Demand-oriented Route Recommendation for Shared Mobility Services

Zhijia Chen[1], Chen Jason Zhang[2], Rui Meng[3], Peng Cheng[4], Libin Zheng[1(✉)], and Jian Yin[1]

[1] The Key Laboratory of Intelligent Assessment Technology for Sustainable Tourism, Ministry of Culture and Tourism, Sun Yat-sen University
chenzhj85@mail2.sysu.edu.cn, {zhenglb6, issjyin}@mail.sysu.edu.cn
[2] The Hong Kong Polytechnic University jason-c.zhang@polyu.edu.hk
[3] BNU-HKBU United International College ruimeng@uic.edu.cn
[4] East China Normal University pcheng@sei.ecnu.edu.cn

Abstract. In the era of the sharing economy, ridesharing has emerged as an efficient and environmentally friendly transportation mode, with route recommendation playing a vital role. Traditional methods primarily rely on shortest-path strategies for route planning, which often overlook opportunities for drivers to slightly extend their routes to capture additional requests. To address this limitation, we introduce the Demand-aware Order Dispatch with Route Recommendation (DODRE) problem, aiming to recommend routes that pass through high-demand areas and improve the overall request fulfillment rate. To solve this problem, we propose a Regional Demand-aware Route Search method, which identifies routes with the highest demand potential by considering future demand distributions across regions. This demand-aware route planning is integrated with an insertion-based allocation strategy, ensuring efficient and effective order dispatch. Extensive experiments on real-world datasets demonstrate that our approach significantly enhances service rates and outperforms traditional shortest-path-based methods in both effectiveness and computational efficiency, highlighting its potential to optimize resource utilization in ridesharing systems.

Keywords: Ridesharing · Route Recommendation.

1 INTRODUCTION

Since the advent of the sharing economy[7], various services have proliferated rapidly, with ridesharing representing a critical component of shared mobility. It is regarded as an efficient and sustainable mode of transportation. Platforms facilitating such services, such as Didi, Lyft, and Uber, play pivotal roles in mediating between drivers and ride requests, optimizing the allocation of ride requests to suitable drivers. From the perspective of passengers, the cost of trips is reduced through shared riding. For platform operators, utilizing a limited fleet to accommodate a larger number of ride requests via ridesharing can lead to increased profitability.

To improve ridesharing efficiency, route recommendation is crucial. After a platform assigns ride requests, the driver follows the suggested route to pick up and drop off passengers. Current research, focused on maximizing ride requests served [15, 4] or enhancing platform revenue [2, 14], typically uses shortest-path strategies [1]. However, this greedy approach prioritizes current requests and overlooks future demand from other regions. We propose a strategy that accounts for upcoming requests, offering drivers routes focused on long-term benefits and improving overall platform performance. Figure 1 clearly illustrates that by taking the shortest path $A \rightarrow B \rightarrow E \rightarrow I \rightarrow J$, the driver d_1 can only serve request r_1, whereas by opting for a slightly longer path $A \rightarrow B \rightarrow C \rightarrow D \rightarrow G \rightarrow J \rightarrow I$, the driver can serve both orders r_1 and r_2.

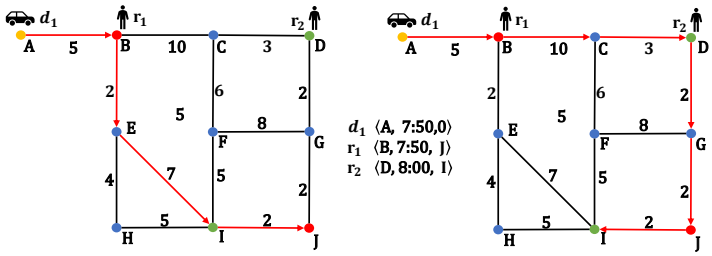

Fig. 1. An Illustrative Example

Therefore, we define a new problem, namely the Demand-aware Order Dispatch with Route Recommendation (DODRE) problem. Specifically, DODRE tackles the challenge of optimizing drivers' routes in ridesharing by selecting the most suitable paths for them, considering the upcoming requests in the near future.

To address the DODRE problem, we propose an innovative route search strategy known as the Regional Demand-Aware Route Search. This strategy is designed to identify one or more optimal routes while considering the predicted distribution of demand across different regions. The strategy aims to achieve two key objectives. First, the recommended routes should be as close as possible to the shortest path to ensure an optimisitic user experience and minimize travel time. Second, the routes should enhance the rate of ride-sharing per journey, thereby increasing both the revenue of platforms and drivers, and improving the overall service rate.The key contributions of our work are as follows:

- We introduce the Demand-aware Order Dispatch with Route Recommendation (DODRE) problem (Section 2).
- We propose a regional demand-aware route search strategy, leveraging predicted regional demand to guide route recommendations (Section 3).
- Extensive experiments conducted on real-world request datasets from New York and Singapore demonstrate the efficiency and effectiveness of the pro-

posed algorithm, showing 5.98% $\sim$ 342.24% improvements in term of number of served requests (Section 4).

2 PROBLEM DEFINITION

In the following, we define the preliminary concepts for the studied route planning and request dispatch in shared mobility applications, including road networks, drivers, requests, and etc. These concepts follow the common formulations in existing works[13, 12, 11].

Definition 1 *(Road Network) A road network is modeled as a directed graph $G = (V, E, dis)$, where V represents nodes (geographical locations), E represents edges (road segments), and $dis : E \to \mathbb{R}$ assigns a length (weight) to each edge.*

The notation $e = (u, v)$ represents a road segment from node u to v, with its length $dis(e)$ as the geographical distance between them. A route is a simple path in the network, meaning it has no cycles. The length of a path $P = \{e_1, \cdots, e_l\}$, consisting of l edges, is $dis(P) = \sum_{e_i \in P} dis(e_i)$. The shortest path length from u to v is denoted as $L(u, v)$.

Definition 2 *(Drivers) Let $D = \{d_1, d_2, \ldots, d_n\}$ be a set of n drivers, where each driver d_i is represented by $\langle l_i, cap_i \rangle$, with l_i as the current location and cap_i as the maximum number of requests d_i can handle.*

Definition 3 *(Requests) Let $R = \{r_1, r_2, r_3, \cdots, r_m\}$ be a set of m requests. Each request r_j can be represented as $\langle s_j, e_j, tc_j, te_j, dt_j \rangle$ consisting of source point s_j, end point e_j, along with the creation time tc_j and the deadline for reaching the destination te_j, and dt_j denotes the maximum allowable travel distance from s_j to e_j.*

Each request r_j is associated with a distinct dt_j. To enhance the representation of dt_j , we define the detour ratio μ for its calculation. The detour ratio is defined as $\mu = \frac{dt_j}{L(s_j, e_j)}$. Typically, μ is set consistently by the platform.

We also denote all the requests served by driver d_i as R_{d_i}. Furthermore, $\tilde{R} = \bigcup_{d_i \in D} R_{d_i}$ and$\hat{R} = R \setminus \tilde{R}$ refers to the total served and un-served requests, respectively.

We define the Demand-aware Order Dispatch with Route Recommendation problem.

Definition 4 *(DODRE Problem) Given a graph G, a driver set D and a request set R, The DODRE problem seeks a set of routes $\mathcal{P} = \{P_{d_1}, P_{d_2}, \cdots, P_{d_{\|D\|}}\}$ to efficiently assgin the requests to the drivers and plan their routes, with the goal of jointly minimizing the platform's service penalty $\sum_{d_i \in D} dis(P_i)$, and the travel cost of drivers $\sum_{r_j \in \hat{R}} p_j$, subjecting to the following constraints:*

- *Path constraint: The distance of the recommended route must not go beyond the maximum allowable detour distance, i.e., $dis(P_{r_j}) <= dt_j$.*

– *Service constraint: Requests must be picked up ahead of dropped off, and the delivery time is ahead of their deadlines, i.e.,* $t_{s_j} < t_{e_j} <= te_j$.
– *Capacity Constraint: The number of requests assigned to a driver must not exceed their capacity at any time.*

In the formulation, the penalty p_j represents the financial loss for un-serving r_j, typically proportional to the distance traveled by the request [13]. We define p_j as $p_o * L(s_j, e_j)$, where p_o is a constant.

Problem Hardness. The problem above is a composition of the route planning and order dispatch problems, either of which is already NP-hard as demonstrated in existing works [15], let alone the composition of both.

3 Demand-aware Route Planning and Order Dispatch

3.1 Demand Heat Map

Definition 5 *(Grid Cell Unit) Let* $U = \{u_{11}, \cdots, u_{xy}\}$ *be a set of grid cell units. Each unit is formed by two components: region* Z_x *and time interval* T_y.

Definition 6 *(Demand Heat Map) Given a grid cell unit set* U *and a prediction model* M*, the demand heat map represents the set of predicted upcoming request counts for each unit* $u_{ab} = \langle Z_a, T_b \rangle$*, based on model* M.

Predicting accurate pickup and drop-off locations and times is challenging due to various uncertainties. Therefore, estimating demand within short time intervals and specific regions is a feasible approach. We use the ST-MGCN model [9] to train and generate demand heat map.

3.2 Reginal Demand-aware route Search

Existing work[1] focuses on the shortest path strategy but overlooks the spatial diversity of comparably short paths, missing the opportunity to select routes that are both short and demand-rich. To address this, we propose generating the top-K shortest paths and selecting the one that passes through the most demand-heavy regions while minimizing additional distance, utilizing Yen's algorithm [16].

Yen's algorithm In Yen's algorithm, Dijkstra's algorithm [6] is first used to find the shortest path $P_1(s,t)$ from source s to destination t. This path is added to a priority queue, which sorts paths by length. The algorithm then iterates, popping a path $P_i(s,t)$ from the queue and mutating it. For each edge (u,u') in $P_i(s,t)$, Dijkstra's algorithm is called again to find the shortest path $P'(u,t)$ from u to t via a different edge (u,u''), avoiding cycles by not using vertices in $P_i(s,u)$. A new candidate path $P''(s,t)$ is formed by concatenating $P_i(s,u)$ and $P'(u,t)$. $P^{'}(u,t), P_i(s,u), u$ and $(u,u^{''})$ are called the branch path, base path, branch node, and branch edge, respectively. All candidate paths are added to the priority queue. The left side of Figure 2 illustrates the above process.

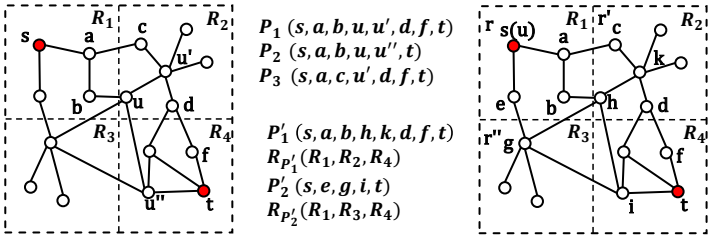

Fig. 2. Left: Yen's Algorithm, Right: our Algorithm

Regional route search Although Yen's method generates three distinct paths, they still cover the same regions, keeping the order probability unchanged and conflicting with our goal. To maximize cumulative demand, we propose a new path similarity metric to identify more effective routes that better reflect regional demand variations.

Definition 7 *(Geographic Path Similarity)The similarity between two paths P_i and P_j, denoted as $GeoSim(P_i, P_j)$, is defined by Equation 1*

Existing work[3, 5] commonly employs the difference in edges between two paths as a measure of similarity. In contrast, given a demand heat map, we define the similarity $GeoSim(P_i, P_j)$ based on the distinct units traversed by the paths.

$$GeoSim(P_i, P_j) = \frac{|r(P_i) \bigcap r(P_j)|}{|r(P_i) \bigcup r(P_j)|} \tag{1}$$

Note that $r(P)$ denotes the set of regions covered by P, with $|r(P_i) \bigcap r(P_j)|$ and $|r(P_i) \bigcup r(P_j)|$ representing the intersection and union, respectively.

According to Definition 7, our search perspective shifts from branch nodes to branch regions, leading to the k-routing search algorithm in Algorithm 1. We start by initializing a priority queue Q and the set K with an initial shortest path (line 1). Then, we iterate while K contains fewer than k paths and Q is not empty (line 2). For each path P', we retrieve its region R_p from the demand map DM (line 3) and explore each region (lines 4-5). If a branch is found (line 6), we calculate new candidate paths P'' by removing edges and ensuring they satisfy the detour ratio constraint (lines 7-11). Once all paths are processed, we check their geo-similarity and insert qualified paths into K (lines 14-16). Finally, the algorithm returns the top-k geo-diverse shortest paths (line 17).An example case is shown on the right side of Figure 2, where $R_{P'_2}$ is generated from $R_{P'_1}$, and P'_2 is created by selecting s as the branch node in R_1

After obtaining k paths, we calculate the demand count $dc_{P_{d_i}}$ for each path using the demand heat map, $\sum_{u \in P_{d_i}} DM(u)$. While we typically recommend the path with the highest demand, this may lead to minimal demand increase

with a significant distance increase. To address this, we use the ratio $\frac{dc_{P_{d_i}}}{dis(P_{d_i})}$, selecting the path with the largest ratio as the recommended route.

It is worth noting that due to the frequent modifications to the graph structure during path search, the Contraction Hierarchy (CH) algorithm [8] is not suitable. However, the A* algorithm [10] and bidirectional Dijkstra algorithm are applicable to dynamic graphs. We utilize the bidirectional Dijkstra algorithm.

Algorithm 1: regional K shortest path search

Input: Graph G, demand heat map DM, source s, target t, detour ratio μ, and ρ
Output: The top-k shortest paths with geo-diversity $\mathbf{K}$

$\mathbf{Q} \leftarrow PriorityQueue$; $P \leftarrow ShortestPath(G, s, t)$; $\mathbf{K} \leftarrow \{\mathbf{P}\}$;
while $|\mathbf{K}| < k$ *and* $!\mathbf{Q}.isEmpty()$ **do**
  $P' \leftarrow \mathbf{K}[-1]$; $R_{P'} \leftarrow getRegion(DM, P')$;
  while $!R_{P'}.isEmpty()$ **do**
    $r = R_{P'}.pop(0)$;
    if (r, r'') *found in* DM **then**
      $n \leftarrow BranchNode(r)$;
      $G' \leftarrow RemoveE(G, r, r')$; $G'' \leftarrow RemoveE(G', P'(s, n))$;
      $P'' \leftarrow P'(s, n) + ShortestPath(G'', n, t)$;
      if $dis(P'') < (1 + \mu) * dis(P)$ **then**
        insert P'' in $\mathbf{Q}$;
  while $!\mathbf{Q}.isEmpty()$ **do**
    $p \leftarrow \mathbf{Q}.pop(0)$;
    if $\forall p' \in \mathbf{K}, GeoSim(p', p) \leq \rho$ **then**
      Insert P into $\mathbf{K}$;
      break;
return $\mathbf{K}$;

Optimization a) Recording explored shortest paths: in Figure 2, paths P_1 and P_2 share a sub-path starting from the source region s. When P_2 is used as the reference path, any candidate path P'' with a shared sub-path through the branch region will already exist in set Q, causing redundant searches. To avoid this, we record and prune repeated sub-paths in subsequent searches. b) edge deletion: Additionally, before searching for a branch path, we remove edges within r' to avoid the branch region. Rather than removing all edges, we optimize by only removing edges connecting adjacent boundary nodes, such as (a, c) and (b, h) in Figure 2.

Complexity Analysis Clearly, each branch path search requires a call to the shortest path algorithm, whose time complexity is $O(|E| + |V| \cdot \log |V|)$ that corresponds to that of the bidirectional Dijkstra algorithm. Furthermore, as our

search process follows the regions traversed by the reference path, the traversal across all regions incurs a time complexity of $O(|R|)$. Finally, since the goal is to generate K distinct paths, the overall process scales with the number of paths.As a result, the worst-case time complexity of our algorithm can be expressed as $O(K \cdot |R| \, (|E| + |V| \cdot \log |V|))$.

3.3 Insertion-based Order Dispatch

In the previous chapter, we proposed a route recommendation strategy . Building on this, we further adopt the online insertion allocation algorithm from [15], replacing the shortest-path approach with our routing strategy, resulting in the Insertion-based Recommended Path (IRP) method.

4 EXPERIMENTS

4.1 Experimental Setup

Dataset We evaluate our methods using two real-world datasets: 2015 NYC yellow taxi trip data and 2013 Singapore taxi trip data [17]. We use data from January 1st to 30th to train the ST-GCN model, which is then applied to forecasting demand for January 31. The evaluation occurs online after 12:00 PM on January 31, focusing on rush hour performance. The ride-request data includes pick-up/drop-off locations and times, while the road network data is sourced from OpenStreetMap.

For the ST-GCN model, we set the grid cell size to 2 km by 2 km and the time interval to 15 minutes. More details on the datasets and ST-GCN configuration are provided in Table 1.

Parameters	NYC	SG	Parameters	Settings
			routes number k	3
Number of vertices	61298	52653	Similarity factor ρ	0.75
Number of edges	141372	86217	Detour ratio μ	0.2,**0.3**,0.4,0.5
Number of requests	324324	250632	Capacity a	2,**3**,5,7
Grid cell unit size	2km × 2km		Penalty ratio	30
Time interval	15 minutes		Divers' number $\|D\|$	1k,**3k**,5k,7k

Table 1. Setting of Parameters

Evaluation set-up We follow the configurations from previous studies [2, 15]. For the road network, edge weights are calculated as travel time (distance divided by average speed). Requests are mapped to the nearest nodes, and driver positions are initialized randomly. Key parameters are summarized in Table 1, with default values in bold. For route search, we set $k = 3$ and $\rho = 0.75$. Driver capacities range from 2 to 7, and the time expectation for requests is defined

as $te_j = (1 + \mu) \times L(s_j, e_j)$. The penalty p_o is set to 30. All experiments were implemented in Java on an Ubuntu server with an Intel Xeon Gold 6258R CPU and 995GB of RAM, using a single-threaded environment.

Compared algorithm We compare our IRP algorithm with the following baselines: **Greedy** [14], a basic insertion method focusing on minimizing additional cost with the shortest-path strategy; **SHARE** [17], which uses historical trip data for route planning in batch mode, simulating real-time allocation with a time window of 0; **DAIF** [15], which incorporates future demand using the DSBScore to minimize additional costs; and **IRP-greedy**, a variant of our method that relies solely on distance costs for allocation decisions.

Metrics All algorithms are evaluated based on the unified total cost (accounts for both travel time cost and the penalties incurred from rejecting requests, as detailed in Definition 4), the number of served requests, and the allocation time cost per request. These metrics are widely used in existing large-scale online ridesharing studies[11, 14].

4.2 Experimental Results

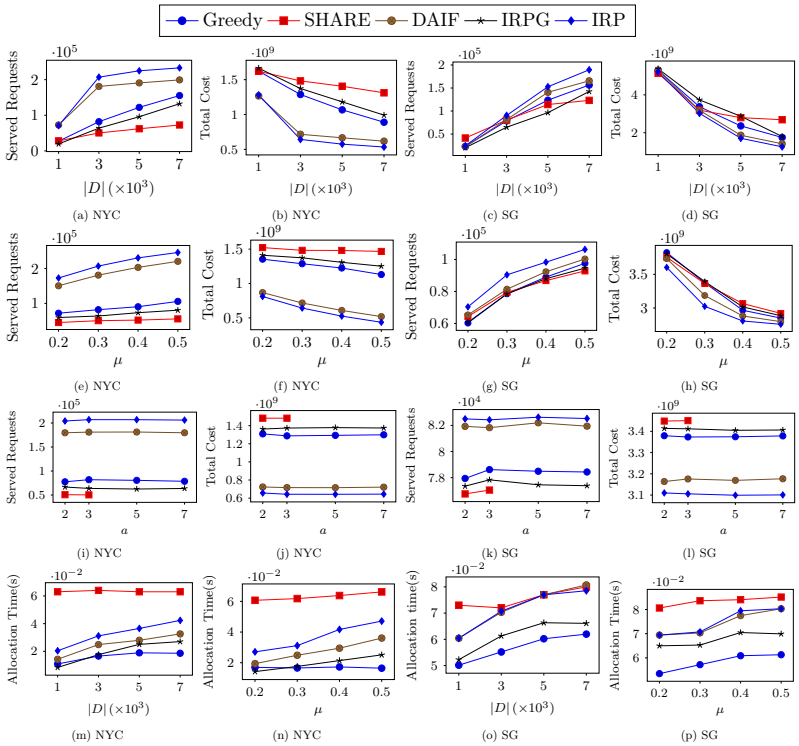

Fig. 3. Performance of varying $|D|, \mu, a$ and Time Analysis

Impact of Number of Drivers $|D|$. The first row of Figure 3 shows that IRP consistently outperforms all methods. In NYC, IRP increases served requests by 14.33% to 310.73% and reduces total cost by 10.24% to 56.62%. In SG, it improves served requests by 14.5% to 53.88% and reduces total cost by 11.30% to 53.29%. As driver numbers increase, all methods serve more requests with lower costs. SHARE performs poorly in NYC due to its 15-minute waiting policy, which is impractical for real-time allocation. In SG, where requests are densely clustered, methods like SHARE, Greedy, and IRPG perform better but still fall short of IRP.

Impact of detour ratio μ. The second row of Figure 3 shows that as the detour ratio increases, the number of served requests grows, with IRP outperforming all other methods. In NYC, IRP serves 13.41% to 342.24% more requests, while in SG, the improvement ranges from 5.98% to 14.47%. IRP also reduces total cost by 13.58% to 64.47% in NYC and 1.43% to 5.58% in SG. In NYC, IRP's growth rate exceeds that of Greedy and SHARE due to its balanced allocation strategy considering both travel cost and future demand. However, in SG, where requests are more densely distributed, the impact of future demand is less significant, resulting in similar growth rates across all methods.

Impact of capacity of Drivers a. The third row of Figure 3 shows that in the NYC dataset, IRP serves 13.63% to 302.78% more requests and reduces total cost by 9.2% to 55.69%. In SG, while IRP still performs best, the difference in served requests is minimal due to the dense request distribution and low detour ratios. Increasing driver capacity doesn't significantly improve service rates, and increasing SHARE's capacity beyond 3 only adds computational overhead without improving performance.

Analysis of Allocation Time. The fourth row of Figure 3 analyzes allocation time, considering the impact of the number of drivers $|D|$ and detour ratio μ on served requests. As both $|D|$ and μ increase, allocation time rises. Among the methods, Greedy shows the shortest allocation time, while others remain within the millisecond range, making them suitable for real-world use.

5 CONCLUSION

This paper tackles the Demand-aware Order Dispatch with Route Recommendation (DODRE) problem, which recommends routes with high demand potential while minimizing travel costs to improve service efficiency. We propose the Regional Demand-aware Route Search method, which uses a demand heat map to suggest one of K diverse routes that pass through high-demand regions, ensuring detour distance constraints are met. Integrated into a real-time allocation framework, our approach, IRP, outperforms several state-of-the-art methods, as demonstrated by extensive experiments on two real-world datasets.

Acknowledgments. Libin Zheng is the corresponding author. This work is supported by the National Natural Science Foundation of China No. 62472455, No. U22B2060 and U2001211, the Guangdong Basic and Applied Basic Research Foundation (2023A1515 110624), the Guangdong Provincial Key Laboratory of IRADS (2022B1212010006),

the Research Foundation of Science and Technology Plan Project of Guangzhou City (2023B01J0001, 2024B01W0004).

References

1. J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus. On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment. *Proc. Natl. Acad. Sci. USA*, 114(3):462–467, 2017.
2. M. Asghari, D. Deng, C. Shahabi, U. Demiryurek, and Y. Li. Price-aware real-time ride-sharing at scale: an auction-based approach. In *SIGSPATIAL/GIS*, pages 3:1–3:10. ACM, 2016.
3. T. Chondrogiannis, P. Bouros, J. Gamper, and U. Leser. Alternative routing: k-shortest paths with limited overlap. In *SIGSPATIAL/GIS*, pages 68:1–68:4. ACM, 2015.
4. B. Cici, A. Markopoulou, and N. Laoutaris. Designing an on-line ride-sharing system. In *SIGSPATIAL/GIS*, pages 60:1–60:4. ACM, 2015.
5. J. Dai, B. Yang, C. Guo, and Z. Ding. Personalized route recommendation using big trajectory data. In *ICDE*, pages 543–554. IEEE Computer Society, 2015.
6. E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.
7. K. Frenken and J. Schor. Putting the sharing economy into perspective. In *A research agenda for sustainable consumption governance*, pages 121–135. Edward Elgar Publishing, 2019.
8. R. Geisberger, P. Sanders, D. Schultes, and D. Delling. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. In *WEA*, volume 5038 of *Lecture Notes in Computer Science*, pages 319–333. Springer, 2008.
9. X. Geng, Y. Li, L. Wang, L. Zhang, Q. Yang, J. Ye, and Y. Liu. Spatiotemporal multi-graph convolution network for ride-hailing demand forecasting. In *AAAI*, pages 3656–3663. AAAI Press, 2019.
10. P.E. Hart, N.J. Nilsson, and B. Raphael. Correction to "a formal basis for the heuristic determination of minimum cost paths". *SIGART Newsl.*, 37:28–29, 1972.
11. Y. Huang, F. Bastani, R. Jin, and X.S. Wang. Large scale real-time ridesharing with service guarantee on road networks. *Proc. VLDB Endow.*, 7(14):2017–2028, 2014.
12. S. Ma, Y. Zheng, and O. Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *ICDE*, pages 410–421. IEEE Computer Society, 2013.
13. Y. Tong, Y. Zeng, Z. Zhou, L. Chen, and K. Xu. Unified route planning for shared mobility: An insertion-based framework. *ACM Trans. Database Syst.*, 47(1):2:1–2:48, 2022.
14. Y. Tong, Y. Zeng, Z. Zhou, L. Chen, J. Ye, and K. Xu. A unified approach to route planning for shared mobility. *Proc. VLDB Endow.*, 11(11):1633–1646, 2018.
15. J. Wang, P. Cheng, L. Zheng, C. Feng, L. Chen, X. Lin, and Z. Wang. Demand-aware route planning for shared mobility services. *Proc. VLDB Endow.*, 13(7):979–991, 2020.
16. J. Y. Yen. Finding the k shortest loopless paths in a network. *management Science*, 17(11):712–716, 1971.
17. C.F. Yuen, A.P. Singh, S. Goyal, S. Ranu, and A. Bagchi. Beyond shortest paths: Route recommendations for ride-sharing. In *WWW*, pages 2258–2269. ACM, 2019.