Customizable Routing with Learnings from Past Recommendations

Kuo-Han Hung
a-jhung@microsoft.com
Microsoft
Taipei, Taiwan

Chiqun Zhang
chizhang@microsoft.com
Microsoft
Mountain View, United States

Dragomir Yankov
dragoy@microsoft.com
Microsoft
Mountain View, CA, USA

Abstract

Finding routes in road networks is a fundamental task for routing services, but most existing methods only consider the network topology and properties and cannot handle semantic queries that express user preferences or constraints. We present a novel method that leverages historical route recommendations to prune irrelevant paths and speed up the search process. Moreover, we introduce a probabilistic modeling for path finding that can incorporate query semantics, such as "route from Seattle to Redmond with less traffic lights", and find optimal routes that satisfy them. We conduct experiments and evaluations on real-world datasets and show that our method outperforms the state-of-the-art methods in terms of runtime efficiency and route quality and can effectively answer semantic queries.

CCS Concepts

• **Applied computing** → **Transportation**; • **Information systems** → **Location based services**.

Keywords

routing, path finding, shortest path, semantic routing

ACM Reference Format:

Kuo-Han Hung, Chiqun Zhang, and Dragomir Yankov. 2024. Customizable Routing with Learnings from Past Recommendations. In *The 32nd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL '24), October 29-November 1, 2024, Atlanta, GA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3678717.3691327

1 Introduction

Routing is a vital component of map services, especially with the recent advances in autonomous driving and smart electric vehicles. Many research efforts have been devoted to finding point-to-point shortest paths on road networks in the last decade, for example, using Dijkstra's algorithm [5], which runs in linear time. Various extensions and variations of Dijkstra's algorithm [4, 9]have been proposed to improve runtime performance and to accommodate more specific scenarios. Meanwhile, algorithms with predictions[7] have emerged as a promising approach to design improved algorithms for optimization problems. Some studies[6, 8] have explored the use of machine learning methods for the fundamental shortest path problem. While several methods have been proposed[8, 10] to leverage machine learning to improve latency, few studies have demonstrated the actual latency improvement considering both the path finding and the machine learning model inference. Therefore, a lightweight and simple model is desirable for practical applications. On the other hand, with the increasing trend of conversational search[12], routing queries have more semantic intent, such as "driving from Seattle to Redmond with less traffic lights". To support such queries, we propose a novel modeling of routing problem from probabilistic perspective which is able to support various preferences and constraints. In this work, we address two aspects of routing that have not been fully investigated: faster path finding with learning from past routes, and general support for customized routing experiences. The contributions of this work are as follows.

- We propose a novel probabilistic modelling for customized routing experiences, which is a general form of classic path finding algorithm but with flexible support for route customization.
- In this paper, we propose a novel and simple method to reduce the search space for path finding by exploiting the information from previous route recommendations. Our method is lightweight and compatible with any classical algorithm, such as Dijkstra, A*, and others.
- Furthermore, we also present a method that can predict missing properties for nodes or edges in a road network without any training. Our model is inspired by the message passing step in graph neural networks, which enables efficient inference on large-scale graphs.

2 Customizable Routing with Learnings from Past

2.1 Probabilistic Modelling for Customizable Routing

Routing problems often aim to find the shortest distance or the least travel time between a source node s and a destination node d in a road network. However, these objectives may not capture the semantic preferences of users who may seek more eco-friendly, less congested, or safer routes. Moreover, the optimal route may depend on a combination of multiple criteria that vary from user to user[3]. To address this challenge, we propose a generic formulation for routing problems from a probabilistic perspective. Given a road network and a source node s and a destination node d, we define the probability of a route L being the desired route as

$$P(L|s,d) = \Pi_{(u,v)\in L} P((u,v)|s,d), \tag{1}$$

where (u,v) denotes a directed edge in the network from node u to node v, and $P((u,v)|s,d)$ represents the probability of an edge (u,v) being passed through along a route L from source s to destination d.

SIGSPATIAL '24, October 29-November 1, 2024, Atlanta, GA, USA

ACM ISBN 979-8-4007-1107-7/24/10
https://doi.org/10.1145/3678717.3691327

The optimal route $\hat{L}$ is the one that maximizes $P(L|s,d)$. However, finding $\hat{L}$ directly is a NP-hard problem[11], as it requires evaluating all possible routes in the network. To simplify the computation, we can apply the Markov property and decompose Eqn 1 into conditional probabilities

$$P(L|s,d) = \Pi_{(u,v)\in L} P((u,v)|u,s,d)P(u|s,d) \quad (2)$$
$$= \Pi_{(u,v)\in L} P((u,v)|u,d)P(u|s,d). \quad (3)$$

In the classic Dijkstra's algorithm, $P(u|s,d)$ is assumed to be a constant from a uniform distribution and $P((u,v)|u,s,d)$ is a normalized value inversely proportional to the distance between u and v. In customizable routing, we can incorporate various additional factors into these probabilities. For example, $P((u,v)|u,s,d)$ can reflect the number of lanes or the environmental impact, and $P(u|s,d)$ can capture the presence of traffic lights or stop signs, or the risk of accidents. Therefore, this probabilistic model offers us flexibility for different routing objectives.In next section, we will describe how learnings from past routes can be modeled into $P((u,v)|u,s,d)$ and accelerate path finding during runtime.

2.2 Path Finding with Learnings from Past

As described in Section 2.1, in a simple situation where $P(u|s,d)$ is a constant, $P((u,v)|u,d)$ captures the basic road segment properties, such as distance or travel time. Common algorithms only rely on road network topology and segment properties. In this work, we propose a simple but effective way to enhance path finding by learning the routing pattern based on a collection of historical trajectories. Given a collection of past route recommendations, we map match the traces to a given road network and learn the route pattern into a frequency-based look-up table. This table records a normalized frequency value for each road segment that is passed through given a destination node, and thus allows us to query the probability of each edge given the destination. Since it is tedious and impractical to collect routes covering every node in the road network, we introduce a tile system to improve the model's efficiency. Instead of conditioning the look-up table solely on the destination node in the network, we partition the map into tiles and condition the table on these tiles. This approach offers several advantages, including reduced memory usage for the look-up table and increased frequency coverage.

Tile Construction To divide the entire region into tiles, we segment both the longitude and latitude axes to create multiple tiles. For example, 32x32 represents a grid including 1024 tiles with 32 segments along each axis.

Frequency Look-up Table Construction We transform the historical trajectory data into a look-up table that is conditioned on the edges and tiles. For each road segment, we count the number of times it is traversed by a trajectory that ends in a certain tile, and normalize it by the total number of trajectories that end in that tile. The normalized value is treated as $P((u,v)|u,d)$.

Frequency-based Path Finding As shown in Algorithm 1, we modify the original Dijkstra algorithm into two parts: in-tile path finding and out-tile path finding. For the in-tile search, which means the query node is inside the tile, we use the standard Astar algorithm. For the out-tile search, we use the product of frequencies

Algorithm 1 Frequency-based Path Finding

procedure FreqDijkstra($src, dest, G, Freq\_table$)
  Initialize $\forall v \in G$, freq$[v] \leftarrow -\infty$, freq$[src] \leftarrow 1$
  Initialize priority queue $Q \leftarrow \{(src, 1)\}$
  Initialize $path$[#nodes]
  while $Q \neq \emptyset$ **do**
    $(u, f_u) \leftarrow$ Extract-Max(Q)
    if Tile(u) = Tile($dest$) **then**
      $p_{\text{out}} \leftarrow$ Reconstruct-Path(path, src, u)
      $p_{\text{in}} \leftarrow$ Astar($u, dest, G$)
      return $p_{\text{out}} + p_{\text{in}}$
    for $v \in$ Neighbors(u) **do**
      $f_{uv} \leftarrow \begin{cases} \text{Freq\_table[Tile}(dest)][(u,v)] & \text{if it exists} \\ \text{FreqPassing}(u, v, Tile(dest)) & \text{otherwise} \end{cases}$
      $f_v \leftarrow f_u \times f_{uv}$
      if $f_v >$ freq$[v]$ **then**
        freq$[v] \leftarrow f_v$; path$[v] \leftarrow u$
        $Q \leftarrow Q \cup \{(v, f_v)\}$

conditioned on the destination tile along the route to find the maximum frequency product from the source node to the destination tile. This way, we favor the routes that are more likely to be chosen by the historical trajectories.

2.3 Message Passing

Collecting a comprehensive dataset that covers all the edges of a huge map is infeasible. Moreover, building a dense look-up table for the entire road network demands enormous memory and computation resources. Furthermore, the historical route recommendations may be sparse for less popular areas. Hence, we introduce a modified message-passing method to estimate the missing frequencies.

Message passing is a key technique in Graph Neural Network (GNN)[1, 2], which allows information exchange and aggregation among nodes in a graph. By passing messages through the graph, the model can exploit the local neighborhood information of each node to refine and update their representations. Algorithm 2 shows the steps of our proposed message passing method for routing. Our method consists of three steps, similar to GNN, as follows.

- Message. Every node or edge in the graph computes a message for each of its neighbors. In our case, the message is the frequency count learned from historical route recommendations.
- Aggregate. A node or edge aggregates the messages it receives, using a permutation-invariant function. We use the average of all received messages in our work.
- Update. Each node or edge updates its attributes based on a function of its current values and the aggregated messages. We use an exponential smoothing function in our work.

In this work, we repeat above steps N times, which is analog of the depth in GNN.

3 Experiments and Discussions

3.1 Experiment Setup and Data Reproducibility

We study the road network of a rectangular region of Puget Sound, defined by the coordinates $((-122.43, 47.43), (-121.99, 47.76))$, using the Open Street Map dataset. This region includes parts of Seattle, Bellevue, and other nearby cities. The road network has 55938

Algorithm 2 Frequency Passing (FreqPassing)

procedure FreqPassing($u, v, tile, Counts\_table, \alpha$)
  Initialize $missing\_edges$ as edges around (u, v) without history counts up to N steps
  Initialize $h^0 \leftarrow Counts\_table[tile]$
  for $i \leftarrow 1$ to N **do**
    $h^i \leftarrow h^{i-1}$
    for each edge (u', v') in $missing\_edges$ **do**
      $h^i_{(u',v')} \leftarrow \alpha * h^{i-1}_{(u',v')} + (1-\alpha) * MEAN(h^{i-1}_{(v',*)})$
  $freq_{(u,v)} \leftarrow h^N_{(u,v)} / SUM(h^N_{(u,*)})$
  return $freq_{(u,v)}$

nodes and 137167 edges, which represent intersections and road segments, respectively. We collected two datasets for our research. The first dataset, used for metric evaluation in Section 3.2.1 and Section 3.3, was obtained by randomly querying the shortest-distance routes in the OSM. This resulted in 100,000 routes, representing 0.003% of all possible queries on the tested map. The second dataset, used for qualitative research in Section 3.2.2, maps real-world use cases by randomly collecting 403,153 fastest routes over a week through one routing API.

Since the density of the frequency table depends on the granularity of the tiles, we compare the coverage for a fine and a coarse tile setting in Figure 1. In Figure 1, given a same destination location but different tile granularity, we color the road segments that are passed by any routes whose destination falls into that destination tile red. We observe that for the fine tile setting, the coverage is mainly concentrated on the major roads. This implies that the guidance for some minor roads is less frequent, leading to more visited nodes that the algorithm has to explore. We discuss the details of the tile selection in Section 3.3.

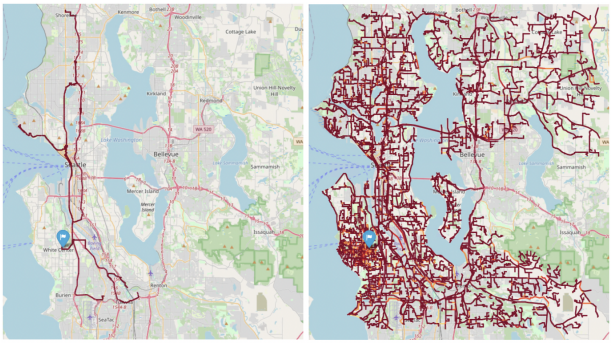

Figure 1: Frequency Coverage Comparison (Left: 1024x1024 tiles, Right: 32x32 tiles).

3.2 Experiment Results

3.2.1 Quantitative results To test the efficacy of our algorithm, we randomly sampled 100 queries with different source and destination points in the tested area and averaged the performance of different algorithms. The statistical results of our algorithms compared to the baselines (such as Dijkstra and A*) are presented in Table 1. The table shows that our algorithm significantly improves the explored area by approximately 98% in terms of visiting nodes, demonstrating its superiority over the baselines, even with only 0.003% of all possible queries used. We also examine the impact of N (the range of message passing) on the algorithm's performance. As illustrated in Table 2, increasing the range for passing frequencies reduces the number of nodes that need to be visited to plan the route without compromising the quality of the planned route, indicating the effectiveness of message passing. However, the improvement is not linear, and there is a trade-off between the range and the communication overhead.

	Visited Nodes	Path Length
Dijkstra	34684	**28609**
A*	12267	28611
Dijkstra Freq	**207**	29333

Table 1: Average number of visited nodes and the average path length (meter) for our algorithm and the baselines.

	Visited Nodes	Path Length
without passing	225	29370
N = 1	228	29314
N = 3	212	29333
N = 5	207	29333
N = 7	205	29371
N = 9	204	29401

Table 2: Average number of visited nodes and the average path length (meter) for our algorithm with different N. More passing range leads to less visited node (more efficient) while maintaining on-par route quality (path length).

3.2.2 Qualitative Analysis for the Planned Route To evaluate the quality of the routes generated by our algorithm, we performed a qualitative analysis based on the road characteristics. We observed that our algorithm tended to select roads with higher speeds (less travel time), such as highways, reflecting its ability to learn real route pattern from historical recommendations. Figure 2 illustrates an example of this behavior. Moreover, we also investigated the role of message passing in our framework in Section 2.3. Figure 3 shows that message passing can facilitate faster exploration in local areas by propagating the historical information of adjacent nodes to the nodes that lack such data. Usually routes follow "local road to major road to local road" pattern, and the proposed method could populate insights from nearby local roads in the dataset to guide the algorithm to reach the major roads faster.

3.3 Study of Selecting Tile Granularity

As mentioned in 3.1, the frequency look-up table depends on a predefined tile grid. We explored the impact of different tile grids, as illustrated in Figure 4. Both the figures for visited nodes and planned length comparisons revealed a U-shaped trend. Larger tiles entail more in-tile searches but compromise out-tile search quality due to a noisy look-up table. Smaller tiles enhance in-tile search efficiency and out-tile search precision, lowering the number of visited nodes. However, when the tiles are too small, the look-up table becomes sparse, leading to fewer routes per tile and less

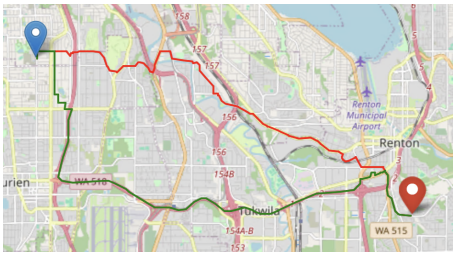

Figure 2: Example of the planned route (red: shortest path, green: our algorithm's path)

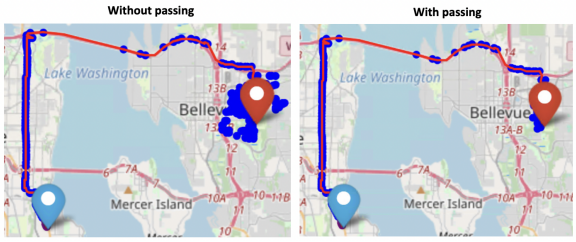

Figure 3: Visualization of the visited nodes of different settings (left: without using message passing, right: with message passing).

direction for finding the right route, which eventually increases visited nodes.

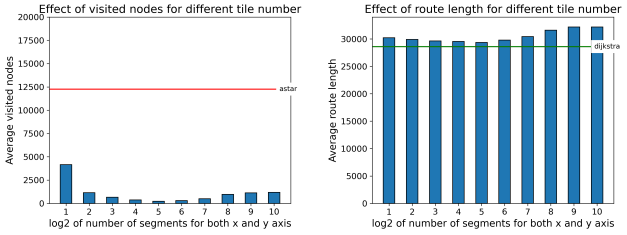

Figure 4: Comparison of the effect of visited nodes and route length for different tile numbers.

3.4 Case Study: Customized Routes with Traffic Light Preference

In this work, we show how our model can generate customized routes based on user preferences. One common preference is to avoid traffic lights. To capture this preference, we use the node property from OSM data. Since traffic lights usually exist at road intersections, it is natural to represent such preference into $P(u|s,d)$. To reflect the preference of less traffic lights, we penalize $P(u|s,d)$ by a factor 0.5 if node u has a traffic light. Figure 5 illustrates the distribution of traffic lights in the study area and an example query with and without the customized $P(u|s,d)$. The orange route is the shortest distance route without considering the preference, while the blue route is the optimal route that balances the distance and the number of traffic lights.

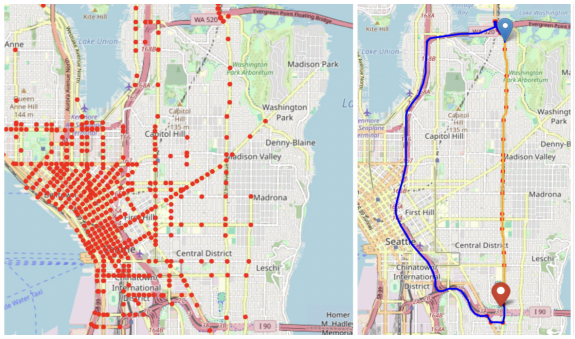

Figure 5: Left: Traffic light on OSM in Seattle. Right: Customized routes with traffic light preference (orange: planning without preference, blue: planning with preference of avoiding traffic light nodes).

4 Conclusion

We have proposed a novel method for finding routes in road networks that can handle semantic queries and leverage historical route recommendations to prune the search space. Our method uses a probabilistic modeling that can capture user preferences and constraints and find optimal routes that satisfy them. We have demonstrated the effectiveness and efficiency of our method on real-world datasets and compared it with the state-of-the-art methods. Our method can provide routing services with more flexibility and functionality and improve the user experience.

References

[1] Michael M Bronstein, Joan Bruna, Taco Cohen, and Petar Veličković. 2021. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478* (2021).
[2] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. 2017. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine* 34, 4 (2017), 18–42.
[3] Daniel Delling, Andrew V Goldberg, Thomas Pajor, and Renato F Werneck. 2017. Customizable route planning in road networks. *Transportation Science* 51, 2 (2017), 566–591.
[4] Daniel Delling, Peter Sanders, Dominik Schultes, and Dorothea Wagner. 2009. Engineering route planning algorithms. In *Algorithmics of large and complex networks: design, analysis, and simulation*. Springer, 117–139.
[5] Edsger W Dijkstra. 2022. A note on two problems in connexion with graphs. In *Edsger Wybe Dijkstra: His Life, Work, and Legacy*. 287–290.
[6] Willem Feijen and Guido Schäfer. 2021. Dijkstras algorithm with predictions to solve the single-source many-targets shortest-path problem. *arXiv preprint arXiv:2112.11927* (2021).
[7] Sen Huang, Kaixiang Yang, Sheng Qi, and Rui Wang. 2024. When Large Language Model Meets Optimization. *arXiv preprint arXiv:2405.10098* (2024).
[8] Lucas Lehnert, Sainbayar Sukhbaatar, Paul Mcvay, Michael Rabbat, and Yuandong Tian. 2024. Beyond A*: Better Planning with Transformers via Search Dynamics Bootstrapping. *arXiv preprint arXiv:2402.14083* (2024).
[9] Christian Sommer. 2014. Shortest-path queries in static networks. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 1–31.
[10] Wei Tian, Jieming Shi, Siqiang Luo, Hui Li, Xike Xie, and Yuanhang Zou. 2023. Effective and Efficient Route Planning Using Historical Trajectories on Road Networks. *Proceedings of the VLDB Endowment* 16, 10 (2023), 2512–2524.
[11] Gang Yu and Jian Yang. 1998. On the robust shortest path problem. *Computers & operations research* 25, 6 (1998), 457–468.
[12] Chiqun Zhang, Antonios Karatzoglou, Helen Craig, and Dragomir Yankov. 2023. Map GPT Playground: Smart Locations and Routes with GPT. In *Proceedings of the 31st ACM International Conference on Advances in Geographic Information Systems*. 1–4.