Semantically Diverse Path Search

Xu Teng*, Goce Trajcevski*, Joon-Seok Kim†, Andreas Züfle†
*Department of Electrical and Computer Engineering
Iowa State University, Ames, USA
Email: {xuteng,gocet25}@iastate.edu
†Department of Geography and Geoinformation Science
George Mason University, Fairfax, USA
Email: {jkim258,azufle}@gmu.edu

***Abstract*—Location-Based Services are often used to find proximal Points of Interest (PoI) – 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. 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 search strategy. Our experimental evaluations conducted on real datasets demonstrate the benefits of proposed methodology over the baseline alternative approaches.**

***Keywords*-Diversity; Trajectories; Road Networks; Indexing;**

I. 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* (kNN) queries [24, 5, 4]. However, in addition to the proximity, the semantics of PoI is often an influential factor when planning one's motion and activities [30].

While modelling and querying of the, so called, semantic or activity trajectories, has been a subject of intense research in the past decade [2, 22, 30], the semantic aspect was typically used to augment the traditional searches used in typical spatial and spatio-temporal queries (range, kNN, 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.

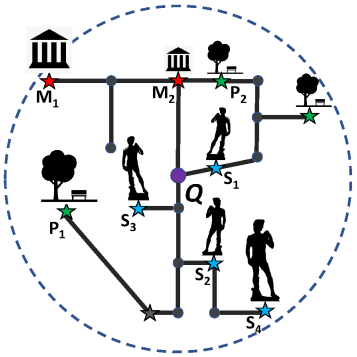


Figure 1. Running Example of Diverse Path Search

Example 1. *Consider the scenario depicted in Figure 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 is willing to travel.*

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 [9, 26, 10]. 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 Ex. 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)

2375-0324/20/$31.00 ©2020 IEEE
DOI 10.1109/MDM48529.2020.00028

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.

To remedy this limitation, this paper introduces a new query type, the k-Diverse Path Query (kDPQ). The goal of kDPQ 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*. Towards processing the kDPQs, we propose three algorithms. While one can always construct a straightforward baseline based on Dijsktra's algorithm, in this work we propose an index structure, called *Diversity Aggregated R-tree* (DAR-tree), devised to improve the efficiency of the kDPQ processing. Specifically, the *DAR*-tree enables the two algorithms that we propose, which are able to navigate the space of possible paths more efficiently, while maximize diversity of PoIs. Our experimental evaluation, where real-world road network and PoI data from Open Street Map 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.

The remainder of this paper is organized as follows. We survey state-of-the-art methods related to diverse nearest PoI search in Section II. Our proposed problem – processing of kDPQ is formally defined in Section III. Section IV presents our solutions in detail, including the *DAR*-tree and query processing algorithm that leverage this index structure. Our experimental evaluation is presented in Section V, and we conclude this work in Section VI.

II. 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 [22, 2] and the query processing aspects [30, 15] 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 highest diversity in terms of their semantic descriptors.

The concept of incorporating diversity into queries answers has its origins in the information retrieval – specifically, in similarity search among documents. The Maximal Marginal Relevance (*MMR*) model [8] 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 kNNs to a given query point q 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 [1]. A k-similar diversification set which optimizes a linear functions combining the similarity (i.e., closeness) and diversity for a given trade-off between them has been studied in [28]. Monitoring the most diverse k-sized set over distributed sets was proposed in [3]. 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 a any path that allows to visit the resulting PoIs.

Other recent works that have combined the diversity and spatial contexts are presented in [9] and [10] in the context of NN queries, tackling the settings of optimizing the weighted sums of the constraints. Our previous work [26] introduced a *k-Diversified Range Query* (kDRQ) 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 kDP queries generate a path that connects multiple PoIs that, ensuring high diversity. More distantly related approaches to spatial diversification include angular diversity [18] – 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 kNN problem in [17].

Relying on the Skyline paradigm [6], finding the set of all optimal solutions for a given linear combination of two diversity notions, spatial and categorical, is presented in [9]. 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 kNNs has been explored in [29]. In that work the keywords are used for filtering data points, i.e., only points that contain *all* query keywords are considered. We, on the other hand, 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. Moreover, differently from the works above, we propose an indexing structure to speedup the processing of kDRQs.

III. Background and Problem Definition

In this section, we introduce the basic terminology and the settings, after which we proceed with the formal definition of the kDPQ problem. We firstly define the problem of finding the k most-diverse path for an abstract diversity metric, and then introduced the diversity function employed in this study.

A. Preliminaries

Definition 1 (Road Network). *A* Road Network $G = (V, E, W)$ *is a weighted directed graph, where* V *is a set of nodes and each node* $v \in V$ *is associated with location-attribute* $v.L$*;* $E \subseteq V \times V$ *represents the set of edges between pairs of nodes* (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*.*

Nodes 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* $p.L \in \{v.L | v \in V\}$ *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 node* $v \in V$*, we let* $v.P$ *denote the (possibly empty) set of PoIs located at node* 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.*

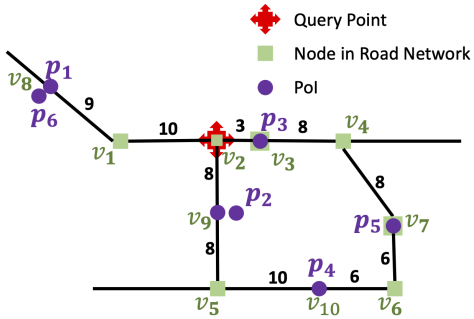
We note that in practice, a particular PoI p may not be directly located at a node of the road network. In such case, we apply map-matching to project the PoI to the nearest point on an edge of the road network [7]. The projected point becomes a new (virtual) node of the network that corresponds to the $p.L$.

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 [25] to store the road network (Lines 3-6) to efficiently retrieve the nearest neighbor edge to a PoI (Line 9). The update in Line 11 adds a new node to the network, and replaces the corresponding edge (i.e., $nearest\_edge$) with two new edges connecting the new (virtual location) node to the nodes of $nearest\_edge$, and replicating the original weight of the $nearest\_edge$ to both new edges.

Fig. 2 presents a small-scale example of a PoI network, having five PoIs $P = \{p_1, p_2, p_3, p_4, p_5, p_6\}$ (shown as purple circles) and a road network having $|V| = 7$ nodes (shown as green rectangles), several bidirectional edges E connecting nodes (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 nodes at the same location. Using Algorithm 1, three new nodes – 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 $v_8.P = \{p_1, p_6\}$.

Algorithm 1: PoI Network Construction

```
   Input: G = (V, E, W), P
1  Copy G as initial G with v.S = ∅ foreach v in V
2  tree ←R-tree()
3  foreach e in E do
4      rect ← rectangle whose diagonal is e
5      tree.insert(rect)
6  end
7  foreach p in P do
8      if p.L = v.L where v ∈ G then v.P.add(p) ;
9      nearest_edge ← tree.nearest_neighbor(p.L)
10     v.L ← Project p onto nearest_edge which minimizes
         distance to p.L
11     Update G with new node (v.L, {p})
12 end
13 return 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}

Figure 2. Example of PoI Network

B. 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 nodes 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 edge 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 nodes 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 node 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 Def. 3 does not require a path to be *simple*, i.e., it allows a path to have cycles and visit the same node 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. Let k denote that limit. 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.,*

$$kDS_{div}(sp, k) = arg\,max_{[P \subseteq sp.collection, |P| \leq k]} div(P)$$

We note that the specification of a diversity function $div(P)$ that maps a set of PoIs P to a diversity score is left abstract in Def. 5, and multiple definitions of diversity have been used in the literature [9, 26]. In this work, we employ the topic-based probabilistic diversity proposed in [26], which is reviewed in detail in Section III-C.

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 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-diverse path query ($kDPQ$) is defined as*

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

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 in Figure 2, let $\varepsilon = 35$ and $k = 2$, two possible paths are 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 $kDPQ(\mathcal{G}, Q, div, \varepsilon, k) = sp_2$ and recommends to visit PoIs p_3 and p_4 on this path.*

Regardless of the diversity function div (cf. Section III-C), we observe the following hardness result:

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

Proof: Let tsp be solution to the traveling salesman problem (TSP) on an arbitrary graph $\mathcal{G}$ starting at an arbitrary node 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 [21]. ■

Due to the complexity of kDPQ, 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.

C. Topic-Based Diversity

In this work, we leverage the topic-based diversity proposed in [26] which extracts K latent topics from textual context of each PoI, where K is a user-specified parameter. Based on textual description $p_i.I$ 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.*

IV. Methodology

To efficiently answer kDPQ, we adopt informed search [23] which, in general, can be considered as greedy algorithm, whereby a node is selected for exploration based on the priority from evaluation function. The evaluation function for solving kDPQ is constructed as the estimated gain of probabilistic diversity, thus the node with the greatest evaluation would be explored first. The quality of the evaluation function is critical for the searching procedure.

Algorithm 2: Swap Algorithm

```
Input: Set of PoIs P, Integer k
res ← ∅
foreach p ∈ P do
    if |res| < k then res ← res ∪ p ;
    else
        C ← res ∪ p
        worst_site ← arg max_{p'∈C} div(C \ p')
        res ← C \ worst_site
return res
```

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 heuristic function. Subsequently, the informed search algorithm is presented.

A. Heuristic and Evaluation Function

For a PoI network $\mathcal{G} = (V, E, W, \mathcal{P})$, node $v \in V$, 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:

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

Note that Euclidean distance, which is used in heuristic function, is always less 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 2. *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 2, 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 graph $\mathcal{G}$.

B. 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 node 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 nodes 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 nodes explored by a path and let δ be the cost of this path. For any adjacent node to extend the path, we evaluate the following function:

$$f(res, v, \varepsilon - \delta, k) = div(Swap(res \bigcup ProbDiv(P_E(v, \varepsilon - \delta) \setminus res), k))$$

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 nodes 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. Alg. 2, proposed in [28]) 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 2 ensures that the diversity of PoIs inside the Euclidean range is $\geq$ to 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.

C. Diversity Aggregated R-Tree

The main point of utilizing Euclidean distance in heuristic function is to leverage an R-Tree [13] to efficiently obtain the set $P_E(v, \varepsilon)$ of PoIs within a Euclidean range around node v while avoiding expensive network exploration to obtain the set $P(v, \varepsilon)$ using, for example, Dijkstra's algorithm. 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 kind of aggregated R-Tree (aR-Tree) [20], storing the information related to probabilistic diversity of each *Minimum Bounding Rectangle* (MBR) in both leaf and inner nodes.

Fig. 3 presents a example of DAR-Tree with 12 PoIs. Each leaf node stores a PoI and its corresponding topic distribution. Besides of the pointer(s) to the child node(s) and the coordinates of MBR, every non-leaf node keeps

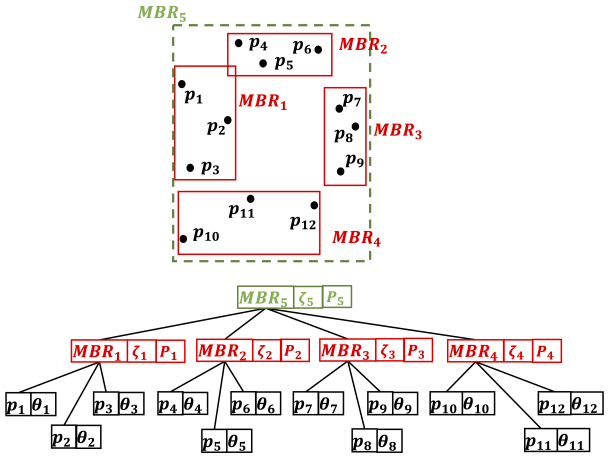


Figure 3. Example of Diversity Aggregated R-Tree

the diversity-related information – 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 [20], we omit details on construction and maintenance of DAR-Tree. However, the benefits of the DAR-Tree, to speed the computation of the heuristic function, are illustrated by Alg. 3. Abstractly 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.

D. Efficient kDPQ Processing – Greedy Best-First Search

Alg. 4 and Alg. 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.

Specifically, Alg. 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 node of degree one). Imposing simple paths, this search algorithm can only visit such a PoI if the path ends there. Alg. 5, in turn, allows to re-visit the nodes but does not allow visiting directed edges more than once.

1) Node-constrained Searching Strategy (NSS-kDPQ): Alg. 4 remembers all the explored nodes to avoid exploring the network redundantly. That is to say, newly generated nodes that match previously explored ones would be discarded so that each node can be visited at most once.

Algorithm 3: DAR-Tree Range Query

Input: PoIs network $\mathcal{G} = (V, E, W, \mathcal{P})$, DAR-Tree $\mathcal{R}$, Node $v \in \mathcal{V}$, Range ε, k-diverse subset res

1 h_val $\leftarrow (1, ..., 1)$
  // |h_val| = the number of categories
2 **Function** *range-search (region, child, res)*
3 **if** *child is a leaf and child* $\notin res$ **then**
4 h_val $\leftarrow$ h_val $\bigodot(1 - \theta_{child})$
  // $\bigodot$ is entrywise product
5 **else if** $region.contains(MBR_{child})$ **then**
6 h_val $\leftarrow$ h_val $\bigodot \zeta_{child}$
7 $dup \leftarrow res \bigcap P_{child}$
8 **if** $dup \neq \emptyset$ **then**
9 h_val $\leftarrow$ h_val $\oslash \prod_{p_i \in dup}(1 - \theta_i)$
  // $\oslash$ is entrywise division
10 **end**
11 **else**
12 **foreach** *gchild of child* **do**
13 **if** $region.intersects(MBR_{gchild})$ **then**
14 $range\text{-}search(region, gchild, res)$
15 **end**
16 **end**
17 **end**
18 **end**
19 $range\text{-}search(circle(v, \varepsilon), \mathcal{R}.root, res)$
20 **return** $(1, ..., 1)-$ h_val // Entrywise subtraction

Algorithm 4: kDPQ Simple Path (node variant)

Input: PoIs network $\mathcal{G} = (V, E, W, \mathcal{P})$, Query point $Q \in \mathcal{G}$, integer k, range ε

1 sp, max_div $\leftarrow$ None, 0
2 queue $\leftarrow$ PriorityQueue()
  // Each node e in queue has 5 components –
  // $e.id$, $e.priority$, $e.dist$, $e.res$, $e.parent$
3 queue $\leftarrow$ Insert($(Q, 0, 0, \{Q.P\}, \text{None})$, queue)
4 explored $\leftarrow \emptyset$
5 **while** queue $\neq \emptyset$ **do**
6 $node \leftarrow$ POP(queue)
7 $div\_score \leftarrow div(node.res)$
  // n is the number of categories
8 **if** $node.priority \leq$ max_div **then return** sp ;
9 **else if** $div\_score >$ max_div **then**
10 sp, max_div $\leftarrow node, div\_score$
11 **end**
12 explored.add($node.id$)
13 **foreach** adj_n *adjacent to* $node.id$ **do**
14 $next\_dist \leftarrow node.dist + W(node.id, adj\_n)$
15 **if** $next\_dist \leq \varepsilon$ **then**
16 $adj\_res \leftarrow$ Swap$(node.res \bigcup adj_n.P, k)$
17 $adj\_prior \leftarrow$ $f(node.res, adj\_n, \varepsilon - next\_dist, k)$
18 $next\_n \leftarrow$ $(adj\_n, adj\_prior, next\_dist, adj\_res, node)$
19 **if** adj_n *is not in* explored *or* queue **then**
20 queue $\leftarrow$ Insert($next\_n$, queue)
21 **else if** adj_n *is in* queue *with lower Priority* **then**
22 replace that queue element with adj_n
23 **end**
24 **end**
25 **end**
26 **end**
27 **return** sp

At the beginning, the priority queue that contains all nodes available for exploration (Line 2) is initialized, and a set for remembering every expanded node (Line 4). For each node in the priority queue, we use a data structure composed of five components: id – the unique identification of the node; *priority* – the potential/approximate diversity measured by evaluation function (Equation 1) if choosing this node to explore; *dist* – the road-network distance from query point Q to this node, *res* – the k-diversified results among the path so far, *parent* – the node in the path that generated this node. Note that $parent$ information enables us to retrieve the whole path from Q to any given point via backtracking.

Our goal is to find the path with greatest diversity, thus an intuitive way to expand first is the node with the highest value from evaluation function $h(v, \varepsilon)$ (Equation 1). The priority queue is sorted by the priority of each nodes in descending order. When a node 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 nodes of the expanded one into priority queue. As mentioned, the explored nodes (recorded in explored set) are not inserted into priority queue again, to avoid duplication. Line 21 is executed when better path is discovered to a node currently in the **queue**.

However, while the searching procedure by Alg. 4 is efficient, as each node 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 nodes.

2) Edge-constrained Searching Strategy (ESS-kDPQ): Alg. 5 is proposed to prioritize diversity rather than efficiency. To achieve that, instead of recording the expanded node globally, we remember the explored directed edges for each path individually. To enforce that capability, a new component – **explored** set – is added for each node in priority queue, and a test (Line 13) takes place to avoid visiting an edges twice. We note that the assumption of visiting each 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 edge more than once [19, 12]. However, Alg. 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 (Alg. 2) may discard a PoI that is part of the optimal path.

3) Analysis: In both algorithms, the searching procedure runs until either termination condition is satisfied: all the paths have been explored or no more path with greater diversity exists. For Algorithm 4, we can guarantee that all paths have been explored after at most $|V|$ iterations –

Algorithm 5: kDPQ Path Search (edge variant)

Input: PoIs network $\mathcal{G} = (V, E, W, \mathcal{P})$, Query point $Q \in \mathcal{G}$, integer k, range ε

sp, max_div $\leftarrow$ None, 0
queue $\leftarrow$ PriorityQueue()
// Each node e in queue has 6 components –
// $e.id$, $e.priority$, $e.dist$, $e.res$, $e.parent$, $e.explored$
queue $\leftarrow$ Insert($(Q, 0, 0, \{Q.P\}, \text{None}, \emptyset)$, queue)
while queue $\neq \emptyset$ **do**
  $node \leftarrow$ POP(queue)
  $div\_score \leftarrow div(node.res)$
  // n is the number of categories
  if $node.priority \leq$ max_div **then**
    return sp
  else if $div\_score >$ max_div **then**
    sp, max_div $\leftarrow node, div\_score$
  end
  foreach adj_n *adjacent to* $node.id$ **do**
    if $(node.id, adj\_n) \notin node.explored$ **then**
      $next\_dist \leftarrow node.dist + W(node.id, adj\_n)$
      if $next\_dist \leq \varepsilon$ **then**
        $adj\_res \leftarrow$ Swap$(node.res \bigcup adj_n.P, k)$
        $adj\_prior \leftarrow f(node.res, adj\_n, \varepsilon - next\_dist, k)$
        $next\_n \leftarrow (adj\_n, adj\_prior, next\_dist, adj\_res, node, node.explored.\text{add}(node.id, adj\_n))$
        if adj_n *is not in* queue **then**
          queue $\leftarrow$ Insert($next\_n$, queue)
        else if adj_n *is in* queue *with lower Priority* **then**
          replace that queue element with adj_n
        end
      end
    end
  end
end
return sp

each issuing a ε-range query at a node 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 $O(\log(n))$ in the average case [14], this algorithm has a run-time complexity of $O(n \cdot \log(n))$.

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.

V. 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 $55,686$ nodes, $140,983$ edges and 622 attractions. On average, each PoI is associated with 27.25 reviews and each reviews contains an average of 29.42 words. To show that our algorithms are generally applicable, 150 nodes are picked uniformly at random.

To demonstrate the effectiveness we use Dijkstra algorithm [11] and *Random Walk with Restart* (RWR) [27] 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 3.5. Both the datasets and code are available at https://github.com/XTRunner/MDM2020.

A. 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, Tab I 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.

B. Comparison of Searching Strategies

Fig. 4 presents the results when the distance range ε is relatively limited, i.e., 500 and $1,000$. 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

Table I
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)

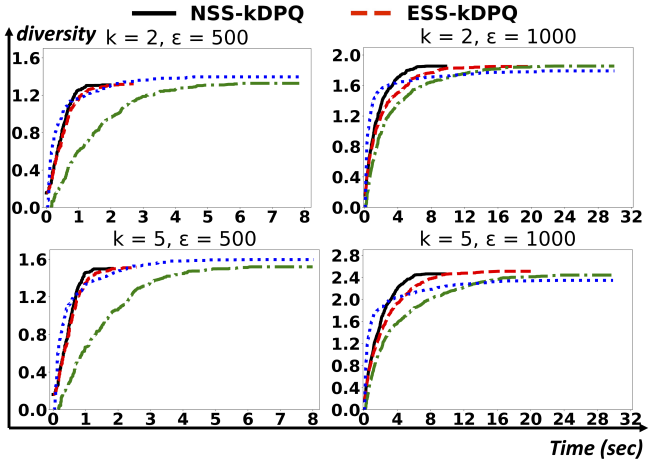


Figure 4. Experimental result of short distance range

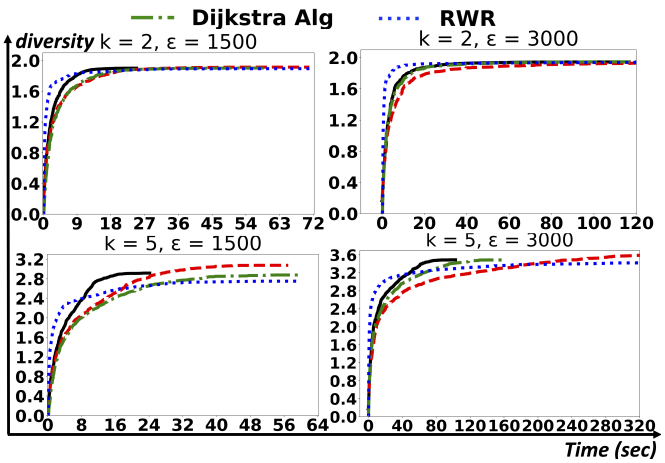


Figure 5. Experimental result of long distance range

in a short time at the very beginning due to its cheap computational complexity. However, our proposed algorithms, either NSS-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 $k = 5$, NSS-kDPQ and ESS-kDPQ terminate in an average of two and three seconds, respectively. Note that in Fig. 4, 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 NSS-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 Figure 5. 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 6 shows the number of explored edges for each algorithm, averaged for $k \in \{2, 3, 4, 5\}$. We observe similar behavior as for the

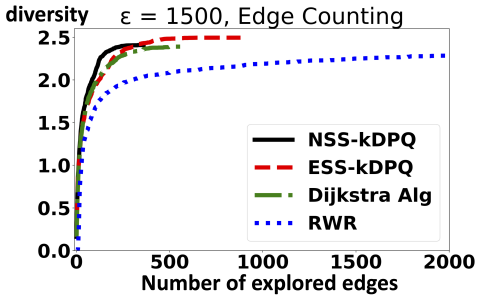


Figure 6. Comparison result regarding of the number of explored edges

run-time experiments: The RWR baseline aimlessly explores edges hoping to accidentally find PoIs of complementary diversity; NSS-kDPQ quickly yields high diversity results, but gets outperformed by ESS-kDPQ after a large number of explored edges. Both approaches the Dijksta baseline.

In sum, NSS-kDPQ consistently obtains high-diversity results in just a few seconds in all settings. Although NSS-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 NSS-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's baseline is dominated by other solutions.

VI. Conclusions

We proposed kDPQ – a novel k diverse path query, which yields a path to visit PoIs with high diversity. Unlike previous works, we can limit the length of the path, thus allowing to satisfy the users mobility constraints. To process kDPQ we leverage the *DAR*-Tree that stores, in each directory node, upper-bounds of diversity achievable by all PoIs inside the node. Our proposed algorithms quickly retrieve high-diversity paths in an A*-like way, by greedily exploring network nodes 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. Our future work investigates 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 CNS 1646107.

References

[1] S. Abbar et al. Diverse near neighbor problem. In *SOCG*, pages 207–214, 2013.
[2] L. O. Alvares, V. Bogorny, B. Kuijpers, J. A. F. de Macedo, B. Moelans, and A. Vaisman. A model for enriching trajectories with semantic geographical information. In *ACM GIS*, page 22. ACM, 2007.
[3] D. Amagata and T. Hara. Diversified set monitoring over distributed data streams. In *DEBS*, pages 1–12. ACM, 2016.
[4] J. Bao, C.-Y. Chow, M. F. Mokbel, and W.-S. Ku. Efficient evaluation of k-range nearest neighbor queries in road networks. In *MDM*, pages 115–124. IEEE, 2010.
[5] R. Benetis, C. S. Jensen, G. Karĉiauskas, and S. Ŝaltenis. Nearest and reverse nearest neighbor queries for moving objects. *The VLDB Journal*, 15(3):229–249, 2006.
[6] S. Börzsönyi et al. The skyline operator. In *ICDE*, pages 421–430, 2001.
[7] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, pages 853–864, 2005.
[8] J. Carbonell and J. Goldstein. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, pages 335–336, 1998.
[9] C. F. Costa and M. A. Nascimento. Towards spatially-and category-wise k-diverse nearest neighbors queries. In *SSTD*, pages 163–181. Springer, 2017.
[10] C. F. Costa, M. A. Nascimento, and M. Schubert. Diverse nearest neighbors queries using linear skylines. *GeoInformatica*, 22(4):815–844, 2018.
[11] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, pages 269–271, 1959.
[12] B. Golden, L. Bodin, T. Doyle, and W. Stewart Jr. Approximate traveling salesman algorithms. *Operations research*, 28(3-part-ii):694–711, 1980.
[13] A. Guttman. R-trees: A dynamic index structure for spatial searching. In *ACM SIGMOD*, 1984.
[14] S. Hwang, K. Kwon, S. K. Cha, and B. S. Lee. Performance evaluation of main-memory r-tree variants. In *SSTD*, pages 10–27. Springer, 2003.
[15] H. Issa and M. L. Damiani. Efficient access to temporally overlaying spatial and textual trajectories. In *MDM*, pages 262–271, 2016.
[16] A. Jain, P. Sarda, and J. R. Haritsa. Providing diversity in k-nearest neighbor query results. In *PAKDD*, pages 404–413, 2004.
[17] O. Kucuktunc and H. Ferhatosmanoglu. λ-diverse nearest neighbors browsing for multidimensional data. *TKDE*, pages 481–493, 2013.
[18] K. C. K. Lee, W.-C. Lee, and H. V. Leong. Nearest surrounder queries. In *ICDE*, pages 85–85, 2006.
[19] C. E. Noon and J. C. Bean. An efficient transformation of the generalized traveling salesman problem. *INFOR*, 31(1):39–44, 1993.
[20] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient olap operations in spatial data warehouses. In *SSTD*, pages 443–459. Springer, 2001.
[21] C. H. Papadimitriou and K. Steiglitz. Some complexity results for the traveling salesman problem. In *ACM TOC*, pages 1–9, 1976.
[22] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, et al. Semantic trajectories modeling and analysis. *ACM Computing Surveys (CSUR)*, page 42, 2013.
[23] J. Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley Longman Publishing Co., Inc., 1984.
[24] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *ACM SIGMOD*, pages 71–79, 1995.
[25] N. Roussopoulos, S. Kelley, and F. Vincent. Nearest neighbor queries. In *ACM sigmod record*, pages 71–79, 1995.
[26] X. Teng, J. Yang, J.-S. Kim, G. Trajcevski, A. Züfle, and M. A. Nascimento. Fine-grained diversification of proximity constrained queries on road networks. In *SSTD*, pages 51–60, 2019.
[27] H. Tong, C. Faloutsos, and J.-Y. Pan. Fast random walk with restart and its applications. In *ICDM*, pages 613–622, 2006.
[28] M. R. Vieira, H. L. Razente, M. C. Barioni, M. Hadjieleftheriou, D. Srivastava, C. Traina, and V. J. Tsotras. On query result diversification. In *ICDE*, pages 1163–1174. IEEE, 2011.
[29] C. Zhang et al. Diversified spatial keyword search on road networks. In *EDBT*, pages 367–378, 2014.
[30] K. Zheng, S. Shang, N. J. Yuan, and Y. Yang. Towards efficient search for activity trajectories. In *ICDE*, pages 230–241. IEEE, 2013.