Journal of Spatial Information Science
Number 26 (2023), pp. 27–52
doi:10.5311/JOSIS.2023.26.221

Research Article

Representative dissimilar path queries: accommodating human movement dynamics in road networks

Tanzima Hashem[a], Matt Duckham[b], Mahathir Monjur[a], and Fariha Tabassum Islam[a]

[a]Department of Computer Science and Engineering, Bangladesh University of Engineering and Technology, Bangladesh
[b]School of Science, RMIT University, Australia

Received: May 3, 2022; returned: July 31, 2022; revised: October 30, 2022; accepted: December 16, 2022.

Abstract: We introduce a representative dissimilar path (RDP) query, a novel type of path query in road networks. The k representative paths (RPs) between a source and a destination locations have k smallest costs for a feature (e.g., length, number of road intersections, or straightness). Given x features and k, an RDP query returns a set of paths for a source-destination pair such that the path set includes at least one of the k RPs for every feature, and the path set's similarity score is minimized. We formulate a novel measure to quantify the similarity of a set of paths. Considering different road features and incorporating the novel similarity measure in the computation of RDPs allow us to accommodate the human movement dynamics between two locations in an effective way. Finding the RDPs is a computational challenge because an RDP query requires computing the RPs for multiple features and then finding the RDPs from an exponential number of path combinations. We develop an efficient solution to answer RDP queries. The underlying ideas behind the efficiency of our algorithms are the refinement of the search space, finding the RPs for multiple features with a single search, and exploiting both the lower and upper bounds of the path set's similarity score while identifying the RDPs. We show the efficacy of the RDP query and the efficiency of our solution to answer the RDP query in extensive experiments using real datasets.

Keywords: Representative paths, representative dissimilar paths, similarity score, shortest paths, road networks

1 Introduction

Travelers on roads have different preferences; some people like to take the shortest or the fastest paths, whereas others prefer straight paths or the paths with the minimum number of intersections [4]. In this paper, we introduce the representative dissimilar path (RDP) query that finds a set of paths with an aim to incorporate different road property preferences of travelers while maintaining a reasonable dissimilarity among the paths. Hence the answer of an RDP query is a set of dissimilar paths that are also representative of different preferences of the travelers.

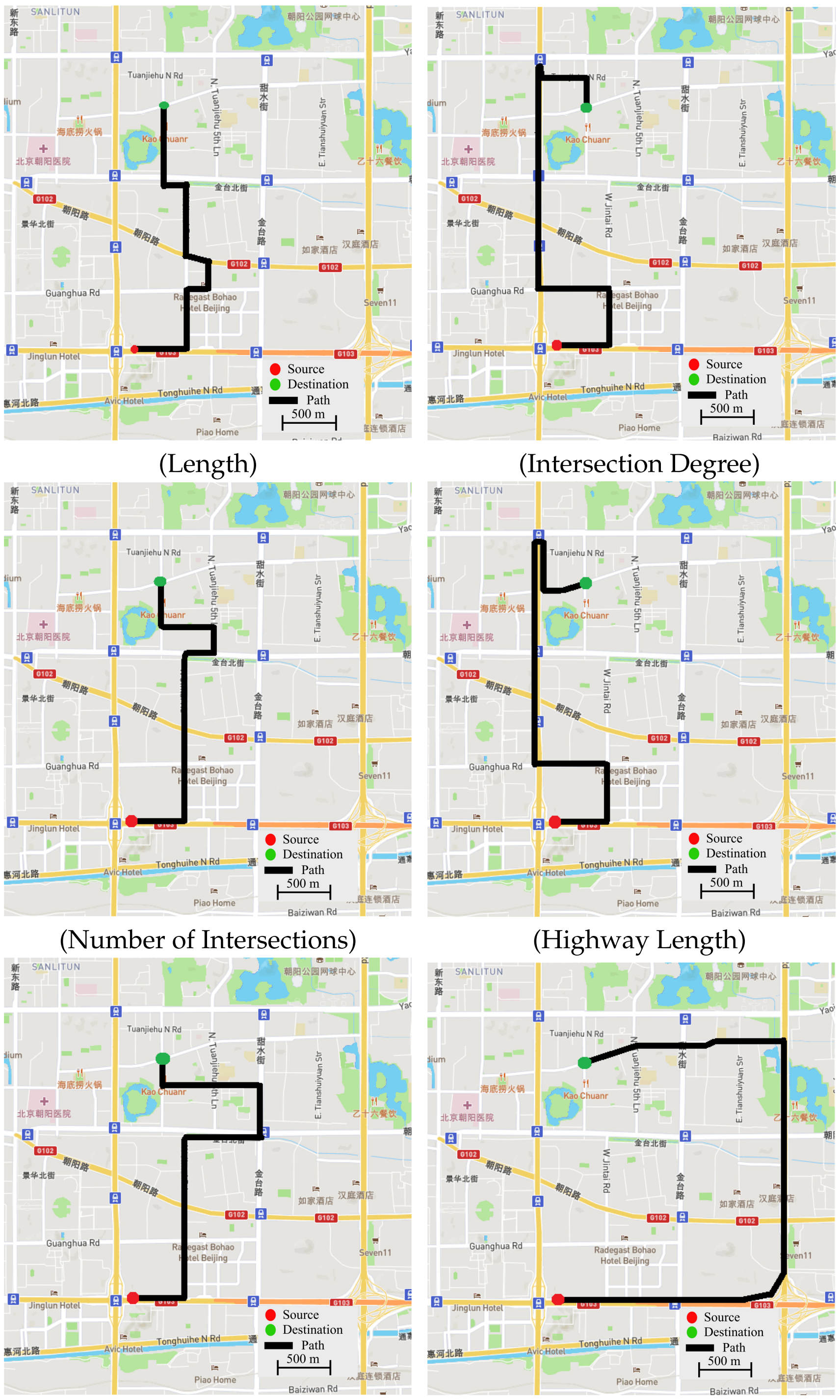
Accommodating human movement dynamics using the RDP query has several important applications, which fall into two broad categories. First, a range of recommendation-type applications can benefit from computing a representative *set* of routes corresponding to a range of route characteristics. Wayfinders, for example, may benefit from the option to select a path that suits their personal preferences from amongst a set of computed paths rather than be required to precisely specify their preferences in advance of computation. Representative but dissimilar paths can ensure coverage of different options, in contrast to other shortest path algorithms such as k-shortest paths. Such dissimilar paths may also be beneficial to the transportation of aid through disaster-affected regions or evacuation during an emergency, for example. Providing the option to recommend the variety of different paths returned by an RDP query may help ensure that the selected route or routes ultimately maximizes the chances of reaching the destination.

Second, a range of spatial analysis and geospatial intelligence (GEOINT) applications may rely on the generation of a set of possible but unknown paths that a person may have used to travel between a known source and destination location. In the case of crime analysis, the likely paths that may possibly have been taken by persons of interest may need to be generated and ranked in order to rule out suspects. In defense applications and GEOINT, similarly, an intelligence analyst may frequently need to estimate and compare the most likely routes a hostile agent could have taken between two locations in order to conduct a threat assessment.

A representative path (RP) for a feature (e.g., length or number of intersections) in the road network has the smallest cost for the corresponding feature (e.g., shortest distance or the smallest number of intersections) among all possible paths between the source-destination pair. The k RPs for a feature between a source-destination pair have k smallest costs for the corresponding feature. Given a set of x features and k RPs for every feature, an RDP query returns a path set that includes one of the k RPs for every feature such that the similarity score of the path set is minimized.

Figure 1 shows an example of representative dissimilar paths (RDPs) returned by an RDP query for $k = 4$ and $x = 6$ (i.e., length, intersection degree, number of intersections, highway length, residential road length and tortuosity). The path costs for features: length, highway length, and residential road length are computed by summing up the length of the roads, highways, and residential roads included in the path, respectively, and the path costs for features: intersection degree and tortuosity are computed by summing up the intersection degree and angle between the consecutive roads in the path, respectively.

In recent years, researchers have focused on finding alternative or dissimilar paths [2,5, 7–10,21]. However, they have major limitations:



(Length) (Intersection Degree)

(Number of Intersections) (Highway Length)

(Residential Road Length) (Tortuosity)

Figure 1: Representative Dissimilar Paths (RDPs) for six features

- The set of alternate or dissimilar paths computed by the existing approaches are not diversified in terms of different road features like path length, number of road intersections and intersection complexity, highway or residential road length, and straightness. For example, it may happen that all paths in the set have a large number of road intersections and include sharp turns (i.e., not straight).
- The existing similarity measures for two paths are only based on the path overlap length. When these measures are extended for a path set by adding the similarity score of every path pair in the set, they fail to capture the actual similarity of the paths as they ignore the location of the path overlap. For example, the similarity score of three paths that overlap at the same road should be higher than that of the scenario where each pair of these three paths overlap at different roads, assuming that the path overlap length is the same. However, if we ignore the location of the path overlap, the similarity scores for these three paths in both scenarios are the same.
- Most existing solutions to find a set of dissimilar paths require the dissimilarity threshold that every pair of paths should satisfy as input. It is not trivial to intuitively determine the appropriate dissimilarity threshold without knowing the surrounding road network structure of the source and destination locations. However, the quality of the set of dissimilar paths depends on the specified dissimilarity threshold.

RDP queries overcome the above limitations. RDP queries consider different road features to find RPs and then determine the RDPs from RPs. We formulate a new similarity measure for a path set by considering both the length and location of the path overlap and use the measure to minimize the similarity score of the path set. Intuitively, the similarity score of a path set increases with the increase of the path overlap length and the number of paths that overlap at the same location. To incorporate the intuition, we use the number of distinct pairwise path overlap, the length of the path overlap, and the number of paths per overlap as the parameters of our similarity measure for a path set (please see Section 3.3 for details). Furthermore, the RDP query does not require the dissimilarity threshold as input.

The major challenges in evaluating an RDP query are twofold: (i) computing k RPs for every feature and (ii) finding the path set that minimizes the similarity score from a huge number of possible combinations of RPs. We cannot adopt efficient shortest path algorithms (e.g., [24]) to compute the RPs for different features due to their dependency on the length measure. Furthermore, evaluating the RPs for every feature independently would traverse the same road network multiple times. We develop an efficient algorithm to find k RPs for x features with a single traversal on the road network (i.e., without exploring the same part of the road network multiple times). Furthermore, there are k^x possible candidate path sets for RDPs, and it would be prohibitively expensive to compute the similarity scores for each of these path sets. Our search space refinement technique using the upper and lower bounds of the path set similarity score prunes the path sets that are guaranteed to be not the answer of the RDP query. To further reduce the processing time for large k, we show a technique to approximate the RDP query answer in return for sacrificing the accuracy slightly.

To the best of our knowledge, we first formulate and develop the solution for RDP queries. The contributions of our paper are summarized as follows:

- We consider both location and length of path overlap and formulate a new measure to quantify the similarity of paths included in a path set.
- We develop an efficient algorithm that finds k RPs for all given features without accessing the same part of the road network multiple times.

- We develop an exact algorithm and a more efficient algorithm to process RDP queries. Our RDP algorithms exploit the lower bound and upper bound properties of our path set similarity score and reduce the number of the candidate path sets for finding the RDP query answer.
- We perform extensive experiments using real datasets to show the effectiveness and efficiency of our solution.

2 Related work

In this section, we discuss the works related to our research problem from two perspectives: (i) alternative and diversified path queries and (ii) path learning and recommendation. Besides, there are many commercial systems like Google Maps or TomTom that have their own confidential techniques to generate alternate paths for travelers.

2.1 Alternative and diversified path queries

Researchers exploited techniques like plateaus [17], alternative route graphs [3], and single via paths [1] to find the paths that are alternative to the shortest paths. Plateau-based solutions build two shortest-path trees rooted at source and destination locations, respectively, and then join the trees to find the common paths between trees known as plateaus. A path between a source and a destination via a plateau consists of the shortest path from the source to one endpoint of the plateau, the plateau, and the shortest path from the destination to another endpoint of the plateau. A plateau goodness metric [17] states that a path through the plateau is good if the plateau length is high and the length not included in the plateau is low. Top k plateaus are selected and used to build k alternate paths. Alternate route graphs encode the union of a set of paths between two vertices to facilitate the computation of the alternate paths. The solution based on single-via paths selects a vertex apart from the source and destination locations. It considers the shortest path via the selected vertex as an alternative path if it satisfies a user's specified constraints like length, local optimality, and stretch. In [19], the authors addressed the problem of continuously finding alternative paths for a user who travels towards a destination. The work exploits the plateau-based method and reuses the already computed information to continuously update the alternative paths as the user moves along a path.

A penalty-based method in [2] iteratively computes the shortest paths by adjusting the edge weights so that the path computed in an iteration is sufficiently dissimilar from the already computed paths. If a generated path does not satisfy the required dissimilarity criteria, the path is not considered as an alternative path. Other approaches [7, 8, 10] focused on finding a set of possible shortest paths such that each pair of paths in the set satisfy the constraint of the maximum allowed similarity threshold in terms of the overlap length. In these works, the authors developed both exact and heuristic solutions for finding the shortest paths with limited overlap. The underlying idea of the exact solutions is to traverse the road network in order of path length until k shortest paths to a destination that satisfy the similarity threshold are found. The heuristic solutions further reduce the exploration of the road network edges in return for sacrificing accuracy. Specifically, the heuristic solutions either do not guarantee that the computed paths are as short as possible or relax the similarity threshold but gain performance improvement in terms of computational time.

[9, 21] minimize the total length of the set of diversified paths, which is NP-hard. Therefore, these works proposed heuristic solutions based on single-via paths and lower bounds, respectively. In contrast, in [5], the authors modeled the problem to minimize both the total length of the paths and the summation of the pairwise path overlap, which is also NP-hard. This work, therefore, proposed a heuristic based on ant colony optimization.

Existing works [2, 5, 7–10, 21] use different measures to quantify the similarity between two paths in terms of the length of the path overlap and find the set of alternative or diversified paths with a guaranteed dissimilarity. However, the similarity measures in these works do not take the location of the path overlap into account. In these works, the similarity of a set of paths is measured by adding the similarity scores for every path pair in the path set. As a result, a set of paths with a small total similarity value can overlap at the same locations and cause a bottleneck in attaining the purpose of using alternate or diversified paths. We develop a similarity measure for a path set by considering both the length and the location of path overlap (Section 3.3).

All of the above works ignore features like path straightness or the number of intersections while finding the alternative and dissimilar paths. An RDP query can incorporate any number of features while finding the dissimilar paths. In addition, the RDP query does not require the specification of the dissimilarity threshold parameter, which is hard to quantify without having knowledge about the surrounding road structure of the source and destination locations. The RDP query is also different from the skyline path query [18, 26], which returns a set of non-dominant paths, where every path is better than others in terms of at least one road feature. Thus, the number of paths returned by a skyline path query can be huge, and more importantly, the skyline path query does not consider minimizing the dissimilarity score of the returned paths.

2.2 Path learning and recommendation

Research works have considered recommending popular or most probable paths [6,20,31], preferred and personalized paths [11,27,28,30] between two locations based on the learning from the historical trajectory data.

[6,31] learns a network graph from historical trajectories and then finds the most popular paths from the derived network. [14] constructs a region graph from sparse trajectories and finds the popular paths from the region graph. Specifically, the work clusters the road network intersections into regions; it learns the historical travel preferences between regions and transfers the learned travel preferences to the region pairs for which there is insufficient trajectory data. DeepST [20] uses recurrent neural networks and variational autoencoders to recommend the most probable paths.

[11] models drivers' preferences from drivers' trajectory data and develops recommendation algorithms to find personalized paths. [30] finds popular road segments from historical trajectories and uses them to find a set of paths between a source and a destination. Then this work uses a scoring function to rank the identified paths based on user preferences and the length of the route. [27, 28] use neural networks to learn the context-based costs for A* search for recommending personalized paths.

None of the above learning frameworks provides a set of representative dissimilar paths for a source-destination pair.

3 Problem formulation

We model the road network with a graph $G(V, E)$, where each vertex in V represents a road intersection and each edge in E represents a road connecting two vertices. A path between two locations is a set of vertices, where each pair of consecutive vertices is connected with an edge.

3.1 Representative paths

We define a representative path (RP) between a source and a destination locations with respect to path features like length, number of intersections, and path straightness.

Definition 3.1. *A representative path (RP).* Given a feature F_x, a cost function c_x for F_x, a source location s and a destination location d, a path p between s and d is called a representative path (RP) for F_x, if $c_x(p) \leq c_x(p')$ for any other path p' between s and d.

The k RPs of a feature F_x are k distinct paths that have the k smallest values for the cost function c_x of F_x. Our solution can find the representative paths for a feature F_x that satisfies the following property: $c_x(p) \geq c_x(p')$, where p' is a subpath of p.

3.2 Representative dissimilar path (RDP) queries

We formulate a representative dissimilar path (RDP) query as follows.

Definition 3.2. *Representative dissimilar path (RDP) queries.* Given a set of x features $F = \{F_1, F_2, \dots F_x\}$, k RPs for each feature in F, a source location s and a destination location d, a function $sim(P)$ that returns the similarity score for a set P of paths, a representative dissimilar path (RDP) query returns a set P_{RD} of x distinct paths such that $sim(P_{RD}) \leq sim(P'_{RD})$ for any other set P'_{RD} of x distinct paths, where both P_{RD} and P'_{RD} have at least one path with cost less than or equal to k^{th} smallest cost for every feature in F.

3.3 Similarity measure

Table 1 shows existing similarity measures for two paths p_i and p_j. All of these similarity measures are based on the length of the path overlap. These measures for two paths can be extended for a set of paths by adding the similarity scores of every pair of paths in the set. However, measuring the similarity score of a set of paths in this way ignores the *location of the path overlaps*, i.e., a set of paths that overlap at the same location are considered the same as the scenario when some of the paths overlap at one location and the remaining paths overlap at different locations.

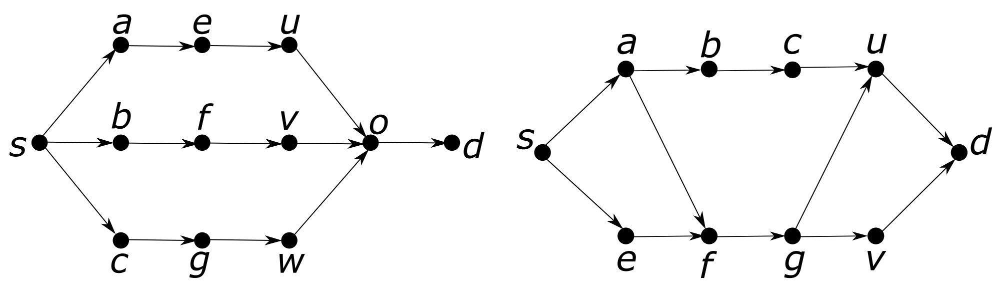
Tables 2 and 3 show two path sets for the road networks shown in Figures 2(a) and 2(b), respectively. For simplicity, we assume that all edges in the road networks have the same length. For the path set in Table 2, three paths go through the same road $\langle o, d \rangle$, whereas, in Table 3, each pair of these three paths overlap at three different roads. Intuitively, the path set in Table 2 should be more similar to that of Table 3. However, according to the existing measures, the similarity scores for these two path sets are the same because the existing measures ignore locations of the path overlap.

We introduce the measure of similarity score for a set of paths by considering both *length and location of the path overlaps*:

Measure	Reference
$\frac{l(p_i \cap p_j)}{l(p_i \cup p_j)}$	[11–13]
$\frac{l(p_i \cap p_j)}{2 \times l(p_i)} + \frac{l(p_i \cap p_j)}{2 \times l(p_j)}$	[2,12,13]
$\sqrt{\frac{l(p_i \cap p_j)^2}{l(p_i) \times l(p_j)}}$	[12,13]
$\frac{l(p_i \cap p_j)}{\max\{l(p_i),l(p_j)}\}$	[12,13]
$\frac{l(p_i \cap p_j)}{\min\{l(p_i),l(p_j)}\}$	[7]

Table 1: Similarity measures for two paths p_i and p_j, where $l(p_i \cap p_j)$ and $l(p_i \cup p_j)$ represent the length of the overlap and the length of the union of p_i and p_j, respectively, and $l(p_i)$ and $l(p_j)$ represent the length of paths p_i and p_j, respectively.

Definition 3.3. *Similarity score.* The similarity score of a set of n paths $P = \{p_1, p_2, \ldots, p_n\}$ that go through the set of m edges $E_P = \{e_1, e_2, \ldots, e_m\}$ is defined as follows: $sim(P) = \sum_{i=1}^{m}\{n_{PO}(e_i) \times (l(e_i))^{n_P(e_i)}\}$, where $n_{PO}(e_i)$ represents the number of distinct pairwise path overlap at e_i, $l(e_i)$ represents the length of the road denoted with e_i and $n_P(e_i)$ represents the number of paths that go through e_i.



(a) (b)

Figure 2: Road network examples

Paths	
p_1	s, a, e, u, o, d
p_2	s, b, f, v, o, d
p_3	s, c, g, w, o, d

Table 2: A path set for the road network in Figure 2(a)

Paths	
p_1	s, a, b, c, u, d
p_2	s, a, f, g, v, d
p_3	s, e, f, g, u, d

Table 3: A path set for the road network in Figure 2(b)

The bigger the score, the higher the similarities among the paths in the set. To increase the similarity score when the paths overlap at the same location (i.e., road network edge) instead of different locations, we use $n_P(e_i)$ as a power of $l(e_i)$. Our measure provides different similarity scores, 24 and 12 for the paths sets in Table 2 and Table 3, respectively, where we assume that the length of each road is 2 units.

With the increase of the overlapped edge length, the contribution of the power factor (i.e., $n_P(e_i)$) in our similarity score increases. In the above examples, if we assume that the length of each road is 1 (or 3 units) instead of 2 units, our measure provides similarity scores of 3 and 3 (or 81 and 27) instead of 24 and 12 for the paths sets in Table 2 and Table 3, respectively. Again, we do not use $n_{PO}(e_i)$ instead of $n_P(e_i)$ as a power factor because $n_P(e_i)$ increases smoothly (i., 2, 3, 4, $\ldots$), whereas the values of $n_{PO}(e_i)$ becomes 1, 3, 6, ... for the overlap of 2, 3, 4, ... paths.

An additional advantage of our similarity score is that the measure is not dependent on the total length of the paths and thus does not allow the longer paths to dominate the shorter paths having the same overlap length while finding the set of dissimilar paths.

3.4 A flexible framework

To accommodate the various preferences of users, a flexible framework to support RDP queries can work as follows. A user can select one or multiple features and a similarity measure from a given set of available options and provides a source location, a destination location, and k as query parameters. Based on these user inputs, the framework shows a path set as the answer of the RDP query. Setting $k = 1$ or selecting a single feature in the framework would allow a user to ignore the constraint of dissimilarity in the generated RDP query answer. For $k = 1$, the RDP query finds the representative paths that have the smallest values for the cost functions associated with the selected feature(s). For $k > 1$ and a single feature, the RDP query finds the k distinct paths that have k smallest values for the cost function associated with the selected feature. If a user does not select any feature, the RDP query considers all available features to find the RDPs.

4 Our approach

Our approach executes an RDP query in two phases. First, for every feature $F_i \in F$, it retrieves a set P_i of k RPs that have k smallest values for the cost function c_i. Then our approach finds P_{RD}, the set of x distinct paths by including one path from every path set in $\{P_1, P_2, \ldots, P_x\}$. P_{RD} represents the path set that minimizes the path similarity score.

4.1 Representative path computation

The straightforward way to compute k RPs for a feature is to apply any existing incremental network expansion based shortest path algorithm [23], where the cost of a path is determined based on the corresponding feature. For example, if the feature represents the number of intersections in a path then the cost of a path is considered as the number of intersections instead of the distance. The downside of this straightforward technique is that it will require x independent searches in the road network for finding k RPs for each of the x features. As a result, the same part of the road network may need to be explored multiple times, and the search would incur an extremely high processing overhead. We propose an efficient algorithm, $Find\_RP$, to compute k RPs between s and d for every feature in $F = \{F_1, F_2, \ldots, F_x\}$ with a single search in the road network.

Algorithm 1: Find_RP(s, d, F, k)

```
1  $Initialize(c_1^k, c_2^k, \dots c_x^k, V_f, first, k')$;
2  $i \leftarrow 1$;
3  $cur \leftarrow i \bmod 2$;
4  $next \leftarrow (i+1) \bmod 2$;
5  $Enqueue(Q_{cur}, p = \{s\}, c_1(p), c_2(p), \dots, c_x(p))$;
6  while $i \leq x$ do
7      $\{p, c_1(p), c_2(p), \dots, c_x(p)\} \leftarrow Dequeue(Q_{cur})$;
8      $v_l \leftarrow ExtractLast(p)$;
9      if $c_i(p) \geq {c_i}^k$ then
10         $Enqueue(Q_{next}, p, c_1(p), c_2(p), \dots, c_x(p))$;
11         while $Q_{cur} \neq \emptyset$ do
12             $\{p, c_1(p), c_2(p), \dots, c_x(p)\} \leftarrow Dequeue(Q_{cur})$;
13             $Enqueue(Q_{next}, p, c_1(p), c_2(p), \dots, c_x(p))$;
14         $i \leftarrow i + 1$;
15         $cur \leftarrow i \bmod 2$;
16         $next \leftarrow (i+1) \bmod 2$;
17         $Initialize(V_f, first, k')$;
18     else
19         $V_f[v_l] \leftarrow V_f[v_l] + 1$;
20         if $v_l = d$ then
21             $j \leftarrow i$;
22             while $j \leq x$ do
23                 $P_j, c_j^k, k', first \leftarrow Update(P_j, p, i, first)$;
24                 $j \leftarrow j + 1$;
25         else
26             if $V_f[v_l] < k'$ then
27                 for each adjacent vertex $w$ of $v_l$ do
28                     $p' \leftarrow p \cup w$;
29                     $Enqueue(Q_{cur}, p', c_1(p'), c_2(p'), \dots, c_x(p'))$;
30             else
31                 $Enqueue(Q_{next}, p, c_1(p), c_2(p), \dots, c_x(p))$;
32 return $P_1, P_2, \dots, P_x$;
```

Algorithm 1 shows the steps of $Find\_RP$. The inputs to the algorithm are a source location s, a destination location d, a set F of x features $\{F_1, F_2, \dots F_x\}$ and parameter k. The algorithm returns x RP sets $P_1, P_2, \dots, P_x$, where each RP set P_i includes k RPs between s and d for Feature F_i. The paths in P_i are ordered in ascending order based on their costs.

The algorithm iterates x times (Lines 6–31) and in the i^{th} iteration, k RPs for Feature F_i are identified. The underlying idea behind the efficiency of the algorithm is that the paths that are explored in an iteration are not explored again in the future iteration. Thus, though our algorithm requires multiple iterations, it does not explore the same part of the road network more than once and finds RP sets for x features with a single search. When an RP between s and d is identified for a feature, the path also becomes a candidate RP for other feature if its cost for other feature F_j is less than c_j^k, the k^{th} smallest cost for the corresponding feature based on already explored paths between s and d.

The algorithm uses two priority queues Q_1 and Q_2, where in the i^{th} iteration one priority queue is used to expand the network to find the RPs for feature F_i and the other one is used to store the paths for the expansion in the $(i+1)^{th}$ iteration. We denote the priority queue that is used for the network expansion in the current iteration as Q_{cur}, and the other priority queue that will be used for network expansion in the next iteration as Q_{next}. The queues alternate their roles in two consecutive iterations. For example, if Q_1 is used for the network expansion in the i^{th} iteration, it is used to store paths for future expansion in the $(i+1)^{th}$ iteration. The values of cur ($next$) alternate between 1 and 2 (2 and 1) in two consecutive iterations. The paths in Q_{cur} and Q_{next} are ordered based on the cost for Features F_i and F_{i+1}, respectively.

In addition, the algorithm uses the following variables:

- $c_1^k, c_2^k, \dots c_x^k$: Variable c_i^k represents the cost of the k^{th} RP for Feature F_i based on already explored paths. The costs $c_1^k, c_2^k, \dots c_x^k$ are initialized to ∞.
- k': Variable k' is initialized to k at the start of every iteration and later represents the number of remaining RPs that need to be identified for Feature F_i in the i^{th} iteration.
- V_f: An array of $|V|$ flags, where $|V|$ is the number of vertices in the road network, each entry $V_f[v]$ in the array corresponds to the flag for a vertex v in the road network. Each entry $V_f[v]$ is initialized to zero at the start of every iteration. $V_f[v]$ stores the number of times v is visited in an iteration i, i.e., the number of paths that go through v and have the smallest costs for feature F_i.
- $first$: Variable $first$ is initialized to 0 at the start of every iteration and becomes 1 when an RP of Feature F_i is updated in the i^{th} iteration.
- v_l: Variable v_l stores the last vertex of the last dequeued path p.

At the start, the algorithm initializes variables: $c_1^k, c_2^k, \dots c_x^k$ to ∞, each entry of V_f and $first$ to 0, k' to k, i to 1, and cur and $next$ to $i \bmod 2$ and $(i+1) \bmod 2$, respectively (Lines 1–4). Then the algorithm enqueues a path that contains source s along with its costs $c_1(p), c_2(p), \dots, c_x(p)$ to the priority queue (Line 5). In each iteration, the algorithm continues to dequeue entries from Q_{cur} until k RPs between s and d with k smallest costs for Feature F_i are identified.

After dequeueing a path p along with its costs $c_1(p), c_2(p), \dots, c_x(p)$ from Q_{cur} in the i^{th} iteration and extracting the last vertex of p in v_l (Lines 7–8), the algorithm checks whether the cost $c_i(p)$ is greater than or equal to c_i^k (Line 9). If the condition is true, then it is guaranteed that k RPs with k smallest costs for Feature F_i have been found. This is because our approach only allows a feature for which the cost of a path is greater than the cost of its subpath (Section 3.1). Thus, the algorithm dequeues the remaining paths along with their costs stored in Q_{cur} and enqueues them and p into Q_{next} (Lines 10–13), then exchanges the values of cur and $next$ (Lines 14–16), and initializes V_f, $first$, k' again (Line 17) for the $(i+1)^{th}$ iteration.

If $c_i(p)$ is smaller than c_i^k (Line 9), then the algorithm increments the visited flag V_f for the last vertex v_l of the dequeued path p by 1 (Line 19). If v_l represents destination d (Line 20), then the algorithm considers every Feature F_j for $j \geq i$ and updates its RP set P_j and c_j^k, if $c_j(p)$ is smaller than c_j^k (Lines 22–24). If $first$ is 0 and t^{th} RP of F_i is updated in P_i, the algorithm also sets k' to $k-t+1$ and $first$ to 1 using Update function (Line 23). For $t>1$, it means that the first $(t-1)$ RPs with t smallest costs have already been identified in the previous iterations.

If v_l does not represent d and $V_f[v_l] \leq k'$ (Line 26), then the algorithm forms new paths by adding every adjacent vertex w of v_l to p and enqueues them along with their costs to Q_{cur} (Lines 27–29). Two vertices are adjacent if they are connected with an edge. If $V_f[v_l] > k'$, then the algorithm enqueues p to Q_{next} as the k' paths that have k' smallest costs for feature F_i and go through v_l have been already considered (Line 31).

4.2 Representative dissimilar path set computation

A naïve solution to find the optimal RDP set is computing the similarity scores for all possible candidate RDP sets and then selecting the one that minimizes the similarity score. However, the number of possible candidate RDP sets is k^x, and considering such a huge candidate RDP sets would be computationally expensive.

We develop an efficient algorithm that does not need to consider all candidate RDP sets to find the RDP query answer. Our algorithm exploits the lower and upper bound properties of the similarity score to refine the search space. Our algorithm first finds an upper bound of the similarity score for the optimal RDP path set and then prunes the partially or completely formed candidate RDP sets whose lower bounds of the similarity score exceed the derived upper bound.

We first discuss the ways to compute the lower and upper bounds of the optimal similarity score and then present our algorithm to find the RDP set.

4.2.1 Lower bounds

Our similarity score measure for a set of paths P has the following property: the similarity score for a path set is greater or equal to the summation of the similarity scores for each pair of paths included in the path set, i.e., $sim(P) \geq \sum_{p_i,p_j \in P} sim(p_i, p_j)$. In addition to this property, we use the following minimum similarity score measures in the computation of lower bounds.

- Given a specified RP p of feature F_i and an RP set P_j of feature F_j, the minimum similarity score between p and any RP in P_j for $i \neq j$ is $sim_{min}{}^j(p) = min_{p' \in P_j} sim(p, p')$.
- Given an RP set P_i of feature F_i and an RP set P_j of feature F_j, the minimum similarity score between any RP in P_i and any RP in P_j for $i \neq j$ is $sim_{min}{}^{ij} = min_{p \in P_i \bigwedge p' \in P_j} sim(p, p')$.

Based on the available information about the paths included in an RDP set P, the lower bound of the similarity score of P is measured as follows:

Rule (i). Let a candidate RDP set P include a specified RP p for feature $F_i \in F$ and any RP for each of the other features in $F \setminus F_i$. The lower bound of the minimum similarity score of P is measured as $sim^l(P) = \sum_{F_j \in F \setminus F_i} sim_{min}{}^j(p) + \sum_{F_j,F_k \in F \setminus F_i} sim_{min}{}^{jk}$.

Rule (ii). Let a path set P' includes $x' < x$ specified paths, where one path is representative of one feature in $F' \subset F$. Let a candidate RDP set P include P' and any RP for each of the other features in $F \setminus F'$. The lower bound of the minimum similarity score of P is measured as $sim^l(P) = \sum_{(p,p') \in P'} sim(p, p') + \bigvee_{p \in P'} \sum_{F_j \in F - F'} sim_{min}{}^j(p) + \sum_{F_j,F_k \in F \setminus F'} sim_{min}{}^{jk}$.

Rule (iii). Let a candidate RDP set P include a specified RP p for every feature $F_i \in F$. The lower bound of the minimum similarity score of P is measured as $sim^l(P) = \sum_{(p,p') \in P} sim(p, p')$.

4.2.2 Upper bounds

The underlying idea behind having the upper bound of the similarity score for an RDP set is to compute actual similarity scores of z candidate RDP sets and take the minimum of these similarity scores as the upper bound. To get a tight upper bound, we select z candidate RDP sets that have the smallest values for the lower bound of the similarity score.

Specifically, the steps to compute the upper bound of the similarity score of the optimal path set are as follows:

Step 1. We first compute one candidate path set for every RP p in P_i, where $P_i \in \{P_1, P_2 \dots P_x\}$. A candidate path set P includes x distinct paths: a path p representing a feature $F_i \in F$ and a path p' for every other feature F_j in $F \setminus F_i$ such that $sim(p, p') = sim^j_{min}(p)$. Since there are x features $F_1, F_2, \dots, F_x$ and k RPs in P_i for every Feature F_i, the total number of candidate path sets is $x \times k$.

Step 2. For each candidate path set P, we compute the lower bound of the similarity score as $sim^l(P) = \sum_{(p,p') \in P} sim(p, p')$ (please see Rule (iii) in Section 4.2.1).

Step 3. We sort the candidate path sets in an ascending order based on their lower bounds of the similarity scores and select z candidate path sets that have z smallest values for the lower bound of the similarity score.

Step 4. We compute the actual similarity scores for z candidate path sets and consider the minimum similarity score as the upper bound S^u of the optimal similarity score.

4.2.3 Algorithms

Algorithm 2: Find_RDP($s, d, F, P_1, P_2, \dots, P_x$)

```
1  $end \leftarrow 0$;
2  $M_{ep}, M_{pp}, M_{pF}, M_{FF} \leftarrow CompM(P_1, P_2, \dots, P_x)$;
3  $S^u, P_{RD} \leftarrow CompUpperBound(s, d, P_1, P_2, \dots, P_x)$;
4  for each path $p \in P_1$ do
5      if $sim^l(p)) \leq S^u$ then
6          $Enqueue(Q_p, \{p\}, sim^l(p))$;
7  while $Q_p$ is not empty and $end = 0$ do
8      $\{P_c, sim^l(P_c)\} \leftarrow Dequeue(Q_p)$;
9      if $sim^l(P_c) > S^u$ then
10         $end \leftarrow 1$;
11     else
12         if $contains(P_c, F)$ then
13             if $sim(P_c) < S^u$ then
14                 $S^u, P_{RD} \leftarrow sim, P_c$;
15         else
16             for each path $p \in P_i$ do
17                 if $sim^l(P_c \cup p) \leq S^u$ then
18                     $Enqueue(Q_p, P_c \cup p, sim^l(P_c \cup p))$;
19 return $P_{RD}$;
```

Algorithm 2, Find_RDP, shows the pseudocode to find the answer of an RDP query. The inputs to the algorithm are a source location s, a destination location d, a set F of x features $\{F_1, F_2, \ldots F_x\}$, and a set of path sets $\{P_1, P_2, \ldots P_x\}$, where P_i is a set of k RPs of feature F_i. The algorithm returns the answer of the RDP query as P_{RD}.

The symbols and notations that we use in our algorithms are listed below:

- $\{e_1, e_2, \ldots, e_m\}$: The paths in $\{P_1, P_2, \ldots, P_x\}$ go through $\{e_1, e_2, \ldots, e_m\}$, the set of m distinct edges.
- M_{ep}: A matrix of $k \times x$ rows and m columns. Each row corresponds to a path p in in P_i, where $P_i \in \{P_1, P_2 \ldots P_x\}$, and each column corresponds to an edge in $\{e_1, e_2, \ldots, e_m\}$. Each cell is set to 1 if the path represented by the row go through the edge represented by the corresponding column, otherwise 0.
- $M_{pp'}$: A matrix of $k \times x$ rows and $k \times x$ columns. Each row (column) corresponds to a path p in P_i, where $P_i \in \{P_1, P_2 \ldots P_x\}$. Each cell has $sim(p, p')$, if the corresponding row represents p and the corresponding column represents p' and $p \neq p'$.
- M_{pF}: A matrix of $k \times x$ rows and x columns. Each row corresponds to a path p in P_i, where $P_i \in \{P_1, P_2 \ldots P_x\}$, and each column corresponds to a feature in $\{F_1, F_2, \ldots, F_x\}$. Each cell has ${sim_{min}}^j(p)$, if the corresponding row represents $p \in P_i$ and the corresponding column represents F_j for $i \neq j$. Note that more than one row may represent the same path p as it might be included in more than one path sets in $\{P_1, P_2, \ldots, P_x\}$. In such scenarios, ${sim_{min}}^j(p)$ of different cells may differ based on which path set p belongs to.
- M_{FF}: A matrix of x rows and x columns. Each row (column) corresponds to a feature in $\{F_1, F_2, \ldots, F_x\}$. Each cell has ${sim_{min}}^{ij}$ if the corresponding row represents F_i and the corresponding column represents F_j and $i \neq j$.
- S^u: the upper bound of the similarity score for an RDP set generated from representative path sets $P_1, P_2, \ldots, P_x$ for a source location s and a destination location d.
- $sim^l(P)$: the lower bound of the similarity score for a path set P.

The algorithm first initializes end to 0, computes the matrices $M_{ep}, M_{pp}, M_{pF}, M_{FF}$ (Lines 1–2). The algorithm then finds the upper bound S^u of the similarity score and the corresponding path set P_{RD} using the steps shown in Section 4.2.2 (Line 3). The algorithm uses a priority queue Q_p, where entries are ordered in ascending order based on the lower bound of the similarity score of the path set of the entries. For every path $p \in P_1$, a path set is enqueued into Q_p, if the lower bound $sim^l(p)$ of the similarity score of the path set that includes p is smaller than or equal to S^u (Lines 4–6). After this step, the algorithm iterates until the actual P_{RD} is identified (Lines 7-19).

In each iteration, a path set P_c with the smallest lower bound of the similarity score $sim^l(P_c)$ is dequeued from Q_p as $\{P_c, sim^l(P_c)\}$ (Line 8). If $sim^l(P_c) > S^u$ then the path set P_{RD} has the smallest similarity score, i.e., the answer is already found and thus, end is set to 1 (Lines 9–10). Otherwise, the algorithm checks whether P_c includes x paths, one path from every feature (Line 12). If yes and the actual similarity score $sim(P_c)$ of P_c is smaller than S^u, then the algorithm updates S^u and P_{RD} with $sim(P_c)$ and P_c, respectively (Lines 12–14). If P_c does not include x paths, then the algorithm enqueues a path set $P_c \cup p$ along with its lower bound $sim^l(P_c \cup p)$ of the similarity score for every $p \in P_i$, if $sim^l(P_c \cup p)$ is smaller than or equal to S^u (Lines 16–18).

Approximation. We select the candidate RDP set that provides the upper bound to approximate the RDP query answer and further reduces the query processing overhead. We use Function CompUpperBound (Line 3 of Algorithm 2) to approximate the RDP set.

5 Experiments

In this section, we present our experimental results to show the effectiveness of RDP queries in accommodating the human movement dynamics in road networks and the efficiency of our solution to process RDP queries.

We introduce the RDP query in this paper and there is no existing solution that can find the answer of an RDP query (please see Section 2 for details). Thus, we evaluate the performance of our solution in experiments by varying different parameters: the number of representative paths (Section 5.1), features (Section 5.2), trip length (Section 5.3) and context (Section 5.4). We also investigate the feasibility of using an approximation of the RDP algorithm in experiments (Section 5.1). In addition, we show that our approach is significantly faster than a naïve approach (Section 5.1).

Furthermore, we compare the performance of RDPs against the shortest paths for an indication of the quality of our RDPs (Section 5.5). We show that RDPs perform significantly better at capturing the actual paths taken by real navigators compared to the shortest paths. The works on finding the alternative or diversified paths (e.g., [9, 10, 21]) focused on minimizing the individual or total path length while ensuring that the pair of paths satisfy the constraint of the maximum similarity threshold. Since the computed paths by these works widely vary for different similarity thresholds, instead of comparing our RDPs with the generated paths by each of these works independently for different similarity thresholds, we compare our RDPs with the shortest paths. Finally, we summarize our experiment results (Section 5.6).

Datasets. We used Beijing taxi trajectory dataset [32, 33] for our experiments. This dataset contains GPS trajectories of 10,357 taxis over seven days (February 2 to February 8, 2008) with an average sampling interval of about 177 seconds with a distance of about 623 meters. We randomly selected 500 taxis and generated 40,027 trips for these taxis.

To generate the trips from the GPS points of taxi trajectories, we considered the maximum sampling threshold as five minutes since, in more than 95 percent of cases, the time interval between two sample GPS points in the dataset is less than or equal to five minutes. Thus, we can assume that a time interval of more than five minutes indicates the beginning of a new trip. We also considered the presence of the same GPS points multiple times continuously as the end of a trip. While extracting the trips from Beijing taxi trajectory dataset, we also identified the context (e.g. weekday or weekend, day or night) of the trips.

In the next step, we matched the GPS points of the extracted trips to the Beijing road network using the algorithm proposed in [22]. We collected the road network of Beijing from OpenStreetMap[1] using the OSMnx tool[2] in the Python language. It contains a total of 34749 vertices and 80602 edges. Each edge of our road network has the following associated information: start and end vertices, length, degree of start vertex, degree of end vertex, and whether the road represented by the edge is a highway, or a residential road.

Parameters. We use six features: length (F_1), intersection degree (F_2), number of intersections (F_3), highway length (F_4), residential road length (F_5) and tortuosity (F_6). All of these features, except the tortuosity, are directly found in the road network data. To compute the angle between two consecutive edges in a path for the tortuosity feature, we convert the GPS coordinates of the endpoints of the edges to the points in the geometric plane and then use the following formula: $cos\theta = \frac{\vec{A}\cdot\vec{B}}{\mathbf{A}\times\mathbf{B}}$, where $\vec{A}$ and $\vec{B}$ represent two

[1]www.openstreetmap.org

[2]https://wiki.openstreetmap.org/wiki/OSMnx

vectors created from the two edges, respectively. The tortuosity of a path is measured as the summation of the angles between every pair of consecutive edges in the path. The costs of a path for other features (F_1–F_5) are found by summing up the costs associated with the edges or the vertices in the path.

Besides features, we vary k, trip length, and the context in our experiments. We vary k as 1, 2, 4, and 8 and set its default value to 4. We divide the generated taxi trips based on trip length L: (0km-5km], (5km-10km] and (10km-∞]. We consider weekdays, weekends, days, nights, peaks, and off-peaks as context. When we vary length range and/or context in our experiments, we consider the endpoints of the taxi trips of the corresponding length range and/or context as source-destination pairs of RDP queries.

To compute the upper bound in Find_RDP, we consider z candidate path sets that have z smallest values for the lower bound of the similarity score. We set z to 10 because we vary z from 1 to 20 and find that the average vertex coverage remains almost the same, whereas the processing time initially decreases slightly up to the value of 10 and then starts increasing again.

Measures. We measure the efficiency of our algorithms in terms of the average processing time. We use a machine with the Intel Core i7-8565U CPU @ 1.80 GHz processor and 16 GB RAM, Windows 10 64-bit operating system, and CLion to run the queries.

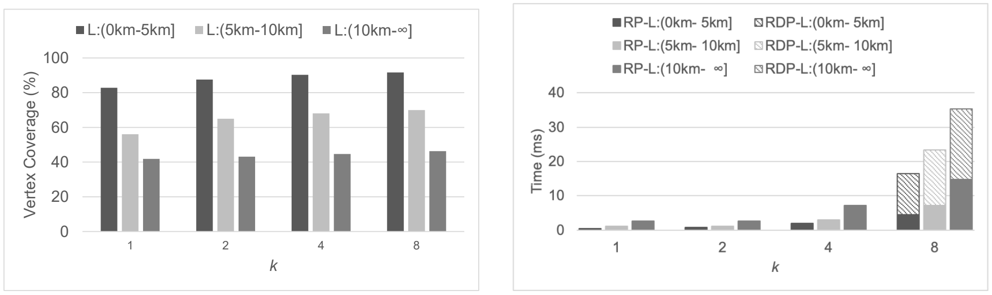
We consider the original taxi trips as representative of actual human movement dynamics between different pairs of source and destination locations and compute the efficacy of the RDP query by measuring the *average vertex coverage* and other statistical measures with respect to the original taxi trips. The source and destination locations of RDP queries come from the endpoints of the taxi trips. Given a taxi trip and an RDP set for a source and destination pair, the vertex coverage of the RDP set is measured as the percentage of vertices of the taxi trip that is included in the paths of the RDP set. The aim of an RDP query is to provide a set of paths that in combination cover the breadth of routes that may be taken by the users to travel between a source and destination locations. Thus, the higher the average vertex coverage rate, the better the performance of the RDP query, because it indicates the returned RDPs better matched the real routes taken. In our experiments, we measure the vertex coverage of an RDP set with respect to the taxi trip from which the source and destination locations of the RDP query sample is derived.

For every experiment, we consider 1000 RDP query samples, and measure the average performance (i.e., average processing time and average vertex coverage) of our solution.

5.1 Effect of the number of representative paths k

Figure 3(a) and Figure 3(b) show the average vertex coverage and processing time, respectively, for varying k for different trip length range L. We observe that the average vertex coverage slightly increases with the increase of k for every L. Our approach considers k RPs for every feature. However, the RPs of a feature normally vary by a few vertices (e.g., paths in Figure 8 for the length feature). Since we select only one path from each feature, the final average vertex coverage changes very slightly if we increase the value of k. The high vertex coverage in our solution mainly comes from considering the dissimilar RPs of different features, not the number of RPs we consider for each feature.

For the processing time, we find that our approach can compute RDPs with a small processing cost (0.250-35.381 seconds). The rate of increase of the processing time of both of our algorithms: Find_RP and Find_RDP (denoted as RP and RDP) is very low for changing

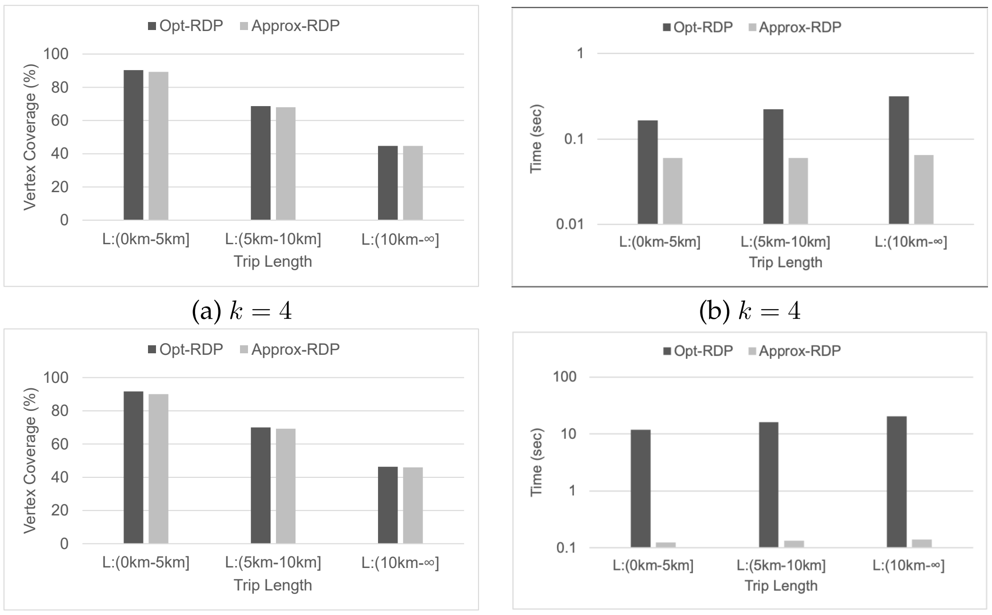


(a) (b)

Figure 3: Effect of varying k on average (a) vertex coverage and (b) processing time

k from 1 to 4 and is very high for changing k from 4 to 8. Figure 3(b) also shows that RDP contributes more than RP in the total processing time.

Approx-RDP vs. Opt-RDP. We compare our optimal algorithm (Algorithm 2) for finding the RDPs with an approximation that considers the RDPs identified as P_{RD} while computing the upper bound (Line 3 in Algorithm 2). In the graphs, we denote them as Optimal-RDP and Approx-RDP, respectively. In Figure 4, we show the results for $k = 4$ and $k = 8$ because for smaller k, the processing time of Optimal-RDP is extremely low and there is no need for an approximation algorithm (Figure 3(b)). We observe that though the average vertex coverage for both Optimal-RDP and Approx-RDP are almost the same, the processing time of Approx-RDP reduces significantly, for example, Approx-RDP is, on average, 120 times faster than Optimal-RDP for $k = 8$. Similar to Optimal-RDP, we set the default value of z to 10 for Approx-RDP.



(a) $k = 4$ (b) $k = 4$

(c) $k = 8$ (d) $k = 8$

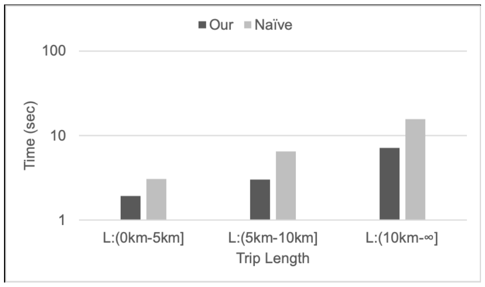
Figure 4: Approx-RDP vs. Opt-RDP

We also analyze how similar the RDPs of Approx-RDP are to those of Optimal-RDP. For every feature, we consider the paths of the corresponding feature included in the optimal and approximate RDP sets, and then, calculate the percentage of common edge length compared to the total length of unique edges included in two paths. Table 4 shows that the percentage of the common path portion is quite high for $k = 4$ and $k = 8$, hence the feature-wise paths included in the optimal and approximate RDP sets are similar.

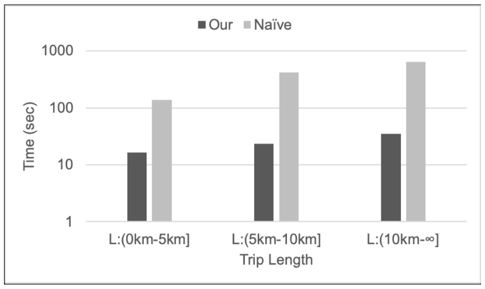
	Length Range	F1	F2	F3	F4	F5	F6
$k = 4$	**L:(0km-5km]**	74.46	74.77	71.56	75.73	79.37	75.75
	L:(5km-10km]	88.30	86.40	88.38	85.95	91.17	86.50
	L:(10km-∞]	92.70	91.44	90.78	89.57	94.31	89.24
$k = 8$	**L:(0km-5km]**	62.70	59.60	59.49	62.03	69.18	62.65
	L:(5km-10km]	80.52	77.92	78.10	74.01	84.18	74.31
	L:(10km-∞]	87.22	82.44	82.72	77.73	89.31	82.09

Table 4: Feature wise common path portion (%) in the optimal and approximate RDP sets

Our approach vs. naïve approach. We compare our approach (Algorithm 1 and Algorithm 2) to find the answer of an RDP query with a naïve approach. The naïve approach first computes k RPs for every feature using Algorithm 1, and then calculates the similarity score of each possible path set that includes at least one RP for each feature. The naïve approach considers the path set with the minimum similarity score as the answer of the RDP query. Since the possible number of path combinations is exponential, the naïve approach is prohibitively expensive. Experiment results show that our approach is on average 8.4 times faster than the naïve approach (Figure 5). Note that both, our approach and the naïve approach, return the optimal answer and thus, the average vertex coverage achieved by them are same.



(a) $k = 4$



(b) $k = 8$

Figure 5: Our approach vs. Naïve approach

5.2 Effect of features

Figure 6 shows the average vertex coverage by each feature's RP in the RDP set for different trip length ranges. We observe that the average vertex coverage is the maximum when all paths of the RDP set are considered. However, the same vertices may be covered by multiple paths in the RDP set. Table 5 shows the average vertex coverage by excluding

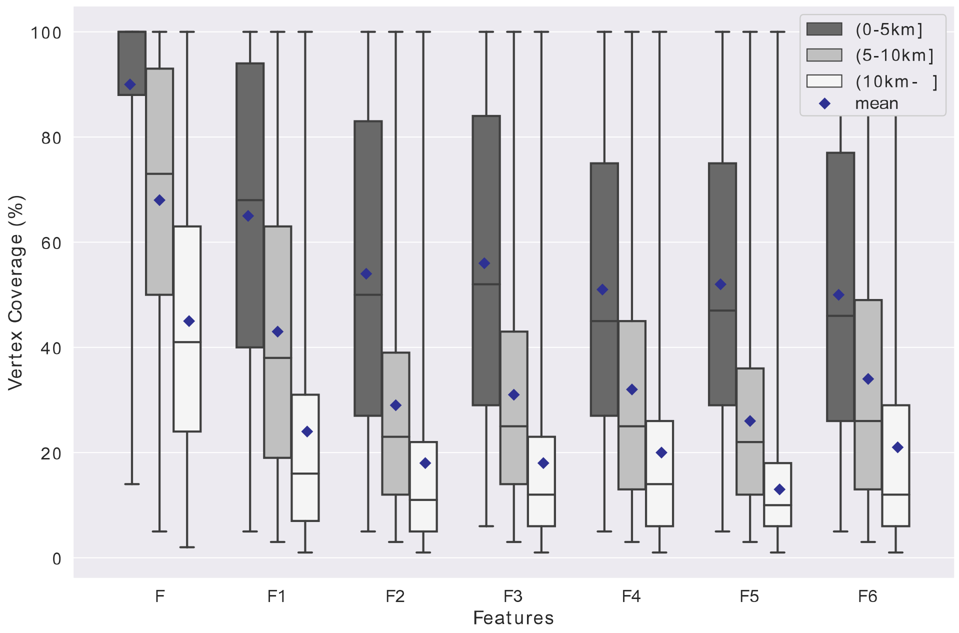


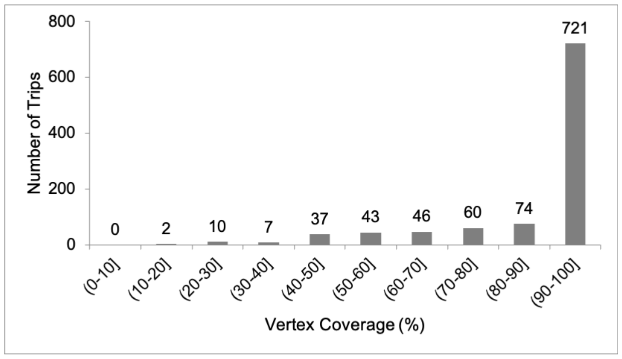
Figure 6: Statistics on vertex coverage: min, median, max, interquartile (25%, 75%) and mean ($k = 4$)

each feature's RP in the RDP set. In our experiment, each RDP set consists of six RPs, one from a feature in F. When we exclude one feature's RP, we find the vertex coverage as the percentage of vertices of the taxi trip that is included in the remaining five paths of the RDP set. In Table 5, we observe that the average vertex coverage achieved by all paths in the RDP set reduces by ignoring any feature's RP in the RDP set, which means that every feature's RP in the RDP set exclusively includes some of the vertices of the taxi trip.

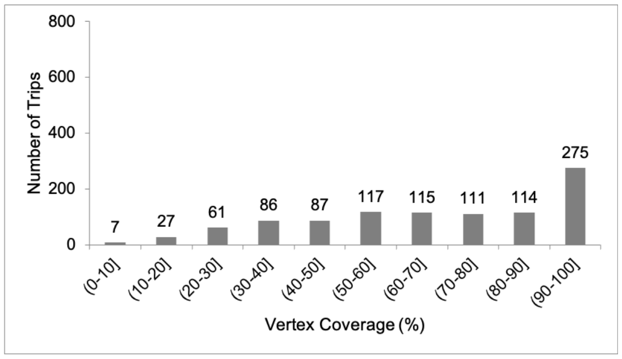
Length Range/Feature	F	$F \setminus F_1$	$F \setminus F_2$	$F \setminus F_3$	$F \setminus F_4$	$F \setminus F_5$	$F \setminus F_6$
L:(0km-5km]	90.37	88.34	89.59	89.53	88.86	88.30	85.95
L:(5km-10km]	68.04	62.10	66.78	66.80	66.18	64.70	59.62
L:(10km-∞]	44.75	39.87	43.71	43.62	43.24	41.88	37.44

Table 5: The average vertex coverage (%) achieved by the RDPs of all features and by excluding the RDP of $F_1/F_2/F_3/F_4/F_5/F_6$ for $k = 4$

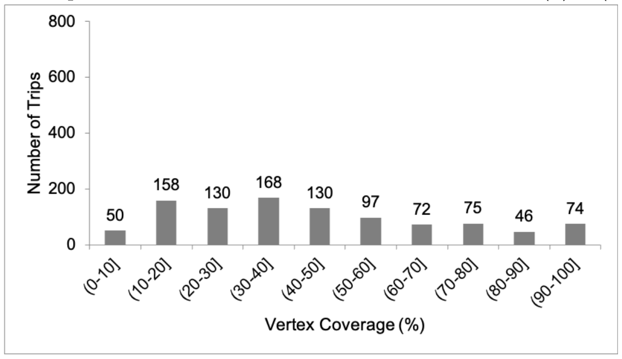
5.3 Effect of trip length



(a) L:(0km-5km]



(b) L:(5km-10km]



(c) L:(10km-∞]

Figure 7: Histograms showing the number of trips for varying vertex coverage (%) for three trip length ranges ($k = 4$)

Figure 3 shows that the average vertex coverage decreases and the processing time increases with the increase of the trip length. The processing time increases because our algorithm needs to explore more edges before reaching the destination while identifying k RPs for the increased trip length. The average vertex coverage decreases because the increased trip length means a higher probability of deviation from the original trip.

In Figure 7, the histograms show the number of trips for varying vertex coverage (%) for three trip length ranges, where we set k to default value 4. For L:(0km-5km], our solution achieves more than 90% vertex coverage for 72.1% trips. For L:(0km-5km], our solution achieves more than 50% vertex coverage for 73.2% trips, among which 27.5% achieve more than 90% vertex coverage. For L:(10km-∞], only 36.4% trips achieve more than 50% vertex coverage.

5.4 Effect of context

Table 6 shows that the average vertex coverage does not show any significant variation in different contexts like weekdays, weekends, day, night, peak, and off-peak. The variation in the average vertex coverage for L:(10km-∞] is (45%-52%), which is higher than those of both L:(0km-5km] and L:(5km-10km]. The variation in the average vertex coverage for L:(0km-5km] and L:(5km-10km] are 89%–90% and 67%–68%, respectively. In this set of experiments, we randomly selected 1000 RDP query samples for every context and set k to default value 4. The variation in the average vertex coverage might change if the same set of RDP query samples (i.e., source-destination pairs) is used for all contexts. However, the dataset does not include sufficient taxi trips for different contexts with the same pairs of source and destination for running experiments.

Length Range/Time	**Weekdays**	**Weekend**	**Day**	**Night**	**Peak**	**Off-peak**
L:(0km-5km]	89.04	89.7	90.41	90.23	90.41	89.55
L:(5km-10km]	66.99	67.49	68.42	69.06	67.98	66.83
L:(10km-∞]	47.88	47.96	45.32	46.46	51.15	52.31

Table 6: Effect of context on average vertex coverage (%) ($k = 4$)

5.5 Representative dissimilar paths vs. shortest paths

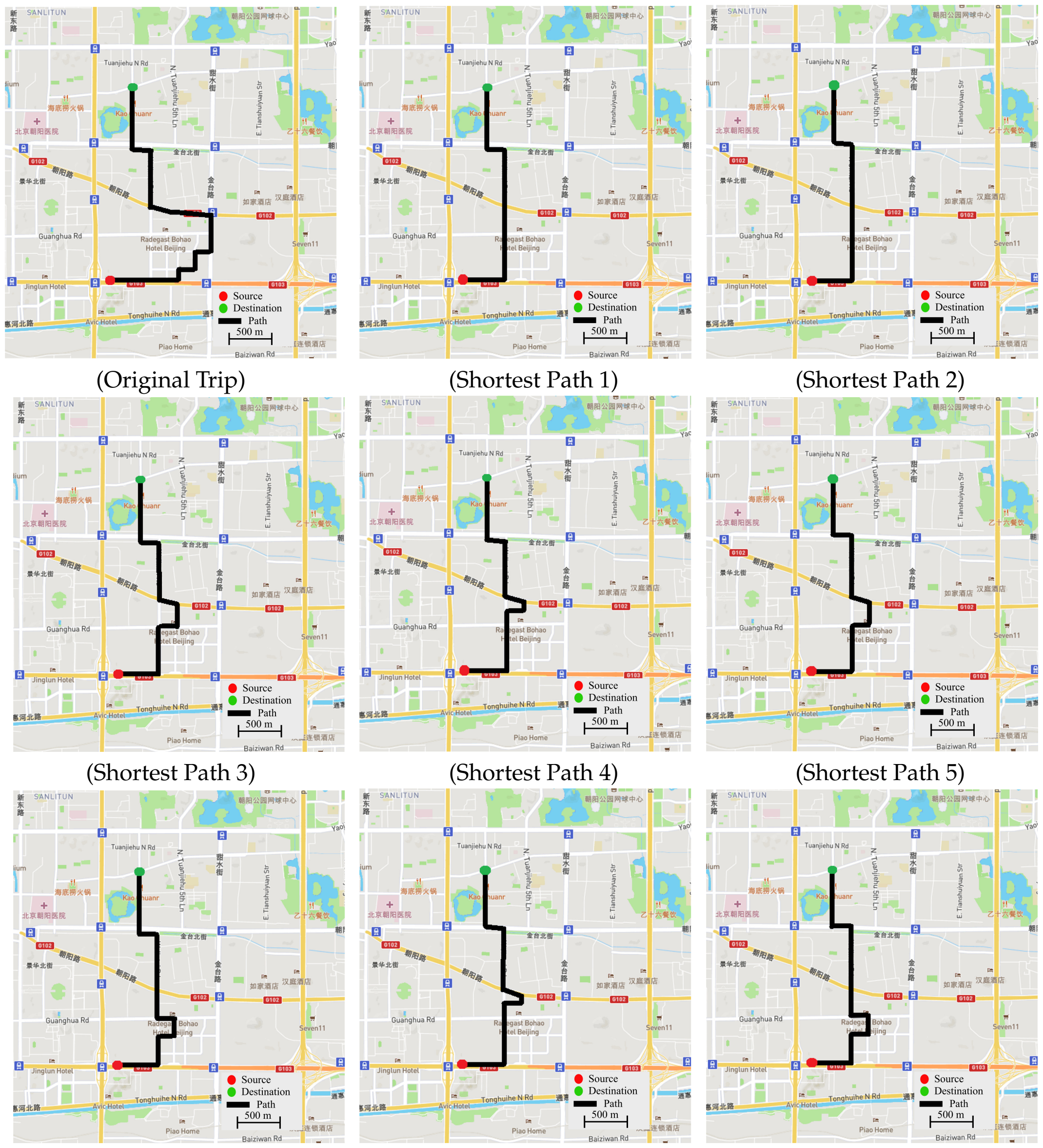
m	Number of trips **L:(0km-5km]**	Number of trips **L:(5km-10km]**	Number of trips **L:(10km-∞]**
1	414	138	55
2	103	28	13
3	61	24	15
4	27	21	9
5	33	20	12
6	19	8	6
7-10	60	40	25
11-20	60	75	22
20+	223	646	843

Table 7: Required m shortest paths to achieve the same vertex coverage as RDPs ($k = 4$)

In these experiments, for each trip length range, we randomly select 1000 taxi trips and consider the endpoints of these trips as source-destination pairs of RDP queries. We compute the top m shortest paths between the source and destination locations of each RDP query, where the vertex coverage achieved by the top $m - 1$ shortest paths with respect to the taxi trip is smaller than the vertex coverage achieved by the RDPs. The vertex coverage of both RDPs and shortest paths are computed with respect to the taxi trip from which the source and destination locations of the RDP query are considered. The vertex coverage of the top m shortest paths is the percentage of vertices of the taxi trip that is included in the top m shortest paths.

In Table 7, we show the number of taxi trips for which the top-m shortest paths achieve the same or more vertex coverage as RDPs for different m. In Figure 9 of Appendix A.1, we plot the vertex coverage (%) achieved by each RDP set and corresponding m. An RDP query returns a set of six paths, where one path corresponds to a feature in F and $|F| = 6$. If m is shown as a range (e.g., 7 to 10 in the seventh row), then it is guaranteed that the top 6 shortest paths achieve smaller vertex coverage than RDPs.

For the trip length range L:(0km-5km], more than six shortest paths ($m > 6$) are required for 34.3% (i.e., 60+60+223=343 trips out of 1000, please see the first column of the last three



(Original Trip) (Shortest Path 1) (Shortest Path 2)

(Shortest Path 3) (Shortest Path 4) (Shortest Path 5)

(Shortest Path 6) (Shortest Path 7) (Shortest Path 8)

Figure 8: Original Taxi Path and Shortest Paths

rows of Table 7) trips to obtain at least the vertex coverage achieved by RDPs. Similarly, for the range L:(5km-10km] and L:(10km-∞], more than six ($m > 6$) shortest paths are required for 76.1% and 89% trips, respectively, to obtain at least the vertex coverage achieved by RDPs. More importantly, for the range L:(5km-10km] and L:(10km-∞], more than 20 ($m > 20$) shortest paths are required for 64.6% and 84.3% trips, respectively, to obtain at least the vertex coverage achieved by RDPs.

Finding such a large number of shortest paths is not realistic as it would incur an extremely high processing overhead. Thus, our solution based on RDPs is significantly better than that based on the shortest paths. The reason behind large m is that the shortest paths do not show significant variation. Figure 8 shows an example taxi trip from the dataset and eight shortest paths considering the source-destination locations of the taxi trip. In this example, the second shortest path is found by only changing a small road in the first shortest path. Figure 1 shows six RDPs considering six features (F_1–F_6), and the paths have the same source and destination locations as the taxi trip shown in Figure 8. The percentage of the vertices of the original trip (Figure 8(a)) that is covered by six RDPs is much higher than that of the six shortest paths.

5.6 Summary

We provide an experimental analysis of the efficiency of our approach, showing that the computation is significantly faster than a naïve approach. We compute the vertex coverage of a returned RDP set in comparison with real human paths (taxi routes), showing that RDPs are able to reasonably accurately represent those real paths. The high average vertex coverage achieved in our experiments validates the effectiveness. We also show that compared with k-shortest paths, RDP result sets are significantly better at capturing real human movement.

6 Conclusion

We introduced a novel query type in road networks called the representative dissimilar path (RDP) query. Unlike existing works on finding the dissimilar paths, a key advantage of the RDP query is that it does not require users to specify the required dissimilarity threshold, which is hard to estimate. We formulated a new measure to quantify the dissimilarity among a set of paths by considering both the length and location of path overlaps. We addressed the problem of RDP queries in two phases: we first compute k RPs for every road feature, and then we find the RDPs from the generated RPs. Our search space refinement techniques allow our algorithms to find the RPs and RDPs with a significantly reduced processing overhead. We also developed an approximation algorithm to further reduce the processing overhead for finding RDPs in return of sacrificing the accuracy slightly, especially for the scenario when k is large. Experiments validated the efficacy of RDP queries in accommodating the human movement dynamics and the efficiency of our proposed solution for RDP queries. Exploiting different road features and our novel path set similarity measure play key roles for RDPs to achieve high coverage of human movements.

In this paper, we have only considered the static features of the roads. In the future, we plan to investigate the impact of dynamic features like road traffic, road safety and weather on the movements of travelers on roads [16,29]. We will also exploit representative dissimilar paths to visit different point of interests (e.g., restaurants, ATM booths and gas stations) and plan trips [15,25].

References

[1] Abraham, I., Delling, D., Goldberg, A. V., and Werneck, R. F. Alternative routes in road networks. *ACM Journal of Experimental Algorithmics 18* (2013), 1.3:1–1.3:17. doi:10.1145/2444016.2444019.

[2] AKGÜN, V., ERKUT, E., AND BATTA, R. On finding dissimilar paths. *European Journal of Operational Research 121*, 2 (2000), 232–246. doi:10.1016/S0377-2217(99)00214-3.

[3] BADER, R., DEES, J., GEISBERGER, R., AND SANDERS, P. Alternative route graphs in road networks. In *Proc. International ICST Conference on Theory and Practice of Algorithms in (Computer) Systems* (2011), pp. 21–32. doi:10.1007/978-3-642-19754-3_5.

[4] CEIKUTE, V., AND JENSEN, C. S. Routing service quality - local driver behavior versus routing services. In *Proc. International Conference on Mobile Data Management* (2013), pp. 97–106. doi:10.1109/MDM.2013.20.

[5] CHANG, C.-W., CHEN, C.-D., AND CHUANG, K.-T. Queries of k-discriminative paths on road networks. *Knowledge and Information Systems* (2019), 1–30. doi:10.1007/s10115-019-01397-4.

[6] CHEN, Z., SHEN, H. T., AND ZHOU, X. Discovering popular routes from trajectories. In *Proc. IEEE International Conference on Data Engineering* (2011), pp. 900–911. doi:10.1109/ICDE.2011.5767890.

[7] CHONDROGIANNIS, T., BOUROS, P., GAMPER, J., AND LESER, U. Alternative routing: k-shortest paths with limited overlap. In *Proc. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2015), pp. 68:1–68:4. doi:10.1145/2820783.2820858.

[8] CHONDROGIANNIS, T., BOUROS, P., GAMPER, J., AND LESER, U. Exact and approximate algorithms for finding k-shortest paths with limited overlap. In *Proc. International Conference on Extending Database Technology* (2017), pp. 414–425. doi:10.5441/002/edbt.2017.37.

[9] CHONDROGIANNIS, T., BOUROS, P., GAMPER, J., LESER, U., AND BLUMENTHAL, D. B. Finding *k*-dissimilar paths with minimum collective length. In *Proc. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2018), pp. 404–407. doi:10.1145/3274895.3274903.

[10] CHONDROGIANNIS, T., BOUROS, P., GAMPER, J., LESER, U., AND BLUMENTHAL, D. B. Finding k-shortest paths with limited overlap. *The VLDB Journal 29*, 5 (2020), 1023–1047. doi:10.1007/s00778-020-00604-x.

[11] DAI, J., YANG, B., GUO, C., AND DING, Z. Personalized route recommendation using big trajectory data. In *Proc. IEEE International Conference on Data Engineering* (2015), pp. 543–554. doi:10.1109/ICDE.2015.7113313.

[12] ERKUT, E., TJANDRA, S. A., AND VERTER, V. Chapter 9 hazardous materials transportation. In *Transportation*, vol. 14 of *Handbooks in Operations Research and Management Science*. Elsevier, 2007, pp. 539–621. doi:10.1016/S0927-0507(06)14009-8.

[13] ERKUT, E., AND VERTER, V. Modeling of Transport Risk for Hazardous Materials. *Operations Research 46*, 5 (1998), 625–642. doi:10.1287/opre.46.5.625.

[14] GUO, C., YANG, B., HU, J., AND JENSEN, C. S. Learning to route with sparse trajectory sets. In *Proc. IEEE International Conference on Data Engineering* (2018), pp. 1073–1084. doi:10.1109/ICDE.2018.00100.

[15] HASHEM, T., HASHEM, T., ALI, M. E., AND KULIK, L. Group trip planning queries in spatial databases. In *International Symposium on Spatial and Temporal Databases* (2013), pp. 259–276. doi:10.1007/978-3-642-40235-7_15.

[16] ISLAM, F. T., HASHEM, T., AND SHAHRIYAR, R. A privacy-enhanced and personalized safe route planner with crowdsourced data and computation. In *Proc. IEEE International Conference on Data Engineering* (2021), pp. 229–240. doi:10.1109/ICDE51399.2021.00027.

[17] JONES, A. H. Method of and apparatus for generating routes (US patent 8,249,810), August 2012.

[18] KRIEGEL, H., RENZ, M., AND SCHUBERT, M. Route skyline queries: A multi-preference path planning approach. In *Proc. IEEE International Conference on Data Engineering* (2010), pp. 261–272. doi:10.1109/ICDE.2010.5447845.

[19] LI, L., CHEEMA, M. A., ALI, M. E., LU, H., AND TANIAR, D. Continuously monitoring alternative shortest paths on road networks. *Proceedings of the VLDB Endowment 13*, 11 (2020), 2243–2255. doi:10.14778/3407790.3407822.

[20] LI, X., CONG, G., AND CHENG, Y. Spatial transition learning on road networks with deep probabilistic models. In *Proc. IEEE International Conference on Data Engineering* (2020), pp. 349–360. doi:10.1109/ICDE48307.2020.00037.

[21] LIU, H., JIN, C., YANG, B., AND ZHOU, A. Finding top-k shortest paths with diversity. *IEEE Transactions on Knowledge and Data Engineering 30*, 3 (2018), 488–502. doi:10.1109/TKDE.2017.2773492.

[22] NEWSON, P., AND KRUMM, J. Hidden markov map matching through noise and sparseness. In *Proc. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2009), pp. 336–343. doi:10.1145/1653771.1653818.

[23] PAPADIAS, D., ZHANG, J., MAMOULIS, N., AND TAO, Y. Query processing in spatial network databases. In *Proc. International Conference on Very Large Data Bases* (2003), pp. 802–813. doi:10.1016/B978-012722442-8/50076-8.

[24] SAMET, H., SANKARANARAYANAN, J., AND ALBORZI, H. Scalable network distance browsing in spatial databases. In *Proc. ACM SIGMOD International Conference on Management of Data* (2008), pp. 43–54. doi:10.1145/1376616.1376623.

[25] SOMA, S. C., HASHEM, T., CHEEMA, M. A., AND SAMROSE, S. Trip planning queries with location privacy in spatial databases. *World Wide Web 20*, 2 (2017), 205–236. doi:10.1007/s11280-016-0384-2.

[26] TIAN, Y., LEE, K. C. K., AND LEE, W. Finding skyline paths in road networks. In *Proc. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2009), pp. 444–447. doi:10.1145/1653771.1653840.

[27] WANG, J., WU, N., ZHAO, W. X., PENG, F., AND LIN, X. Empowering A* search algorithms with neural networks for personalized route recommendation. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2019), pp. 539–547. doi:10.1145/3292500.3330824.

[28] WANG, J., WU, N., AND ZHAO, X. Personalized route recommendation with neural network enhanced A* search algorithm. *IEEE Transactions on Knowledge and Data Engineering* (2021). doi:10.1109/TKDE.2021.3068479.

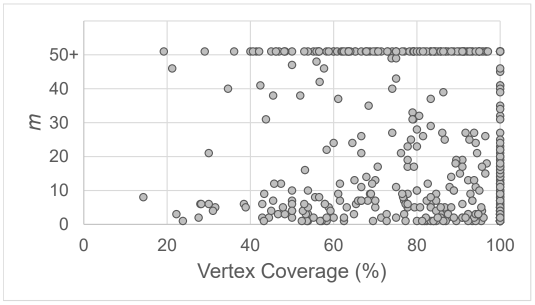
[29] WANG, Y., LI, G., AND TANG, N. Querying shortest paths on time dependent road networks. *Proceedings of the VLDB Endowment 12*, 11 (2019), 1249–1261. doi:10.14778/3342263.3342265.

[30] WEI, L., CHANG, K., AND PENG, W. Discovering pattern-aware routes from trajectories. *Distributed Parallel Databases 33*, 2 (2015), 201–226. doi:10.1007/s10619-013-7139-1.

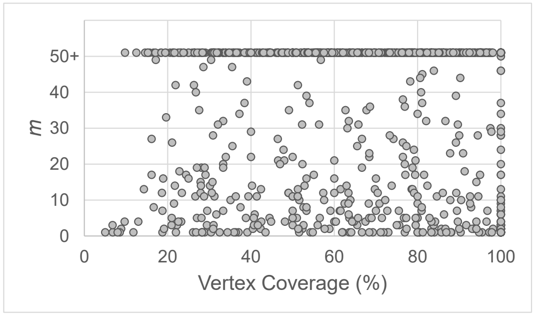
[31] WEI, L., ZHENG, Y., AND PENG, W. Constructing popular routes from uncertain trajectories. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2012), pp. 195–203. doi:10.1145/2339530.2339562.

[32] YUAN, J., ZHENG, Y., XIE, X., AND SUN, G. Driving with knowledge from the physical world. In *Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2011), pp. 316–324. doi:10.1145/2020408.2020462.

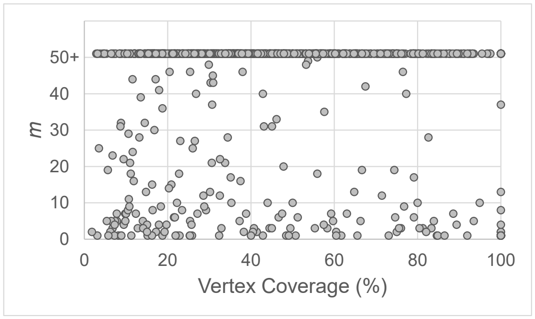
[33] YUAN, J., ZHENG, Y., ZHANG, C., XIE, W., XIE, X., SUN, G., AND HUANG, Y. T-drive: driving directions based on taxi trajectories. In *Proc. ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* (2010), pp. 99–108. doi:10.1145/1869790.1869807.



(a) L:(0km-5km]



(b) L:(5km-10km]



(c) L:(10km-∞]

Figure 9: Vertex coverage (%) by each RDP set vs. m

A Appendix

A.1 Vertex coverage of RDPs vs. top m shortest paths

In Figure 9, a point represents the vertex coverage (%) achieved by each RDP set and the value of m for the required top shortest paths. We observe that there is no specific correlation between m and the percentage of vertex coverage.