the proposed algorithm selects the one with minimum increase in taxi's scheduled travel distance. Moreover, the system computed recommendations on the basis of driver's objective only. The monetary constraints proposed in [9] exhibit limitations in terms of fare reduction computation of existing and new passenger(s). The new passenger is compensated by a fixed amount in fare regardless of the increase in travel time. *coRide* [10] handled static ride requests in which both passengers and taxicabs are at the same pick up location. Although algorithms proposed by coRide [10] maintain balance between incentives of drivers and passengers, they are not able to handle dynamic (real-time) ride requests. Cloud-based public vehicle system [11] proposed multi-hop ridesharing system, in which a passenger can transfer between different vehicles and may be served by multiple vehicles. The algorithms proposed for the transfer problem in [11] calculate a transfer point for each ride request that can most reduce the traveling distance of vehicle. Thus, benefit of only drivers is taken into account while computing transfer points of passengers.

All of the above mentioned systems consider either the objectives of drivers or passengers while computing vehicle recommendations. To address the above mentioned challenges, we propose a dynamic and unified carpooling system that provides vehicle recommendations to real-time ride requests. The recommended vehicle in the proposed system can be occupied or vacant. We propose a heuristic based vehicle recommendation framework, *HASVR*, which considers multiple objectives of drivers and passengers rather than a single objective while generating vehicle recommendations. We summarize the contribution of this paper as follows:

- A recommendation frameworks is presented that combines multiple parameters of both drivers and passengers while computing recommendations. The parameters of passengers are (a) average time delay, (b) vehicle's capacity, and (c) fare reduction. The drivers' parameters are: (a) profit increment and (b) driving distance. The vehicle recommended by our proposed framework is an overall preferred vehicle that takes into account the objective of each party participating in dynamic carpooling i.e., existing passengers, new passenger, and driver.
- A heuristic based scheduling algorithm; variant of nearest neighbor *(NN)*algorithm [12] is used to schedule the ride requests.
- A policy is used to reduce fare of each passenger, whether existing or new. The fare of each passenger reduces in proportion to increase in travel distance. The extra charges to pick up a new passenger are included in the regular fare of new passenger to benefit drivers. Our recommendation framework improves the dynamic carpooling by maintaining balance among the incentives of all parties involved in carpooling.
- The experiments are conducted with real-world dataset from *T-Drive* trajectory data sample [13] and the results show effectiveness of *HASVR* as compared to vacant taxicab service.

The remainder of this paper is organized as follows. Section 2 reviews the related work The system architecture is described in Section 3. In Section 4, we present our proposed model for vehicle recommendation. Section 5 evaluates our system with a large-scale dataset. Finally, the conclusions and future work are reported in the last Section 6.

II. LITERATURE REVIEW

The problem of providing taxicab recommendations has attracted the attention of many researchers over the last decades. Recommender systems in taxicab industry play a significant role in helping passengers and drivers in a variety of ways. Yuan *et al.* [14] determined fast driving directions that are learned from the historical (GPS) trajectories of taxis. The proposed approach provided a taxi's driver with fastest route to a given destination at a given travel start time. Zhang *et al.* [15] proposed a system for taxicab drivers that finds optimal routes to pick up passengers with an objective to reduce drivers' cruising miles. The aforementioned systems were proposed only for non-sharing taxicabs in which capacity of taxicabs is usually not fully utilized. In addition, these systems considered the objectives of only drivers while our system considers the perspectives of both passengers and drivers.

Zhang *et al.* [16] introduced a novel parameter called detour ratio, which is defined as the ratio of a passenger's

detour distance (extra distance due to carpooling) and the distance of the passenger's direct route. The proposed system recommends either a vacant taxicab with zero detour ratio or an occupied taxicab with minimum detour ratio. Moreover, the detour ratio is calculated at real-time. Zhang *et al.* [8] extended the earlier work [16] by adding price mechanism. The proposed mechanism reduces fare of an individual passenger and increases profit of drivers at the same time. However, the proposed pricing model did not compensate the travel time delay of a passenger in the form of fare reduction. Orey *et al.* [17] proposed a distributed and dynamic taxi-sharing algorithm enhanced by wireless communications and distributed computing capabilities to perform coordination between customers' requests.

Setzke *et al.* [18] proposed and evaluated a dynamic ridesharing matching algorithm for crowd sourced delivery platforms that assigns items to potential drivers. The aforementioned algorithm automates and optimizes the assignment of transportation requests to drivers by matching them on the basis of transportation routes and time constraints. Cao *et al.* [19] proposed an efficient and scalable ridesharing service that allows riders to provide the maximum price they can pay for the service and the maximum time they can pay before being picked up. The aforementioned technique employs a cost model that estimates the cost of the ride sharing service for each driver. The proposed technique identifies those drivers that can satisfy the rider request within its cost limits and temporal constraints on the basis of cost model.

Agatz *et al.* [20] proposed a dynamic carpooling technique that considers matching drivers and riders on a short notice. The paper proposed optimization-based approaches with an objective to minimize the total system- wide vehicle miles incurred by system users, and their individual travel costs.

Ma *et al.* [3] proposed a system that takes real-time ride requests as an input and generates ridesharing schedules. These schedules reduce total travel distance of taxis. Ma *et al.* [9] extended the earlier work [3]. The proposed ridesharing system [9] introduced the mechanism of taking the agreement of existing passengers before generating a ridesharing schedule. The aforementioned technique added monetary constraints in the scheduling algorithm. However, the scheduling algorithm in the aforementioned systems do not consider the increase in travel time of existing passengers while scheduling a new ride request.

Zhu *et al.* [11] proposed a cloud-based public vehicle system where passengers can transfer among public vehicles according to the scheduling decisions made using cloud computing. The aim of aforementioned approach is to reduce the travel distance of all public vehicles with service assurance for passengers e.g., transfer time and detour ratio. Although the proposed approach improves the ride availability but lacks pricing mechanism i.e., how to charge passengers when they are transferred among vehicles. Zhang *et al*. [10] calculated cost-efficient carpool routes for taxicab drivers in response to delivery requests of passengers. This reduces the total mileage of taxicabs. The proposed approach provides the concept of delivery graph, which indicates the schedule of a carpool route to deliver passengers. The pricing model compensates increase in travel time in the form of fare reduction for the individual passenger. However, the proposed algorithm [10] can only handle the static ride requests in which both passengers and taxicabs are present at the same pick-up location. Moreover, the proposed technique is not able to handle the new requests (requests that arrive 0.25 or 1 hour before the delivery start time) efficiently.

Moreover, the carpooling services [8], [9] consider either the objectives of passengers or drivers while finding an optimal vehicle to serve the requesting passenger. In addition, these carpooling systems do not consider the objectives of existing passengers while computing vehicle recommendations.

Huang [21] proposed a branch and bound algorithm, a mixed-integer-programing algorithm, and an optimized kinetic tree algorithm to dynamically match real-time ride requests to servers vehicles) in a road network to allow ridesharing. The goal of before mentioned approach is to schedule requests in real-time and minimize the servers' travel times i.e., objective of only drivers is considered.

Asghari *et al.* [22] proposed a fair pricing model with an objective to satisfy both the constraints of drivers and riders simultaneously. The aforementioned model is an auction-based framework where each driver automatically bids on every nearby request by considering a number of parameters such as both the driver's and the riders' profiles, their destinations, the pricing model, and the current number of riders in the vehicle. The server determines the driver that generates highest profit and assigns the rider to that driver. However, the proposed model considers only objectives of only drivers and platform owner while assigning the request to a driver.

To address these limitation, our proposed heuristics based recommendation framework, *HASVR,*presents a solution for maintaining a balance among the objectives of all the parties involved in carpooling systems while finding an optimal vehicle to serve the new request.

III. SYSTEM ARCHITECTURE

As shown in Fig. 1, we consider a city road network map and divide it into two components: (a) the pick and drop locations of passengers and (b) the routes among those locations. It is possible to have multiple routes between two locations. We estimate travel time and driving distance between any two locations using Google maps API [23]. The API computes distance and time between any two locations by also considering the distances of in-between road segments. Fig. 2 shows the major components of our proposed architecture.

The following are the major components of our proposed system architecture.

A. INPUT MODULE

The input module has two major components, namely ride requests and vehicles' current state.

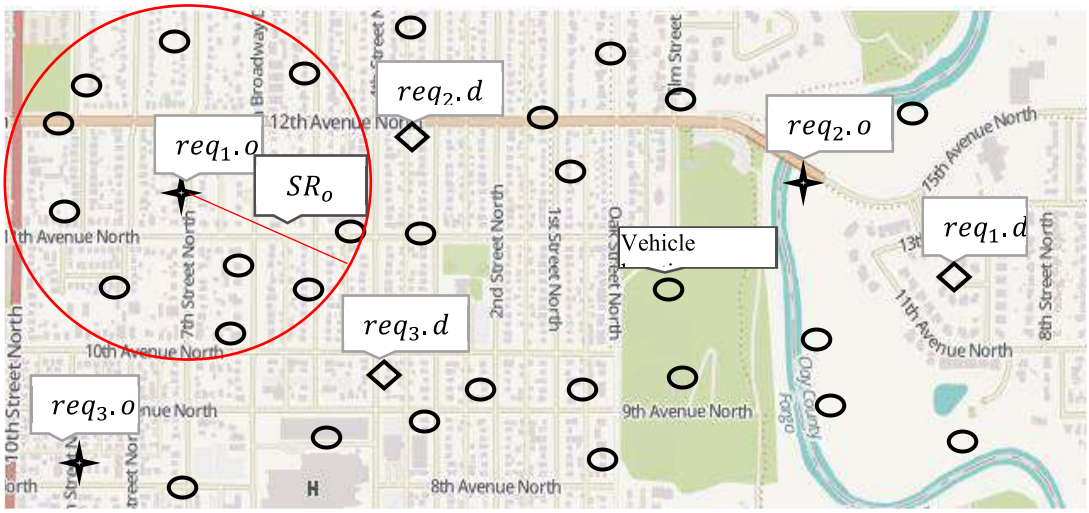


FIGURE 1. Road Network. Here *req* represents request, *o* represents origin of a request, and *d* represents destination of a request.

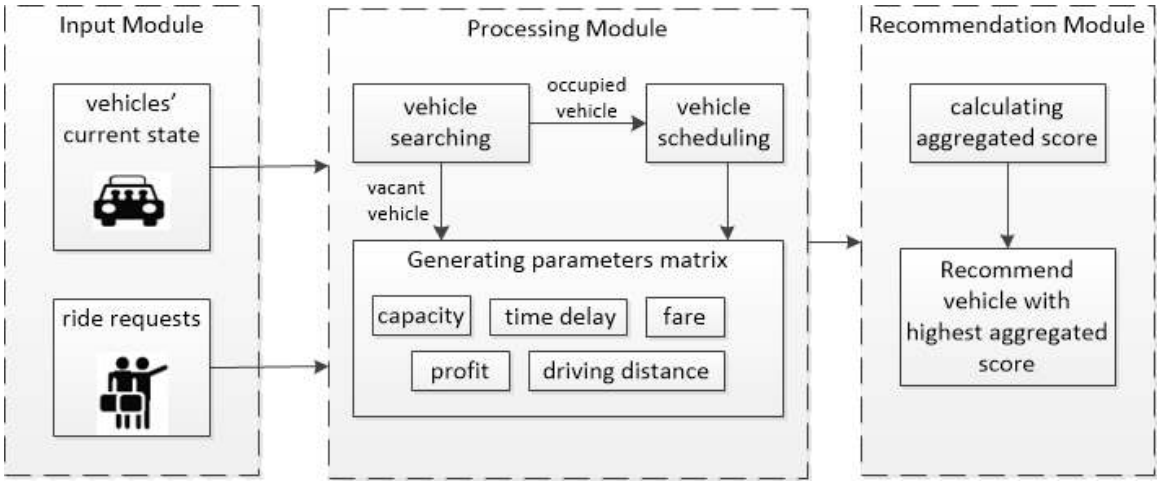


FIGURE 2. System architecture.

1) RIDE REQUEST

As shown in Fig. 2 (bottom left), a passenger sends a ride request *req* to the recommendation system that contains the following attributes:

- *req.t* : Time when *req* is submitted.
- *req.o* : Pick-up location of *req*.
- *req.d* : Destination of *req*.
- *req.ept* : Earliest pick-up time of *req* representing earliest possible time when passenger wants to be picked up.
- *req.edt* : Earliest drop-time of *req* representing earliest possible time when passenger can be dropped off.
- *req.sdt* : Scheduled drop-time indicating latest possible time when passenger can be dropped off according to a generated schedule of a vehicle.

We have considered two types of requests in our model, namely (a) new ride request req_{new}, that arrives at current time t_{cur} and (b) existing ride requests req_e that are already assigned to an occupied vehicle. We further assume that two ride requests cannot arrive at exactly the same time period as still there will be a difference of microseconds or nanoseconds in the two requests. However, in extreme case if two requests arrive at exactly the same time, then the requests will be sorted and prioritized using the First in First Out (FIFO) mechanism, based on the request ID associated with each request. However, still if we strictly consider that two or more requests arrive at exactly the same time, then sorting them in different orders may lead to different results. Moreover, we have used the indexing on various table fields, including, cab id, latitude, and longitude in database to speed up the query processing time. To reduce the complexity of the simulator, we further assume that if a request cannot be satisfied due to the computed cab rankings below threshold, the request will be discarded, and not shifted to the next time stamp. A new passenger only submits request time, origin,

and destination. For simplicity, we consider $req_{new}.ept$ to be equal to $req_{new}.t$. The $req_{new}.edt$ can be calculated using the following formula, where T indicates travel time of the fastest route from $req_{new}.o$ to $req_{new}.d$.

$$req_{new}.edt = req_{new}.ept + T\,(req_{new}.o, req_{new}.d)\,. \quad (1)$$

2) VEHICLE STATE

A vehicle state s_v denotes the current status of the vehicle v at time t as shown in Fig. 2 (top left). It contains the following fields.

- $v.ID$: Identification number of vehicle.
- $v.t$: Time stamp associated with the current state of vehicle.
- $v.l$: Geographical location (longitude, latitude) of vehicle associated with $v.t$.
- $v.ep$: Number of existing passengers in vehicle at $v.t$.
- $v.c$: Current seat capacity of vehicle at $v.t$

$v.ED$: Set of destinations of existing passengers in vehicle, such that $v.ED = \{req_1.d, \ldots, req_{n-1}.d\}$, where $n-1 = ep$.

B. PROCESSING MODULE

Ride requests and vehicles' current states are imported to the processing module as shown in Fig. 2 (middle) that further comprises of three stages that need to be executed in the following sequence: vehicle searching, vehicle scheduling, and generating parameters matrix. The current location of a vehicle v at arrival time of new request is represented as $v.l$.

1) VEHICLE SEARCHING

This is the first step of processing module which extracts a nearby vehicle set V^N where vehicles lie within searching radius SR_o around the request origin. Fig. 1 shows a road network of a city where three ride requests have been generated at time t. The locations of vehicles, origin and destination of ride requests are randomly distributed over the network as shown in Fig. 1. For instance, the SR_o (represented by a circle in Fig. 1) around origin of req_1 contains ten vehicles. While finding the nearby vehicles around request origin, it is also important to consider the seat capacity of vehicle to check whether or not the vehicle can accommodate requesting passenger. Therefore, we represent nearby vehicle set by V_c^N. The current seat capacity of a vehicle can be calculated using following equation:

$$v.c = capacity_{max} - v.ep, \quad \forall v \in V_c^N. \quad (2)$$

Where $capacity_{max}$ is maximum seat capacity of a vehicle.

2) VEHICLE SCHEDULING

After identifying the set of candidate vehicles i.e.,V_c^N for arrived request req_{new}, the scheduling module next calculates the schedules of occupied vehicles in the set. The calculation of shared route for each occupied vehicle in nearby vehicle set is modelled as travelling salesman problem (TSP) [24] that states: *"Given a list of cities and distances between each pair of cities, what is the shortest possible route that visits each city exactly once"*. It is an NP-hard problem and finding an optimal solution results in long running time. Thus, a heuristic algorithm should be used to calculate the shared route within reasonable time.

If the vehicle $v \in V_c^N$ is occupied, then a variant of nearest neighbor *(NN)* heuristic algorithm [12] is used to schedule all ride requests (new and existing). Nearest neighbor procedure is one of the commonly used heuristics of travelling salesman problem (*TSP*). NN algorithm builds a trip on the basis of travelling distance from the currently visited node to the closest node in the network. However, the heuristic produces an approximately optimal solution from the distance matrix.

3) GENERATING PARAMETERS MATRIX

Given the set V_c^N (each vehicle in the set has its own current state) retrieved for new ride request req_{new}, the purpose of generating parameters matrix is to find vehicle $v \in V_c^N$ that is preferred from the perspective of driver, requesting passenger, and existing passengers. We have considered the following five parameters of drivers and passengers in our methodology, namely: (a) average time delay of passengers, (b) vehicle capacity, (c) fare reduction of passengers, (d) total driving distance, and (e) profit increment to driver. If vehicle is occupied, then it needs to be scheduled before calculation of aforementioned parameters. All parameters except vehicle capacity are calculated on the basis of *NN* schedule.

C. RECOMMENDATION MODULE

The last module of our system architecture as shown in Fig. 2 (right most) is the recommendation module where each req_{new} that arrives at current time t_{cur} is recommended a preferred vehicle from the set V_c^N. The vehicle needs to be ranked with respect to *average time delay, vehicle capacity, fare reduction, driving distance, and profit increment*. Therefore, we need to sum all the aforementioned parameters to calculate an aggregated score.

IV. PROPOSED VEHICLE RECOMMENDATION FRAMEWORK

In this section, we discuss in detail the proposed heuristic based vehicle recommendation framework *HASVR*.The framework has two major modules: (a) scheduling module for occupied vehicles and (b) calculation of aggregated score for each vehicle.

A. SCHEDULING MODULE

The scheduling algorithm produces a temporally-ordered sequences of locations that an occupied vehicle $v \in V_c^N$ will visit when req_{new} is assigned to it. The locations include origin of new passenger and destinations of all (new and existing) passengers. The schedule indicates the order of serving the ride requests if the requesting passenger is assigned to an occupied vehicle $v \in V_c^N$.

Algorithm 1 illustrates the procedure of nearest neighbor scheduling. The algorithm takes as input the following parameters: (a) currently arrived ride request req_{new} that

Algorithm 1 *NN* Scheduling

Input: Ride request arrived at $\boldsymbol{t_{cur}}$:$\boldsymbol{req_{new}}$, state of vehicle $\boldsymbol{v}$ at $\boldsymbol{t_{cur}}$: $\boldsymbol{s_v}$
Output: Schedule of $\boldsymbol{v}$ if $\boldsymbol{req_{new}}$ is assigned to $\boldsymbol{v}$: $\boldsymbol{S}$
Definitions: $\boldsymbol{t_{cur}}$ = *current time*, $\boldsymbol{S}$= *list to store visited locations,* $\boldsymbol{U_{vs}}$ = *list to store unvisited locations, DT = list to store travel time* $\boldsymbol{T}$ *and distance* $\boldsymbol{d}$ *from one location to another,* ***select*** = *list to store DT along with each unvisited location*

1. $S \leftarrow \{\}$; $U_{vs} \leftarrow \{\}$; $DT = (T, d)$
2. $S\,[first]\,.loc_v \leftarrow v.l$; $S\,[first]\,.d \leftarrow 0$; $S\,[first]\,.ETA \leftarrow t_{cur}$
3. $U_{vs} \leftarrow U_{vs}.append\ (v..ED, , req_{new}..o)$
4. while $length(U_{vs}) \geq 0$ do
5. $l_s \leftarrow length(S)$
6. $Select \leftarrow \{\}$ // Initialize an empty list *select*
7. if *search* $(S, req_{new}.o)$ returns *TRUE* then //search $req_{new}.o$ in S and if found then
8. $P \leftarrow GetPosition\ (S, , req_{new}..o)$ // find position of $req_{new}.o$in$S$
9. if P equals l_s then
10. $U_{vs} \leftarrow U_{vs}.append\ (req_{new}..d)$
11. end if
12. end if
13. if U_{vs} gets $\{\}$ then
14. break
15. end if
16. $vs_{curr} \leftarrow S\,[l_s]\,.loc_v$
17. for each *unvisitedlocation* $loc_{uv} \in U_{vs}$ do
18. $DT(T, d) = Gettraveltime\&distance(vs_{curr}, loc_{uv})$ // from vs_{curr} to loc_{uv}
19. $select \leftarrow select.append(list\ (loc_{uv}, , DT)\ )$
20. end for
21. $DT_{min} \leftarrow DT$ from *select* with *MIN* distance
22. $location_{near} \leftarrow loc_{uv}$ from *select* such that $DT = DT_{min}$
23. $distance_{near} \leftarrow DT_{min}.d$
24. $ETA_{near} \leftarrow S[l_s].ETA + DT_{min}.T$
25. $S \leftarrow S.append\ (list(location_{near}, distance_{near}, ETA_{near})\ )$
26. $U_{vs} \leftarrow U_{vs}.remove\ (location_{near})$
27. end while
28. return S

wants recommendation of preferred vehicle and (b) current state of vehicle s_v from which location of vehicle at t_{cur}, and set of destinations of existing passengers can be extracted. The term ''visited'' and ''unvisited'' is used to differentiate whether the location has been inserted into the schedule (termed as ''visited'') or not (termed as ''unvisited''). Following text explains the steps involved in the algorithm.

- *Initializations (Line 1−Line 3):* In *Line 1*, various lists (data structures) used by algorithm are declared. S consists of a list where each index stores three elements, (i) visited location loc_v, (ii) estimated time of arrival ETA at the visited location according to the created schedule (also referred as *scheduled arrival time*), and (iii) distance travelled d to the location from current (last or previous) visited location. The distance and travel time are computed by using Google maps distance matrix API [23]. The current location of vehicle $v.l$ at t_{cur} is the starting point from where *NN* schedule is created. Therefore, $v.l$ is first location to be inserted in S. Distance travelled to $v.l$ is 0 and estimated time of arrival at $v.l$ is equal to request time (*Line 2*). Origin of requesting passenger and destinations of existing passengers are appended to U_{vs}(*Line 3*). Destination of requesting passenger is appended to U_{vs} after visiting corresponding origin.
- *Schedule Creation (Line 4- Line 28):* On each iteration of while loop, nearest location from current visited location vs_{curr} is inserted into S. While loop continues to insert locations into S unless each location in U_{vs} gets visited (inserted into S)*(Line 4).*Both lists S and U_{vs} are updated on each iteration of while loop. A location is added to S and removed from U_{vs} on each while loop iteration. Length of S is calculated to refer the current visited location in S *(Line 5).*A list *select* is initialized as empty on each iteration of while loop *(Line 6).*This list stores each unvisited $loc_{uv} \in U_{vs}$ along with travel time and distance from vs_{curr} to loc_{uv}.
- *Precedence rule of origin and destination (Line 7-Line 12):* The origin of new request must be inserted before the destination in the schedule. The algorithm first checks whether the origin has been visited or not (*Line 7*). If origin has been visited, then

algorithm checks whether it is the last visited location in S or not i.e., finds position of new request's origin in S (*Line 8).*If origin is found to be last visited location in S, then destination of new request is appended to list of unvisited locations U_{vs} (*Line 9-Line 12).* The destination is included in the decision "*where to go next*" only if the corresponding origin has been visited. If each unvisited location in U_{vs} has been inserted into S (U_{vs} gets empty), then control goes out of the *while* loop and algorithm terminates returning schedule S *(Line 13-Line15, and Line 28).*

- *Finding "Nearest" location (Line 16- Line 27):* In this step, spatial closeness (distance) from the current visited location to each unvisited location $loc_{uv} \in U_{vs}$ is measured. The last location inserted into S is the currently visited location *(Line 16).* The unvisited location loc_{uv} that has highest spatial closeness to vs_{curr} is scheduled to be next location to visit *(Line 17 –Line 22).* In *line 24,* the estimated arrival time at the nearest location $location_{near}$ is calculated by adding the estimated arrival time at previous visited location and travel time from vs_{curr} to $location_{near}$. The nearest location along with its spatial closeness from vs_{curr} and estimated arrival time is appended as a list to S *(Line 25).* The nearest location needs to be removed from U_{vs} after its insertion into S *(Line 26).*

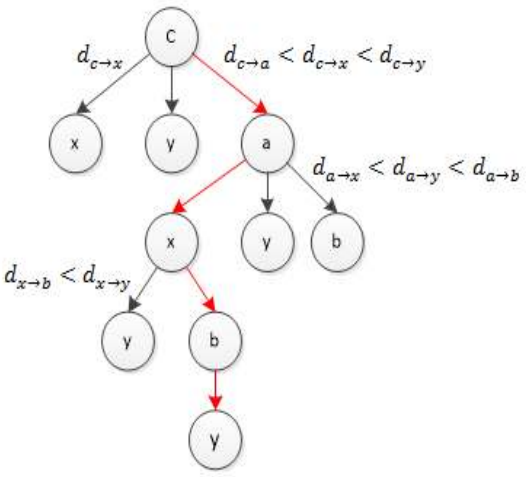


FIGURE 3. Constructing *NN* graph for three ride requests.

Fig. 3 gives an example of how to create a *NN* schedule serving three ride requests (one new and two existing) from the vehicle current location c. The origin and destination nodes of new request are a and b, respectively. The destinations nodes of two existing passengers are x and y. A weight on an edge (e.g., $d_{c\to x}$) indicates real-world mileage of the fastest route from node c to node x. Therefore, *NN* route will be $c \to a \to x \to b \to y$.

B. AGGREGATE SCORE CALCULATION

For each vehicle $v \in V_c^N$, our proposed model calculates the values of various parameters as discussed previously. The recommendation module combines all the calculated parameters to calculate an aggregate score of each vehicle $v \in V_c^N$. This aggregated score is used as a rating scale that is generated by the recommender system by considering multiple objectives of passengers (new and existing) and drivers. Each parameter is normalized before aggregating so that each parameter's value is proportional to its original value. In this section, various equations to calculate the parameters will be discussed.

The average time delay, capacity, fare reduction, driving distance, profit increment, and aggregated score AS of a vehicle $v \in V_c^N$, is represented by $v.td_{avg}$, $v.c$, $v.\sum_{i=1}^{n} \Delta fare_i$, $v.D_{Total}$, $v.\Delta Profit$, and $v.AS$ respectively.The aggregated score of a vehicle $v.AS$ is calculated by (3). Finally, our model recommends the vehicle with highest AS to the requesting passenger.

$$v.AS = \frac{1}{v.td_{avg}} + v.c + v.\sum_{i=1}^{n} \Delta fare_i, + \frac{1}{v.D_{Total}} + v.\Delta Profit \quad , \forall v \in V_c^N. \quad (3)$$

Where $n = v.ep + 1$. Next, we explain various equations to calculate the parameters.

1) VEHICLE CAPACITY c

It represents the number of seats available in $v \in V_c^N$ at current time t_{cur}. The current capacity of vehicle at t_{cur} is simply calculated by (2).

2) TOTAL DRIVING DISTANCE D_{Total}

It indicates the total distance of the route that the vehicle v (vacant or occupied) will follow if req_{new} is assigned to it.

a) If vehicle is vacant, then it will follow a direct route to serve the new passenger. The direct route is represented as follows.

$$Route_{Direct} = v.l \to req_{new}.o \to req_{new}.d. \quad (4)$$

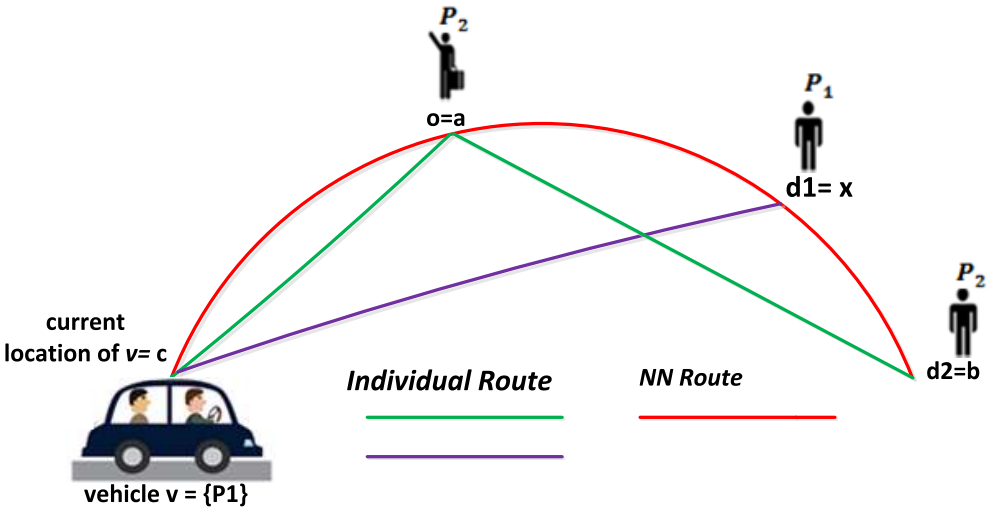
The total distance can be calculated by using the following equation.

$$D_{Total} = d_{v.l\to req_{new}.o} + d_{req_{new}.o\to req_{new}.d}. \quad (5)$$

Where $d_{v.l\to req.o}$ represents the pick-up distance d_{pick} and $d_{req_{new}.o\to req_{new}.d}$ is the direct distance between origin and destination d_{direct}.

b) In case of an occupied vehicle, D_{Total} is the distance of nearest neighbor route (also referred as D_{NN}). The vehicle will follow the route of the *NN* schedule to serve new and existing passengers. First location is always the vehicle's current location in the created route. We can represent *NN* route as a sequence defined as follows.

$$Route_{NN} = (loc_i)_{i=1}^{L}. \quad (6)$$



$$Total\ Distance\ Reduced = \Delta D_T$$
$$d_{c\to a} = 3; d_{a\to b} = 5.5; d_{c\to x} = 6,$$
$$D_R = 3 + 5.5 + 6 = 14.5;\ D_{NN} = 11.5$$
$$\Delta D_T = D_R - D_{NN} = 14.5 - 11.5 = 3 \geq 0$$

FIGURE 4. Reduced total distance due to carpooling.

In (6) loc_i represents *ith* location in the created route and L is number of locations in the route. D_{Total} is simply calculated as follows.

$$D_{Total} = D_{NN} = \sum_{i=1}^{L-1} d_{loc_i \to loc_{i+1}}. \tag{7}$$

In our proposed model, we define a matching criteria to find whether the requesting passenger P_{new} is able to share an occupied vehicle or not. We define a regular distance D_R to be sum of distances of individual route of each passenger. The individual route of requesting passenger is from vehicle's current location at $req.t$ to requesting passenger's origin and then directly from origin to destination. For an existing passenger, the individual route is direct measurement from vehicle current location to existing passenger's destination. The matching criteria is measured by the total distance reduced by ridesharing ΔD_T, defined as difference between regular distance D_R and D_{NN}. An occupied vehicle $v \in V_c^N$ can be a candidate for a preferred vehicle if and only if it satisfies the following distance constraint.

$$\Delta D_T = D_R - D_{NN} \geq 0. \tag{8}$$

If ΔD_T in (8) is greater, then this indicates that the vehicle's current location, origin and destinations in *NN* schedule lie near to each other, leading to a greater reduction in total distance. This implies that the new request matches to the destinations of existing passengers in $v \in V_c^N$ and assigning the request to v will provide benefit to both drivers and passengers (new and existing). However, if ΔD_T negative, then the vehicle's current location, origin and destinations in *NN* schedule lie far apart from each other such that sum of distances of individual routes of passengers is less than *NN* distance. This indicates that new request is not able to carpool existing passengers in $v \in V_c^N$. Therefore, the nearby occupied vehicle v needs to be removed from nearby vehicle set V_c^N.

Fig. 4 shows an example where at current time t_{curr}, a new request arrives and a nearby vehicle v is occupied with passenger P_1. The new passenger is represented as P_2. The origin of new passenger is represented by o whereas destinations of P_1 and P_2 are represented by $d1$ and $d2$, respectively. The distance of *NN* schedule is 11.5 units whereas the sum of the individual distances is 14.5 units. The vehicle v satisfies (8) with reduction of total distance of 3 units.

3) AVERAGE TIME DELAY TD_{AVG}

Time delay is the difference of earliest drop time and scheduled drop time of a passenger. The average time delay gives an idea about delay that each passenger will tolerate on average if the requesting passenger is assigned to a vehicle $v \in V_c^N$. Average time delay can be computed by using the following formula.

$$td_{avg} = \frac{1}{n} \sum_{i=1}^{n} (req_i.sdt - req_i.edt). \tag{9}$$

Where $n = v.ep + 1$. We have assumed 1 passenger per ride request in our model. Earliest drop time of new request is calculated by using (1). Scheduled drop time of a passenger

is simply the estimated arrival time at the corresponding destination according to *NN* schedule. The waiting time of a requesting passenger is decided from the time when the passenger submits request till the time when the vehicle reaches the pick-up location. The travel time delay is the delay incurred to passenger due to traveling detour distance (as compared to direct distance). The calculation of time delay also includes the waiting time of a new passenger (also referred as pick-up delay).

For existing passengers, our model estimates that how much delay they have to tolerate as compared to direct travel time from vehicle current location $v.l$ at $req.t$ to their corresponding destinations. Obviously, pick-up delay of existing passengers is 0. If vehicle is vacant, then travel time delay is equal to 0 (only pick-up delay is incurred to requesting passenger).

$$td_{avg} = delay_{pick} = T\left(v.latreq.t, req_{new}.o\right). \quad (10)$$

4) PRICE MECHANISM (FARE REDUCTION AND PROFIT INCREMENT)

We have proposed a variant of win-win fare model [10]. In order to motivate drivers and passengers to participate in carpooling, the pricing mechanism is designed to provide monetary incentives for all involved parties. Since the time delay caused by detouring is the major concern of carpooling systems, it is important to design the pricing mechanism while considering the detouring of each passenger participating in carpooling. The pricing scheme is used to calculate total fare reduction and profit increment score of each vehicle $v \in V_c^N$. The fare reduction score estimates that how much fare of new passenger as well as existing passengers in vehicle v can be reduced if req_{new} is assigned to v. Our proposed pricing scheme works as follows.

- The passenger pays the regular fare RF while travelling alone. The regular fare is the fare proportional to the distance travelled by vehicle to serve the passenger alone. Regular fare rate ∂_r is a constant price for unit distance. The regular fare RF corresponding to travelled distance d is represented as $rf(d) = \partial_r \times d$. The value of ∂_r can be decided by the vehicle service company.
- The RF of a requesting passenger also includes the pick-up charge in our model. Pick-up charge is the fare for the distance that vehicle travels to pick the passenger from its previous location in the created route. This strategy helps to avoid loss to drivers.
- The passenger P whose travel distance is increased due to sharing the ride should be compensated in the form of fare reduction and the reduction should be proportional to increase in travel distance of P.
- The driver's profit is the sum of fares estimated for all passengers. It is important to mention that total fare paid by all passengers equals the profit collected by driver in our model. The expected profit of a driver $Profit_{exp}$ is the regular fare corresponding to the total distance travelled by vehicle to serve all passengers. The increase in driver's profit indicates that how much extra profit the driver can earn as compared to expected profit. The increase in profit can be calculated as follows.

$$\Delta\text{Profit} = \text{collected profit} - \text{Profit}_{\text{exp}}. \quad (11)$$

$$\text{Collected profit} = \sum_{\text{i}=1}^{\text{n}} \text{EF}_{\text{i}} \text{where, n} = \text{v.ep} + 1. \quad (12)$$

$$\text{Profit}_{\text{exp}} = \text{rf}\left(\text{D}_{\text{Total}}\right). \quad (13)$$

In (12) EF_i represents the fare estimated for request i. The fare constraint can be represented by (14); representing regular fare of request i by RF_i.

$$\text{EF}_{\text{i}} \leq \text{RF}_{\text{i}} \quad \forall \text{i} = 1, \ldots, \text{n}. \quad (14)$$

Each nearby vehicle $v \in V_c^N$ can be vacant or occupied at arrival time of new request. Therefore, we consider two cases in our proposed mechanism.

a) Vacant vehicle : Vacant vehicle v will follow a direct route if new request is assigned to it. The requesting passenger is regularly charged. Fare of requesting passenger is estimated by (15).

$$\begin{aligned} \text{EF}_{\text{new}} &= \text{RF}_{\text{new}} = \text{rf}(\text{d}_{\text{loc}_{\text{i}-1} \rightarrow \text{loc}_{\text{i}}} + \text{d}_{\text{loc}_{\text{i}} \rightarrow \text{req}_{\text{new}}.\text{d}}), \\ \text{loc}_{\text{i}} &= \text{req}_{\text{new}}.\text{o}. \end{aligned} \quad (15)$$

Where loc_i represents ith location in the created route. There will be no reduction in fare as vehicle will follow a direct route to serve the passenger so total fare reduction associated with passenger of a vacant vehicle is 0. Profit increment score of a vacant vehicle is also 0 as the driver's expected and collected profit is same.

b) Occupied vehicle: The total saving due to carpooling is shared among all the involved parties (new passenger, existing passengers, and driver). Total carpool saving CS is defined by (16).

$$\text{CS} = \text{rf}\left(\text{D}_{\text{R}} - \text{D}_{\text{NN}}\right), \quad \text{where } \text{D}_{\text{R}} \geq \text{D}_{\text{NN}}. \quad (16)$$

To optimize the objectives of drivers and passengers, we need to share the carpool saving between driver and all passengers as a party. The percentage of carpool saving given to party of passengers is then shared between new passenger and existing passengers of vehicle on the basis of their detour distance (increase in travel distance) in NN route. We represent new ride request by req_n and existing ride requests by $req_1, \ldots, req_{n-1}$.

To calculate the fare reduction score of a vehicle v, our model first calculates regular fare and then estimates fare of each passenger P on the basis of Rf, CS and $P^{'}s$ detour distance. The Rf of new passenger is calculated by using (15). However, the regular fare of an existing passenger is calculated using (17).

$$\text{RF}_{\text{i}} = \text{rf}\left(\text{d}_{\text{v.l} \rightarrow \text{req}_{\text{i}}.\text{d}}\right) \quad \forall \text{i} = 1, \ldots, \text{n} - 1. \quad (17)$$

The fare of each passenger in scenario of same origin and same destinations (same origin means when the request origin

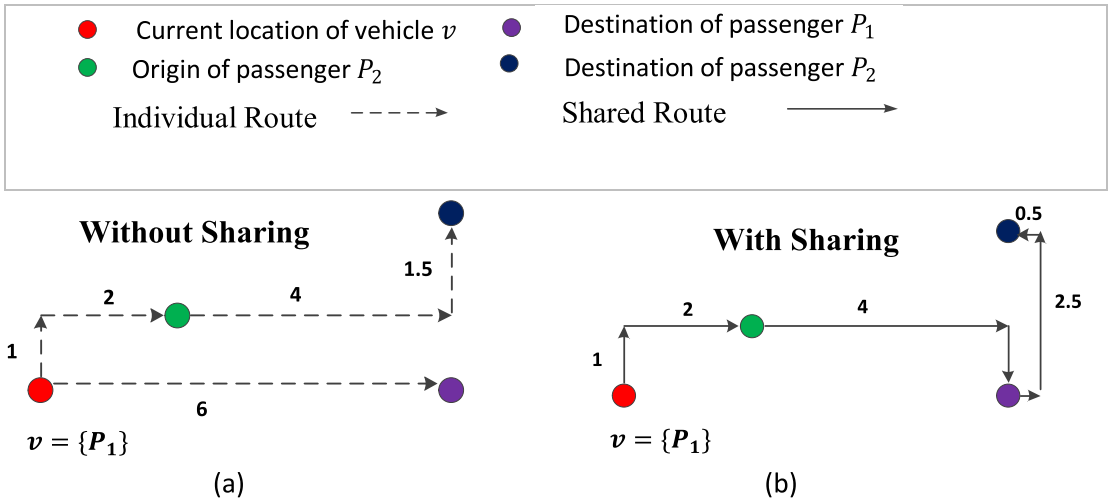


FIGURE 5. Example of price strategy.

is in the way of cab) is estimated by the following equation.

$$EF_i = RF_i - \mu \times CS \times \frac{1}{n}. \tag{18}$$

In this case, detour distance is 0 as all passengers have common origin and common destination. This scenario happens when the new request's origin matches to vehicle's current location and destination of new request matches to common destination of all existing passengers. The parameter μ in (18) and (19) indicates the percentage of total carpool saving given to passengers group. The percentage of carpool saving given to passengers is then shared equally between new passenger and existing passengers. Whereas the fare of each passenger in case of different origins and different destinations is estimated by the following equation.

$$\mathrm{EF_i} = \mathrm{RF_i} - \mu \times \mathrm{CS} \times \frac{\Delta \mathrm{d_i}}{\sum_{\mathrm{i=1}}^{\mathrm{n}} \Delta \mathrm{d_i}}. \tag{19}$$

Where $n = v.ep + 1$ and $0 < \mu < 1$. Δd_i represents the detour distance of passenger i that can be calculated by (20). For an existing passenger P, detour distance represents extra distance that P has to travel as compared to direct distance from the vehicle's current location to the destination of P. However, for a new passenger, detour distance indicates extra distance as compared to direct distance from origin to destination.

$$\text{Detour distance} = \text{travel distance in Route}_{\mathrm{NN}} - \text{direct distance}. \tag{20}$$

The term $\mu \times CS \times (\Delta d_i / \sum_{i=1}^{n} \Delta d_i)$ indicates the saving given to an individual passenger. The expression $\Delta d_i / \sum_{i=1}^{n} \Delta d_i$ is used to share the percentage of total carpool saving given to passengers group on the basis of their detour distance. The passenger having maximum detour distance will be rewarded maximum in the form of fare reduction. The total fare reduction associated with passengers of a nearby vehicle $v \in V_c^N$ when new request assigned to it can be computed by using the following formula, where $n = v.ep + 1$.

$$\sum_{\mathrm{i=1}}^{\mathrm{n}} \Delta \mathrm{fare_i} = \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{RF_i} - \sum_{\mathrm{i=1}}^{\mathrm{n}} \mathrm{EF_i}. \tag{21}$$

Fig. 5 illustrates pricing strategy for the example given in Fig. 4. If the passenger P_1 agrees to tolerate the delay caused by the arrival of new passenger P_2, then from the vehicle's current location c, we can calculate the distance of individual routes and shared route on the basis of *NN* schedule as shown in Fig. 5a and Fig. 5b, respectively. As vehicle v in Fig. 4 satisfies the distance constraint of (8), therefore, fares and profit are calculated as follows. Given that $\mu = 0.5$ and $\partial_r = 10$, and the existing passenger P_1 tolerates a detour distance of 2 units whereas new passenger P_2's detour distance is 3 units in shared route.

The total carpool saving CS is calculated to be 30. The remaining fare of P_1 is estimated as $60 - 0.5 \times 30 \times (2/(2+3)) = 54$. The fare of P_2 is estimated as $(3+5.5) \times 10 - 0.5 \times 30 \times (3/(2+3)) = 76$. The passenger with more detour has more fare reduction. Passengers can save a total amount of 15 and driver can also earn an amount of 15, which is greater than the expected profit when the new request is assigned to v. Hence, our price model optimizes monetary benefits of all involves parties.

Algorithm 2 illustrates the complete recommendation mechanism for each request that arrives at current time. The algorithm takes as input the following parameters: (a) road map of a city, (b) set of vehicles in the city, and (c) Queue of ride requests.

1. *Searching for Candidate Vehicles (Line 2-Line 3):* In this step, the recommendation framework searches for candidate vehicles that can serve new request. Two constraints are validated: (a) candidate should lie

Algorithm 2 Vehicle Recommendation

Input: A road network : $\boldsymbol{G}(\boldsymbol{N}, , \boldsymbol{E})$, vehicles present in $\boldsymbol{G}$:$\boldsymbol{V}$, Queue of ride requests: $\boldsymbol{R_Q}$
Output: Recommended vehicle for each request $\boldsymbol{req} \in \boldsymbol{R_Q}$
Definitions: $\boldsymbol{t_{cur}}$ = *current time*, $\boldsymbol{SR_o}$ =*searching radius around origin*,$\boldsymbol{V_c^N}$ = *nearby vehicle set having seat capacity*, $\boldsymbol{capacity_{max}}$ = *maximum seat capacity of a vehicle*, $\boldsymbol{Fare_R}$ = *regular fare*, $\boldsymbol{Fare_E}$ = *estimated fare*,$\boldsymbol{T_d}$ = *list to store time delay of each passenger*, $\boldsymbol{S}$ = *created schedule*, $\boldsymbol{F_r}$ = *list to store regular fare of each passenger*, $\boldsymbol{F_e}$ = *list to store estimated fare of each passenger*, $\boldsymbol{i}$ = *counter variable for number of passengers in carpooling*

1. for each request $req_{new} \in R_Q$ that arrives at t_{cur} do
2. $SR_o \leftarrow radius(req_{new}.o)$
3. $V_c^N \leftarrow GetVehicles(SR_o, capacity)$
4. for each vehicle $v \in V_c^N$ do
5. $v.c \leftarrow capacity_{max} - v.ep$
6. if $v.c$ equals $capacity_{max}$ then // cab is vacant
7. $v..D_{Total} \leftarrow d_{v..l \rightarrow req_{new}..o} + d_{req_{new}..o \rightarrow req_{new}..d}$
8. $v.td_{avg} \leftarrow T(v.l, req_{new}.o)$
9. $v.\sum_i \Delta fare \leftarrow 0$
10. $v.\Delta Profit \leftarrow 0$
11. else // cab is occupied
12. $T_d \leftarrow \{\}; F_r \leftarrow \{\}; F_e \leftarrow \{\}$
13. $S \leftarrow NNscheduling(req_{new}, s_v)$
14. $v..D_{Total} \leftarrow GetDistance(Route_S)$
15. $D_R \leftarrow AggregateIndividualRouteDistance(req_{new}, s_v)$
16. if $D_R - v..D_{Total} < 0$ then
17. $V_c^N \leftarrow V_c^N - v$
18. else
19. for each request $req \in \{req_{new} \cup v.ED$ do
20. $td \leftarrow sdt - edt$
21. if req is *new* then
22. $Fare_R \leftarrow CalculateFare$ using (15)
23. else
24. $Farre_R \leftarrow CalculateFare$ using (17)
25. end if
26. $Fare_E \leftarrow CalculateFare$ using (18)
27. $T_d \leftarrow T_d.append(td)$
28. $F_r \leftarrow F_r.append(Fare_R)$
29. $F_e \leftarrow F_e.append(Fare_E)$
30. end for
31. $v.td_{avg} \leftarrow Mean(T_d)$
32. $v.\sum_i \Delta fare \leftarrow F_r) - Sum(F_e)$
33. $v.\Delta Profit \leftarrow Sum(F_e) - Profit_{exp}$
34. end if
35. end if
36. $v.AS \leftarrow Sum(v.c, \frac{1}{v..D_{Total}}, \frac{1}{v..td_{avg}}, v.\sum_i \Delta fare_i, v.\Delta Profit)$
37. end for
38. recommend vehicle with max $v..AS$
39. end for

within searching radius around origin and (b) candidate has enough available seats to hold new passenger.

2. *Calculating Aggregated Score for Each Candidate Vehicle (Line 4-Line 36):* In this step, a combined score is calculated for each candidate on the basis of average time delay, capacity, driving distance, fare reduction, and profit increment. *Line 5* calculates capacity of each candidate vehicle. Two cases may arise: (a) vacant vehicle and (b) occupied vehicle.
   (a) If the vehicle is vacant at arrival time of new request, then average time delay associated with passenger of a vacant cab is only the pick-up

delay, total distance is the distance of direct route, total fare reduction is 0 as passenger has to pay regular fare, and profit increment is also 0 as driver collects the regular fare for the total travel distance which is same as profit *(Line 6- Line 10).*

(b) If the vehicle is occupied at arrival time of new request, then different lists are initialized with null for each occupied vehicle. The lists store time delay and fare information of an individual passenger *(Line 12).* A *NN* schedule is created to assign priority to each request (new and already assigned requests to the vehicle) using *NNscheduling* function in*Line 13*that is defined in Algorithm 1.

Line 14 calculates total distance of the route defined by *NN* schedule *S*.*Line 15*calculates regular distance by adding the distances of the individual routes of passengers as illustrated in Fig. 4. After that, the candidate vehicle is tested with one more condition. If the test condition at *line 16* is true, then there is no benefit of assigning the new request to the candidate vehicle. Therefore, the candidate is removed from the nearby set V_c^N *(Line 17)* and control goes to the next vehicle in the set for calculating its parameters. Otherwise, if the test condition is false, then this means there is benefit of assigning new request to the candidate vehicle. Therefore, the framework calculates remaining parameters for this candidate (*Line 19–Line 30)*. Time delay of a passenger is calculated at *Line 20*. *Line 22* calculates regular fare of new passenger by using (15).

Line 24 calculates regular fare of an existing passenger in candidate vehicle by using (17). Afterwards, fare of each passenger is estimated on the basis of regular fare, detour distance, and carpool saving *(Line 26).* Carpool saving is calculated by (16). Each calculated parameter of a passenger is appended to its corresponding list *(Line 27-Line 29).*

Once all the parameters have been calculated for each passenger; average time delay, total fare reduction, and profit increment associated with occupied vehicle is calculated *(Line 31-Line 33). Line 36* calculates an aggregated score on the basis of all the calculated parameters. The candidate vehicle with highest aggregated score is recommended to the requesting passenger *(Line 38).*

V. PERFORMANCE EVALUATION

In this section, we perform the experimental validation of our proposed heuristic based vehicle recommendation framework *HASVR.*

A. EXPERIMENTAL SETTINGS

We have created a customized simulation framework in *R* programming language and utilized gmapsdistance [25] package. The framework is capable of visualizing all simulation modules (e.g., tracking vehicles and ride requests). We have conducted trace driven experimental analysis using *T-Drive* trajectory data sample [13]. The dataset contains GPS trajectories of 10,357 taxis of Beijing during the period of February 2 to February 8, 2008. We draw a sample with one day GPS traces of 250 taxis from the dataset to test our recommendation system. The framework takes as input a total of real-world 61,136 taxicab traces. From the dataset, we generate a passenger request (request time, origin, and destination) using uniform and Poisson distribution.

B. INITIAL TAXI STATES

The timestamp and location of taxis at the corresponding timestamp is taken from the GPS traces. However, the number of existing passengers in a taxi at a certain timestamp is randomly chosen between 0 and $capacity_{max}$. The trajectory points of each taxicab in the dataset are arranged with respect to ascending order of timestamps. This feature of *T-Drive* dataset helps to determine the direction of each taxi's traversal. Each trajectory point p in the dataset is followed by the points that are in forward direction to p. Therefore, the destination of existing passengers in a taxicab can be selected from the taxi's trajectory points that are in forward direction to taxicab's current location. Fig. 6 shows a trajectory followed by a taxicab. For example in Fig. 6, at arrival time of new request, taxi's current location is p_{17}; so the destination of an existing passenger can be selected on the same path as p_{23}.

To show the effectiveness of our proposed unified and dynamic carpooling service (recommended vehicle can be occupied or vacant); we compare our proposed approach, namely *Highest Aggregated Score Vehicle Recommendation (HASVR)*with *Nearest Vehicle Recommendation (NVR),* and *No-Carpooling.* We also compare our fare calculation model with the win-win fare model of *coRide* [10] to evaluate the maximum fare reduction of an individual passenger. *NVR* is a heuristic approach that first extracts the occupied vehicles with $\Delta D_T < 0$ from nearby set V_c^N and then greedily recommends the nearest vehicle to the passengers to minimize their waiting time. In *No-Carpooling* system, each requesting passenger is recommended a vacant vehicle from the set V_c^N. A ride request may be rejected if the system cannot find a vacant vehicle at the requested time. The passenger may resend the request to get the recommended vehicle.

TABLE 1. Default values of parameters used in simulation.

Parameter	Description	Value
∂_r	taxi fare per kilometer	0.62 Yuan
$capacity_{max}$	No. of seats in a taxi	3
SR_o	radius for searching nearby taxis	6 km
μ	% of carpool saving given to passengers group	0.5

C. EXPERIMENTAL RESULTS

Table 1 lists the default values for experimental parameters used in simulation. To study the benefits of our proposed carpooling recommendation system, different performance

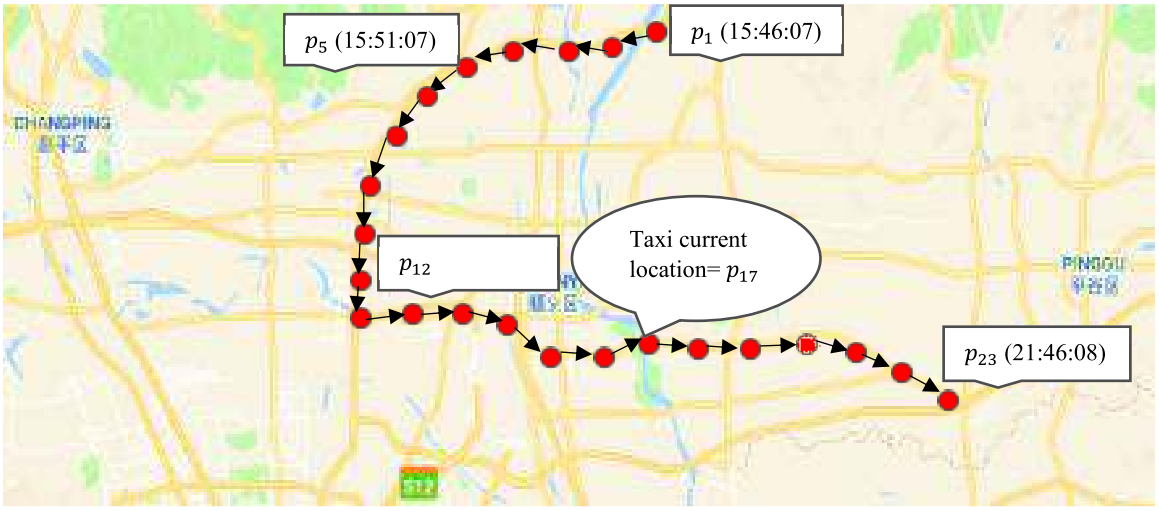


FIGURE 6. Trajectory followed by taxicab. Here *p* represents each trajectory point along with associated timestamp.

metrics are analyzed. A total of 23 simulations is considered in our study to evaluate different parameters. In our analysis, we have used two distinct perspectives to analyze the overall system performance, namely (a) driver's perspective and (b) passenger's perspective.

Driver's perspective:

- ***Percentage of reduced total mileage:*** It evaluates the effectiveness of our system by measuring how much total miles can be reduced by the taxis recommended by our unified carpooling system. It can be represented as follows.

$$\%\ \text{of reduced total mileage} = (\mathrm{M_v} - \mathrm{M})/\mathrm{M_v}. \quad (22)$$

Where M_v is the total mileage used to deliver all passengers separately using vacant taxicab services and M is the total mileage used to deliver all passengers using the taxis recommended by our proposed system (either vacant or occupied taxis).

- ***Percentage of increased profit:*** It evaluates that how much total extra profit can be earned by the taxis recommended by our system as compared to expected profit for the total distance travelled by all the recommended taxis. It can be computed as:

$$\%\ \text{of increased profit} = \frac{\text{collectedprofit} - \text{Profit}_{\text{exp}}}{\text{Profit}_{\text{exp}}}. \quad (23)$$

Where *collected profit* is the total profit collected by drivers of all the recommended taxis.

Passenger's perspective:

- ***Percentage of satisfied ride requests:*** It is defined as percentage of ride requests that get recommendation of taxis as compared to the total number of requests. A passenger request may be rejected at the requested time if each candidate in the nearby vehicle set V_c^N does not satisfy distance constraint (8). The passenger may resend the request to get recommendation of an optimal vehicle.
- ***Percentage of reduced total fares:*** It measures how much total fares can be reduced by the taxis recommended by our system as compared to vacant taxicab fares.
- ***Average detour ratio (%):*** It measures the percentage of detour distance to an individual passenger as compared to direct distance in a taxicab recommended by our system. Detour distance can be computed by (20).

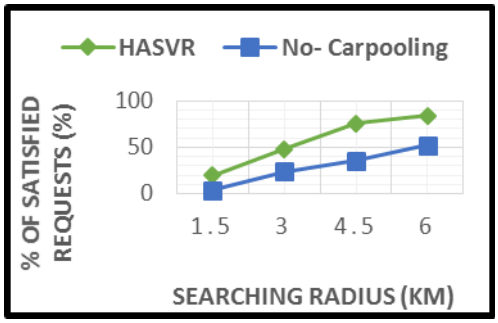


FIGURE 7. Percentage of satisfied ride.

1) PERCENTAGE OF SATISFIED RIDE REQUESTS

Fig. 7 shows the effect of searching radius on the percentage of satisfied requests. With the increase of radius from 1.5 to 6 km, the performance of both recommendation strategies increases. This is because the larger radius has more nearby taxis to satisfy ride requests. *No-Carpooling* scheme simply rejects the request if there is no vacant taxi available at requested time whereas our proposed scheme recommends either a vacant or occupied taxi. Therefore, *HASVR* outperforms the *No-Carpooling* scheme by an average of 28%. However, when the radius gets close to 6 km, there is little

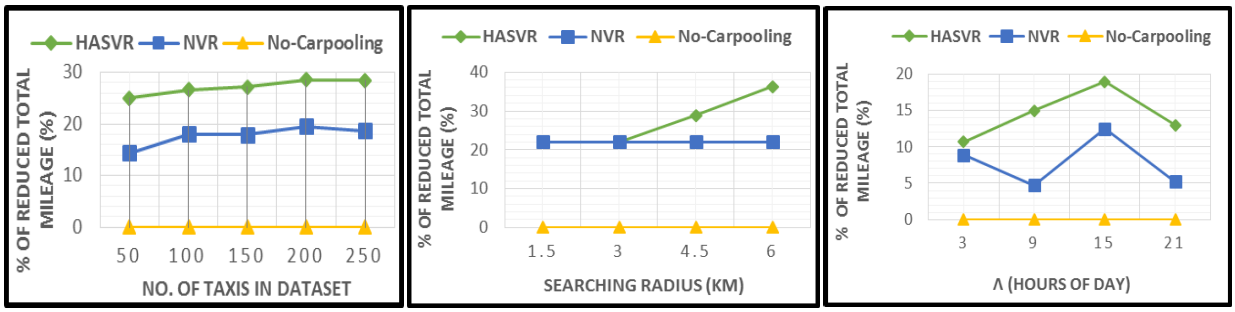


FIGURE 8. Percentage of reduced miles. (a) No. of taxis in dataset. (b) Searching radius. (c) Mean of poisson distribution.

increase for *HASVR,* as the radius becomes large enough to have sufficient nearby taxis and increasing the radius further does not improve the performance.

2) PERCENTAGE OF REDUCED TOTAL MILEAGE

Fig. 8 (a) shows the impact of number of taxis in the dataset on the percentage of reduced miles by keeping the searching radius constant. With the increase of taxis from 50 to 250 the performance of HASVR increases since increasing the number of taxis in the dataset increases the probability of finding an optimal taxi in terms of reduced mileage. However, *NVR* shows a fluctuation in performance. This behavior is due to the fact that increasing the taxis in the dataset finds the closer taxi to the request origin but that taxi may or may not be optimal in terms of reduced mileage.

HASVR outperforms *NVR* by 9 % on average and in comparison to *No-Carpooling*, it performs better by 27 % on average. The difference in performance of *HASVR* and NVR is due to the fact that requesting passenger's origin and destination may not match to the destinations of existing passengers in the nearest cab, since *NVR* only considers waiting time of passengers while recommending cabs.

Fig. 8 (b) shows the impact of searching radius on the percentage of reduced miles by keeping the number of taxis constant. The performance of *NVR* stays constant as it always recommends the nearest taxi from the nearby taxi set. At first, the performance *of HASVR* and *NVR* stays the same, then *HASVR* starts to perform better than *NVR* as the radius increases from 3 km. *HASVR* outperforms *NVR* by 5 % and as compared to *No-Carpooling*, its performance is better by an average of 27 %. Therefore, we conclude that increasing the number of taxis or increasing the searching radius, increases the number of taxis in the nearby set, thereby increasing the probability of finding the most appropriate taxi.

Fig. 8 (c) shows the impact of hours of the day on the percentage of reduced miles. For instance, $\lambda = 9$ represents the arrival time of requests around 9 am. During rush hours of the day e.g., 9 and 15, the percentage of reduced mileage is higher as compared to non-rush hours e.g., 3 and 21 for *HASVR.*This difference is due to the fact that during rush hours, vacant taxicabs become deficient and hence more occupied taxis are recommended to passengers leading to higher reduction of total mileage. As compared to *No-Carpooling, HASVR* outperforms during rush hours by an average of 17% and during non-rush hours it surpasses by 12% on average. However, *NVR* does not differentiate between rush and non-rush hours as it applies heuristics of nearest taxi for recommendation.

3) PERCENTAGE OF REDUCED FARES

Fig. 9 (a) illustrates the effect of different values of sharing percentage μ on the percentage of total fare reduction. When μ increases from 0.1 to 0.9, more share of total carpool saving is given to passengers group. Therefore, both *HASVR* and *NVR* show an increasing trend. However, the %-age of total fare reduction for *NVR* is less as compared to *HASVR* since NVR does not consider fare reduction of passengers while recommending a taxicab.

We evaluate the performance of *HASVR* in terms of maximum fare saving for an individual passenger as compared to win-win fare model of *coRide*[10] in Fig. 9 (b). Both schemes provide 0 % fare saving when number of passenger is 1 (when passenger travels alone). The performance of both schemes increases as we increase the number of passengers sharing the ride. In *coRide*, with more number of passengers, greater distance can be shared with other passengers, therefore, leading to larger carpool benefit for individual fare reduction. Since our scheme *HASVR* reduces each passenger's fare on the basis of detour distance, with greater number of passengers in carpool, a passenger has to tolerate more detour distance, leading to greater fare saving. As compared to our price model, win-win fare model [10] reduces fare of passengers on the basis of their regular fares. Our scheme outperforms than *coRide* by an average of 7% in a scenario when one of the passenger in carpool has to tolerate highest detour distance whereas the other passengers have comparatively very less detour. In *coRide,* each passenger will get some portion of the carpool saving whereas *HASVR* provides maximum share to the passenger with highest detour.

Fig. 10(a) illustrates monetary incentives in HASVR for the scenario of same origin and same destinations of all passengers whereas Fig. 10(b) illustrates the scenario of different origin and different destinations. For percentage of profit

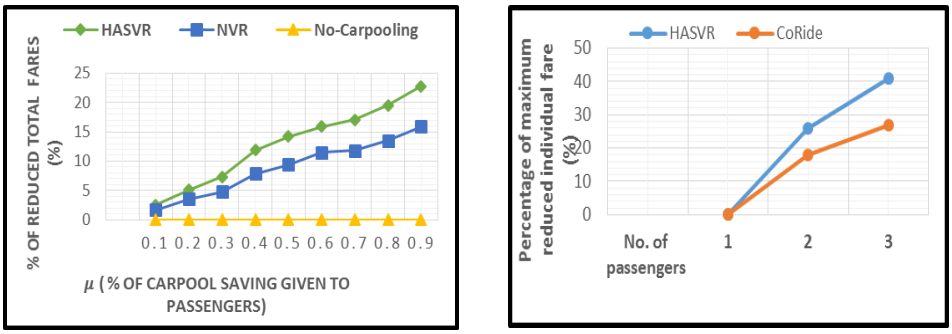


FIGURE 9. Percentage of reduced fares. (a) Reduced total fares . (b) Reduced individual fare.

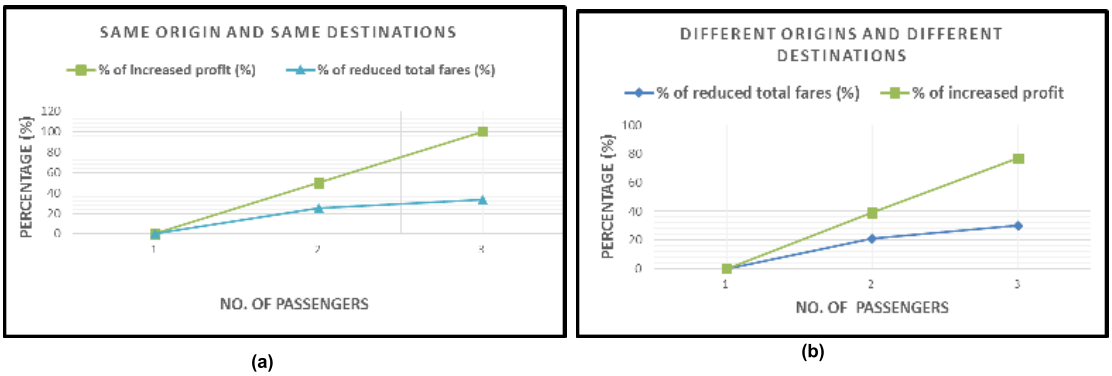


FIGURE 10. Monetary incentives in *HASVR*. (a) Same origin and same destinations. (b) Different origin and different destinations.

increase, the scenario of same origin and same destinations outperforms the scenario of different origin and different destinations by an average of 11% (in case of no detouring, carpooling gives max profit). For percentage of total fare reduction, the scenario of same origin and same destinations outperforms the scenario of different origin and different destinations by an average of 2%. In the scenario of same origin and same destinations, there will be maximum reduced total distance due to carpooling as per (8). This implies that the new request matches to the destinations of existing passengers in $v \in V_c^N$ and assigning the request to v will provide maximum benefit to both drivers and passengers (new and existing).

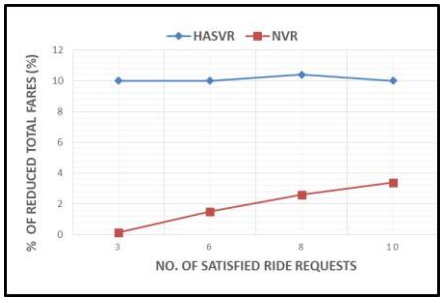


FIGURE 11. Scalability (% of reduced total fares).

Fig. 11 shows the impact of number of satisfied requests on the percentage of total fare reduction by fixing the number of vehicles in dataset. The performance of HASVR almost remains constant even if we increase the number of requests. On the other hand, the performance of NVR increases with the increase of number of requests. In worst scenario, the origins and destinations of new passengers will not match to the destinations of existing passengers that may result in lesser fare reduction passengers. However, HASVR outperforms NVR by an average of 8%. From the plot of HASVR, it is shown that HASVR is able to serve large number of requests without affecting the performance in terms of percentage of total fare reduction. Hence, our proposed scheme is scalable enough to serve large number of requests.

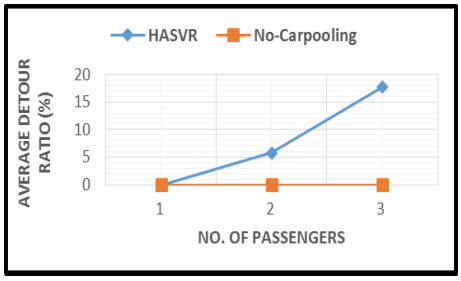


FIGURE 12. Average detour ratio (%).

4) AVERAGE DETOUR RATIO (%)

Fig. 12 plots the effect of number of passengers in a taxi on the detour ratio of an individual passenger. With increase

in number of passengers, a high distance is added to an individual delivery since after adding more passengers in carpool, most of the passengers will have to travel a longer route as compared to *No-Carpooling*.Although the percentage of individual detour ratio increases, the fare of individual passenger is reduced by our fare rewarding price mechanism as more passengers will share the common route leading to increased carpool saving. As a result, fare saving increases for an individual passenger.

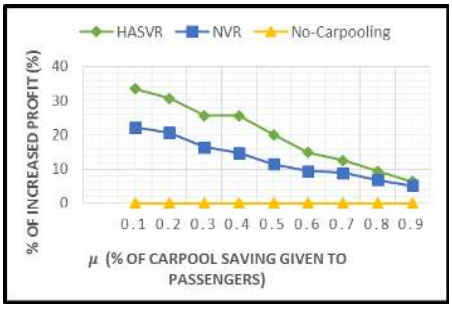


FIGURE 13. Percentage of increased profit.

5) PERCENTAGE OF INCREASED PROFIT (%)

Fig. 13 shows the percentage of increased profit of all schemes as μ increases from 0.1 to 0.9. When μ increases, less share $(1-\mu)$ of total carpool saving is given to drivers group. Therefore, both *HASVR* and *NVR* show a declining trend. Since *NVR* does not consider profit increment of drivers while recommending a taxicab, it falls behind *HASVR*.

We evaluate the performance of HASVR in terms of driver's increased income as compared to win-win fare model of coRide [10] in Fig. 14. The performance of HASVR increases by increasing number of vehicles in dataset. With the increase of vehicles from 50 to 150, the performance of HASVR stays same and then increases when number of vehicles in dataset is increased to 200. Then again, the performance remains same when number of vehicles in dataset is increased to 300. This shows increasing number of vehicles may or may not increase the probability of finding an optimal vehicle in terms of increased profit. If we use fare equation of CoRide, then driver will have to bear the loss as indicated in the graph. As compared to our price model, win-win fare model [10] reduces fare of passengers on the basis of their regular fares. If a passenger has greater distance between origin and destination in comparison to detour distance, then there will be more fare reduction as compared to our fare equation. According to our price model, a passenger also has to pay pick-up charge. This indicates that using our fare equation results in less fare reduction. Due to greater reduction in fares and absence of pick-up charge in fares, fare equation of CoRide will result in loss to drivers. On the other hand, our proposed fare equation reduces fares on the basis of detour distance and also adds pick-up charge in fare, so it ensures that the driver never gets loss and earns increased profit in the vehicle recommended by HASVR.

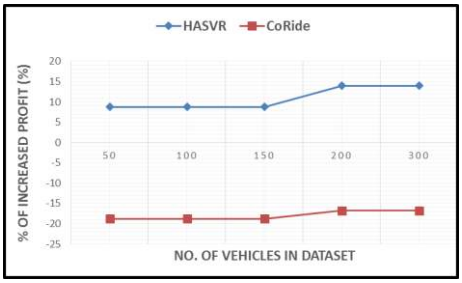


FIGURE 14. Percentage of increased profit.

Fig. 15 gives the impact of increase in total distance reduced due to carpooling on the monetary incentives of both passengers and drivers. The increase in ΔD_T is due to the reduction in distance of nearest neighbor route D_{NN}. This indicates that the new request matches to the destinations of existing passengers in taxi recommended by our system. This will lead to increased total carpool saving. As a result, the drivers' income (as compared to vacant taxicab service) increases and fare saving for passengers also increases.

To summarize the results, it is evident that our heuristic based dynamic vehicle recommendation strategy *HASVR* has on overall better performance, as the proposed framework considers the multiple objectives of drivers and passengers while recommending a vehicle to a passenger.

D. TIME COMPLEXITY

In this subsection, we present a time complexity analysis of the optimal ride sharing recommendation framework. We compute the time complexity of the NN scheduling algorithm and vehicle recommendation algorithm presented as Algorithm 1 and Algorithm 2 in this paper.

The NN scheduling algorithm utilizes a loop from Line 4 until Line 27. The loop is dependent on the length of the list to store unvisited locations. In worst case scenario, all the u locations appears in the list and the loop has to operate for all the locations. In the Line 17 to Line 20, there is another loop that calculates the distance from a certain location to all the other unvisited locations. Therefore, in the worst case scenario, the overall computation is u^2.

In the vehicle recommendation algorithm, for each vehicle NN scheduling is performed if they are not vacant. In worst case scenario, all the vehicles are occupied. Therefore, the time complexity will be vu^2 for all the v vehicles for one request. For r number of requests, the total time complexity of the optimal ride sharing algorithm is rvu ^{2}.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we have analyzed, designed, and evaluated a unified and dynamic carpooling recommendation system *HASVR* on the basis of large-scale historical GPS traces. The objectives of existing passengers in vehicle is also an important issue that needs to be addressed by the carpooling systems while recommending vehicles to requesting passengers.

HASVR considers the objectives of drivers, requesting passengers as well as existing passengers during vehicle recommendation. Achieving a trade-off between objectives of all the parties participating in carpooling is a major challenge that has been addressed in our work. Our main contribution is the mathematical model that is not dependent on any particular simulation framework. From the simulation study, it is evident that our proposed recommendation system has the potential in enhancing the system performance as compared to vacant taxicab services. We verify *HASVR* with a real-world dataset containing 61,136 taxicab traces.

In future, we plan to refine our dynamic carpooling model by introducing gender and passenger's detour distance constraints. In this study, we assign priority to each ride request on the basis of spatial closeness to the current location of vehicle. However, a requesting passenger might be in hurry and direct ride towards destination is not available. Moreover, we aim to provide solution to this issue by proposing an optimization scheduling technique. In such situations, a passenger may prefer to switch rides to reach as much closer to destination as possible.

REFERENCES

[1] Energy, Oil. *Consumption: Countries Compared*. Accessed: May 12, 2017. [Online]. Available: http://www.nationmaster.com/country-info/stats/Energy/Oil/Consumption
[2] NYC Taxi and Limousine Commission. *Taxi of Tomorrow Survey Results*. Accessed: Oct. 13, 2018. [Online]. Available: http://www.nyc.gov/html/tlc/downloads/pdf/tot_survey_results_02_10_11.pdf
[3] S. Ma, Y. Zheng, and O. Wolfson, "T-share: A large-scale dynamic taxi ridesharing service," in *Proc. IEEE 29th Int. Conf. Data Eng.*, Apr. 2013, pp. 410–421.
[4] *uberPOOL Together*. Accessed: Oct. 11, 2018. [Online]. Available: https://www.uber.com/en-PK/ride/uberpool/
[5] *Careem Sawa*. Accessed: Oct. 23, 2018. [Online]. Available: https://blog.careem.com/en/careem-sawa/
[6] *Car-Pooling Declines as Driving Becomes Cheaper*. Accessed: Nov. 21, 2017. [Online]. Available: http://www.nytimes.com/interactive/2011/01/29/us/20110129-nat-CARPOOL.html
[7] M. Furuhata, M. Dessouky, F. Ordóñez, M.-E. Brunet, X. Wang, and S. Koenig, "Ridesharing: The state-of-the-art and future directions," *Transp. Res. B, Methodol.*, vol. 57, pp. 28–46, Nov. 2013.
[8] D. Zhang, T. He, Y. Liu, S. Lin, and J. A. Stankovic, "A carpooling recommendation system for taxicab services," *IEEE Trans. Emerg. Topics Comput.*, vol. 2, no. 3, pp. 254–266, Sep. 2014.
[9] S. Ma, Y. Zheng, and O. Wolfson, "Real-time city-scale taxi ridesharing," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 7, pp. 1782–1795, Jul. 2015.
[10] D. Zhang *et al.*, "Carpooling service for large-scale taxicab networks," *ACM Trans. Sensor Netw.*, vol. 12, no. 3, p. 18, 2016.
[11] M. Zhu, X.-Y. Liu, M. Qiu, R. Shen, W. Shu, and M.-Y. Wu, "Transfer problem in a cloud-based public vehicle system with sustainable discomfort," *Mobile Netw. Appl.*, vol. 21, no. 5, pp. 890–900, 2016.
[12] M. M. Solomon, "Algorithms for the vehicle routing and scheduling problems with time window constraints," *Oper. Res.*, vol. 35, no. 2, pp. 254–265, 1987.
[13] Y. Zheng. (2011). *T-Drive Trajectory Data Sample*. Accessed: Sep. 13, 2017. [Online]. Available: www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/
[14] J. Yuan *et al.*, "T-Drive: Driving directions based on taxi trajectories," in *Proc. 18th Int. Conf. Adv. Geogr. Inf. Syst. (GIS)*, 2010, pp. 99–108.
[15] D. Zhang, T. He, S. Lin, S. Munir, and J. A. Stankovic, "Online cruising mile reduction in large-scale taxicab networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 11, pp. 3122–3135, Nov. 2015.
[16] D. Zhang, T. He, Y. Liu, and J. A. Stankovic, "CallCab: A unified recommendation system for carpooling and regular taxicab services," in *Proc. IEEE Int. Conf. Big Data*, Oct. 2013, pp. 439–447.
[17] P. M. Orey, R. Fernandes, and M. Ferreira, "Empirical evaluation of a dynamic and distributed taxi-sharing system," in *Proc. 15th Int. IEEE Conf. Intell. Transp. Syst.*, Anchorage, AK, USA, Sep. 2012, pp. 121–134.
[18] D. S. Setzke, C. Pflügler, M. Schreieck, S. Fröhlich, M. Wiesche, and H. Krcmar, "Matching drivers and transportation requests in crowdsourced delivery systems," in *Proc. 23rd Amer. Conf. Inf. Syst.*, Boston, MA, USA, 2017, pp. 1–10.
[19] B. Cao, L. Alarabi, M. F. Mokbel, A. Basalamah, S. Engineering, and S. Arabia, "SHAREK: A scalable dynamic ride sharing system," in *Proc. 16th IEEE Int. Conf. Mobile Data Manage.*, Pittsburgh, PA, USA, Jun. 2015, pp. 234–240.
[20] N. Agatz, A. L. Erera, M. W. P. Savelsbergh, and X. Wang, "Dynamic ride-sharing: A simulation study in metro Atlanta," *Transp. Res. B, Methodol.*, vol. 45, no. 9, pp. 1450–1464, 2011.
[21] X. S. Wang, "Large scale real-time ridesharing with service guarantee," *Proc. VLDB Endowment*, vol. 7, no. 14, pp. 2017–2028, 2017.
[22] M. Asghari, D. Deng, C. Shahabi, U. Demiryurek, and Y. Li, "Price-aware real-time ride-sharing at scale—An auction-based approach categories and subject descriptors," in *Proc. 24th ACM SIGSPATIAL Int. Conf. Adv. Geogr. Inf. Syst.*, Burlingame, CA, USA, Nov. 2016, Art. no. 3.
[23] *Developer's Guide|Google Maps Distance Matrix API|Google Developers*. Accessed: Aug. 7, 2017. [Online]. Available: https://developers.google.com/maps/documentation/distance-matrix/intro
[24] K. L. Hoffman, M. Padberg, and G. Rinaldi, "Traveling salesman problem," in *Encyclopedia of Operations Research and Management Science*. New York, NY, USA: Springer, Springer, 2013, pp. 1573–1578.
[25] T. Package, T. Distance, A. Rodrigo, A. Melo, and D. Zarruk. (2016). *Package Gmapsdistance*. [Online]. Available: https://cran.r-project.org/package=gmapsdistance

HAJRA QADIR received the bachelor's degree in computer engineering and the master's degree in computer sciences from the COMSATS Institute of Information Technology at Abbottabad Campus, Islamabad, Pakistan. She is currently a Lecturer with the COMSATS Institute of Information Technology. Her research areas include recommender systems, data mining, and natural language processing.

OSMAN KHALID received the Ph.D. degree from North Dakota State University, USA, and the master's degree from the Center for Advanced Studies in Engineering. He is currently an Assistant Professor with the COMSATS Institute of Information Technology. His research areas include recommender systems, network routing protocols, Internet of Things, and fog computing.

MUHAMMAD U. S. KHAN (M'13) received the Ph.D. degree in electrical and computer engineering from North Dakota State University, USA, in 2015, and the master's degree in information security from the National University of Science and Technology, Pakistan, in 2008. He is currently an Assistant Professor with the COMSATS Institute of Information Technology at Abbottabad Campus, Pakistan. His research interests include text analysis, data mining, social network analysis, recommendation systems, computer security, IoT, and big data.

ATTA UR REHMAN KHAN (SM'14) is currently an Associate Professor with the Faculty of Computing and Information Technology, Sohar University (SU), Oman. Prior to joining SU, he was the Director of National Cybercrime Forensics Lab, Pakistan, and the Head of the Air University Cybersecurity Center. He also serves as a Domain Expert for multiple international research funding bodies. His areas of research interest include cybersecurity, mobile cloud computing, ad hoc networks, and IoT. He is a steering committee member/the track chair/a technical program committee member of over 60 international conferences. He has received multiple awards, fellowships, and research grants. He was the Conferences Chair of the IEEE Islamabad Section. He is currently serving as an Associate Editor for the IEEE Access and the *Elsevier Journal of Network and Computer Applications*, an Associate Technical Editor for the *IEEE Communications Magazine*, and an Editor for the journal of *Cluster Computing* (Springer), *The Computer Journal* (Oxford), the IEEE SDN Newsletter, the *KSII Transactions on Internet and Information Systems*, the Springer Open *Human-Centric Computing and Information Sciences*, *SpringerPlus*, and *Ad hoc and Sensor Wireless Networks* journal.

RAHEEL NAWAZ is currently a Reader in text and data mining with Manchester Metropolitan University. He is also the Head of the Digital Transformations Research Cluster. He is also the Founding Head of the Text and Data Mining Lab. He holds adjunct and honorary positions at several research organizations, both in U.K. and in Pakistan. He regularly makes media appearances and speaks on a range of topics, especially, artificial intelligence and higher education. Before becoming a full-time academic, he served in various senior leadership positions at the private, higher, and further education sectors; and was an army officer before that.