Beyond Shortest Paths: Route Recommendations for Ride-sharing

Chak Fai Yuen, Abhishek Pratap Singh, Sagar Goyal, Sayan Ranu, Amitabha Bagchi
Dept. of Computer Science and Engineering, Indian Institute of Technology
New Delhi, India
[Chak.Fai.Yuen.mcs17,Abhishek.Pratap.Singh.cs513,Sagar.Goyal.cs115,sayanranu,bagchi]@cse.iitd.ac.in

ABSTRACT

In taxi ride-sharing, multiple customers are allotted to the same taxi as long as they are *compatible*, i.e., if none of them suffers a detour beyond a permissible threshold. To attract customers to ride-sharing, taxi operators promise a reduced fare upfront. As a result, if the taxi fails to pair the initial customer with additional compatible passengers, the taxi operator incurs a financial loss. Hence, it is important to ensure that the taxi finds compatible customers once it has picked up the initial customer. In the current scenario, the appearance of subsequent compatible customers is left to luck: a taxi moves along the shortest (or quickest) path for the existing customer and hopes to find additional compatible customers on its way. In this paper, we ask: *Is the shortest path the optimal path for ride-sharing?* To investigate this question, we develop a route recommendation algorithm called Share, which predicts the route with the highest probability of finding compatible customers, while staying within the detour limits. Through extensive evaluations on real datasets, we show that Share reduces failure rate in ride-sharing by up to 40%, while also improving waiting time for customers by 20%. Technically, the route recommendation problem is NP-hard. We overcome this bottleneck, by reducing the search space of the entire road network into a smaller directed acyclic graph, on which we perform *dynamic programming* to identify the optimal route. This strategy allows us to answer queries within 0.2 seconds, and thereby making Share deployable on real road conditions.

1 INTRODUCTION

With well known brands such as Uber and Didi Chuxing promoting *ride-sharing*, the taxi-service industry has become a prominent part of the internet-based "Sharing economy" [11]. In contrast to traditional ride-hailing services where a taxi is assigned at most one *ride-order* (request for a taxi), in ride-sharing, multiple *compatible* ride-orders are allocated to the same taxi. Two ride-orders are compatible if both of them can be serviced by a single taxi without taking a significant detour in either of their travel routes.

Ridesharing offers several benefits. From the riders' perspectives, their travel costs are reduced since they share a vehicle. For the ride-hailing service providers, ride-sharing allows them to better utilize their limited vehicles by servicing more customers, which results in more profits [42]. This is particularly useful during peak hours. In addition, ride-sharing is more environment-friendly as it helps reduce fuel consumption, carbon footprint and vehicular traffic congestion in cities. Owing to the numerous benefits of ride-sharing, the problem has received significant interest both from academia [10, 15, 25, 41] and industry [24, 37, 42]. In this work, we study the problem of *route recommendation* in ride-sharing environments.

The context is this: In the current technology scenario, when a taxi is allocated to a ride-order opted for ride-sharing, it picks the ride-order from the source location and travels to its destination in the shortest (or quickest) route. While travelling along this shortest route, the taxi may get lucky and be allocated to another compatible ride-order from a nearby location by the taxi service provider, and the taxi is re-routed to pick up the new request. To entice customers into opting for ride-sharing, taxi service providers such as Uber and Didi show the exact ride-sharing fare upfront even before the trip is started. Typically, this fare is much lower than what the customer would pay if an exclusive access to the taxi is requested. Since the fare would not hike even if the taxi is not allocated to another compatible ride-order, from the service providers' perspective, they risk incurring a loss with the ride-sharing model. This risk depends on the probability of being allocated to a compatible ride-order while travelling on the shortest route of the first ride-order. Therefore, an important question arises: *Can we move beyond shortest paths by recommending a route that is close in length to the shortest path and offers much higher chance of finding a compatible ride-order?* In this paper, we investigate this question and present substantial evidence that moving beyond shortest paths is indeed a promising approach.

Existing works on ride-sharing optimization are in two primary aspects: taxi assignment [10, 15, 24, 25, 32, 41] and route planning [37, 42]. Given the current status of all taxis and the ride-orders, the taxi assignment problem looks into the aspect of mapping each ride-order to a taxi such that a particular objective function is maximized. This objective function can be maximizing profit, minimizing detour, etc. The route planning problem identifies the sequence of customer pick-ups and customer drop-offs that maximizes an objective function. In both aspects of works, it is assumed that the taxi travels on the shortest path between two consecutive pick-up/drop-off points. Furthermore, there is no predictive element: all computations are deterministic in nature and based solely on the known ride-orders and taxi locations.

In our problem, we attack the *route recommendation* problem. Specifically we ask: Can we *predict* a slightly longer path that significantly enhances the chances of finding compatible customers? Our recommendation achieves three goals. First, by enhancing the likelihood of finding compatible customers, it reduces the risk associated with ride-sharing for taxi service providers. Second, by ensuring the length of recommended path is close to the shortest path, the customer experience is not compromised. Third, by anticipating where future ride-orders are likely to come from, the recommended

WWW '19, May 13–17, 2019, San Francisco, CA, USA

ACM ISBN 978-1-4503-6674-8/19/05.
https://doi.org/10.1145/3308558.3313465

route reduces the waiting time for customers and thereby improves their experience. The key contributions of our work are as follows:

- We formulate a novel problem of route recommendation for taxi in the ride-sharing environment.
- We show that the proposed problem is NP-hard. We overcome the computational bottleneck by developing an algorithm called SHARE:*Ride-Sharing with History-Aware Route Recommendations*. SHARE reduces the exponential search space of all possible paths in the entire road network to a directed acyclic graph, on which we perform dynamic programming to identify the optimal path.
- We perform extensive experiments on real taxi datasets from New York and Singapore across a range of metrics. We show that the routes recommended by the proposed model receives up to 40% reduction in orders without ride-sharing than the shortest path. In addition, the recommended routes reduce waiting time for customers by 20% while improving number of passengers per kilometer for the taxi operator.

2 PROBLEM STATEMENT

Definition 1 (Road Network). *A road network is a directed graph $G = (V, E, \delta, \lambda)$, where V is the set of nodes representing geographical sites, $E \subseteq V \times V$ is the set of edges representing road segments between sites, a distance function $\delta : E \rightarrow \mathbb{R}$ representing the length (weight) of each road segment, and a spatial mapping λ representing the geographic locations of sites in the form (latitude, longitude) tuples.*

We use the notation $e = (u, v)$ to denote a road segment (edge) from node u to v. The length $\delta(e)$ of an edge e is the Haversine distance from u to v. A *route* corresponds to a *simple path* in the network, i.e., path without cycles. In the rest of this paper, we use the word "path" to refer to a simple path. The length of a path $P = \{e_1, \cdots, e_k\}$ over k edges is $\delta(P) = \sum_{\forall e_i \in P} \delta(e_i)$. A path may also be represented as a sequence of connected nodes. We use the notation $e_i \in P$ and $v_i \in P$ to denote the fact that path P goes through edge e_i and node v_i respectively. Finally, we use the notation $SP(u, v)$ to denote the length of the *shortest path* from node u to v.

Definition 2 (Ride Order). *A ride-order o is represented as $\langle o^s, o^d, o^t \rangle$ where $o^s \in V$ denotes the pick-up point, $o^d \in V$ denotes the drop-off point, and o^t denotes the time at which the order request is made.*

A path P *covers* order o if there exists a *sub-path* $P' \subseteq P$, such that P' originates at o^s and ends at o^d. We use the notation $\delta_o(P) = \delta(P')$ to denote the distance travelled to fulfill order o through path P. The *detour ratio* $dr(o, P)$ of P with respect to order o is the proportion of extra distance travelled compared to the shortest path, i.e.,

$$dr(o, P) = \frac{\delta_o(P)}{SP(o^s, o^d)}. \tag{1}$$

When two ride-orders are allocated to the same taxi, they may incur a detour. To ensure that these detours are not too long, taxi service providers allocate two orders to the same taxi only if both their detours are at most α [37, 42], where $\alpha \geq 1$ is a *detour threshold*. We call such ride-orders *compatible*.

Definition 3 (Compatible Ride Orders). *Let $o_1 = \langle o_1^s, o_1^d, o_1^t \rangle$ be a ride-order being serviced by taxi C and $o = \langle o^s, o^d, o^t \rangle$ is a new ride-order request. Furthermore, let $P_1 = \{o_1^s, \cdots, n\}$ be the path taken by C so far where n is its current location at time o^t. We call o compatible with o_1 if there exists a path P_2 originating at n such that P_2 covers o, $P_C = P_1 + P_2$ is a path that covers o_1 and both $dr(o_1, P) \leq \alpha$, $dr(o, P_2) = dr(o, P_C) \leq \alpha$.*

To generalize the above definition for arbitrary number of passengers, we assume each taxi is associated with a capacity of X passengers. Furthermore, for some m s.t. $0 \leq m \leq X$, $\mathbb{O} = \{o_1, \cdots, o_m\}$ is the set of orders currently being serviced by the taxi and P_1 is the path it has taken till now. A new request o can be allotted to this taxi if $Compatible(\mathbb{O}, o) = 1$.

$$Compatible(\mathbb{O}, o) = \begin{cases} 0 & \text{if } m = X \\ 1 & \text{if } \exists \text{ path } P_2 \text{ from current node } n, \\ & \text{such that } P_2 \text{ covers } o, \\ & \forall o_i \in \mathbb{O},\ P_C = P_1 + P_2 \text{ covers } o_i, \\ & \forall o_i \in \{\mathbb{O} \cup o\},\ dr(o_i, P_C) \leq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

Example 1. *Consider the scenario shown in Fig. 1. A taxi is at v_1 servicing $\mathbb{O} = \{o\}$, where $o = \langle v_1, v_{10}, o^t \rangle$. There are two other orders, $o_1 = \langle v_5, v_8, o^t \rangle$ and $o_2 = \langle v_9, v_3, o^t \rangle$, that are yet to be allotted to a taxi. Now, we will calculate $Compatible(\mathbb{O}, o_1)$ and $Compatible(\mathbb{O}, o_2)$.*

For o_1, there exists a path $P_C = \{v_1, v_3, v_5, v_6, v_8, v_{10}\}$ of distance 23. This path P_C covers both o and o_1. Assume $\alpha = 1.5$. A sub-path $P'_C = \{v_5, v_6, v_8\} \subseteq P_C$ is the shortest path for o_1 so the detour ratio is 1. Furthermore, $SP(v_1, v_{10}) = 20$, so the detour ratio for o is $23/20 = 1.15 \leq 1.5$. Hence, $Compatible(\mathbb{O}, o_1) = 1$.

For o_2, there does not exist a path that satisfies Eq. 2 without violating the detour constraint. Hence, $Compatible(\mathbb{O}, o_2) = 0$.

Let C be a taxi of capacity X servicing a set of ride-orders $\mathbb{O}$, $|\mathbb{O}| < X$. Taxi C has just picked up the order request $o_m = \langle o_m^s, o_m^d, o_m^t \rangle \in \mathbb{O}$. Without loss of generality (*WLOG*), we assume the taxi has taken path P_1 till now. We now need to recommend the *best feasible* route to the taxi. Formally, this problem is defined as follows.

Problem 1 (Route Recommendation in Ride-sharing). *A path P in the network is feasible if (i) P originates at o_m^s, and (ii) $P_1 + P$ does not exceed the detour ratio for any of the orders in $\mathbb{O}$. The best feasible path is the one that has the highest number of expected compatible orders with respect to $\mathbb{O}$ at time o^t. More specifically, find the path P^* where*

$$P^* = \arg\max_P \left\{ E\left(P|\mathbb{O}, o^t\right) \mid P \text{ is a feasible path} \right\} \tag{3}$$

$E\left(P|\mathbb{O}, o^t\right)$ denotes the expected number of compatible passengers in P at time o^t.

The next task therefore is to mathematically quantify $E\left(P|\mathbb{O}, o^t\right)$ for any given feasible path P.

2.1 Computing Expected Number of Passengers

To compute the expected number of compatible passengers, we use the historical data $\mathbb{H}$ containing past ride-orders. We use the notation $\mathbb{H}_D \subseteq \mathbb{H}$ to denote the set of ride-orders made on a particular date D. Given a node v and current time o^t, we extract the set of compatible orders originating at v from $\mathbb{H}_D$ for each date D covered in the

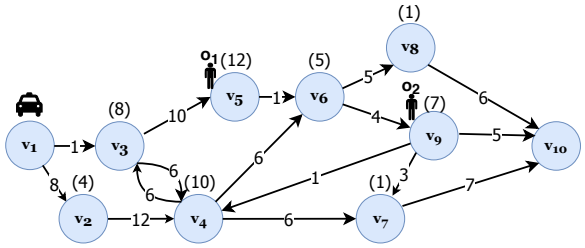


Figure 1: An illustration of a road network G. In this graph, the length of each edge is shown as edge weights. The numbers in brackets above each node denotes the expected number of compatible passengers, $E(v|\mathbb{O}, o^t)$, which we define in Sec. 2.1.

history.

$$\mathbb{H}_{D,v,\mathbb{O},o^t} = \{o' | o' \in \mathbb{H}_D,\ o'^s = v, \\ \|o^t - o'^t\| \leq \Delta, Compatible(\mathbb{O}, o') = 1\} \quad (4)$$

Here, Δ is a smoothing parameter on the time dimension. It is reasonable to set Δ to any value between 5 to 15 minutes. Therefore, the *expected* number of ride-orders arising from node v at time o^t is the average count across all dates in $\mathbb{H}$. Specifically,

$$E(v|\mathbb{O}, o^t) = \frac{1}{Y} \sum_{\forall \mathbb{H}_{D_i} \in \mathbb{H}} \left| \mathbb{H}_{D_i,v,\mathbb{O},o^t} \right| \quad (5)$$

where Y is the total number of days spanning $\mathbb{H}$.

Let $P = \langle v_1, \cdots, v_n \rangle$ be a feasible path. The expected number of compatible passengers in P is therefore

$$E(P|\mathbb{O}, o^t) = \sum_{\forall v_i \in P} E(v_i|\mathbb{O}, o^t) \quad (6)$$

Eq. 6 completes the problem formulation.

3 BASELINE: SHORTEST PATH

The baseline algorithm is to select the shortest path [37, 42]. There is no predictive element in this approach. Formally, let taxi C be servicing order set $\mathbb{O}$. We now introduce the concept of a *route plan*.

Definition 4 (Route Plan). *A* route plan, *RP, is a sequence of drop-off locations corresponding to each order in* $\mathbb{O}$.

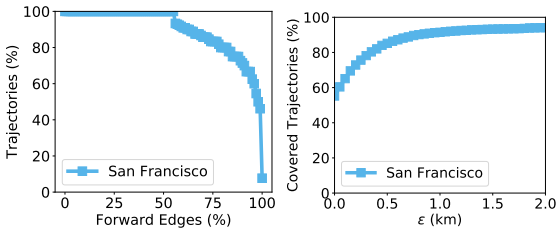
A path P *satisfies* route plan RP if it visits the drop-off locations in the same sequence as in RP. Let $SP(RP)$ denote the shortest path satisfying route plan RP. The *shortest route plan* for $\mathbb{O}$, RP^*, is the route plan with the smallest shortest path. Mathematically,

$$RP^* = \arg\min_{RP} \{SP(RP) \mid RP \in \mathbb{RP}\} \quad (7)$$

where $\mathbb{RP}$ is the set of all possible route plans for $\mathbb{O}$. In the shortest path approach, the taxi therefore simply follows $SP(RP^*)$.

4 SHARE:RIDE-SHARING WITH HISTORY-AWARE ROUTE ReCOMMENDATIONS

The shortest path approach is blind to the need of finding compatible passengers on its way. Our formulation addresses this weakness.



(a) (b)

Figure 2: (a) Frequency distribution of trajectories against the percentage of forward edges. (b) Distribution of covered trajectories against the extra distance threshold ϵ.

4.1 Complexity Analysis

Definition 5 (Longest Path Problem). *Given a weighted graph $G = (V, E)$ with non-negative edge weights w_e, the Longest Path problem is to find the longest simple path from a source node (s) to a destination node (d).*

Definition 6 (Max-Weight Path Problem). *Given a graph $G = (V, E)$ with non-negative node weights w_v, the Max-Weight path problem is to find the maximum weight simple path from a source node (s) to a destination node (d).*

The route recommendation problem for ride-sharing is essentially the *Max-Weight Path* problem, where the weight of each node v is the expected number of compatible passengers at v (Eq. 5). The route with the highest number of expected compatible customers is therefore the Max-Weight path.

We also know that the Longest Path problem is NP-hard [6].

Theorem 1. *Finding the Max-Weight path is NP-Hard*

Proof : Given an arbitrary instance of the Longest Path Problem, we reduce it to an instance of the Max-Weight Path problem through the following procedure. Consider the given graph $G = (V, E)$ in the Longest Path Problem, we design a graph $G' = (V', E')$ such that $\forall v \in V$, we have a node N_v in V' and $\forall e = (u, v) \in E$ we have a node N_e in V'. Furthermore, $\forall e = (u, v) \in E$ we have the edges $e_1 = (u, N_e)$ and the edge $e_2 = (N_e, v) \in E'$. We assign the weight w_e to all $N_e \in V'$, the weight 0 to all N_v and all the edges $e \in E'$ are un-weighted.

In G' the Max-Weight problem is to identify the path from a source s to a destination d with maximum weight. Since each node N_e selected in the path in G' corresponds to selecting the edge $e \in E$, if Max-Weight problem is solved, the resultant path is the longest path from s to d in graph G and hence, we would have solved the Longest Path Problem. □

Owing to the NP-hardness, an optimal algorithm is not feasible. This result motivates us to study real-world taxi trajectories and identify properties that may help reduce the search space.

4.2 An inspection of real-life trajectories

When a passenger is in a taxi, the taxi is expected to progressively get closer to the destination o^d through *forward* edges.

Definition 7 (Forward and Backward Edge). *Given a destination node o^d, we call an edge (u, v) to be a* forward edge *if $SP(u, o^d) > SP(v, o^d)$; otherwise, it is a* backward *edge.*

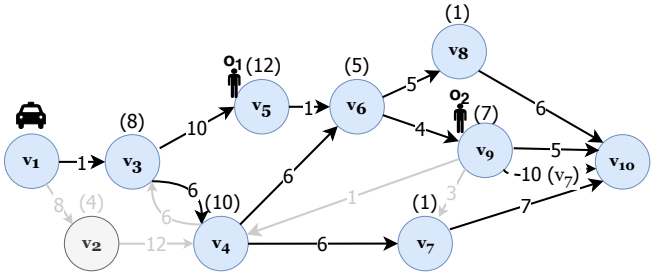


Figure 3: The subgraph D of the graph G in Fig. 1. The solid edges represent those that are part of E_D. The greyed-out edges and nodes represent those that are part of the original graph G but omitted from the D (See Ex. 2 for further details). The only dashed edge (v_9, v_{10}) represents an edge that is part of the extended DAG X (See Sec. 4.4) but not D.

Taking a backward edge is a sub-optimal decision as it detours away from the destination, so we expect most real-life taxis move predominantly through forward edges. To verify this hypothesis, we analyze taxi trajectories in San Francisco [31] and compute the distribution of trajectories containing at least $x\%$ of forward edges. The details of the San Francisco dataset are provided in Sec. 5.1.1.

Fig. 2(a) presents the result. As can be seen, more than 55% of the trajectories have only forward edges. Even when a taxi travels on backward edges, the proportion of backward edges is generally small. For example, more than 80% of the trajectories have at least 80% forward edges. This result gives credence to our hypothesis that taxis progressively move closer to its destination in general.

4.3 Inducing a Directed Acyclic Graph on the Road Network

Motivated by the above result, we reduce the search space from all feasible paths to only those feasible paths that are comprised entirely of forward edges. To formalize this idea, let C be a taxi servicing order set $\mathbb{O}$ and have just picked up order $o \in \mathbb{O}$ from node o^s. Furthermore, let node o^{d*} be the next drop-off point according to the shortest route plan RP^* for $\mathbb{O}$. Then we extract the subgraph $D(V_D, E_D)$ from the entire road network $G(V, E)$, where

$$V_D = \left\{ v \in V \mid SP(v, o^{d*}) \leq SP(o^s, o^{d*}) \right\} \tag{8}$$

$$E_D = \{(u, v) \in E \mid u, v \in V_D, (u, v) \text{ is a forward edge}\} \tag{9}$$

Example 2. *Let us revisit graph G from Fig. 1. Fig. 3 presents the subgraph $D = (V_D, E_D)$ constructed from G corresponding to the order set $\mathbb{O} = \{o = \langle v_1, v_{10}, o^t \rangle\}$. Furthermore, only the solid edges in Fig. 3 are part of E_D. We illustrate the importance of the dashed edge (v_9, v_{10}) later in Sec. 4.4.*

In D, according to Eq. 8, $v_2 \notin V_D$ since $SP(v_2, v_{10}) > SP(v_1, v_{10})$. Moreover, according to Eq. 9, the edge (v_9, v_7) is omitted since $SP(v_7, v_{10}) = 7 > SP(v_9, v_{10}) = 5$.

Theorem 2. *$D(V_D, E_D)$ is a directed acyclic graph (DAG).*

Proof by contradiction. D is directed since G is directed. Let us assume that the set of edges $E_c = \{(v_1, v_2), (v_2, v_3) \cdots, (v_m, v_1)\}$ form a cycle in D. In D, an edge (u, v) from G is retained if and only if $SP(u, o^d) > SP(v, o^d)$. This leads to both $SP(v_1, o^d) > SP(v_m, o^d)$ and $SP(v_m, o^d) > SP(v_1, o^d)$, which is a contradiction. □

Thm. 2 has deep implication on our problem. Specifically, although the Max-Weight Path problem is NP-hard on a graph, on a DAG, it can be solved in polynomial time [33]. Specifically, let (d_1, o^{d*}) and (d_2, o^{d*}) be the two incoming edges to destination node o^{d*}. The longest path, $LP(o^s, o^{d*})$, from o^s to o^{d*} in a node-weighted graph is $LP(o^s, o^{d*}) = \max\{LP(o^s, d_1), LP(o^s, d_2)\} + weight(o^{d*})$. Since the Max-Weight Path problem in a DAG can be broken down into smaller problems, this can be solved in polynomial time using *dynamic programming (DP)*.

Unfortunately, this DP formulation cannot directly be applied to our problem. Specifically, we have two kinds of weights: the edge weights denoting the road distance, and the node weights denoting the expected number of compatible customers. We need to find the Max-Weight path from o^s to o^{d*} in the domain of expected customers, such that the path, in terms of distance, does not violate the detour ratio (Eq. 1) of any of the orders in $\mathbb{O}$. We satisfy these dual needs through the following DP policy.

For each order $o_i \in \mathbb{O}$, let l_i be the length of the path they have travelled so far. The maximum amount of distance o_i can further travel in its path towards o^{d*} without violating the detour constraint is therefore $\overrightarrow{l_i} = \alpha \times SP(o_i^s, o_i^d) - l_i - SP(RP^*(o^{d*}, o_i^d))$, where $RP^*(o^{d*}, o_i^d)$ is the subsequence of the shortest route plan from o^{d*} to o_i^d. Therefore, any path P from o^s to o^{d*} is feasible if $\delta(P) \leq \overrightarrow{l_{min}}$, where $\overrightarrow{l_{min}} = \min\left\{\overrightarrow{l_i} \mid o_i \in \mathbb{O}\right\}$.

Next, we introduce the notation $c_{v,j}$ to denote the maximum expected number of compatible passengers among all paths of length j from o^s to some node v. Mathematically,

$$c_{v,j} = \arg \max_{\forall P \in \mathbb{P}_{v,j}} \left\{ E(P|\mathbb{O}, o^t) \right\} \tag{10}$$

where $\mathbb{P}_{v,j} = \{P | \delta(P) = j, P \text{ is a path from } o^s \text{ to } v \text{ in } D\}$. We are now ready to define the DP. The initialization condition in the DP is

$$c_{v,j} = \begin{cases} 0 & \text{if } v = o^s, j = 0 \\ -\infty & \text{otherwise} \end{cases} \tag{11}$$

Now, the recurrence is defined as:

$$c_{v,j} = \max_{\forall (u,j') \in N_j(v)} \left\{ c_{u,j'} \right\} + E\left(v \mid \mathbb{O}, o^t\right) \tag{12}$$

where $N_j(v) = \left\{(u, j') \mid (u, v) \in E_D,\ j = j' + \delta(u, v) \leq \overrightarrow{l_{min}}\right\}$ is the set of all incoming edge-length pairs that result in a path of length j without violating the detour ratio. The recurrence is computed using *memoization* for all nodes. The recommended path to o^{d*} corresponds to the path leading to $c^* = \max_{\forall j} \left\{ c_{o^{d*},j} \right\}$. In addition, we also store the incoming neighbor from $N_j(v)$ that maximizes $c_{v,j}$. This is used to backtrack and identify the path providing the highest expected number of compatible passengers.

Example 3. *Table 1 presents the $c_{v,j}$ values for each node in the DAG D in Fig. 3. We assume source $o^s = v_1$ and destination $o^{d*} = v_{10}$. The shortest path from o^s to o^{d*} is of length 20. Assuming $\alpha = 1.5$, $l_{min} = 30$. Thus, $c_{v,j}$ matters only if $j \leq 30$.*

Consider calculating $c_{v_6,12}$, the possible (u, j') pairs are $N_{12}(v_6) = \{(v_5, 11), (v_4, 6)\}$. The maximum $c_{u,j'}$ is $c_{v_5,11} = 20$. Thus $c_{v_6,12} = c_{v_5,11} + E\left(v_6 \mid \mathbb{O}, o^t\right) = 20 + 5 = 25$.

At o^{d*}, the maximum $c_{v_{10},j}$ out of the feasible js is 32, which is highlighted in bold. The row corresponding to $\rightarrow$ in Table 1 indicates

Table 1: The DP table corresponding to the DAG in Fig. 2(a) ($c_{v,j} = -\infty$ is omitted). Refer to *Ex.* 3 for further details.

v	v_1	v_3	v_4	v_5	v_6	v_6	v_7	v_8	v_8	v_9	v_9	v_{10}	v_{10}	v_{10}	v_{10}	v_{10}
j	0	1	7	11	12	13	13	17	18	16	17	20	**21**	22	23	24
$c_{v,j}$	0	8	18	20	25	23	19	26	24	32	30	19	**32**	30	26	24
$\rightarrow$	-	v_1	v_3	v_3	v_5	v_4	v_4	v_6	v_6	v_6	v_6	v_7	v_9	v_9	v_8	v_8

Algorithm 1 Path finding using Dynamic Programming in DAG

Input: Starting location o^s, destination o^{d*}, detour threshold α, number of bins Ω, Graph G
Output: Optimal path p^* in DAG
1: $D \leftarrow$ Construct DAG from G
2: $\forall v \in V_D$, compute $top-order(v)$
3: $c_{o^s,0} \leftarrow 0$
4: **for** $i \leftarrow 1$ **to** $top-order(o^{d*})$ **do**
5: $\quad V_i \leftarrow \{v \in V_D \mid top-order(v) = i\}$
6: $\quad$ **for each** $v \in V_i$ **do**
7: $\quad\quad \forall b,\ 1 \le b \le \Omega$, Compute $c_{v,b}$ (Eq. 12)
8: $c^* \leftarrow \max_{\forall b}\{c_{o^{d*},b}\}$
9: $P^* \leftarrow$ backtrack with c^* to get optimal path
10: **return** P^*

the incoming neighbor that maximizes the $c_{v,j}$ value. By backtracking using the incoming neighbors, we conclude that the optimal path P^* is $\{v_1, v_3, v_5, v_6, v_9, v_{10}\}$.

The computational complexity of the proposed DP remains exponential since, for each node v, the number of possible js can be as large as the number of possible ways to reach v from o^s, which is exponential. This implies an exponential number of columns in the table. The exponential cost can be avoided by moving from the continuous space to a discrete space. Particularly, each value of j is assigned to bin $b = \left\lceil \frac{j}{\omega} \right\rceil$, where ω is the width of each bin. All aspects of the DP remains identical except (1) instead of computing $c_{v,j}$, we compute $c_{v,b}$; (2) instead of $N_j(v)$, we use $N_b(v) = \left\{(u, b') \mid (u,v) \in E_D,\ b = \left\lceil \frac{b' \times \omega + \delta(u,v)}{\omega} \right\rceil \le \Omega \right\}$, where Ω is the total number of bins and the bin width ω is set to $\frac{\overrightarrow{l_{min}}}{\Omega}$. As we will see in our empirical evaluation (Sec. 5.4), $\Omega = 100$ provides a good balance between efficiency and quality.

Alg. 1 presents the pseudocode of the DP. After constructing the DAG D (line 1), we compute the *topological order* of each node in D (line 2). From the construction of D, it is guaranteed that the source o^s will have a topological order of 1 and o^{d*} will have the largest order. The nodes in D are processed according to their topological order (line 4) and $c_{v,b}$ for each node is computed (lines 5-7). By processing nodes in *topological order*, we ensure that all incoming neighbors of a node has already been processed. Finally, the path corresponding to the highest expected number of compatible passengers is returned (lines 8-10).

4.3.1 Properties: We next outline the key properties of Share.

Theorem 3. *The shortest path is one of the possible solutions.*

Proof. Consider the shortest path $P = \{v_1, \cdots, v_n\}$, where $v_1 = o^s$ and $v_n = o^{d*}$. Since it is the shortest path, $SP(v_i, o^{d*}) > SP(v_{i+1}, o^{d*})$ for $i < n$. Therefore, this path exists in D. □

Time Complexity: Topological sorting of all nodes in the DAG consumes $O(|V|+|E|)$ time. The number of states in the dynamic programming is $|V|\Omega$. To compute the value of each state, we consider all incoming edges. Hence, each edge is considered at most once per discrete distance leading to a complexity of $O((|V| + |E|) \cdot \Omega)$.

Space Complexity: Since we have $|V|\Omega$ states in the worst case, the space complexity is $O(|V|\Omega)$.

4.4 Improving the Search Space through Extended DAG

Our search space in the DAG is restricted to only paths formed by forward edges. Fig. 2(a) reveals that $\approx 55\%$ of the taxis conform to this assumption. Therefore, we ask: *Is it possible to incorporate more paths in the DAG by slightly relaxing the forward edge assumption?*

Whenever a taxi takes a backward edge from some node u, it moves further away from the destination, and therefore, incurs some *extra distance* over what would have been the shortest path from u. We hypothesize that the extra distance incurred by a taxi owing to a backward edge is small since no taxi would want to burn unnecessary fuel. *Extra distance* is formally defined as follows.

Definition 8 (Extra Distance). *Let (u, v) be a backward edge. Furthermore, let $P = \{u, v, \cdots, x\}$ be the sequence of nodes following the backward edge till the taxi reaches some node x such that $SP(x, o^{d*}) \le SP(u, o^{d*})$. The extra distance, denoted as $eDist(u, v, o^{d*}) = \delta(P)$, is the distance travelled by the taxi to reach a node that is at least as close to the destination as u.*

To verify our hypothesis that backward edges are taken by a taxi only when the resulting extra distance is small, we revisit the San Francisco dataset and call a taxi trajectory T *covered* if $\forall (u, v) \in T$, either (1) (u, v) is a forward edge, or (2) $eDist\left(u, v, o^{d*}\right) \le \epsilon$, where ϵ is a threshold and o^{d*} is the next drop-off point of T. ϵ controls when we call the extra distance owing to a backward edge *small*; when $\epsilon = 0$, only forward edges are allowed. Fig. 2(b) presents the results as ϵ is varied. As can be seen, at $\epsilon = 0.5km$, more than 85% trajectories are covered. We take inspiration from this result and propose the idea of an *extended* DAG $X = (V_X, E_X)$.

The nodes in the extended DAG is same as in the original DAG, i.e., $V_X = V_D$. The edge set $E_X = \{E_D \cup E_\epsilon\}$, where E_D is the set of forward edges (Eq. 9), and E_ϵ is the set of *extended* edges that capture paths through backward edges incurring an extra distance of at most ϵ. Formally, an extended edge is defined as follows.

Definition 9 (Extended Edge). *We add an* extended *edge between two nodes $u,\ x \in V_X$ if the following conditions hold:*
(1) $SP(x, o^{d*}) \le SP(u, o^{d*})$, where o^{d*} is the next drop-off point according to RP^* for $\mathbb{O}$.
(2) There exists a path $P = \{u, v, \cdots, x\}$ in the road network G such that (i) (u, v) is a backward edge, and (ii) $\delta(P) \le \epsilon$. It is possible that multiple paths from u to x satisfy both conditions (i) and (ii). Consequently, multiple edges may exist between u and x.

Since multiple edges may exist between two nodes in the extended DAG, we denote each edge with the tuple (u, x, P). For extended edges, P corresponds to the path satisfying condition (2) in Def. 9. On the other hand, in a forward edge (u, x, P), $P = \{u, x\}$. The *length* of any edge (u, x, P) is therefore $\delta(P)$. Condition (1) in Def. 9 ensures that the extended DAG remains acyclic. This enables us to extend Share from the forward edge assumption.

Example 4. *The extended DAG X of the network in Fig. 1 at $\epsilon = 10$ is shown in Fig. 3. The solid edges correspond to the forward edge, and the dashed edge from v_9 to v_{10} corresponds to an extended edge. This extended edge exists due to the path $P = \{v_9, v_7, v_{10}\}$.*

Now we can obtain a path $\{v_1, v_3, v_5, v_6, v_9, v_7, v_{10}\}$ with an optimal $c^* = 33$, which is higher than 32 obtained in plain DAG D.

The size of the extended DAG is exponential in ϵ: All extended edges originating from node u can be identified by initiating a depth-first exploration from u, traversing a backward edge to v first, then exploring further edges until the total length exceeds ϵ or a node x is reached that is closer to the destination than u. Since the number of paths between any two nodes is potentially exponential, the worst case number of edges in E_ϵ is exponential in ϵ. Hence, constructing the extended DAG and solving the DP are both exponential in ϵ. However, by limiting ϵ to a small value such as $0.5km$, we can obtain a good balance between running time and quality. We further explore this aspect in Sec. 5.

Searching Algorithm: The dynamic programming (DP) applied on the DAG can also be applied on the extended DAG to identify the path having the largest number of expected compatible customers. However, there are two issues that need to be treated differently. First, each extended edge (u, x, P) corresponds to path P from the network. Thus, each extended edge (u, x, P) is replaced with P in the optimal path identified by the DP. Second, although any path in the extended DAG is simple, once extended edges are replaced with their corresponding paths, they may introduce repeated vertices. To give an example, consider two connected extended edges (u, x, P_1) and (x, z, P_2). It is possible that $\exists v \in V_X,\ v \in P_1$ and $v \in P_2$. To reject such paths, the recurrence is allowed to proceed only through incoming edges that do not introduce a cycle. To accommodate this additional constraint, we modify the definition of feasible incoming neighbors $N_j(x)$ of node x in Eq. 12 as follows.

$$N_j(x) = \{(u, j') \mid (u, x, P) \in E_X,\ j = j' + \delta(P) \leq \overrightarrow{l_{min}},\ P \cap path(c_{u,j}) = \emptyset\} \tag{13}$$

$path(c_{u,j})$ denotes the path that leads to $c_{u,j}$. Recall that this path can be identified through backtracking. Computing the intersection between two paths can be done in $O(|V|)$ time through hashing.

To summarize, the extended DAG includes paths with backward edges under the constraint that the backward edge does not cause an additional detour beyond ϵ than a path consisting of only forward edges. As shown in Fig. 2(b), more than 85% of real-life taxis do not take any backward edge that results in an extra distance of more than $0.5km$. The online recommendation pipeline is therefore as follows: every time a new order o is picked up by a taxi, we first compute the nearest drop-off location o^{d*} according to the shortest route plan RP^* (Recall Eq. 7). The extended DAG X is constructed with o^s and o^{d*} as the source and destination nodes and the feasible path with the highest number of expected customers is computed.

Time Complexity: As discussed earlier, the construction time of the extended DAG is exponential in ϵ and the time for executing the DP on this DAG is also similarly exponential in ϵ. Consequently, by limiting ϵ to a small value, we obtain a good balance between efficiency and quality.

Space Complexity: Since each backward edge corresponds to a path from the road network, and there could potentially be exponential number of paths, the number of extended edges is also exponential. However, in a practical scenario, under a small ϵ in the range $[0.1km, 0.7km]$, the space consumption is less than 0.4 GB on large metropolitan cities like New York and Singapore.

4.5 Infusing Additional Intelligence

The proposed prediction model is generic enough to accommodate further optimizations. We discuss some possibilities below. These optimizations are not part of SHARE, but may be added if preferred. We empirically evaluate their impact in Sec. 5.6

4.5.1 Non-linear weighting of nodes: The expected number of compatible customers in a path is simply the summation of the expected customers in each of the constituent nodes. This formulation does not incorporate the factor that finding a customer early in a path is much more profitable than in one of the later nodes. This aspect can be modeled by applying a non-linear weighting factor to each node. Specifically, we may modify Eq. 6 as:

$$E\left(P|\mathbb{O}, o^t\right) = \sum_{\forall v_i \in P} E\left(v_i|o^t\right) \cdot w_i \tag{14}$$

where w_i is the weight importance of node v_i. Under an exponential decay function, $w_i = e^{-\lambda \frac{dist_i}{\delta(P)}}$, where $dist_i$ is the distance travelled till node v_i in path P and λ is the decay parameter.

4.5.2 Regional Expectations: The expected number of compatible customers in a node v is derived from the historical pick-up data of v . In reality, a taxi can always deviate slightly from the path, pick up a customer, and then continue on its journey. To model the expected number of compatible customer from the region around a node instead of the node itself, we may modify Eq. 5 as:

$$E(v|\mathbb{O}, o^t) = \frac{1}{Y} \sum_{\forall \mathbb{H}_{D_i} \in \mathbb{H}} \sum_{\forall u \in R(v,\theta)} \left|\mathbb{H}_{D_i,u,\mathbb{O},o^t}\right| \tag{15}$$

where $R(v, \theta) = \{u \in V \mid SP(v, u) \leq \theta\}$ is the set of all nodes reachable within a distance of θ from v.

4.5.3 Incorporating Expected Time: If an order o is picked up by a taxi at time o^t, the expected number of customers at all nodes in a path $P = \{v_1, \cdots, v_n\}$ is computed conditioned on o^t. In reality, the taxi reaches any node $v_i \in P$ some time after o^t. To reflect the time component more accurately, we can compute the average commute time t_e for each edge e in the network. Next, one may modify Eq. 6 as follows to obtain better estimates.

$$E\left(P|\mathbb{O}, o^t\right) = \sum_{\forall v_i \in P} E\left(v_i \mid (o^t + time(P, v_i))\right) \tag{16}$$

where $time(P, v_i) = \sum_{j=1}^{i-1} t_{(v_j, v_{j+1})}$

5 EXPERIMENTS

In this section, we benchmark SHARE and establish that

- **Quality:** The path recommended by SHARE is better than the shortest path across a range of practical metrics.
- **Route Planning:** Using SHARE as the base approach, instead of shortest paths, reduces ride-sharing failure rates by up to 40% in the route planning problem for ride-sharing [42].
- **Efficiency:** SHARE consumes only ≈ 0.2 seconds to answer recommendation queries, and is therefore fast-enough for real-time recommendations on large metropolitan cities.

The code and datasets used can be downloaded from https://github.com/Ride-SHARE/SHARE.

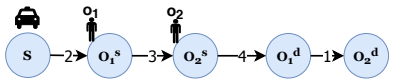


Figure 4: Illustrates the Passengers per km metric.

5.1 Experimental Setup

All experiments are implemented in Java and performed on a machine with Intel(R) Xeon(R) CPU 2.50GHz E5-2680 with 32GB RAM on Red Hat Enterprise Linux Server release 6.9.

5.1.1 Datasets. Table 2 summarizes the real taxi datasets used for our empirical evaluation. The road networks of all three cities have been obtained from OpenStreetMap [29].

•**New York [8]:** This dataset contains records of yellow and green taxis in the city of New York. Each record includes fields capturing pick-up and drop-off dates/times, locations and trip distances. From this dataset, we extract all pick-ups for yellow taxis in January, 2013 and construct our dataset.

•**Singapore [38]:** This dataset contains records of taxis from Singapore. Each record includes a series of tuples of the form $\langle Latitude, Longitude, TaxiID, DriverID, TaxiStatus\rangle$, which are recorded at 1-minute intervals. The $TaxiStatus$ field indicates whether the taxi is occupied or idling. We extract all pick-ups in January, 2012.

•**San Francisco [31]:** The dataset has been collected over a duration of one month and contains the trajectories and taxi pick-up and drop-off information from taxis in San Francisco.

In all these datasets, the capacity of a taxi is not mentioned. We assume it to be 3 passengers.

5.1.2 Evaluation Framework: To evaluate the performance of Share, we partition each taxi pick-up dataset into two portions: the training set and the testing set. The recommendation model is learned from the training set and evaluated on the testing set. The training set contains a random 80% of all orders and remaining portion is assigned to test set. All taxis in the test set are initially randomly positioned in the road network. Next, each order $o = \langle o^s, o^d, o^t\rangle$ from the test set is picked sequentially according to o^t and added to a queue of pending orders. A pending order o is removed from the queue when a compatible taxi, i.e., a taxi servicing order set $\mathbb{O}$ such that $compatible(\mathbb{O}, o) = 1$ (Recall Eq. 2), $|\mathbb{O}| \geq 1$ passes through o^s. If it is detected that no compatible taxi can reach o^s within 5 minutes, then the closest empty taxi to o is assigned. To compute the expected time to reach a particular location, we use the mechanism described in Sec. 4.5.3.

For fast computation of shortest path distances, the road networks are indexed using the *two-hop* index structure[16].

5.1.3 Baselines: As discussed in Sec. 1, existing techniques for taxi ride-sharing primarily look into either route planning or taxi assignment. In both these problems, there is no element of prediction. Thus, a direct comparison is not possible. Therefore, We consider the following baselines.

Table 2: Summary of the datasets used

City	# nodes	# edges	# ride-orders
New York	61,298	141,373	14.77 million
Singapore	52,653	86,410	12.57 million
San Francisco	3,527	7,964	0.372 million

•**Shortest Path:** Most major app-based taxi services such as Uber and Didi Chuxing move according to the shortest route plan [37, 42]. We therefore compare the performance of Share with shortest path.

•**Route Planning [42]:** Our goal in this comparison is to show that route planning algorithms can be improved by assuming the taxi moves along the path recommended by Share instead of the shortest path. Towards that end, we compute route based on Share and compare the performance with the state-of-the-art *Greedy* approach by Zheng et al. [42].

•**Optimal path:** Since the route recommendation problem is NP-hard, the extended DAG is used to approximate the optimal solution. To understand the approximation quality, we compare the path output by Share with the optimal path. We perform this evaluation only on the smaller San Francisco dataset since finding the optimal path has exponential computation cost.

5.1.4 Metrics: We use the following well-established metrics to quantify the performance of Share and other baselines.

Percentage of orders without ride-sharing [1]: Whenever a taxi is unable to find a compatible co-passenger for a ride-order, the taxi operator incurs a financial loss. Thus, we measure the percentage of ride-orders without any co-passenger. The lower the value, the better is the performance.

Passengers per kilometer (km) [9]: This metric measures the mean occupancy level per km. More specifically, let D_k be the total distance covered by all taxis with k passengers. Then the average passengers per km is $\frac{\Sigma_{\forall k} kD_k}{\Sigma_k D_k}$. A simple example is illustrated in Fig. 4. An empty taxi starts at S. Then, it picks up passengers at o_1^s and o_2^s after travelling 2km and 5km respectively, and finally, drops them off at o_1^d and o_2^d. In this scenario, the average number of passengers per km is $\frac{0\times2+1\times3+2\times4+1\times1}{2+3+4+1} = 1.2$. Clearly, the higher the value, the better is the performance.

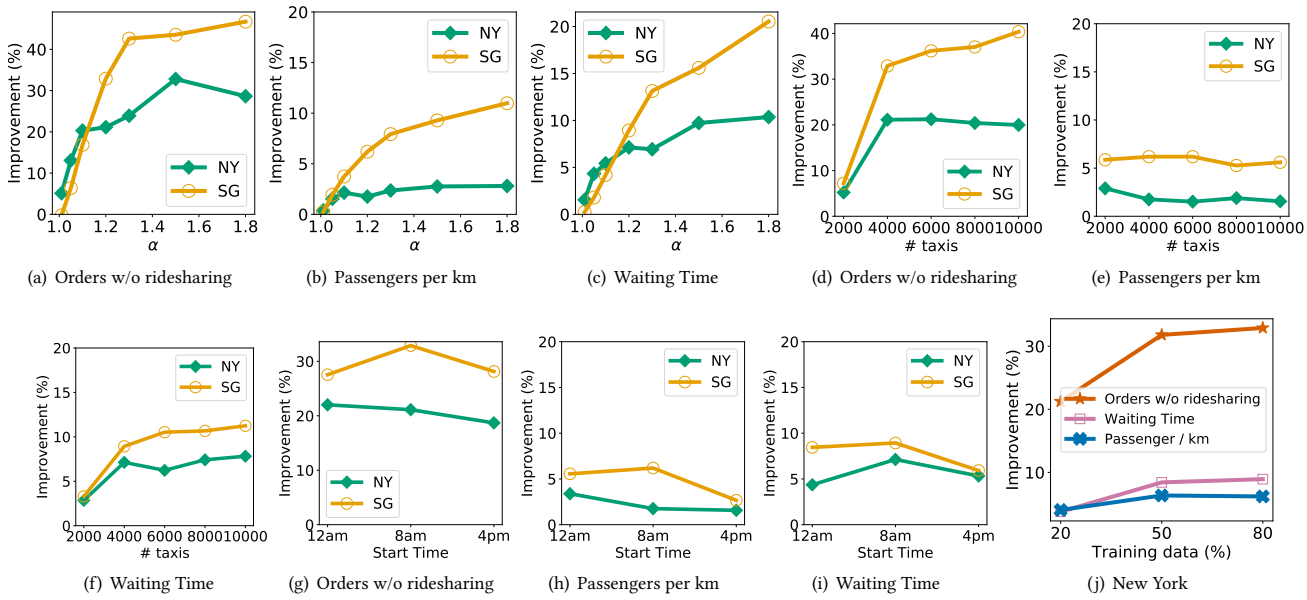
Average waiting time [1]: Average waiting time measures the average number of minutes between an order being placed and a taxi reaching the pick-up location.

Rejection rate [42]: Rejection rate measures the number of ride-orders who remain in the queue of pending orders for more than a certain time threshold. Under a 15 minute time threshold, which is often used [42], we observe a rejection rate of less than 5% for both Share as well as the other baseline algorithms. Hence, we do not present results with respect to rejection rate.

5.1.5 Parameters: Share requires two key parameters, the extension threshold ϵ to construct the extended DAG, the number of bins Ω for discretization in the dynamic programming. Unless specifically mentioned, their default values are set to $\epsilon = 0.5km$ and $\Omega = 100$. Nonetheless, we also include experiments where we vary them explicitly and study their impact on the performance of Share.

5.2 Route Recommendations

First, we analyze the quality of the recommended routes. The performance of Share is compared with the shortest path in terms of the *Improvement (%)* over each metric in Sec. 5.1.4. Intuitively, *Improvement* measures the percentage by which Share improves upon shortest path. A higher *Improvement* signifies a better performance of Share. Mathematically, let $SHARE_M$ and SP_M be the values obtained by Share and shortest path respectively in metric



(a) Orders w/o ridesharing (b) Passengers per km (c) Waiting Time (d) Orders w/o ridesharing (e) Passengers per km

(f) Waiting Time (g) Orders w/o ridesharing (h) Passengers per km (i) Waiting Time (j) New York

Figure 5: Improvement of SHARE over Shortest Path (SP) against (a-c) α, (d-f) number of taxis and (g-i) time slot of the day (j) training dataset size.

M. The *Improvement* in M is

$$Improvement\ (\%) = \frac{SP_M - SHARE_M}{SHARE_M} \times 100^1 \tag{17}$$

Figs. 5(a)-5(c) present the results as the detour parameter α is varied. Across all three metrics, SHARE improves upon shortest path. The improvement is most stark in the reduction of orders without ride-sharing (Fig. 5(a)); in New York (NY) SHARE is up to 30% better, while in Singapore (SG) the reduction is as high as 42%. This result highlights that SHARE is indeed able to learn from the historical data and recommend paths that lead to more ride-sharing. In contrast, in shortest paths, taxis solely depend on their luck since the route selection is done purely based on distance.

In its effort to maximize the number of expected customers, SHARE may recommend slightly longer routes. It is therefore important to ensure that this extra distance adds value to the taxi operators. The second metric, which compares the distance travelled per customer, evaluates this aspect. Fig. 5(b) reveals that SHARE leads to more customer density per km.

The third dimension that we measure is the amount time customers spend waiting for the taxis to arrive at their pick-up locations. We expect the waiting time to reduce in SHARE since we predict the locations where more requests are likely to originate from, and accordingly select routes that provides strategic advantage to taxis. This behavior is prominently visible in Fig. 5(c), where the reduction is as high as 20% in SG.

Two key patterns emerge from the results in Figs. 5(a)-5(c). First, the improvements are consistently higher in SG than in NY. A deeper look into the data reveals that the number of ride-orders is higher in NY and more importantly, highly concentrated in the smaller geographical area of Manhattan. In contrast, the orders are much more spread out in SG. When the density of ride-orders is very high in a particular region, like in Manhattan, any route has almost equal chance of finding compatible passengers. Thus, the improvement of SHARE in NY over shortest paths is not as prominent as in SG.

The second pattern that emerges from Figs. 5(a)-5(c) is that the improvement in general increases with α. Naturally, with higher α, the search space increases and SHARE has more scope to improve upon shortest paths.

Next, we benchmark the performance of SHARE as the number of taxis is explicitly varied. In this experiment, we randomly position C taxis in the city and initiate the simulation on the test set. Next, C is varied on the x-axis and *Improvement* is measured. We set $\alpha = 1.2$ in this experiment. As visible in Figs. 5(d)-5(f), the results follow a similar trend as earlier: SHARE is consistently better than shortest path and the improvement is higher in SG. As number of taxis increases, the improvement in orders without ridesharing and waiting time increases. When the number of taxis is small, the ratio of orders to taxis is high and hence there is a higher chance that a taxi would find a customer. But when the number of taxis increases, the taxis compete among each other for the same passenger. Hence route recommendation becomes important for a taxi to travel on a route which has a higher probability of picking up passengers. Thus, the improvement of SHARE over shortest paths increases.

In the next experiment, we analyze the performance at three different time slots of the day: 12AM-1:30AM (least busy), 8AM-9:30AM (office going traffic), and 4PM-5:30PM (home-bound). As expected, SHARE consistently outperforms shortest path. While the improvement of SHARE is relatively stable across different time slots of the day, a slight dip is observed in the last time slot of 4PM-5:30PM. This slot has the highest customer density and as we observed earlier, the competitive advantage of SHARE decreases when the ratio of orders to number of taxis is higher.

[1]For the Passengers per km metric, a higher value is better. Thus, the numerator is changed to $SHARE_M - SP_M$

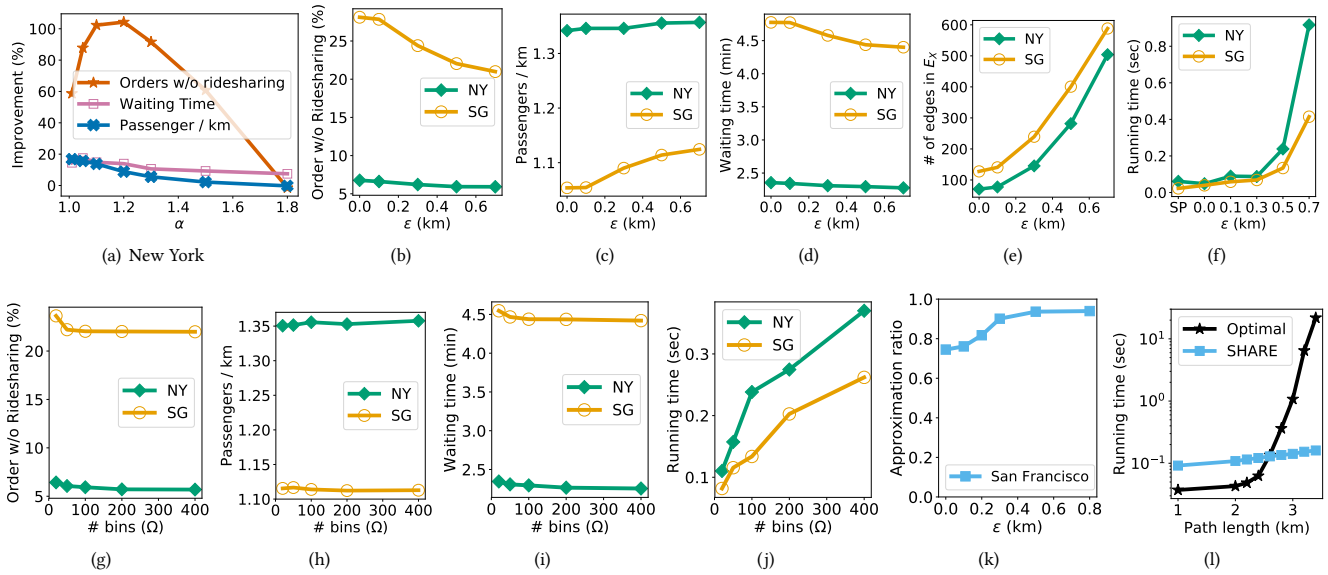


Figure 6: (a) Performance comparison of SHARE with the algorithm proposed by Zheng et al. [42]. Impact of extension threshold on (b-d) quality, (e) size of extended DAG, and (f) running time. Impact of number of bins on (g-i) quality and (j) running time. Comparison of SHARE with the optimal algorithm in term of (k) quality and (l) running time.

Finally, we analyze the performance of SHARE against the size of the training dataset in Fig. 5(j). As expected, the performance improvement begins to saturate as the training dataset size increases.

5.3 Taxi Assignment and Route Planning

The entire ride-sharing pipeline can be divided into three key stages: assigning taxis to orders, constructing the route plan that decides the sequence of order pick-ups and drop-offs, and finally, selecting the routes between each of the order pick-up and drop-off locations as decided in the route plan. While SHARE directly targets the third stage of the ride-sharing pipeline, we show that SHARE also makes the task of the other two stages easier.

In route planning, given the current positions of all taxis, and the pending order requests, the task is to dispatch the order into an optimal route plan of a taxi. The algorithm must ensure all constraints are met such as respecting the taxi capacity and detour ratio. While we allot the closest empty compatible taxi to a pending ride-order, in which only simple taxi assignment and no route planning is involved, more sophisticated algorithms have been proposed [42]. In this experiment, we compare the performance of SHARE with the state-of-the-art algorithm by Zheng et al. [42], where the assumption is that taxis always move along the shortest path.

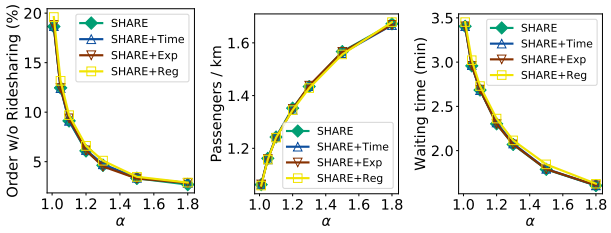
Fig. 6(a) presents the results. As can be seen, SHARE has better performance across all three metrics. Note that the improvement is more prominently visible at lower values of α, which is the detour limit. When the detour limit is low, a taxi does not have much flexibility to re-route and pick up a passenger that is not already close to its current position. With SHARE, a taxi is more likely to be close to a new order since SHARE strategically chooses the routes based on prior historical patterns rather than travels through the shortest path. Consequently, the percentage of ride-sharing improves, waiting time decreases and passengers per km goes up, when compared to the shortest path route in conjunction with the route planning algorithm of Zheng et al. [42].

We also note that while the improvement of SHARE increases with α in Figs. 5(a)-5(c), in Fig. 6(a) the improvement decreases. This follows from the fact that in the route planning experiment, a taxi is allowed to re-route and pick up a pending order that is not directly on its path, but can be serviced without violating the detour limit of α. In the previous experiments, an order is serviced only by taxis that goes directly through the pick-up location. All in all, the results reveal that regardless of the model used, SHARE leads to substantially improved performance over shortest paths.

5.4 Impact of Input Parameters

Figs. 6(b)-6(d) present the impact of ϵ on the quality of SHARE as measured by the three metrics. We observe an improvement across all three metrics with higher values of ϵ. This is natural since the search space increases with increase in ϵ and this property is evident from Fig. 6(e), where we measure the number of edges in the extended DAG against ϵ. This improvement in quality, however, comes at the cost of higher running time since more paths need to be processed. Specifically, the number of edges increases exponentially with ϵ, which leads to an exponential increase in running time as well (Fig. 6(f)). Given all these factors, $\epsilon = 0.5km$ is suggested as the optimal choice since the quality saturates beyond this point, while the running time increases exponentially. Fig. 6(f) also shows that at $\epsilon = 0.5$, recommendations can be made within ≈ 0.2 seconds. This result substantiates our claim that SHARE is effective and fast enough to handle the load of any metropolitan city.

Next, we analyze the impact of number of bins, Ω, on the quality and running time. Figs. 6(g)-6(i) show that the quality remains stable against Ω. The running time, however, increases with number of bins, and thereby, justifying the choice of $\Omega = 100$.



(a) New York (b) New York (c) New York

Figure 7: Impact of additional intelligence (Time = Incorporate expected time, Exp = Exponential decay, Reg = Regional expectations).

5.5 Approximation Quality

SHARE approximates the longest path problem by searching only within the extended DAG. To diagnose the approximation quality, we plot the approximation ratio against the extension threshold ϵ. The approximation ratio is quantified as $\frac{w_{SHARE}}{w^*}$, where w^* and w_{SHARE} are the expected number of compatible passengers on the optimal path and the path recommended by SHARE respectively. Since computing w^* is NP-hard, we perform this analysis on the smaller San Francisco dataset. Fig. 6(k) shows that at $\epsilon = 0.5$, the approximation quality is near-optimal with a ratio larger than 0.95.

The utility of the extended DAG is further highlighted by the running time analysis in Fig. 6(l). In this plot, we study the running times of the optimal algorithm and SHARE as a function of the distance between the source and destination nodes. As visible, SHARE is more than two orders of magnitude faster at $\epsilon = 0.5$, while providing near-optimal quality.

5.6 Impact of Additional Intelligence

In Sec. 4.5, we discussed some strategies to further improve the prediction model. Figs. 7(a)-7(c) present the impact of these various strategies. In these figures, SHARE+Time, SHARE+Exp, and SHARE+Reg correspond to Sec 4.5.3, Sec. 4.5.1, and Sec. 4.5.2 respectively. In our experiments, we do not observe a noticeable improvement with any of these strategies with SHARE+Reg giving marginally improved performance, particularly in the metric of Passengers per km.

6 RELATED WORK

Historical trajectory data has been used for an array of smart-city applications such as anomaly detection[2], infrastructure improvement [20, 26–28] and community detection [34, 35]. In this work, we use historical data to recommend routes to partially occupied taxis such that their chances of finding ride-sharing passengers improve. Existing work on ride-sharing has focused on primarily two aspects: *taxi assignment* and *route planning*.

The taxi assignment approach works to optimize metrics such as profit, detour, etc. by assigning the right taxi with spare capacity to compatible orders [10, 15, 24, 25, 41]. Most recently, Ta et. al. work with a typical solution paradigm for this approach: they solve the assignment problem using an algorithm based on bipartite matching [36]. The focus on assignment makes the ride-sharing problem basically a generalization of the classical *dial-a-ride problem (DARP)* [5]One key difference of DARP with taxi ride-sharing is that in the ride-sharing partially occupied taxis can also be assigned to fulfill orders. The route planning approach assumes that a number of orders and taxis are available and it seeks to dispatch taxis to orders in a sequence computed such that the taxis can fulfill the orders while optimizing the appropriate metrics [37, 42]. This approach is a contemporary version of the classical vehicle routing problem [21] and typically involves solving a variant of the travelling salesman problem. Zheng et. al. apply a greedy method to overcoming this difficulty in [42], whereas Tong et. al. take a dynamic programming approach [37]. In the same line is the work by Cheng et. al. [4], which adds the dimension of rider utility and social connections. Some works take a hybrid approach of choosing an assignment and then repositioning empty taxis after order drop offs [1]. All these works, however, share one feature: An order once picked up is dropped off by following the shortest path between pick-up point and destination. And this is where our approach diverges.

Our method seeks to recommend good routes based on historical data. Historical data has been used to address various other problems related to taxis in the literature [13, 40, 41]. We refer the reader to den Hoedt's Masters thesis for a fuller discussion [7].

Our problem is a variant of the NP-hard longest path problem which cannot be approximated to with a factor of $\exp(\theta(\log^{1-\epsilon} n))$ for any $\epsilon > 0$ and a graph with n nodes [17]. So, we turn to heuristics without good approximation guarantees. Our basic optimization problem most resembles the orienteering problem which is a variant of the longest path that seeks to collect the maximum benefit on a graph using a path of given length [14]. Given the approximability constraints of the problem research on this problem has focused on meta-heuristics that work for small instances [3, 19, 25]. A recent work in this line also uses a dynamic programming approach like we do, but the formulation and approach is very different [25].

In the domain of route recommendations, it has been studied for both the cab service industry [12, 38] as well as itinerary recommendations [18, 22, 23, 30, 39]. These techniques, however, do not apply to our problem due to the recommendation setting being different.

7 CONCLUSION

In this paper, we established that it is indeed possible to learn from historical data and recommend routes for ride-sharing in taxis that significantly enhances the likelihood of finding more customers.The key weakness of the current taxi-sharing paradigm is that discounted fares are offered to promote sharing, which in turn, leads to a risk of revenue loss if sufficient number of orders are not allocated for ride-sharing. Till now, no effort has been made on the routing front to mitigate this risk, a shortcoming that we have endeavoured to address. Our route recommendation algorithm, SHARE, reduces failure rates in finding ride-sharing partners by as much as 40% when compared to the shortest paths used as default currently. In addition, it improves mean occupancy level and waiting times for customers by up to 20%. These excellent results are obtained by reducing the route recommendation problem to the classical problem of finding the *longest path* in a network. SHARE overcomes the scalability challenges posed by this NP-hard problem through an approximation algorithm based on *dynamic programming*. Extensive evaluation on real datasets demonstrate that SHARE provides near-optimal accuracy, while limiting the running time to within 0.2 seconds and thereby making it deployable on any major metropolitan city.

REFERENCES

[1] Javier Alonso-Mora, Samitha Samaranayake, Alex Wallar, Emilio Frazzoli, and Daniela Rus. 2017. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proc. Nat. Acad. Sci. United States* 114, 3 (2017), 462–467.

[2] Prithu Banerjee, Pranali Yawalkar, and Sayan Ranu. 2016. Mantra: a scalable approach to mining temporally anomalous sub-trajectories. In *ACM SIGKDD*. 1415–1424.

[3] I-Ming Chao, Bruce L. Golden, and Edward A. Wasil. 1996. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research* 88, 3 (1996), 475 – 489.

[4] P. Cheng, H. Xin, and L. Chen. 2017. Utility-Aware Ridesharing on Road Networks. In *Proc. SIGMOD '17*. 1197–1210.

[5] Jean-François Cordeau and Gilbert Laporte. 2007. The dial-a-ride problem (DARP): Models and algorithms. *Annals OR* 153 (06 2007), 29–46.

[6] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms, Third Edition* (3rd ed.). The MIT Press.

[7] Martijn den Hoedt. 2016. *Taxi ride scheduling and pricing using historical data*. Master's thesis. Delft University of Technology.

[8] Dan Donovan, Brian; Work. 2016. New York City Taxi Trip Data (2010-2013). https://doi.org/10.13012/J8PN93H8.

[9] P. M. d'Orey, R. Fernandes, and M. Ferreira. 2012. Empirical evaluation of a dynamic and distributed taxi-sharing system. In *15th Intl. IEEE Conf. Intelligent Transportation Systems*. 140–146.

[10] Esteban Feuerstein and Leen Stougie. 2001. On-line single-server dial-a-ride problems. *Theoretical Computer Science* 268, 1 (2001), 91 – 105. https://doi.org/10.1016/S0304-3975(00)00261-9 On-line Algorithms '98.

[11] Koen Frenken and Juliet Schor. 2017. Putting the sharing economy into perspective. *Environmental Innovation and Societal Transitions* 23 (2017), 3 – 10. Sustainability Perspectives on the Sharing Economy.

[12] Nandani Garg and Sayan Ranu. 2018. Route Recommendations for Idle Taxi Drivers: Find Me the Shortest Route to a Customer!. In *ACM SIGKDD*. ACM, 1425–1434.

[13] Yong Ge, Hui Xiong, Alexander Tuzhilin, Keli Xiao, Marco Gruteser, and Michael Pazzani. 2010. An Energy-efficient Mobile Recommender System. In *Proc. 16th ACM Intl. Conf. Knowledge Discovery and Data Mining (KDD '10)*. ACM, New York, NY, USA, 899–908.

[14] Aldy Gunawan, Hoong Chuin Lau, and Pieter Vansteenwegen. 2016. Orienteering Problem: A survey of recent variants, solution approaches and applications. *European Journal of Operational Research* 255, 2 (2016), 315 – 332.

[15] Yan Huang, Favyen Bastani, Ruoming Jin, and Xiaoyang Sean Wang. 2014. Large Scale Real-time Ridesharing with Service Guarantee on Road Networks. *Proc. VLDB Endow.* 7, 14 (Oct. 2014), 2017–2028.

[16] Minhao Jiang, Ada Wai-Chee Fu, Raymond Chi-Wing Wong, James Cheng, and Yanyan Xu. 2014. Hop Doubling Label Indexing for Point-to-Point Distance Querying on Scale-Free Networks. *CoRR* abs/1403.0779 (2014). arXiv:1403.0779 http://arxiv.org/abs/1403.0779

[17] D. Karger, R. Motwani, and G. D. S. Ramkumar. 1997. On approximating the longest path in a graph. *Algorithmica* 18 (1997), 82–98.

[18] Arzoo Katiyar, Arnab Bhattacharya, and Shubhadip Mitra. 2014. Efficient and Effective Route Planning in Road Networks with Probabilistic Data Using Skyline Paths. In *IKDD*. 3:1–3:10.

[19] Liangjun Ke, Claudia Archetti, and Zuren Feng. 2008. Ants Can Solve the Team Orienteering Problem. *Comput. Ind. Eng.* 54, 3 (April 2008), 648–665.

[20] Vinay Kolar, Sayan Ranu, Anand Prabhu Subramainan, Yedendra Shrinivasan, Aditya Telang, Ravi Kokku, and Sriram Raghavan. 2014. People In Motion: Spatio-temporal Analytics on Call Detail Records. In *COMSNETS*. 1–4.

[21] G. Laporte. 1992. The vehicle routing problem: An overview of exact and approximate algorithms. *Eur. J. Operations Research* 59, 3 (1992), 345–358.

[22] Wengen Li, Jiannong Cao, Jihong Guan, and Shuigeng Zhou. 2016. Retrieving Routes of Interest Over Road Networks. In *Web-Age Information Management*. 109–123.

[23] Yujiao Li, Weidong Yang, Wu Dan, and Zhipeng Xie. 2015. Keyword-Aware Dominant Route Search for Various User Preferences. In *Database Systems for Advanced Applications*. 207–222.

[24] S. Ma, Y. Zheng, and O. Wolfson. 2013. T-share: A large-scale dynamic taxi ridesharing service. In *29th Intl. IEEE Conf. Data Engineering (ICDE '13)*. 410–421.

[25] Zhibei Ma, Kai Yin, Lantao Liu, and Gaurav S. Sukhatme. 2017. A spatio-temporal representation for the orienteering problem with time-varying profits. In *Proc. IEEE/RSJ Intl. Conf. Intelligent Robots and Systems (IROS '17)*. 6785–6792.

[26] Sourav Medya, Sayan Ranu, Jithin Vachery, and Ambuj Singh. 2018. Noticeable network delay minimization via node upgrades. *Proceedings of the VLDB Endowment* 11, 9 (2018), 988–1001.

[27] Shubhadip Mitra, Sayan Ranu, Vinay Kolar, Aditya Telang, Arnab Bhattacharya, Ravi Kokku, and Sriram Raghavan. 2015. Trajectory aware macro-cell planning for mobile users. In *INFOCOM*. 792–800.

[28] Shubhadip Mitra, Priya Saraf, Richa Sharma, Arnab Bhattacharya, Sayan Ranu, and Harsh Bhandari. 2017. NetClus: A scalable framework for locating top-k sites for placement of trajectory-aware services. In *ICDE*. 87–90.

[29] OpenStreetMap contributors. 2017. Planet dump retrieved from https://planet.osm.org . https://www.openstreetmap.org.

[30] Shiladitya Pande, Sayan Ranu, and Arnab Bhattacharya. 2017. SkyGraph: retrieving regions of interest using skyline subgraph queries. *Proceedings of the VLDB Endowment* 10, 11 (2017), 1382–1393.

[31] Michal Piorkowski, Natasa Sarafijanovic-Djukic, and Matthias Grossglauser. 2009. CRAWDAD dataset epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.org/epfl/mobility/20090224. https://doi.org/10.15783/C7J010

[32] P. Santi, G. Resta, M. Szell, S. Sobolevsky, S. H. Strogatz, and C. Ratti. 2014. Quantifying the benefits of vehicle pooling with shareability networks. *Proc. Nat. Acad. Sci. US* 111, 37 (2014), 13290–13294.

[33] R. Sedgewick and K. D. Wayne. 2011. *Algorithms* (4th ed.). Addison Wesley Professional.

[34] Jieming Shi, Nikos Mamoulis, Dingming Wu, and David W Cheung. 2014. Density-based place clustering in geo-social networks. In *SIGMOD*. 99–110.

[35] Shivam Srivastava, Shiladitya Pande, and Sayan Ranu. 2015. Geo-social clustering of places from check-in data. In *ICDM*. 985–990.

[36] Na Ta, Guoliang Li, Tianyu Zhao, Jianhua Feng, Hanchao Ma, and Zhiguo Gong. 2018. An Efficient Ride-Sharing Framework for Maximizing Shared Route. *IEEE T. Knowl Data Eng* 30, 2 (February 2018), 219–233.

[37] Yongxin Tong, Yuxiang Zeng, Zimu Zhou, Lei Chen, Jieping Ye, and Ke Xu. 2018. A Unified Approach to Route Planning for Shared Mobility. *Proc. VLDB Endow.* 11, 11 (July 2018), 1633–1646.

[38] Tanvi Verma, Pradeep Varakantham, Sarit Kraus, and Hoong Chuin Lau. 2017. Augmenting Decisions of Taxi Drivers through Reinforcement Learning for Improving Revenues. In *ICAPS*, Vol. 27. 409–417.

[39] Pranali Yawalkar and Sayan Ranu. 2019. Route Recommendations on Road Networks for Arbitrary User Preference Functions. In *ICDE*.

[40] Nicholas Jing Yuan, Yu Zheng, Liuhang Zhang, and Xing Xie. 2012. T-Finder: A Recommender System for Finding Passengers and Vacant Taxis. *IEEE T. Knowl. Data Eng* 25, 10 (September 2012), 2390–2403.

[41] Mingyue Zhang, Jianxun Liu, Yizhi Liu, Zhenyang Hu, and Liang Yi. 2012. Recommending Pick-up Points for Taxi-drivers Based on Spatio-temporal Clustering. In *Proc. 2nd Intl. Conf. Cloud and Green Computing (CGC '12)*. IEEE Computer Society, Washington, DC, USA, 67–72.

[42] Libin Zheng, Lei Chen, and Jieping Ye. 2018. Order dispatch in price-aware ridesharing. *Proc. VLDB Endow.* 11, 8 (April 2018), 853–865.