Distributed and Parallel Databases (2023) 41:603–638
https://doi.org/10.1007/s10619-022-07413-x

Searching semantically diverse paths

Xu Teng[1] · **Goce Trajcevski**[1] · **Andreas Züfle**[2]

Accepted: 5 May 2022 / Published online: 15 June 2022

Abstract

Location-Based Services are often used to find proximal Points of Interest (PoIs)—e.g., nearby restaurants and museums, police stations, hospitals, etc.—in a plethora of applications. An important recently addressed variant of the problem not only considers the distance/proximity aspect, but also desires semantically diverse locations in the answer-set. For instance, rather than picking several close-by attractions with similar features—e.g., restaurants with similar menus; museums with similar art exhibitions—a tourist may be more interested in a result set that could potentially provide more diverse types of experiences, for as long as they are within an acceptable distance from a given (current) location. Towards that goal, in this work we propose a novel approach to efficiently retrieve a path that will maximize the semantic diversity of the visited PoIs that are within distance limits along a given road network. Our approach allows to specify both a start and terminal location to return a (non-necessarily shortest) path that maximizes diversity rather than only minimizing travel cost, thus providing ample applications in tourist route recommendation systems. We introduce a novel indexing structure—the *Diversity Aggregated R-tree*, based on which we devise efficient algorithms to generate the answer-set—i.e., the recommended locations among a set of given PoIs—relying on a greedy searching strategy. Our experimental evaluations conducted on real datasets demonstrate the benefits of the proposed methodology over the baseline alternative approaches.

Keywords Diversity · Trajectories · Road networks · Indexing

✉ Xu Teng
xuteng@iastate.edu

✉ Goce Trajcevski
gocet25@iastate.edu

✉ Andreas Züfle
azufle@gmu.edu

[1] Department of Electrical and Computer Engineering, Iowa State University, Ames, Iowa 50011, USA

[2] Department of Geography and Geoinformation Science, George Mason University, Fairfax, Virginia 22030, USA

1 Introduction

Since the late 1990s, many applications relying on *Location-Based Services* (LBS) have targeted the search for *Points of Interest* (PoIs)—e.g., tourist attractions and restaurants—in the vicinity of their users. Since traveling cost, in terms of distance or travel-time, is an important factor when selecting PoIs, significant amount of research efforts have been invested into distance-oriented queries such as range queries and *k-Nearest Neighbor* (*k*NN) queries [1–3]. However, in addition to the proximity, the semantics of PoI is often an influential factor when planning one's motion and activities [4].

While modeling and querying of the, so-called, semantic or activity trajectories has been a subject of intense research in the past decade [4–6], the semantic aspect was typically used to augment the traditional searches used in typical spatial and spatio-temporal queries (range, *k*NN, etc).

In this work we are taking up a novel variant of the problem—namely, coupling the proximity constraints (with respect to the the querying user's location) with the diversity of the semantic descriptors of the PoI, in a manner that considers the cost of the travel.

Although traveling cost, i.e., distance/time, is a significant factor to choose PoIs, people tend to consider incorporating semantically diverse options within acceptable distance. For instance, a tourist might prefer selecting multiple attractions which not only are within a given distance range but also exhibit different types of tourist experiences. To provide an intuition in the realm of LBS, consider the following:

Example 1 Consider the scenario depicted in Fig. 1, illustrating a user at location Q who is searching for three tourist attractions to visit. The user specifies a maximum distance, indicated by the dashed circle, that he/she is willing to travel.

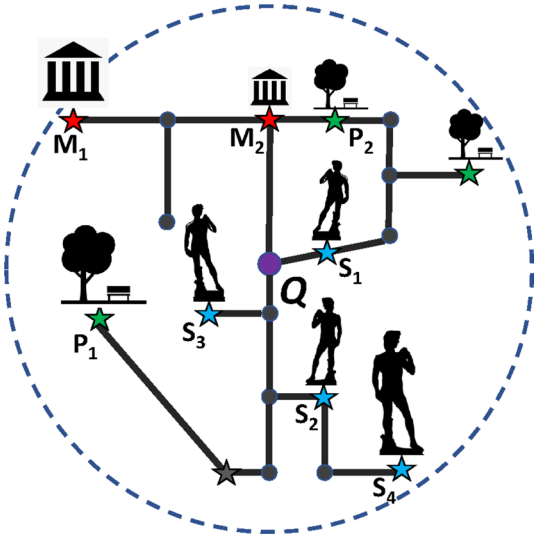


Fig. 1 Running example of diverse path search

Processing this query would return the answer set $T_1 = \{S_1, S_2, S_3\}$, consisting of 3 nearest PoIs as the user indicated that $k = 3$ is a limit of the number of PoIs. However, one can readily see that in this case, all three returned PoIs are monuments/statues. If the user would like a more diverse experience, recommending these three sites would likely not be satisfactory. To cater to situations described by the above example, recent works introduced the concept of *diversity* in the spatial queries [7–9]. We note that due to the hardness of the problem, the works propose approximated solutions (with slightly different variations of the constraint). As a concrete illustration, in the context of Example 1, the user may have a preference for the answer set $T_2 = \{M_1, P_1, S_4\}$, which includes a statue (S_4), a museum (M_1) and a park (P_1)—within the desired distance bound.

What motivates this work is the observation that the existing approaches assume that the user will choose only one of the results, aiming at maximizing the diversity of the options of the user. However, no guarantee is provided that there exists a path between all the PoIs that satisfies the range constraints as path. In this example, it is clear that, while all three PoIs in T_2 are within the spatial range, visiting all three of them relying on the existing road network (cf. Figure 1) will exceed the distance limitation in terms of total travel. Although answers like T_2 may be useful to provide diverse options such as different types of restaurants, they do not properly consider the traveling cost between the PoIs. Moreover, in practice, in addition to the distance budget itself, the user may have other preferences that could constrain the answer set. For instance, a user does not mind walking 10km, however he/she is a fan of in-depth tours of popular venues. Thus, he/she may want to limit the visits to no more than 3 attractions (although there might be more PoIs in the surrounding areas) because the time plan to be spent in each of them is 1 hour. Another rationale for explicitly incorporating a limit on is that the user would like to limit the budget of expenses (e.g., for entrance fee). Hence, in our work, we use the maximum number of PoIs in the answer to reflect such preferences.

To combine all these considerations, in this paper we introduce a new query type, called the *k-Diverse Path Query* (*k*DPQ). The goal of *k*DPQ is to find a *path that maximizes diversity of PoIs along it, subject to the constraint that the length of the whole path is within user-specified limits, and the number of PoIs is k.*

We also consider settings in which the user would like to have a very specific location as the terminal one for the trip. Such settings would correspond to scenarios in which the user wants to have subsequent activities at/near a particular location, e.g., have a meeting with a collaborator in a restaurant, after the tour. In an extreme case, the scenario would correspond to the user wanting to be back to his/her hotel, which is a starting location of the trip. We denote this variant with k_TDPQ, to indicate that the terminal point of the trip is fixed.

We note that the proposed approaches can be generalized to different classes of problems in which: (a) there exists a collection of states, each specified with a set of values from different domains; (b) there exists a limited set of transitions between certain pairs of states; (c) there is a cost associated with each transition; and (d) the desired properties of the states along an "execution path" of a process can be semantically specified. One specific example comes from the domain of workflows and *Business Process Management* (BPM) domains, where attempts have been made to semantically

characterize the enactments [10, 11]. From a broader perspective, the generalized traveling salesman problem also appears in genomic research [12] when considering a graph of possible transition among states in which one would like to couple a constraint on the types of states and the reaction time, with specific initial conditions/states.

In this work, we focus on the LBS settings and towards processing the above diverse path queries, we propose two searching algorithms. While one can always construct a straightforward baseline based on Dijkstra algorithm, in this work we propose an index structure, called *Diversity Aggregated R-tree* (DAR-tree), devised to improve the efficiency of the *k*DPQ processing. Specifically, the DAR-tree enables the two algorithms that we propose to navigate the space of possible paths more efficiently, while maximizing diversity of PoIs. We also introduce adaptations of the criteria for using DAR-tree structure towards efficient processing of k_TDPQ.

Our experimental evaluation, where real-world road network and PoI data from OpenStreetMap are used to generate applicable scenarios, demonstrates that our proposed algorithms can provide highly-diverse paths, while being efficient in terms of running time. We also provide a discussion, illustrating how each of our algorithms has advantages in specific scenarios.

In summary, our main contributions are as follows:

- We identify and formalize a novel type of path planning query, *k*DPQ, enabling the users to generate a visit of a sequence of PoIs that are within certain distance bound and provide maximal diversity, as well as its variant with a fixed terminal, k_TDPQ.
- We devise novel data structure and processing algorithms to enable efficient processing of the *k*DPQ and k_TDPQ variants. The DAR-tree augments the traditional *R*-tree by embedding aggregated semantic information in its nodes.
- We conduct experimental evaluations over real-world datasets to demonstrate the benefits of the proposed methodologies and the trade-offs between two complementary solutions.

We note that an earlier version of this work was presented in [13]. The present article extends the prior publication with the k_TDPQ variant, along with the corresponding experimental results and analysis.

The remainder of this paper is organized as follows. We survey state-of-the-art methods related to diverse nearest PoI search in Sect. 2. After introducing the necessary background, our proposed novel queries—*k*DPQ and k_TDPQ—are formally defined in Sect. 3. Section 4 presents our solutions in detail, including the DAR-tree and query processing algorithm that leverage this index structure. In Sect. 5 we discuss in detail the processing of k_TDPQ variant. The experimental evaluations are presented in Sect. 6, and we conclude this work in Sect. 7.

2 Related work

Coupling motion and semantics has already been considered in the literature, bringing about the concepts of semantic and activity trajectories. Both the modelling aspects [5, 6] and the query processing aspects [4, 14] combining spatial, temporal

and descriptor contexts of the PoIs, along with transition mode (e.g., walk, drive) have been tackled. What separates the present work from the aforementioned ones is that we are focusing on constructing a path that will be limited in its length, be it travel-time or distance along a road network, and will visit a collection of PoIs with the highest diversity in terms of their semantic descriptors.

The concept of incorporating diversity into queries answers has its origins in information retrieval—specifically, in similarity search among documents. The *Maximal Marginal Relevance* (MMR) model [15] is one of the earliest proposals to consider diversity to re-rank documents in the answer set, where at each step, the element with higher marginal relevance is selected. A document has high marginal relevance if it is both relevant to the query and has minimal similarity to previously selected documents.

Several approaches have been proposed for coupling spatial and diversity contexts. Finding the *k*NNs to a given query point such that the distance between any two points is greater than a predefined minimum diversity was introduced in [16], and selecting the most diverse set within a predefined radius in Hamming space is addressed in [17]. A *k*-similar diversification set which optimizes a linear function combining the similarity (i.e., closeness) and diversity for a given trade-off between them has been studied in [18]. Monitoring the most diverse *k*-sized set over distributed sets was proposed in [19]. All these works have in common that their goal is to find a *k*-cardinality subset of size *k*, among a set of candidates PoIs, that maximizes diversity. However, these works do not consider the constrained travel along road networks, and thus, cannot return any path that allows to visit the resulting PoIs.

Other recent works that have combined the diversity and spatial contexts are presented in [7] and [9] in the context of NN queries, tackling the settings of optimizing the weighted sums of the constraints. Our previous work [8] introduced a *k-Diversified Range Query* (*k*DRQ) on road networks, which maximizes the semantic diversity of the answer set from spatial range queries on road network. While this work does consider road networks, it selects a diverse set within a network range regardless of the length of the path between the PoIs. The rationalé is to give users merely a set of diverse options, from which the user is expected to choose one, however, it is restricted within a path from a query location to a single PoI. The main difference with the present work is that *k*DP queries generate a path that connects multiple PoIs that, ensuring high diversity. More distantly related approaches to spatial diversification include angular diversity [20]—which defines the nearest Surrounder Query to find the nearest objects from a query point from different angles; and the angular similarity—which have been used for diversified *k*NN problem in [21].

Relying on the Skyline paradigm [22], finding the set of all optimal solutions for a given linear combination of two diversity notions, spatial and categorical, is presented in [7]. The categorical diversity is modeled by the difference between categories of data points—e.g., two restaurants are diverse if they are from different ethnicities. The idea of using keywords, i.e., a finer granularity in order to distinguish categories, to find diverse *k*NNs has been explored in [23]. In that work, the keywords are used for filtering data points, i.e., only points that contain *all* query keywords are considered. More recently, the problem of finding *k* shortest trajectories that contain the most relevant keywords to the query was addressed in [24],

where a hybrid index structure was proposed. Complementary to these works, we use the concept of Latent Dirichlet Allocation in order to consider a more sophisticated notion of diversity based on the set of keywords that describe each object. To speedup the processing of kDRQs, we propose an indexing structure which augments the spatial data in a node with aggregated diversity value for the sub-trees.

We note that the k_TDPQ version can be perceived as an extended variant of the *Generalized Traveling Salesman Path* (GTSP) [25]. In the original setting, the query required at least one node from each semantic category. In [26], a special case of GTSP problem was introduced—finding the optimal path with a given set of node constraints—and two heuristics were proposed. Another variant of GTSP query was recently considered in [27] which, complementary to this work, was pondering the settings with a numerical preference of each PoI category, without any pre-established bound on the starting or terminating locations. Preliminary results were presented on solving the variant over real-world dataset by adopting multiple benchmarks. In this work, the spatial constraints pertain to the starting and ending/terminating location—however, what separates it from the above works is that we are focusing on concomitantly optimizing the diversity of the PoIs.

We close this section with observations regarding two bodies of related work. Recently, *Machine Learning* (ML) based approaches have been proposed for recommending the (next) PoI to visit, incorporating features such as popularity, preferences, starting time, etc. [28–32]. Some recent works have also targeted the problem of recommending a sequence of PoIs [33]. However, we note that, at present, any kind of a ML based approach for kDPQ/k_TDPQ variants is hindered by the lack of proper training data—for which our work may generate enabling source. Complementary to this, diversification has been studied in social sciences—e.g., for variations of economic and racial groups, and mobility across spatial areas (cf. [34, 35]). However, those applications—while of societal importance (and subject of future work), are outside the scope of the current problem domain.

3 Background and problem definition

In this section, we introduce the basic terminologies and the settings, after which we proceed with the formal definition of the kDPQ and k_TDPQ problems. We first define the problem of finding the k most-diverse path for an abstract diversity metric, and then introduce the topic-based diversity employed in this article.

3.1 Preliminaries

Definition 1 (Road Network) A *Road Network* $G = (V, E, W)$ is a weighted directed graph, where V is a set of vertices and each vertex $v \in V$ is associated with location-attribute $v.L$; $E \subseteq V \times V$ represents the set of edges between pairs of vertices (v_i, v_j) $(v_i, v_j \in V)$; $W : E \mapsto \mathbb{R}^+$ is a function which maps each edge $e \in E$ to a positive real value representing the cost of traversing e.

Vertices on a road network may contain *Points of Interest* (PoIs). Each PoI is associated with two attributes: *location* (such as latitude and longitude) and *descriptors* (such as keywords, categories, and etc.), formally defined as follows:

Definition 2 (PoI Network) Let $G = (V, E, W)$ be a road network. A *PoI* p is represented as a pair $p = (L, I)$, where $p.L \in \{v.L \mid v \in V\}$ is the spatial location of p on the road network, and $p.I$ is the semantic information of p. A *PoI Database* $\mathcal{P} = \{p_1, ..., p_{|\mathcal{P}|}\}$ is a collection of PoIs and for any vertex $v \in V$, we let $v.P$ denote the (possibly empty) set of PoIs located at vertex v. We denote the road network enriched with the PoI information as $\mathcal{G} = (V, E, W, \mathcal{P})$, and call it a PoI network.

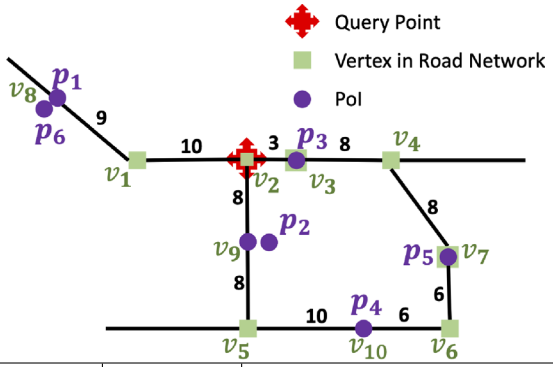
We note that in practice, a particular PoI p may not be directly located at a vertex of the road network. In such a case, we apply map-matching to project the PoI to the nearest point on an edge of the road network [36]. The projected point becomes a new (virtual) vertex of the network that corresponds to the $p.L$.

The process of constructing a PoI network from a given road network graph G and a set of PoIs $\mathcal{P}$ is formalized in Algorithm 1. Note that we leverage an R-tree [1] to store the road network (Lines 3 ∼ 6) to efficiently retrieve the nearest neighbor edge to a PoI (Line 11). The update in Line 13 adds a new vertex to the network, and replaces the corresponding edge (i.e., *nearest_edge*) with two new edges connecting the new (virtual location) vertex to the vertices of *nearest_edge*, and replicating the original weight of the *nearest_edge* to both new edges.

Algorithm 1 PoI Network Construction

Input: Road Network $G = (V, E, W)$, PoI Database $\mathcal{P}$
Output: PoI Network $\mathcal{G}$

1: Copy G as initial $\mathcal{G}$ with $v.S = \emptyset$ **for each** v in V
2: $tree \leftarrow$ R-tree()
3: **for each** e in E **do**
4: $rect \leftarrow$ rectangle whose diagonal is e
5: $tree$.insert($rect$)
6: **end for**
7: **for each** p in $\mathcal{P}$ **do**
8: **if** $p.L = v.L$ where $v \in \mathcal{G}$ **then**
9: $v.P$.add(p)
10: **else**
11: $nearest\_edge \leftarrow tree$.nearest_neighbor($p.L$)
12: $v.L \leftarrow$ Project p onto $nearest\_edge$ which minimizes distance to $p.L$
13: Update $\mathcal{G}$ with new vertex $(v.L, \{p\})$
14: **end if**
15: **end for**
16: **return** $\mathcal{G}$



Attractions	Types	Descriptors
p_1	Museum	{Historical, Cultural, Art}
p_2	Park	{Fountain, Forest, Playground}
p_3	Aquarium	{Seal, Fish, Sea}
p_4	Park	{Green, Fish, Monument}
p_5	Zoo	{Turtle, Tiger, Safari}
p_6	Museum	{Literature, Art, Painting}

Fig. 2 Example of poI network

To illustrate the concepts, we provide another small-scale example in Fig. 2—slightly more focused on the terms and their relationship than the intuitive motivation in Fig. 1. Specifically, we present a PoI network, having six PoIs $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ (shown as purple circles) and a road network having $|V| = 7$ vertices (shown as green rectangles), several bidirectional edges E connecting vertices (shown as solid black lines) and a weight function W mapping edges to annotated weights. PoIs p_3 and p_5 are trivially mapped to vertices at the same location. Using Algorithm 1, three new vertices—v_8, v_9, v_{10}—are added into the PoI network, as well as the related edges and the updated corresponding weights. Note that Algorithm 1 will also map p_6 to v_8, thus yielding $v_8.P = \{p_1, p_6\}$.

3.2 The k-most diverse path query

Definition 3 (Semantic Path) Let $\mathcal{G} = (V, E, W, \mathcal{P})$ be a PoI network. A semantic path $sp = (sp_1, ..., sp_{|sp|})$ is a sequence of adjacent vertices in $\mathcal{G}$, i.e., $\forall i\ (1 \leq i \leq |sp|) : sp_i \in V$ and $\forall i\ (1 \leq i < |sp|) : (sp_i, sp_{i+1}) \in E$. The *cost* of a given path sp is defined as sum of edges weight $sp.cost := \sum_{i=1}^{|sp|-1} W(sp_i, sp_{i+1})$. The attribute *collection* of a given path sp is defined as $sp.collection = \bigcup_{i=1}^{|sp|} sp_i.P$—i.e., the union of all the PoIs contained in the vertices along sp (ones for which $sp_i.P \neq \emptyset$).

In Fig. 2, $sp = (v_8, v_1, v_2, v_9, v_5, v_{10})$ is a semantic path having cost $sp.cost = 9 + 10 + 8 + 8 + 10 = 45$ that includes the set of PoIs $sp.collection = \{p_1, p_6, p_2, p_4\}$.

Definition 4 (Range Path Search Query) Let $\mathcal{G} = (V, E, W, \mathcal{P})$ be a PoI network and $Q \in V$ be a query location. Given a positive value $\varepsilon \in \mathbb{R}^+$, a network range path search query $RPS(\mathcal{G}, Q, \varepsilon)$ returns all semantic paths starting at Q having a cost no greater than ε Formally:

$$RPS(\mathcal{G}, Q, \varepsilon) = \{sp \mid sp_1 = Q \ \wedge \ sp.cost \leq \varepsilon\}$$

We note that the assumption that $Q \in V$ comes without loss of generality, as we can project any query location to a (potentially new) network vertex using Algorithm 1.

The concepts introduced so far are illustrated in Fig. 2, showing the query point Q (red cross) located at v_2. Given a distance range $\varepsilon = 30$, answers to *RPS* include (Q, v_1, v_8, v_1), (Q, v_9, v_5, v_{10}), (Q, v_3, v_4, v_7, v_6). We note that Definition 3 does not require a path to be *simple*, i.e., it allows a path to have cycles and visit the same vertex more than once. This is necessary in order to enable a path to collect PoIs located in dead ends—which is, nodes of degree 1—and still continue collecting additional PoIs.

In addition to limiting the distance for a user to travel on a path, we further assume that a user may have other kinds of constraints (e.g., a limited spending budget, or limited stay-time) which, in turn, may impose a limit on the maximum number of PoIs along a semantic path and we denoted it by k in this work. For a set of PoIs collected by a path, the following definition finds the most diverse subset of PoIs of cardinality k:

Definition 5 (k-Diverse Subset of Semantic Path) Let sp be a semantic path, and $div : P \mapsto \mathbb{R}_0^+$ be a function that maps a set of PoIs to a non-negative diversity score. The k-diverse subset of sp, $kDS_{div}(sp, k)$, is defined as the subset of $sp.collection$ with cardinality at most k, maximizing the diversity score, i.e.,

$$kDS_{div}(sp, k) = \underset{P \subseteq sp.collection, |P| \leq k}{\arg\max} \ div(P)$$

The specification of a diversity function $div(P)$ that maps a set of PoIs P to a diversity score is left abstract in Definition 5, and multiple definitions of diversity have been used in the literature [7, 8]. In this work, we employ the topic-based probabilistic diversity proposed in [8], which is reviewed in detail in Sect. 3.3.

Example 2 Returning to the scenario in Fig. 2, consider the semantic path $(v_2, v_3, v_2, v_9, v_5, v_{10})$, which collects the set of three PoIs $\{p_2, p_3, p_4\}$. Assume that a user only has time/budget to visit two PoIs, thus setting $k = 2$. In this case, we see that both PoIs p_2 and p_4 are a park, having similar textual descriptors. Intuitively, to

maximize diversity, p_3 should be chosen as the only non-park PoI, and it should be chosen together with p_2, as p_4 shares keyword similarity (i.e., *Fish*) with p_3.

Given a measure of diversity of a semantic path in Definition 5, we can now proceed to define our proposed k diverse path query as finding the semantic path that starts at a specified query location and maximizes the diversity of collected paths subject to a maximum length of the path and a maximum number of PoIs to be collected. This query is formally defined as follows.

Definition 6 (k-Diverse Path Query) Let $\mathcal{G} = (V, E, W, \mathcal{P})$ be a PoI network and $Q \in V$ be a query location. Furthermore, let $div : P \mapsto \mathbb{R}_0^+$ be a function that maps a set of PoIs to a non-negative diversity score, let k be a positive integer, and let $\varepsilon \in \mathbb{R}^+$ be a cost constraint. Then, a $k - diversepathquery$ ($kDPQ$) is defined as

$$kDPQ(\mathcal{G}, Q, div, \varepsilon, k) = \underset{sp \in RPS(\mathcal{G},Q,\varepsilon)}{\arg\max} \; div(kDS_{div}(sp, k))$$

where $RPS(\mathcal{G}, Q, \varepsilon)$ is the set of all semantic paths starting at Q having a cost no greater than ε as defined in Definition 4, and $kDS_{div}(sp, k)$ returns the k-subset of PoIs among all PoIs collected by path sp that maximizes the diversity function div as defined in Definition 5.

Example 3 Given the PoI network $\mathcal{G}$ in Fig. 2, let $\varepsilon = 35$ and $k = 2$, two possible paths could be $sp_1 = (Q, v_2, v_9, v_5, v_{10})$ with $sp_1.collection = \{p_2, p_4\}$ and $sp_2 = (Q, v_2, v_3, v_4, v_7, v_6, v_{10})$ with $sp_2.collection = \{p_3, p_5, p_4\}$. Since both p_2 and p_4 are parks and most textual descriptors are semantically similar, a k-diverse path query returns path $kDPQ(\mathcal{G}, Q, div, 35, 2) = sp_2$ and recommends to visit PoIs p_3 and p_4 on this path.

As mentioned, in certain scenarios users may want to impose an additional constraint—finishing (i.e., terminating) their trips at a specific terminal, denoted T. To cater to such settings, we have the following:

Definition 7 (k-Diverse Path Query With Fixed Terminal) Let $\mathcal{G} = (V, E, W, \mathcal{P})$ be a PoI network, $Q \in V$ be a query location and $T \in V$ be a vertex corresponding to the desired end of trip location. As before, let $div : P \mapsto \mathbb{R}_0^+$ be a function that maps a set of PoIs to a non-negative diversity score, let k be a positive integer, and let $\varepsilon \in \mathbb{R}^+$ be a cost constraint. Then, a *k-diverse path query with fixed terminal* ($k_T DPQ$) is defined as

$$k_T DPQ(\mathcal{G}, Q, T, div, \varepsilon, k) = \underset{sp \in RPS(\mathcal{G},Q,T,\varepsilon)}{\arg\max} \; div(kDS_{div}(sp, k))$$

where $RPS(\mathcal{G}, Q, T, \varepsilon)$ is the subset of $RPS(\mathcal{G}, Q, \varepsilon)$, corresponding to all semantic paths starting at Q, terminating at T, and having a cost no greater than ε (cf. as Definition 4). Similarly to Definition 6, $kDS_{div}(sp, k)$ returns the k-subset of PoIs among all PoIs collected by path sp that maximizes the diversity function div (cf. Definition 5).

We observe that for any strictly monotonic diversity function *div* (cf. Sect. 3.3), the following hardness result holds:

Lemma 1 *The problem of finding the most diverse path* $kDPQ(\mathcal{G}, Q, div, \varepsilon, k)$ *is NP-hard.*

Proof Let *tsp* be a solution to the traveling salesman problem (TSP) on an arbitrary graph $\mathcal{G}$ starting at an arbitrary vertex Q, that is, the shortest path that collects all PoIs. Let *tsp.cost* denote the cost of this path. Let *div* be any strictly monotonic diversity function, that is, adding additional PoIs to a set will increase the diversity of the set. Since *div* is strictly monotonic, the set P, which contains all PoIs, maximizes *div*. Then, by Definition 6, $kDPQ(\mathcal{G}, Q, div, tsp.cost, \infty) = tsp$. This is evident, as a kDPQ query starting at Q, having a range of $\varepsilon = tsp.cost$, will return the most diverse path (collecting all PoIs due to a strictly monotonic diversity function) having a length of at most *tsp.cost*. By definition, this path exists and is the solution to the TSP on $\mathcal{G}$ starting at Q. Thus, any instance of TSP can be written as an instance of kDPQ, implying that answering kDPQ queries is at least as hard as TSP, which is known to be NP-hard [37]. □

Lemma 2 *The problem of finding the most diverse path* $k_TDPQ(\mathcal{G}, Q, T, div, \varepsilon, k)$ *is NP-hard.*

Proof Analogously to the proof of Lemma 1 we let *tsp* be the solution to the traveling salesman problem (TSP) on an arbitrary graph $\mathcal{G}$ starting at an arbitrary vertex Q and we additionally let T' be the last vertex of *tsp*. Then we have $k_TDPQ(\mathcal{G}, Q, T', div, \varepsilon, k) = tsp$. Again, this shows that any instance of TSP can be reduced to an instance of k_TDPQ. □

Due to the complexity of $kDPQ$ and k_TDPQ, we resort to heuristics to find (approximate) solutions that return high, but not necessarily optimal, diversity. Next, we briefly explain the diversity function *div* that we employ.

3.3 Topic-based diversity

In this work, we leverage the topic-based diversity proposed in [8] which extracts K latent topics from textual context of each PoI, where K is a user-specified parameter. Based on textual descriptor $p_i.I$ of a PoI $p_i \in P$, p_i is mapped to a topic distribution θ_i that maps each topic to the probability $\theta_{i,j}$ that p_i covers the topic $1 \leq j \leq K$. Then, the diversity of a set P of PoIs is defined as the expected number of topics that is covered by any PoI in P.

Based on the attached descriptive items, the semantic description of each PoI p_i is illustrated by a vector of probability (topic) distribution θ_i whose length is the number of latent topics K. $\theta_{i,j}$ $(1 \leq j \leq n)$ represents the probability of p_i belonging to

topic j. For a set of PoIs $P = \{p_1, ..., p_{|P|}\}$, we define a vector $ProbDiv(P)$ that stores, for each topic j, the probability that it is covered by P as

$$ProbDiv(P)_j := 1 - \prod_{p_i \in P} (1 - \theta_{i,j})$$

which is then aggregated into a diversity score via expected number of topics covered:

$$div(P) := \sum_{j=1}^{K} ProbDiv(P)_j$$

Intuitively, the probability $1 - \theta_{i,j}$ is the probability that PoI p_i does not cover topic j. Exploiting that PoIs are stochastically independent, $\prod_{p_i \in P}(1 - \theta_{i,j})$ is the probability that none of the PoIs in P covers topic j. We define $ProbDiv(P)_j$ as the counter-probability, i.e., the probability of the complementary event that at least one PoI in P covers topic j. Finally, these probabilities are aggregated into the expected number of topics covered by P via $div(P)$.

Example 4 Let $P = \{p_1, p_2, p_3\}$, and each p_i allocated a topic distribution having $K = 3$ topics, e.g, $\theta_1 = (0.1, 0.0, 0.9), \theta_2 = (0.4, 0.3, 0.3), \theta_3 = (1.0, 0.0, 0.0)$, respectively. One can observe that p_1 is very likely to cover the third category and p_3 is guaranteed to belong to the first category, while p_2 obtains a high uncertainty since its distribution among different categories is close to uniform. To compute $ProbDiv(P)$ using above equations, we get $ProbDiv(P)_1 = 1 - (1 - 0.1) \times (1 - 0.4) \times (1 - 1.0) = 1.0$ since p_3 is certain to cover the first category; $ProbDiv(P)_2 = 1 - (1 - 0.0) \times (1 - 0.3) \times (1 - 0.0) = 0.3$ indicating a 30% likelihood that P can cover the second category; and $ProbDiv(P)_3 = 1 - (1 - 0.9) \times (1 - 0.3) \times (1 - 0.0) = 0.93$ showing a high probability that the third category is covered due to the probability distribution of p_1. In sum, $div(P) = 1.0 + 0.3 + 0.93 = 2.23$ implies that an expected 2.23 topics are covered by P.

4 Efficient processing of *k*DPQ

To efficiently answer kDPQ we adopt the informed search [38] approach which, in general, can be considered as a greedy algorithm, whereby a vertex is selected for exploration based on the priority from an evaluation function. The evaluation function for solving kDPQ is constructed as the estimated gain of probabilistic diversity, thus the vertex with the greatest evaluation would be explored first. The quality of the evaluation function is critical for the searching procedure.

We firstly introduce the heuristic function, which is an important component of the evaluation along with the proposed supporting index structure: *Diversity Aggregated R-Tree*, to efficiently compute the result from a heuristic function. Subsequently, the informed search algorithm is presented.

4.1 Heuristic and evaluation function

For a PoI network $\mathcal{G} = (V, E, W, \mathcal{P}\Rightarrow$, vertex $v \in V$, and value $\varepsilon \geq 0$, let $P[v, \varepsilon]$ denote the set of all PoIs in P having a network distance from v of at most ε. Furthermore, let $P_E[v, \varepsilon]$ denote the set of all PoIs in P having Euclidean distance from v of at most ε. To conservatively bound the category-wise diversity vector $ProbDiv(P[v, \varepsilon])$, we propose the following heuristic function:

$$\begin{aligned} h(v, \varepsilon) &:= ProbDiv(P_E[v, \varepsilon]), \\ \text{where } P_E[v, \varepsilon] &= \{p \mid \|v.L, p.L\|_2 \leq \varepsilon\} \end{aligned} \tag{1}$$

Note that the Euclidean distance, which is used in the heuristic function, is always less than or equal to the road-network distance—thus $P[v, \varepsilon] \subseteq P_E[v, \varepsilon]$ holds. We can leverage this relation to obtain an upper bound of the diversity $div(P[v, \varepsilon])$ using the following lemma:

Lemma 3 *For any topic* $1 \leq j \leq K$ *it holds that*:

$$div(P_E[v, \varepsilon]) \geq div(P[v, \varepsilon])$$

Proof Because of $0 \leq \theta_{i,j} \leq 1$ for any PoI p_i and category j, we also have $0 \leq 1 - \theta_{i,j} \leq 1$. Due to $P[v, \varepsilon] \subseteq P_E[v, \varepsilon]$, it holds that $\prod_{p_i \in P_E[v,\varepsilon]}(1 - \theta_{i,j}) \leq \prod_{p_i \in P[v,\varepsilon]}(1 - \theta_{i,j})$ and thus $1 - \prod_{p_i \in P_E[v,\varepsilon]}(1 - \theta_{i,j}) \geq 1 - \prod_{p_i \in P[v,\varepsilon]}(1 - \theta_{i,j})$. Summarizing over all categories j, this implies that $\sum_{i=1}^{K} 1 - \prod_{p_i \in P_E[v,\varepsilon]}(1 - \theta_{i,j}) \geq \sum_{i=1}^{K} 1 - \prod_{p_i \in P[v,\varepsilon]}(1 - \theta_{i,j})$ i.e., $div(P_E[v, \varepsilon]) \geq div(P[v, \varepsilon])$. □

According to Lemma 3, the Euclidean forward estimation using all PoIs $P_E[v, \varepsilon]$ in the Euclidean range allows to derive an upper bound of $ProbDiv(P[v, \varepsilon])$ without having to consider the network topology of the entire graph $\mathcal{G}$.

Algorithm 2 Swap Algorithm

Input: Set of PoIs P, Integer k
Output: $k-$Diverse Subset P^*

```
1:  P* ← ∅
2:  for each p ∈ P do
3:      if |P*| < k then
4:          P* ← P* ∪ p
5:      else
6:          C ← P* ∪ p
7:          p⁻ ← arg max_{p'∈C} div(C \ p')
8:          P* ← C \ p⁻
9:      end if
10: end for
11: return P*
```

4.2 Informed search

Starting at Q, the idea of our proposed algorithm is to iteratively expand paths that yield the highest potential diversity using the heuristic of Equation 1. In a nutshell, if we reach a vertex v on a path of cost δ, then we have at most a distance of $\varepsilon - \delta$ left to explore from v. If the path leading to v has already collected the set of PoIs res, then the maximum diversity of exploring v can be upper-bounded by computing the maximum k-diversity of any k-subset of the set $res \bigcup P_E(v, \varepsilon - \delta)$—that is, via extending res by all PoIs still reachable from v using Euclidean distance. Our algorithm greedily processes vertices using a priority queue sorted by this upper bound. Once the currently most diverse result exceeds the diversity of the largest unexplored upper bound, we can terminate computation.

Formally, let $res \subseteq V$ be the set of vertices explored by a path and let δ be the cost of this path. For any adjacent vertex to extend the path, we evaluate the following function:

$$\begin{aligned} &f(res, v, \varepsilon - \delta, k) \\ &= div(Swap(res \bigcup ProbDiv(P_E(v, \varepsilon - \delta) \setminus res), k)) \end{aligned} \tag{2}$$

The rationale of $f(res, v, \varepsilon - \delta, k)$ is to consider the set of all PoIs $P_E(v, \varepsilon - \delta) \setminus res$ reachable from v at a Euclidean distance of $\varepsilon - \delta$, except the PoIs in res which are already collected. Then, the result of the heuristic function $ProbDiv(P_E(v, \varepsilon - \delta) \setminus res)$ is treated as the topic distribution of a single PoI. Because of the limit on cardinality, to estimate the potential gain of following a specific direction to extend a semantic path, we employ the *Swap Algorithm*, (cf. Algorithm 2, proposed in [18]) to heuristically find k *subset* obtaining greatest diversity among its k-diverse subset res and $P_E(v, \varepsilon - \delta) \setminus res$. We note that Lemma 3 ensures that the diversity of PoIs inside the Euclidean range is no less than the diversity of the PoIs in the network range, thus that $f(res, v, \varepsilon - \delta, k)$ provides an upper bound of the diversity obtainable by extending an existing path by node v. Our algorithm will exploit the evaluation function $f(res, v, \varepsilon - \delta, k)$ to direct the searching process to the node having the highest upper bound diversity.

4.3 Diversity aggregated R-tree

The main point of utilizing the Euclidean distance as a heuristic function is to be able to leverage an R-Tree [39] to efficiently obtain the set $P_E(v, \varepsilon)$ of PoIs within a Euclidean range around node v. Since the Euclidean distance is a lower bound of the network distance, it allows us to prune any PoI (or R-Tree node) having a Euclidean distance already greater than ε while avoiding expensive network exploration to obtain the set $P(v, \varepsilon)$

using, for example, Dijkstra's algorithm. Although shortest path algorithms can retrieve all PoIs whose network distance is within a certain budget, the execution time is prohibitive (e.g., the complexity for Dijkstra algorithm is $O(|E| + |V| \log |V|)$) and we have to run it once whenever we want to explore an vertex in graph. On the opposite, we leverage R-Tree and retrieve P_E as a candidate set in Euclidean space. The worst-case complexity of searching in R-Tree is linear to the number of nodes, which in our case is the number of PoIs. In addition, the number of PoIs is always (much) smaller than the number of vertices in the road network. This, along with the linear complexity, enables speeding up the query processing, especially in comparison with Dijkstra-based exploration.

To help our search for diverse paths, we introduce a *Diversity Aggregated R-Tree* (DAR-Tree) to accelerate the computation of heuristic function $div(P_E[v, \varepsilon])$. DAR-Tree is a variant of *aggregated R-Tree* (aR-Tree) [40], storing the information related to probabilistic diversity of each *Minimum Bounding Rectangle* (MBR) in both leaf and inner nodes.

Figure 3 presents a example of DAR-Tree with 12 PoIs. Each leaf node stores a PoI and its corresponding topic distribution. Every non-leaf node contains:

1. The pointer(s) to the child node(s) and the coordinates of *MBR*;
2. The diversity-related information, which is a vector representing the probability of each category not being covered of all its children, i.e., $\zeta_{m,j} = \prod_{p_i \in m}(1 - \theta_{i,j})$, where m is an MBR and $1 \leq j \leq K$.

Furthermore, each MBR m memorizes the set P of PoIs inside m, e.g., $P_1 = \{p_1, p_2, p_3\}$ and $P_3 = \{p_7, p_8, p_9\}$.

Since the DAR-Tree inherits the structure of an *aR*-Tree [40], we omit details on construction and maintenance of DAR-Tree—however, for reproducibility, we do provide the source code (cf. Sect. 6). The benefits of the DAR-Tree in terms of speeding up the computation of the heuristic function, are illustrated by Algorithm 3. Broadly speaking, instead of always recursively iterating all the way down to the leaf node, we can terminate the search if an MBR of some non-leaf node has already been fully contained by the searching region. For a set of approximated PoIs, Line 4 and Line 6 calculate the probability of each category being uncovered, thus Line 20 returns the complementary probability.

Algorithm 3 DAR-Tree Range Query

Input: PoI network $\mathcal{G} = (V, E, W, \mathcal{P})$, DAR-Tree $\mathcal{R}$, Vertex $v \in V$, Range ε, k-diverse subset res
Output: Heuristic score h_val

$h\_val \leftarrow (1, ..., 1)$ ▷ $|h\_val|$ = the number of categories
function RANGE-SEARCH($region$, $child$, res)
  if $child$ is a leaf and $P_{child} \notin res$ **then**
    $h\_val \leftarrow h\_val \odot (1 - \theta_{child})$ ▷ $\odot$ is entrywise product
  else if $region$.contains(MBR_{child}) **then**
    $h\_val \leftarrow h\_val \odot \zeta_{child}$
    $dup \leftarrow res \bigcap P_{child}$
    if $dup \neq \emptyset$ **then**
      $h\_val \leftarrow h\_val \oslash \prod_{p_i \in dup}(1 - \theta_i)$ ▷ $\oslash$ is entrywise division
    end if
  else
    for each $gchild$ of $child$ **do**
      if $region$.intersects(MBR_{gchild}) **then**
        RANGE-SEARCH($region$, $gchild$, res)
      end if
    end for
  end if
end function
RANGE-SEARCH($circle(v, \varepsilon)$, $\mathcal{R}.root$, res)
return $(1, ..., 1) - h\_val$ ▷ Entrywise subtraction

4.4 Efficient kDPQ processing—greedy best-first search

Algorithms 4 and 5 are two searching strategies that we propose, utilizing the heuristic function (Equation 1) and DAR-Tree. The main idea is to greedily explore the network, while two different variations are introduced to balance the efficiency and diversity.

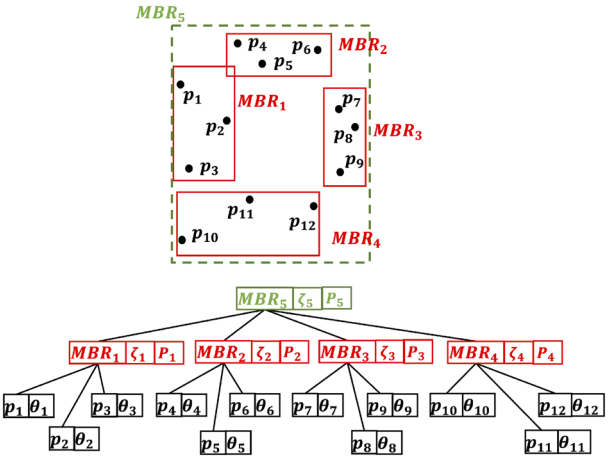


Fig. 3 Example of diversity aggregated R-Tree

Specifically, Algorithm 4 prunes the search space by only considering simple paths, i.e., paths that do not visit the same node twice. While this constraint yields gains in efficiency, it must be noted that the most diverse path may very well be non-simple, for example if the path visits a PoI located in a dead end (a vertex of degree one). Such cases may occur in practice—for example the Field Museum in Chicago is located on lake shore. Thus, when continuing the search for other PoIs, the intersection immediately leading to the Field Museum will inevitably be visited again (as a vertex of a road network). If simple paths are imposed, this search algorithm can only visit such a PoI if the path ends there. Algorithm 5, in turn, allows to re-visit the vertices but does not allow visiting directed edges more than once.

4.4.1 Vertex-constrained searching strategy (VSS-*k*DPQ)

Algorithm 4 remembers all the explored vertices to avoid exploring the network redundantly. That is to say, newly generated vertices that match previously explored ones would be discarded so that each vertex can be visited at most once.

Algorithm 4 Vertex Variant of kDPQ Path Searching Strategy

Input: PoI network $\mathcal{G} = (V, E, W, \mathcal{P})$, Query point $Q \in \mathcal{G}$, Integer k, Range ε
Output: Semantic path sp

$sp, max\_div \leftarrow$ None, -1
$pq \leftarrow$ Max-Heap() ▷ Each element e in pq has 5 components – $e.id$, $e.priority$, $e.dist$, $e.res$, $e.path$
pq.Insert$(Q, 0, 0, Q.P, [Q])$
$explored \leftarrow \emptyset$
while $pq \neq \emptyset$ **do**
    $v, prior, d, res, path \leftarrow pq$.Pop()
    $div\_score \leftarrow div(res)$
    if $prior \leq max\_div$ **then**
        return sp
    else if $div\_score > max\_div$ **then**
        $sp, max\_div \leftarrow path, div\_score$
    end if
    $explored$.Add(v)
    for each adj_v adjacent to v **do**
        $adj\_d \leftarrow d + W(v, adj\_v)$
        if $adj\_d \leq \varepsilon$ **then**
            $adj\_res \leftarrow$ Swap$(res \bigcup adj_v.P, k)$
            $adj\_prior \leftarrow f(res, adj\_v, \varepsilon - adj\_d, k)$
            $adj\_path \leftarrow path$.Append(adj_v)
            if adj_v is not in $explored$ or pq **then**
                pq.Insert$(adj_v, adj_prior, adj_d, adj_res, adj_path)$
            else if adj_v is in pq with lower Priority **then**
                replace that pq element with updated adj_v
            end if
        end if
    end for
end while
return sp

At the beginning, the priority queue that contains all vertices available for exploration (Line 2) is initialized, and a set for remembering every expanded vertex (Line 4). For each vertex in the priority queue, we use a data structure composed of five components: *id*—the unique identification of the vertex; *priority*—the potential/approximate diversity measured by evaluation function (Equation 1) if choosing this vertex to explore; *dist*—the road-network distance from query point Q to this vertex, *res*—the k-diversified results among the path so far, *path*—the path from query point Q to this vertex.

Our goal is to find the path with greatest diversity, thus an intuitive way to expand first is the vertex with the highest value from evaluation function $h(v, \varepsilon)$ (Equation 1). The priority queue is sorted by the priority of each vertices in descending order. When a vertex is popped out for expansion, stop computation if the highest diversity result found so far exceeds the upper bound diversity in the priority queue (Line 8). If a better solution might still exist, the searching procedure will continue and add the adjacent vertices of the expanded one into priority queue. As mentioned, the explored vertices (recorded in explored set) are not inserted into priority queue again, to avoid duplication. Line 22 is executed when better path is discovered to a vertex currently in the **queue** *pq*.

However, while the searching procedure by Algorithm 4 is efficient, as each vertex must be visited at most once—the diversity of the result may be low, especially in sparse network where an optimal path may need to backtrack to previously visited vertices.

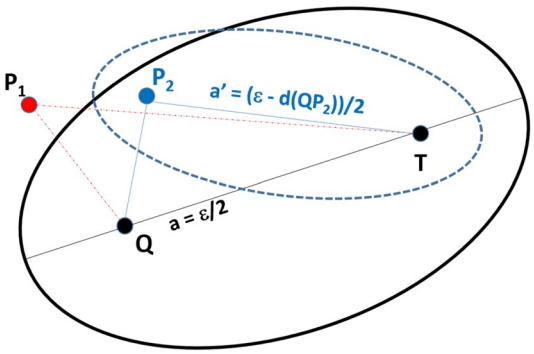


Fig. 4 Distance pruning with ellipse

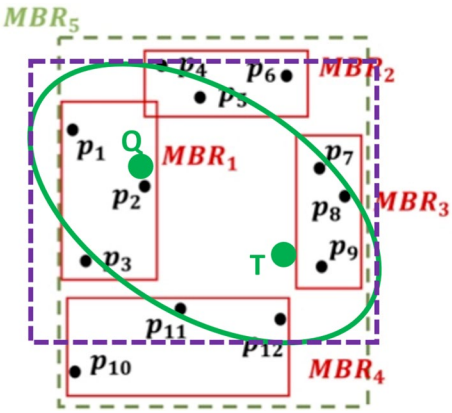


Fig. 5 Distance pruning with ellipse

Table 1 Top-10 most probably keywords for 6 latent topics (from TripAdvisor, with Natural Language Tookit)

Topic	Top-10 most probably Keywords (Probabilities in %)
1	'memorial'(1.8), 'see'(1.7), 'visit'(1.1), 'statue'(1.1), 'walk'(0.9), 'monument'(0.8), 'Lincoln'(0.8), 'take'(0.8), 'war'(0.7), 'great'(0.7)
2	'park'(2.2), 'beach'(1.9), 'walk'(1.7), 'place'(1.6), 'great'(1.5), 'nice'(1.4), 'view'(1.2), 'beautiful'(1.2), 'area'(1.0), 'go'(1.0)
3	'great'(1.5), 'show'(1.4), 'see'(1.3), 'go'(1.2), 'good'(1.2), 'get'(1.1), 'seat'(1.0), 'theater'(1.0), 'time'(0.7), 'would'(0.7)
4	'shop'(2.3), 'place'(2.0), 'restaurant'(1.9), 'great'(1.7), 'food'(1.5), 'area'(1.4), 'good'(1.3), 'nice'(1.1), 'lot'(1.1), 'go'(1.0)
5	'dog'(5.7), 'beer'(4.1), 'great'(2.0), 'good'(1.6), 'brewery'(1.5), 'place'(1.4),'friendly'(1.0), 'taste'(1.0), 'food'(0.9), 'fun'(0.9)
6	'museum'(1.7), 'tour'(1.5), 'visit'(1.4), 'see'(1.2), 'house'(1.1), 'art'(0.9), 'history'(0.9), 'beautiful'(0.9), 'interest'(0.9), 'building'(0.8)

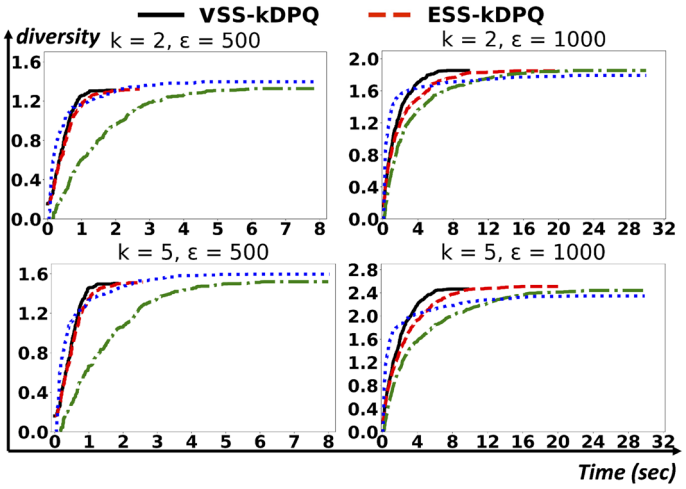


Fig. 6 Experimental result of short distance range

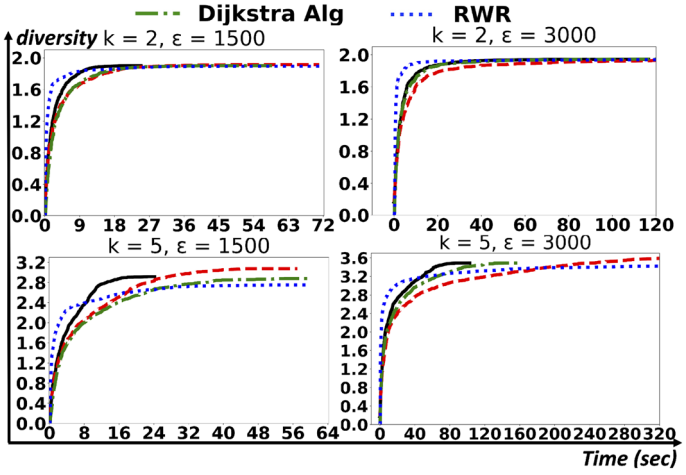


Fig. 7 Experimental result of large distance range

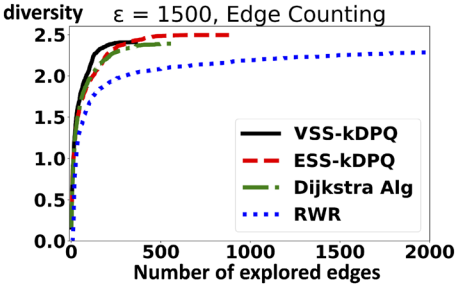
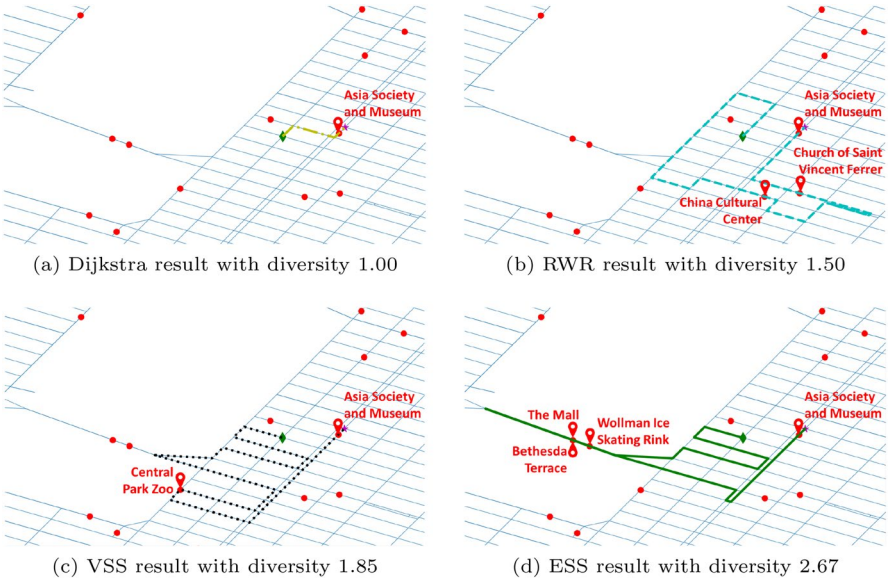


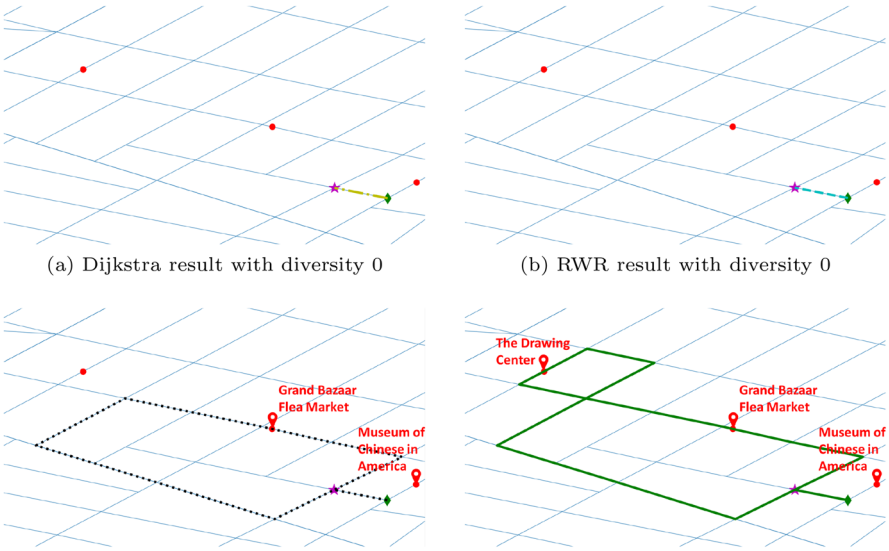
Fig. 8 Comparison result regarding of the number of explored edges



(a) Dijkstra result with diversity 1.00 (b) RWR result with diversity 1.50

(c) VSS result with diversity 1.85 (d) ESS result with diversity 2.67

Fig. 9 First path example retrieved from baselines and our proposed algorithms



(a) Dijkstra result with diversity 0 (b) RWR result with diversity 0

(c) VSS result with diversity 1.00 (d) ESS result with diversity 1.80

Fig. 10 Second path example retrieved from baselines and our proposed algorithms

Table 2 Number of unsuccessful k_TDPQ queries (out of 300 queries)

	Vertex variant	Edge variant	RWR	Dijsktra
k = 2, d = 500	181	185	219	175
k = 2, d = 1000	83	88	154	64
k = 2, d = 1500	41	48	107	21
k = 2, d = 3000	8	14	35	0
k = 2, d = 5000	11	7	9	0
k = 4, d = 500	181	185	221	175
k = 4, d = 1000	83	88	152	64
k = 4, d = 1500	41	48	107	21
k = 4, d = 3000	8	14	33	0
k = 4, d = 5000	11	9	9	0
k = 6, d = 500	181	185	224	175
k = 6, d = 1000	83	88	153	64
k = 6, d = 1500	41	48	111	21
k = 6, d = 3000	9	14	37	0
k = 6, d = 5000	11	8	9	0
k = 8, d = 500	181	185	226	175
k = 8, d = 1000	83	88	154	64
k = 8, d = 1500	41	48	97	21
k = 8, d = 3000	9	14	35	0
k = 8, d = 5000	11	8	9	0

Algorithm 5 Edge variant of kDPQ Path Searching Strategy

Input: PoI network $\mathcal{G} = (V, E, W, \mathcal{P})$, Query point $Q \in \mathcal{G}$, Integer k, Range ε
Output: Semantic path sp

1: $sp, max\_div \leftarrow$ None, -1
2: $pq \leftarrow$ Max-Heap() ▷ Each element e in pq has 6 components – $e.id$, $e.priority$, $e.dist$, $e.res$, $e.path$, $e.explored$
3: pq.Insert$(Q, 0, 0, Q.P, [Q], \emptyset)$
4: **while** $pq \neq \emptyset$ **do**
5: $v, prior, d, res, path, ex \leftarrow pq$.Pop()
6: $div\_score \leftarrow div(res)$
7: **if** $prior \leq max\_div$ **then**
8: **return** sp
9: **else if** $div\_score > max\_div$ **then**
10: $sp, max\_div \leftarrow path, div\_score$
11: **end if**
12: **for each** adj_v adjacent to v **do**
13: **if** $(v, adj\_v) \in ex$ **then** Skip **end if**
14: $adj\_d \leftarrow d + W(v, adj\_v)$
15: **if** $adj\_d \leq \varepsilon$ **then**
16: $adj\_res \leftarrow$ Swap$(res \bigcup adj_v.P, k)$
17: $adj\_prior \leftarrow f(res, adj\_v, \varepsilon - adj\_d, k)$
18: $adj\_path \leftarrow path$.Append(adj_v)
19: $adj\_ex \leftarrow ex$.Add(v, adj_v)
20: **if** adj_v is not in pq **then**
21: pq.Insert$(adj_v, adj_prior, adj_d, adj_res, adj_path, adj_ex)$
22: **else if** adj_v is in pq with lower Priority **then**
23: replace that pq element with updated adj_v
24: **end if**
25: **end if**
26: **end for**
27: **end while**
28: **return** sp

4.4.2 Edge-constrained searching strategy (ESS-*k*DPQ)

Algorithm 5 is proposed to prioritize diversity rather than efficiency. To achieve that, instead of recording the expanded vertex globally, we remember the explored directed edges for each path individually. To enforce that capability, a new component—**explored** set—is added for each vertex in priority queue, and a test (Line 13) takes place to avoid visiting an directed edge twice. We note that the

assumption of visiting each directed edge at most once does not exclude the optimal solution from the search space, as the cost-minimizing path between a set of PoIs is a Hamilton cycle, which does not visit any directed edge more than once [41, 42]. However, Algorithm 5 is not guaranteed to find this optimal path. While it eventually explores all possible paths, and thus the optimal path, the greedy Swap algorithm (Algorithm 2) may discard a PoI that is part of the optimal path.

4.4.3 Analysis

In both algorithms, the searching procedure runs until either termination condition is satisfied: (1) all the paths have been explored, or (2) no more path with greater diversity exists. For Algorithm 4, we can guarantee that all paths have been explored after at most $|V|$ iterations—each issuing a ϵ-range query at a vertex v for the informed search forward estimation. Assuming that ϵ is small, and assume that an R-Tree can support range queries on two-dimensional data in

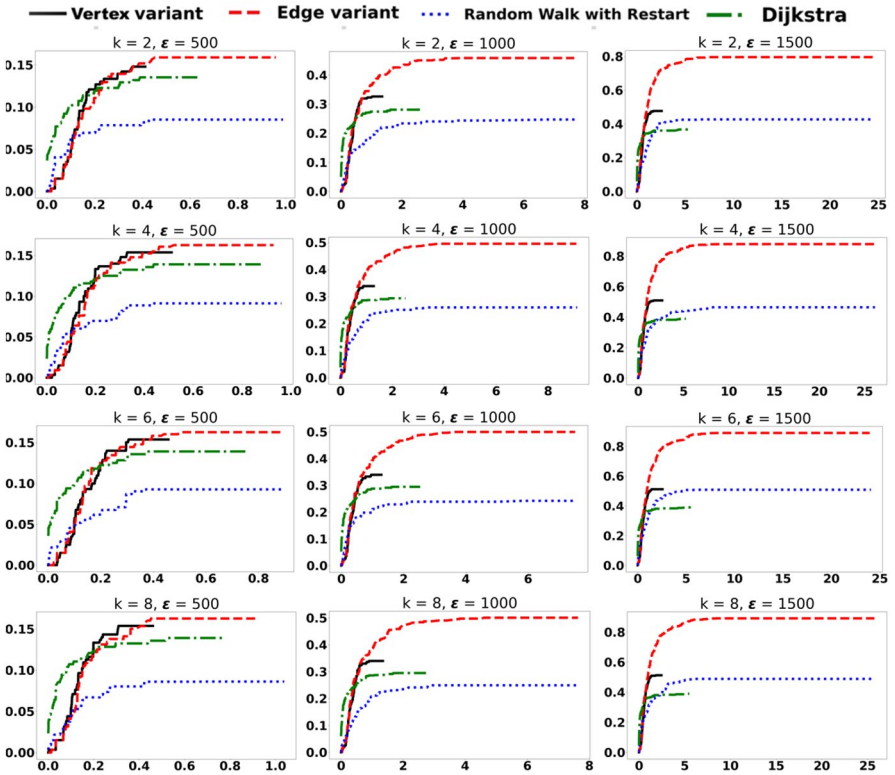


Fig. 11 Experimental results with short distance budget: k_TDPQ. Each subfigure shows the average diversity (y-axis) of 300 queries given the specified (x-axis) processing time (in seconds)

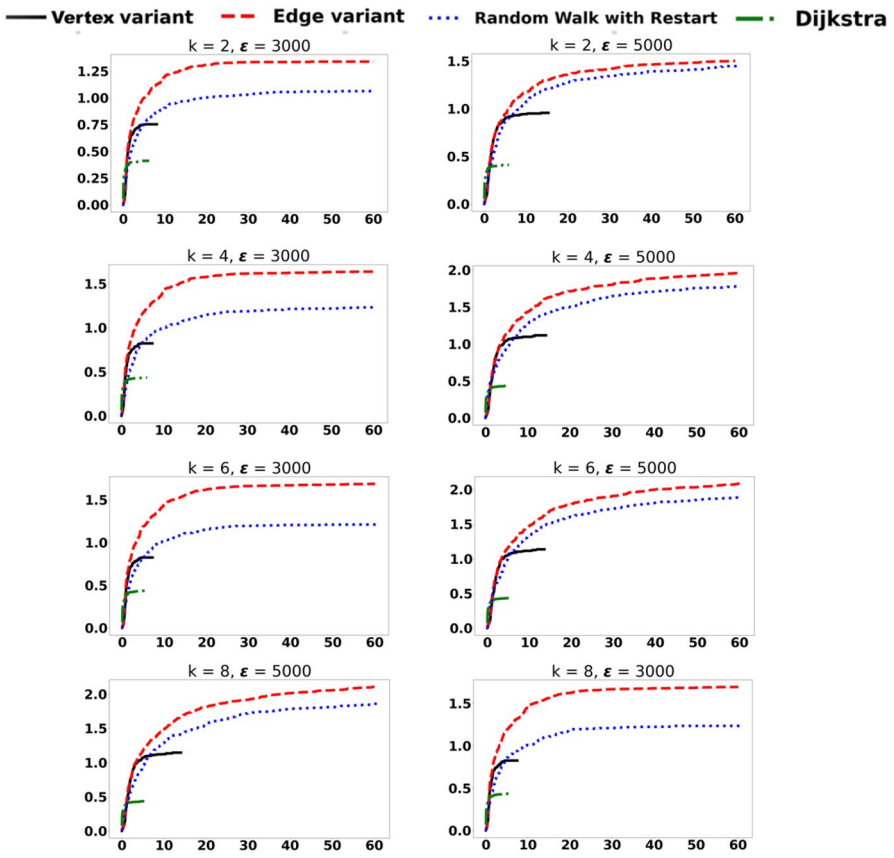


Fig. 12 Experimental results with large distance budget: k_TDPQ. Each subfigure shows the average diversity (y-axis) of 300 queries given the specified (x-axis) processing time (in seconds)

$O(\log |V|)$ in the average case [43], this algorithm has a run-time complexity of $O(|V| \cdot \log |V|)$.

For Algorithm 5, we can not guarantee a polynomial run-time. This algorithm explores the set of all possible simple paths, which is exponential in the range ε. In the worst-case, where the network is a single clique connecting all nodes at the same cost, the early termination criterion using Equation 1 cannot hold, such that all paths must be explored. Despite the exponential worst-case complexity, our experiments show that this algorithm terminates early in real-world settings.

5 Terminal constraint (k_TDPQ)

We now discuss in detail the processing of the k_TDPQ variant.

As mentioned, we assume that in addition to the initial location Q, we now have a targeted terminal location T. The main observation is that now the (calculation of the) lower distance bound changes. Namely, instead of considering only PoIs which are inside the disk centered at Q and with radius ε—now we have an ellipse $E_{k_T\text{DPQ}}$ with:

1. Focal points at Q and T;
2. Major axis $a = \varepsilon/2$

The reason for the value of the major axis a is that, as is well known from the definition of an ellipse as a locus of points, the sum of distances $d(Q,P) + d(T,P) \leq 2a$ for every point in the interior or along the boundary of the ellipse.

Figure 4 illustrates the basic properties and their use in the processing of K_TDPQ. As shown, the PoI P_1, even if it yields a good semantic diversity, cannot be incorporated as part of the path, since the it is outside of the ellipse $E_{k_T\text{DPQ}}$. In contrast, the PoI P_2 is in its interior—hence, it is feasible to incorporate it in the path, and still have a sufficient budget to travel to T. What is also shown in Fig. 4 is that starting at P_2, the pruning can be recursively repeated, except now we have another ellipse bounding the feasible PoIs on the sequence from P_2 towards T. Namely, travelling from Q to P_2 has already "consumed" $d(Q,P_2)$ of the overall ε. Thus, the distance budget from P_2 towards T is $\varepsilon' = \varepsilon - d(Q,P_2)$. This defines a new value for the major axis of the subsequent ellipse $a' = \varepsilon'/2$, and the focal points are now P_2 and T.

In general, the equation of an ellipse [44] which is centered at a point (x_c, y_c) with major axis a and minor axis b, when the axes are rotated for an angle θ with respect to the main coordinate axes is specified as:

$$\frac{[(x-x_c)\cos\theta + (y-y_c)\sin\theta]^2}{a^2} + \frac{[(x-x_c)\sin\theta + (y-y_c)\cos\theta]^2}{b^2} = 1 \qquad (3)$$

To determine the parameters in Equation 3, we rely on the known (values of the) arguments from the k_TDPQ, which include the location of the starting point Q, terminal point T, and the distance budget ε. We have the following:

$$\begin{aligned} x_C = (x_Q + x_T)/2, \quad y_C = (y_Q + y_T)/2 \\ c = \sqrt{(x_Q - x_T)^2 + (y_Q - y_T)^2}/2 \\ a = \varepsilon/2, \quad b = \sqrt{a^2 - c^2} \end{aligned} \qquad (4)$$

One last parameter to determine is the angle θ. Towards that, let s denote the slope of the line between Q and T (i.e., $s = (y_T - y_Q)/(x_T - x_Q)$). Then, θ is simply arctan(s).

The main impact of the fixed terminal point is, in a sense, in the flexibility of the (execution of) Algorithm 2—more specifically, in the (boundary of the) available PoIs and the value of δ in Equation 2. Specifically, before conducting the "if—else" test (Lines 3 ∼ 9) in Algorithm 2, we need properly update the leftover budget ε', and set the next focal point (note that one focal point is always fixed to be the terminal PoI T).

The way this observation is translated more explicitly in the processing algorithms (cf. Sect. 4) is that instead of simple `if` test whether $adj\_d \leq \varepsilon$ (Line 16 in Algorithm 4 and Line 15 in Algorithm 5), we need to validate if the candidate vertex is actually inside the ellipse with foci in the starting point and terminal, and with major axis equal to the half of distance budget.

We now turn the attention to the changes regarding the use of `DAR`-Tree for processing k_TDPQ. Its creation (as part of pre-processing) is not affected—what is changed is the *MBR* of the range query (cf. Algorithm 3). Specifically, what we need now is an axes-parallel *MBR* for $E_{k_T\text{DPQ}}$. Figure 5 shows the corresponding modification of the top-portion of Fig. 3 from Sect. 4.3, which illustrates the ellipse $E_{k_T\text{DPQ}}$ with the foci Q and T and its corresponding axes-parallel *MBR* (dashed purple edges). The construction of that MBR is based on calculating the horizontal and vertical tangents (i.e., the extremal points of E_{k_T}DPQ with respect to each of the axes) [44, 45]. But one possible approach is to convert Equation 3 for a general (i.e., rotated and translated) ellipse into its parametric form:

$$\begin{aligned} x &= x_c + a \times \cos(t)\cos(\theta) - b \times \sin(t)\sin(\theta) \\ y &= y_c + b \times \sin(t)\cos(\theta) + a \times \cos(t)\sin(\theta) \end{aligned} \tag{5}$$

where $t \in [-\pi, \pi]$.

Taking the derivatives dx/dt and dy/dt, setting them to 0 and finding the corresponding solutions in terms of t will yield the corresponding two values of the extreme points that can be used for determining the axes-parallel MBR of $E_{k_T\text{DPQ}}$.

We close this section with a note that when the user would like to have the terminal point coincide with the starting point of the trip (e.g., starting at the hotel where the user is staying, and returning there), the initial ellipse $E_{k_T\text{DPQ}}$ degenerates to a circle with raduis $\varepsilon/2$. However, once the first PoI is selected, the subsequent $k-2$ of them (with the last one being T) will use the iterative construction of the ellipses with foci at the most recently selected PoI and T, appropriately modifying the residual budget (i.e., the shape of the ellipses).

6 Experimental evaluation

We now present a comparative study of our proposed algorithms against two baseline approaches, using real-world datasets. The datasets used for constructing the PoI Network consist of two main components: (1) Road network obtained from OpenStreetMap; (2) Attractions as well the related reviews crawled from

TripAdvisor. Obtaining this data for Manhattan, New York City, USA, yields a road network having 55, 686 vertices, 140, 983 edges and 622 attractions. We note that only the PoIs listed on the first 20 pages at TripAdvisor are utilized in our experiment due to further PoIs not having sufficient textual reviews to generate a latent topic distribution. On average, each PoI is associated with 27.25 reviews and each reviews contains an average of 29.42 words (Table 1).

To demonstrate the effectiveness, we use Dijkstra algorithm [46] and *Random Walk with Restart* (RWR) [47] as the alternative baselines. In order to fairly compare the performances of each algorithms, we set the timer for RWR as the maximum computation time of the other three algorithms under the same experiment settings.

The experiments are conducted on a PC with Intel(R) Xeon(R) CPU E3-1240 v6 @3.70GHz, 32 GB RAM and 512 GB disk storage. Windows 10 Enterprise 64-bit is the operating system, and all the algorithms are implemented by Python 3. Both the datasets and code are available at https://github.com/XTRunner/k-Most-Diverse-Path-Query.git.

6.1 Latent topic model

The dataset used for training latent topic-based diversity model is as well from TripAdvisor, which includes 1, 626 attractions in four cities across the U.S.—Chicago, Miami, Washington D.C. and San Diego. Each of them, on average, is associated with 18.54 reviews and each reviews contains 30.68 words. To demonstrate the validity of our trained model, Tabble 1 shows the ten largest probability values of each topic.

Intuitively, we can observe that the six topics clustered by our learning model are reasonable, which, based on the keywords in each topic, correspond to memorial building, beach park, theater, shop & restaurant, bar, and museum. Some keywords appear in most topics, such as “great” and “see”, but with distinct frequencies. Moreover, there are many discriminative and informative keywords, e.g. “memorial”, “theater” and “museum”, appearing with relatively high probability in specific topic only. Furthermore, in each topic, all the keywords are holding a close relationship between each others, such like “memorial”, “monument” and “war” in topic 1, as well “museum”, “art” and “history” in topic 6.

6.2 Comparison of searching strategies—*k*DPQ

To show that our proposed searching strategies are generally applicable, 150 vertices are picked uniformly at random. Figure 6 presents the results when the distance range ε is relatively limited, i.e., 500 and 1000. The x-axis shows the computation time in seconds and the y-axis is the diversity score. For brevity, we show the results for $k = 2$ and $k = 5$. As can be clearly observed, RWR (dotted blue line) is always able to find some good paths in a short time at the very beginning due to its cheap computational complexity. However, our proposed algorithms, either VSS-kDPQ (Algorithm 4, black solid line) or ESS-kDPQ (Algorithm 5, red dashed

line), outperform RWR in terms of diversity of the result after a few seconds. Dijkstra's algorithm (green dash-dot line), as a breadth-first searching strategy, always achieves the lowest diversity, as it explores parts of the network that may have few (or not) points of interest. Specifically, for $\varepsilon = 500m$, we observe that for both $k = 2$ and $k = 5$, VSS-kDPQ and ESS-kDPQ terminate in an average of two and three seconds, respectively. Note that in Fig. 6, we discontinue drawing the achieved diversity of an algorithm once it has terminated.

We further observe that for the case of $\varepsilon = 500m$ the random walk approach is able to achieve the highest diversity. This is due to relatively low number of possible paths having this cost, allowing RWR to converge on any of them. While ESS-kDPQ is also guaranteed to find the best path (as it explores all possible paths), the order in which it processes PoIs may lead to discard PoIs that are part of the optimal solution in the Swap heuristic used to select the k-most diverse subset (Algorithm 2). The RWR baseline suffers from the same problem (as it also uses the Swap heuristic to select PoIs), but RWR is able to restart to possibly find the same PoIs in a different order to correct the Swap heuristic.

For a range of $\varepsilon = 1000m$, we see that the much large set of possible paths prevents RWR from finding better solutions than our proposed approaches. In this case, we see that VSS-kDPQ finds a solution in about 8 seconds, whereas ESS-kDPQ takes about 20 seconds. We also note that in this case, all competitor approaches yield the approximately same diversity among the 150 queries.

For large query ranges ε, our results are shown in Fig. 7. We observe that our proposed solutions more clearly outperform the baselines when we enlarge the distance range to 1, $500m$ or 3, $000m$. Note that when $\varepsilon = 3,000m$ and $k = 5$, the computation time for ESS-kDPQ is around $600s$. Yet, we observe that the result of the ESS-kDPQ outperforms all other approaches in terms of result diversity after about $200s$, yielding even more diversity beyond that.

We further observe that for the case of $k = 2$, the RWR approach yields the highest diversity (approaching a diversity of 2.0) in the least amount of time. This is because there may be many combinations of attractions that are perfectly diverse, i.e., have (near-) zero overlap among their topics. RWR is able to randomly find any such pair of attractions quickly. However, for $k = 5$, we observe that RWR has a much harder time, i.e., it requires more time to randomly run into a good combination of five PoIs. Yet, the random walk does converge such that, given infinite time, RWR with almost certainly (i.e., with a probability approach 1) find the optimal path, but for large search ranges and $k > 2$, this may take a very long time.

In addition to comparing wall-clock time, we analyzed the number of network edges explored by each algorithm as a system-independent measure of I/O operation. Figure 8 shows the number of explored edges for each algorithm, averaged for $k = 2, 3, 4, 5$. We observe similar behavior as for the run-time experiments: The RWR baseline aimlessly explores edges hoping to accidentally find PoIs of complementary diversity; VSS-kDPQ quickly yields high diversity results, but gets outperformed by ESS-kDPQ after a large number of explored edges. Both outperform the Dijkstra baseline.

In sum, VSS-kDPQ consistently obtains high-diversity results in just a few seconds in all settings. Although VSS-kDPQ outperforms the competitors during its

whole running time, it eventually terminates not finding any more diverse results. ESS-kDPQ continues searching and is able to discover more diverse results given longer time. Thus, it is fair to state that VSS-kDPQ is our best choice if fast response time is required, while ESS-kDPQ retrieves a better path with greater diversity given enough time. We note that RWR is a good choice for very small ranges or $k \leq 2$, while the Dijkstra baseline is dominated by other solutions.

6.3 Comparison of searching strategies—k_TDPQ

We now present the experimental evaluations regarding the processing of K_TDPQ. As it turns out, adding the terminal location as a constraint has interesting implications on the behavior of the processing algorithms and their standing with respect to the baselines. For this experiment, we have 300 different pairs (Q, T) of initial location Q and terminal location T on the Manhattan road network. For additional details and reproducibility, details and visualization of initial and terminal locations can be found at the github, https://github.com/XTRunner/k-Most-Diverse-Path-Query.git. To justify the efficiency and effectiveness of our algorithms, around 2/3 of (Q, T) pairs are randomly selected from the Manhattan area with high PoI density and the rest are distributed in other area. Moreover, recall that we have distance limit ε in our query settings, and thus the distance between Q and T is also significant. In our experiment, 126 pairs of (Q, T) have network distances between 0 and 500 meters and 110 pairs have network distances from 500 meters to 1k meters. The remaining 64 pairs have network distances above 1k meters but less than 2k meters. For each of the 300 resulting (Q, T)-pairs we demonstrate diversity and run-time for $k_T DPQ(\mathcal{G}, Q, T, div, \varepsilon, k)$ using the Manhattan PoI-network $\mathcal{G}$, and the topic-based diversity function div described in Sect. 3.3. We report our results for $k = 2, 4, 6, 8$ (i.e., diverse PoIs to be returned) and having distance threshold $\varepsilon = 500, 1000, 1500, 3000, 5000$m.

6.3.1 Qualitative evaluation of returned paths

In this section, we first present two example paths retrieved from baselines and our proposed algorithms to demonstrate the practicality of our proposed query.

Figure 9 shows the first example with distance limit 3.5k meters and $k = 3$, where green diamond and purple star indicate initial and terminal location, respectively. Dijkstra algorithm (Fig. 9a) happened to meet one PoI along its shortest path, while RWR, after randomly searching for long enough time, discovered a great path with 3 PoIs on it. However, as the names of PoIs in Fig. 9b indicate, “Asia Society and Museum” and “China Cultural Center” are semantically similar to each other and thus the diversity was lower than the tuple combination collected by VSS-k_TDPQ in Fig. 9c. Eventually, Fig. 9D presents the path found by ESS-k_TDPQ with the greatest diversity. Compared Fig. 9c and D, we can observe the idea behind our proposed Algorithm 5—restrict visiting each vertex no more than once will prune the optimal result.

Figure 10 shows another example where initial and terminal location are geographically near to each other. We have distance budget 1.5k meters and assume the visitor prefers not to visit more than 2 PoIs. In such scenario, we can see from Fig. 10a and b that both Dijkstra and RWR did not have luck on finding any PoI at this time. Our proposed algorithms was able to further explore the network beyond the terminal location (shown as purple star). Due to the restriction of visitng each vertex at most once, VSS-k_TDPQ was not able to collect "The Drawing Center" as ESS-k_TDPQ did in Fig. 10d.

6.3.2 Evaluation of successful searches

Due to the additional constraint of having to terminate at location T, under certain settings of k and ε it may not be possible to find any path between Q and T, regardless of diversity.

For consistency, we use VSS-k_TDPQ and ESS-k_TDPQ to denote the vertex and edge variants (adaptations of Algorithms 4 and 5). Table 2 shows the number of unsuccessful k_TDPQ queries for the two proposed approaches, as well as for the two baseline approaches RWR and Dijkstra. Each experiment was terminated after 60s of searching time, and a query was consider successful if any path between Q and T (regardless of the diversity) was returned.

First, we observe that Dijkstra algorithm, whose result is only related to the given distance budget, is able to succeed in finding a path (regardless of diversity) for all cases having a distance budget $\varepsilon \geq 3000$m. For $\varepsilon = 1500$m, there are 21 unsuccessful cases for Dijkstra algorithm to find a path, which is because 21 of the 300 generated (Q, T) of initial location Q and terminal T have a network distance greater than 1500m and there simply exists no path having a distance of 1500m or less. While Dijkstra algorithm has the best success in terms finding (any) path between Q and T, it turns out that the returned paths often do not exhibit high semantic diversity. For RWR we discover that for a large number of (Q, T)-pairs it is not able to find a path. In particular when $\varepsilon = 1500$m, for the cases where the distance between Q and T is large, it become exceedingly unlikely that RWR randomly chooses the correct direction to reach T before the distance budget ε is exceeded and RWR is forced to restart.

For both VSS-k_TDPQ and ESS-k_TDPQ we observe that the number of unsuccessful searches is higher than the ones for Dijkstra algorithm, but much lower than RWR. The reason that less successful searches are achieved by our proposed strategies than using Dijkstra algorithm is that both VSS-k_TDPQ and ESS-k_TDPQ prioritize expanding paths greedily towards area of high diversity, which may not necessary lead into a direction from which the terminal T can still be reached in a manner that will retain the total distance bound ε. Since VSS-kDPQ and ESS-kDPQ are not allowed to reuse the same vertex and directed edge, respectively, more than once, choices may lead to explore partial paths from which T can not longer be reached.

Table 2 only presents how many unsuccessful cases of the 300 generated queries any path could not be found, but without assessing the quality in terms of semantic diversity. The diversity of the returned paths is evaluated in the following.

6.3.3 Evaluation of diversity and run-time

Figures 11 and 12 show the run time in seconds (x-axis) versus the average achieved diversity of returned paths (y-axis) for our experiments having $k = 2, 4, 6, 8$ and $\varepsilon = 500, 1000, 1500, 3000, 5000$m. Each individual subfigure can be found at the github, https://github.com/XTRunner/k-Most-Diverse-Path-Query/tree/main/experiment_related/Figures. While all algorithms were given a time limit of 60s to find diverse paths from Q to T, we limit the x-axis to exclude time intervals where we could not observe any major changes in the values of the diversity. For these experiment, we treat any query for which no path between Q and T (see Table 2) was returned as a diversity of zero.

We can observe that with limited distance budget $\varepsilon = 500, 1000, 1500$m (Fig. 11), the diversity on y-axis is always below 1.0, while the diversity is increased above 1.0 with greater ε (Fig. 12). One thing that becomes apparent is that the edge variant ESS-k_TDPQ (red dashed line in both Figs. 11 and 12) yields the highest diversity paths across all settings if given sufficient processing time. Particularly, for $\varepsilon = 1000, 1500, 3000$m, the diversity difference between ESS-k_TDPQ and other algorithms is substantial—after more than 1s of time, ESS-k_TDPQ outperforms all others in terms of diversity. However, the gap is relatively small when $\varepsilon = 500$m because the searching space is limited and when $\varepsilon = 5000$m because RWR is given more flexibility on both budget and running time. Another interesting observation is the performance of RWR, which yields low diversity for the case of $\varepsilon = 1500$m. But as the distance budget ε increases, which gives RWR more options to explore the network before it has to restart because of exceeding ε, the chance of finding random paths between Q and T which may have (by chance) high diversity is enlarged. Due to this observation, we realize that for $\varepsilon = 5000$m, RWR yields comparable results as ESS-k_TDPQ, in particular for lower values of k. For VSS-k_TDPQ, we discover that the algorithm quickly terminates, but the resulting diversity of paths is not very ideal. In our experiments, VSS-k_TDPQ and Dijkstra algorithm are dominated by ESS-k_TDPQ for $k_T DPQ$ queries, such that there are no settings of distance budget ε, number of returned PoIs k, and run-time budget (x-axis in Figs. 11 and 12) where these approaches perform better.

We conclude this section with the observation that ESS-k_TDPQ is consistently able to retrieve high diversity paths between starting location Q and terminal T. While Dijkstra is always able to find a path (if a path satisfying ε exists), the diversity of such a path is generally low, as this approach does not aim at visiting high-diversity PoIs across the generated path.

7 Conclusions

We introduced two novel types of queries—kDPQ and k_TDPQ—which enable the users to generate a path that will ensure a visit of k PoI's with high diversity, while ensuring that the total trip to visit them, from a given starting vertex Q, is within a user-specified bound ε. The main difference between them is that k_TDPQ also provides the option of specifying a desired terminal location for the trip.

We also introduced algorithmic solutions for their processing which, for efficiency, rely on a novel indexing structure—DAR-Tree. In addition to the spatial component of the R-tree, DAR-Tree also stores in each directory node the respective upper-bounds of diversity achievable by all PoIs whose spatial locations are inside that node.

Our proposed algorithms quickly retrieve high-diversity paths in an A*-like way, by greedily exploring network vertices that promise the highest potential gain using a forward estimation using the maximum possible diversity retrieved from the index. Our experimental evaluation using real-world data from OpenStreetMap demonstrated that the proposed algorithms outperform the baseline based on a breadth-first search and random walks, and provide a trade-off between run-time and path diversity. We also observed that for the k_TDPQ variant—while enabling the flexibility of selecting a terminal—the benefits of the vertex variant (cf. Algorithm 4) are not manifested when the distance limit ε is close to the distance between the starting point and the terminal, while the edge variant is still able to discover great-diversity semantic path.

Our future work will focus on three aspects: (1) investigate the impact of different diversity measures; (2) investigate extensions to kDPQ and k_TDPQ that will enable returning a collection of trajectories that will capture other constraints (e.g., price limit for PoI visits), along with different aggregating indexing structures; and (3) develop data structures and algorithms which will enable efficient updates to the active paths when traffic conditions change or PoI descriptors (e.g., different lunch/dinner menu; special exhibits) are updated.

Acknowledgements Dr. Züfle is supported by National Science Foundation AitF Grant CCF-1637541. Dr. Trajcevski is supported by National Science Foundation Grant SWIFT 203024.

References

1. Roussopoulos, N., Kelley, S., Vincent, F.: Nearest neighbor queries. SIGMOD Rec. **24**, 71–79 (1995)
2. Benetis, R., Jensen, C.S., Karĉiauskas, G., Ŝaltenis, S.: Nearest and reverse nearest neighbor queries for moving objects. VLDB J. **15**(3), 229–249 (2006)
3. Bao, J., Chow, C.-Y., Mokbel, M.F., Ku, W.-S.: Efficient evaluation of k-range nearest neighbor queries in road networks. In: 2010 Eleventh International Conference on Mobile Data Management, pp. 115–124 (2010)
4. Zheng, K., Shang, S., Yuan, N.J., Yang, Y.: Towards efficient search for activity trajectories. In: 2013 IEEE 29th International Conference on Data Engineering (ICDE), pp. 230–241 (2013)

5. Alvares, L.O., Bogorny, V., Kuijpers, B., de Macedo, J.A.F., Moelans, B., Vaisman, A.: A model for enriching trajectories with semantic geographical information. In: Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems, pp. 1–8 (2007)
6. Parent, C., Spaccapietra, S., Renso, C., Andrienko, G., Andrienko, N., Bogorny, V., Damiani, M.L., Gkoulalas-Divanis, A., Macedo, J., Pelekis, N., Theodoridis, Y., Yan, Z.: Semantic trajectories modeling and analysis. ACM Comput. Surv. **45**(4), 1–32 (2013)
7. Costa, C.F., Nascimento, M.A.: Towards spatially-and category-wise k-diverse nearest neighbors queries. In: Advances in Spatial and Temporal Databases, pp. 163–181 (2017)
8. Teng, X., Yang, J., Kim, J.-S., Trajcevski, G., Züfle, A., Nascimento, M.A.: Fine-grained diversification of proximity constrained queries on road networks. In: Proceedings of the 16th International Symposium on Spatial and Temporal Databases, pp. 51–60 (2019)
9. Costa, C.F., Nascimento, M.A., Schubert, M.: Diverse nearest neighbors queries using linear skylines. GeoInformatica **22**(4), 815–844 (2018)
10. Grambow, G., Oberhauser, R., Reichert, M.: Semantically-driven workflow generation using declarative modeling for processes in software engineering. In: 2011 IEEE 15th International Enterprise Distributed Object Computing Conference Workshops, pp. 164–173 (2011)
11. Wong, P.Y.H., Gibbons, J.: A process semantics for BPMN. Formal Methods Softw Eng **5256**, 355–374 (2008)
12. Kelci, M., Pratt, R., Galati, M.: The traveling salesman traverses the genome: using sas® optimization in jmp® genomics to build genetic maps. In: In SAS Global Forum (2012)
13. Teng, X., Trajcevski, G., Kim, J., Züfle, A.: Semantically diverse path search. In: 21st IEEE International Conference on Mobile Data Management (MDM), pp. 69–78 (2020)
14. Issa, H., Damiani, M.L.: Efficient access to temporally overlaying spatial and textual trajectories. In: 17th IEEE International Conference on Mobile Data Management (MDM), pp. 262–271 (2016)
15. Carbonell, J., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pp. 335–336 (1998)
16. Jain, A., Sarda, P., Haritsa, J.R.: Providing diversity in k-nearest neighbor query results. In: Advances in Knowledge Discovery and Data Mining, pp. 404–413 (2004)
17. Abbar, S., Amer-Yahia, S., Indyk, P., Mahabadi, S., Varadarajan, K.R.: Diverse near neighbor problem. In: Proceedings of the Twenty-Ninth Annual Symposium on Computational Geometry, pp. 207–214 (2013)
18. Vieira, M.R., Razente, H.L., Barioni, M.C., Hadjieleftheriou, M., Srivastava, D., Traina, C., Tsotras, V.J.: On query result diversification. In: IEEE 27th International Conference on Data Engineering, pp. 1163–1174 (2011)
19. Amagata, D., Hara, T.: Diversified set monitoring over distributed data streams. In: Proceedings of the 10th ACM International Conference on Distributed and Event-Based Systems, pp. 1–12 (2016)
20. Lee, K.C.K., Lee, W.-C., Leong, H.V.: Nearest surrounder queries. In: 22nd International Conference on Data Engineering, p. 85 (2006)
21. Kucuktunc, O., Ferhatosmanoglu, H.: λ-diverse nearest neighbors browsing for multidimensional data. IEEE Trans. Knowl. Data Eng. **25**(3), 481–493 (2013)
22. Borzsony, S., Kossmann, D., Stocker, K.: The skyline operator. In: Proceedings 17th International Conference on Data Engineering, pp. 421–430 (2001)
23. Zhang, C., Zhang, Y., Zhang, W., Lin, X., Cheema, M.A., Wang, X.: Diversified spatial keyword search on road networks. In: Advances in Database Technology-EDBT 2014: 17th International Conference on Extending Database Technology, pp. 367–378 (2014)
24. Zheng, B., Zheng, K., Scheuermann, P., Zhou, X., Nguyen, Q.V.H., Li, C.: Searching activity trajectory with keywords. World Wide Web **22**(3), 967–1000 (2019)
25. Rice, M.N., Tsotras, V.J.: Exact graph search algorithms for generalized traveling salesman path problems. In: International Symposium on Experimental Algorithms, pp. 344–355 (2012)
26. Yang, Y., Li, Z., Wang, X., Hu, Q.: Finding the shortest path with vertex constraint over large graphs. Complexity **2019**, 8728245–1872824513 (2019)
27. Teng, X., Trajcevski, G., Züfle, A.: Semantically diverse paths with range and origin constraints. In: Proceedings of the 29th International Conference on Advances in Geographic Information Systems, pp. 375–378 (2021)
28. Gao, R., Li, J., Li, X., Song, C., Zhou, Y.: A personalized point-of-interest recommendation model via fusion of geo-social information. Neurocomputing **273**(C), 159–170 (2018)

29. Han, P., Li, Z., Liu, Y., Zhao, P., Li, J., Wang, H., Shang, S.: Contextualized point-of-interest recommendation. In: Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, pp. 2484–2490 (2020)
30. Liu, Y., Pham, T.-A.N., Cong, G., Yuan, Q.: An experimental evaluation of point-of-interest recommendation in location-based social networks. Proc. VLDB Endow. **10**(10), 1010–1021 (2017)
31. Zhou, F., Yin, R., Zhang, K., Trajcevski, G., Zhong, T., Wu, J.: Adversarial point-of-interest recommendation. In: The World Wide Web Conference, pp. 3462–34618 (2019)
32. Zhao, S., King, I., Lyu, M.R.: A survey of point-of-interest recommendation in location-based social networks. CoRR (2016) 1607.00647
33. Zhou, F., Wu, H., Trajcevski, G., Khokhar, A., Zhang, K.: Semi-supervised trajectory understanding with poi attention for end-to-end trip recommendation. ACM Trans. Spatial Algorithms Syst. **6**(2), 1–25 (2020)
34. Schaake, K., Burgers, J., Mulder, C.H.: Ethnicity, education and income, and residential mobility between neighbourhoods. J. Ethn. Migr. Stud. **40**(4), 512–527 (2014)
35. Yao, J., Wong, D.W.S., Bailey, N., Minton, J.: Spatial segregation measures: a methodological review. Tijdschr. Econ. Soc. Geogr. **110**(3), 235–250 (2019)
36. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: Proceedings of the 31st International Conference on Very Large Data Bases, pp. 853–864 (2005)
37. Papadimitriou, C.H., Steiglitz, K.: Some complexity results for the traveling salesman problem. In: Proceedings of the Eighth Annual ACM Symposium on Theory of Computing, pp. 1–9 (1976)
38. Pearl, J.: Heuristics: Intelligent Search Strategies for Computer Problem Solving. Addison-Wesley Longman Publishing Co., Inc, USA (1984)
39. Guttman, A.: R-trees: A dynamic index structure for spatial searching. In: Proceedings of the 1984 ACM SIGMOD International Conference on Management of Data, pp. 47–57 (1984)
40. Papadias, D., Kalnis, P., Zhang, J., Tao, Y.: Efficient OLAP operations in spatial data warehouses. In: Advances in Spatial and Temporal Databases, pp. 443–459 (2001)
41. Noon, C.E., Bean, J.C.: An efficient transformation of the generalized traveling salesman problem. INFOR: Inform. Syst. Operat. Res. **31**(1), 39–44 (1993)
42. Golden, B., Bodin, L., Doyle, T., Stewart, W., Jr.: Approximate traveling salesman algorithms. Oper. Res. **28**(3–part–ii), 694–711 (1980)
43. Hwang, S., Kwon, K., Cha, S.K., Lee, B.S.: Performance evaluation of main-memory r-tree variants. In: Advances in Spatial and Temporal Databases, pp. 10–27 (2003)
44. Serdarushich, V.: Analytic Geometry. Nabla Ltd (2014)
45. Silverman, R.A.: Modern Calculus and Analytic Geometry. Dover (2012)
46. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik, 269–271 (1959)
47. Tong, H., Faloutsos, C., Pan, J.-Y.: Fast random walk with restart and its applications. In: Sixth International Conference on Data Mining, pp. 613–622 (2006)