Diversified Spatial Keyword Search On Road Networks

Chengyuan Zhang[†], Ying Zhang[‡,†], Wenjie Zhang[†],
(✉) Xuemin Lin[♯,†], Muhammad Aamir Cheema[§,†], Xiaoyang Wang[†]
[†] The University of New South Wales, [‡] QCIS, University of Technology, Sydney
[♯] Shanghai Key Laboratory of Trustworthy Computing, East China Normal University
[§] Clayton School of Information Technology, Monash University
{zhangc, yingz, zhangw, lxue, xiaoyangw}@cse.unsw.edu.au, aamir.cheema@monash.edu

ABSTRACT

With the increasing pervasiveness of the geo-positioning technologies, there is an enormous amount of *spatio-textual* objects available in many applications such as location based services and social networks. Consequently, various types of spatial keyword searches which explore both locations and textual descriptions of the objects have been intensively studied by the research communities and commercial organizations. In many important applications (e.g., location based services), the closeness of two spatial objects is measured by the road network distance. Moreover, the result diversification is becoming a common practice to enhance the quality of the search results. Motived by the above facts, in this paper we study the problem of diversified spatial keyword search on road networks which considers both the *relevance* and the *spatial diversity* of the results. An efficient signature-based inverted indexing technique is proposed to facilitate the spatial keyword query processing on road networks. Then we develop an efficient diversified spatial keyword search algorithm by taking advantage of spatial keyword pruning and diversity pruning techniques. Comprehensive experiments on real and synthetic data clearly demonstrate the efficiency of our methods.

1. INTRODUCTION

With advances in geo-positioning technologies, there is a rapidly growing amount of *spatio-textual* objects collected in many applications such as location based services and social networks, in which an object is described by its spatial location and a set of keywords (terms). For instance, in the local search services, an online business directory (e.g., yellow pages) provides the location information as well as short descriptions of the businesses (e.g., hotels, restaurants). Consequently, the study of spatial keyword search which explores both location and textual description of the objects has attracted great attention from the commercial organizations and research communities.

Motivation. Due to the massive amount of *spatio-textual* objects in many important applications, various spatial keyword query models, query processing techniques, and indexing mechanisms have emerged in recent years (surveyed in [6, 2]) such that users can effectively exploit both spatial and textual information of these *spatio-textual* objects. Most of the existing work focuses on Euclidean space. However, in practice, the road network distance (cost) is employed in many key applications (e.g., location based services) of the spatial keyword search. Therefore, it is critical to develop efficient indexing techniques and query algorithms to support spatial keyword search on road networks. In this paper, we focus on the boolean spatial keyword search on road networks which aims to find a set of objects each of which contains all query keywords and is close to the query in terms of road network distance (cost).

Moreover, it has been widely recognized [1] that the usefulness of a retrieved object depends not only on its relevance to the query (i.e., distance and keyword constraint) but also on other objects in the results. Intuitively, the retrieved objects should be dissimilar to each other (i.e., diversified) since in some scenarios it is less interesting for users to retrieve two highly similar objects at the same time. In the context of *spatio-textual* objects, it is reported in [20, 19] that users have strong preference on **spatially diversified** result; that is, the pair-wise distance (i.e., dissimilarity) between two objects in the result should be reasonably large. Nevertheless, to the best of our knowledge, there is no existing work on the diversified spatial keyword search on road networks.

Below is a motivating example in which both the relevance and the spatial diversity of the results are considered.

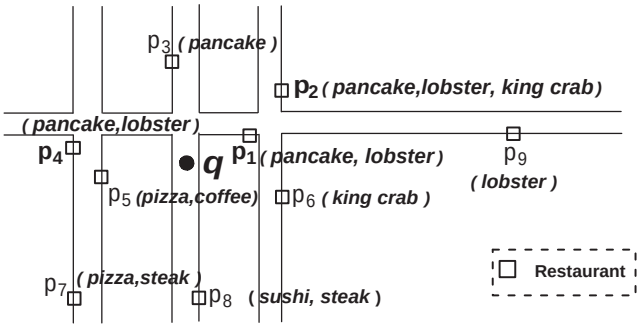


Figure 1: Online Yellow Pages Map

EXAMPLE 1 (**Motivation**). *In Fig. 1, there is a set of restaurants in Sydney CBD whose locations (represented by squares) and service lists (a set of keywords) are registered in the online yellow pages map of a local search service provider. Suppose a tourist with a GPS-enabled smartphone wants to: 1) have a nice dinner and then 2) visit nearby attractions or shops after the dinner. Assume she decides to enjoy the lobster and the pancake which are famous food in*

Sydney, but has no idea what kinds of attractions or shops to explore until a set of candidates near the restaurants are readily available. This implies that she expects to see a limited number (say $k = 2$) of restaurants, denoted by $\mathcal{S}$, which are close to her current location (e.g., the point q in Fig. 1), and each of which serves both lobster and pancake. Meanwhile, these k restaurants should be well spatially distributed on the area in the sense that a reasonable number of attractions or shops within walking distance to the restaurants can be considered for her post-dinner activity. Intuitively, although p_1 and p_2 in Fig. 1 are the two closest restaurants each of which serves both pancake and lobster, $\mathcal{S}_1 = \{p_1, p_2\}$ is not a good result because two restaurants are very close to each other, and the attractions or shops close to p_1 is highly likely to be reported by p_2 as well. In this scenario, $\mathcal{S}_2 = \{p_1, p_4\}$ might be a better choice since it provides more attractions or shops for consideration with a slight sacrifice in the relevance (i.e., closeness) compared with $\mathcal{S}_1$.

Motivated by the above example, in this paper we study the problem of diversified spatial keyword search on road networks. Given a set of *spatio-textual* objects in a road network, a spatial keyword query consists of a location and a set of query keywords, which aims to retrieve nearby objects each of which contains all query keywords. Moreover, we also consider the *spatial diversity* of the results. In order to meet a user's preference on the *relevance* and the *spatial diversity* of the results, we employ a bi-criteria objective function f which effectively combines both aspects. Specifically, we aim to find a set $\mathcal{S}$ of objects with $|\mathcal{S}| = k$ such that each object in $\mathcal{S}$ contains *all* query keywords and $f(\mathcal{S})$ is maximized. In this paper, we adopt the popular *max-sum diversification* function [12], which is formally defined in Section 2.1, based on the network distances of the objects to the query location (**relevance**) and the pairwise network distances among objects (**spatial diversity**) in $\mathcal{S}$.

Challenges. The challenges for the problem of diversified spatial keyword search on road networks are two-fold.

Firstly, effective indexing technique is required for diversified spatial keyword query on road networks due to the large size of road networks and *spatio-textual* objects in many applications. As shown in our empirical study, the performance of the algorithm is poor if we simply keep textual information with objects in the existing road networking indexing structure because a large number of irrelevant objects may be loaded before we apply the keyword based pruning. Similarly, since most of the existing s patial keyword indexing techniques are proposed in Euclidean space which are independent to the underlying road network structure, it is cost expensive to conduct the spatial keyword based pruning during the network expansion. We adopt the inverted indexing technique on the edges of the road network to significantly improve the performance of spatial keyword search since objects which do not contain *any* query keyword can be immediately excluded from computation due to the nature of the inverted indexing technique. Observe that the performance of the inverted indexing technique is affected by the number of *false hits* (i.e., the I/Os invoked by objects containing part of the query keywords), we propose the signature-based inverted indexing technique such that a small summary is built for each keyword and the number of *false hits* can be significantly reduced by exploiting the **AND** semantics. We further enhance the pruning capability of the signature technique by partitioning objects on the same edge.

Secondly, it is a challenge to develop novel technique to prune objects based on the spatial keyword constraints (i.e., distance based and keyword based constraints) and the objective function at the same time. As shown in our empirical study, it is cost expensive if we first retrieve all *spatio-textual* objects which satisfy spatial keyword constraints and then apply the existing diversification algorithm. This is because many non-promising objects may be loaded for the diversification computation, and the pairwise network distance computation on road networks is cost expensive. This motivates us to develop incremental diversified spatial keyword search algorithm so that the spatial keyword pruning and diversity pruning techniques can be seamlessly integrated and hence significantly reduce the overall cost.

Contributions. Our main contributions can be summarized as follows.

- We formally define the problem of diversified spatial keyword search on road networks.
- We develop an efficient signature-based inverted indexing technique as well as an efficient incremental network expansion algorithm for spatial keyword search on road networks. We further propose a partition based method to enhance the effectiveness of the signature technique.
- An effective incremental diversified spatial keyword search algorithm is proposed based on the spatial keyword pruning and diversity based pruning techniques.
- Comprehensive experiments on real and synthetic datasets demonstrate the effectiveness and efficiency of our methods.

Roadmap. The rest of the paper is organized as follows. Section 2 formally defines the problem of diversified spatial keyword search on road network, followed by some preliminary work. Section 3 introduces the signature-based inverted indexing technique as well as efficient spatial keyword search algorithm on road network. An incremental diversified spatial keyword search algorithm is developed in Section 4. Experimental results are reported in Section 5. Section 6 introduces related work, and Section 7 concludes the paper.

2. PRELIMINARY

In this section, we first formally define the problem of diversified spatial keyword search on road networks in Section 2.1. Section 2.2 introduces the disk-based road network data structure, and Section 2.3 presents a general greedy based diversification algorithm. Table 1 summarizes the mathematical notations used throughout this paper.

Notation	Definition
o (q)	a *spatio-textual* object (query)
$\mathcal{G}$	a road network
$q.T$	a set of query keywords for query q
$\mathcal{V}$, $\|\mathcal{V}\|$	vocabulary, size of the vocabulary
n	a node in the road network
e, (n_1, n_2)	an edge, an edge with two end-nodes n_1 and n_2
m	the number objects on an edge
$\delta(o_1, o_2)$	the *network distance* (cost) between o_1 and o_2
δ_T	*network distance* threshold in network expansion
δ_{max}	maximal *network distance* in the search
$\theta(o_1, o_2)$	*diversification distance* between o_1 and o_2
θ_T	*diversification distance* threshold
$\mathcal{CP}(\mathcal{CO})$	core pairs (objects)

Table 1: The summary of notations.

2.1 Problem Definition

Road Networks. In this paper, a road network is modeled as a weighted graph $\mathcal{G} = (N, E, W)$ where a road node $n \in N$ represents a road intersection, an edge $e = (n_1, n_2) \in E$ corresponds to a road segment which connects two road nodes n_1 and n_2, and a non-negative weight $w(n_1, n_2) \in W$ stands for the cost (e.g., distance or travel time) associated with the road segment. We assume each edge is bi-directional. Whenever there is no ambiguity, we assume the end-node n_1 has smaller id than n_2 where n_1 is called the *reference node* of the edge. Let p be a spatial point lying on the edge (n_1, n_2), we assume the cost from the node n_1 to p, denoted by $w(n_1, p)$, is proportional to the distance between them[1]. Clearly, we have $w(n_1, p) + w(p, n_2) = w(n_1, n_2)$ for an edge (n_1, n_2) and a point p on the edge. Then for two given points u and v in the road network, we use $\delta(u, v)$ to represent the *network distance (cost)* between u and v which is the sum of the edge weights along the least costly path from u to v. Note that the least costly path corresponds to the shortest path if the weight represents the distance of an edge. It is immediate that we have $\delta(u, v) = \delta(v, u)$. Given a point p lying on the edge (n_1, n_2), following equation shows how to derive network distance between p and a query point q when q does not lie on (n_1, n_2).

$$\delta(q, p) = min(\delta(q, n_1) + w(n_1, p)\ ,\ \delta(q, n_2) + w(n_2, p)) \quad (1)$$

Note that we have $\delta(q, p) = w(q, p)$ if both q and p lie on the same edge.

Spatio-textual object. A *spatio-textual* object o is described by a spatial point in a 2-dimensional space and a set of keywords (terms) from a vocabulary $\mathcal{V}$, denoted by $o.loc$ and $o.T$ respectively. For presentation simplicity, we assume objects lie along the edges (i.e., road segments) of a road network $\mathcal{G}$. In this paper hereafter, whenever there is no ambiguity, "*spatio-textual* object" is abbreviated to "object" and o (q) is used to represent its location $o.loc$ ($q.loc$).

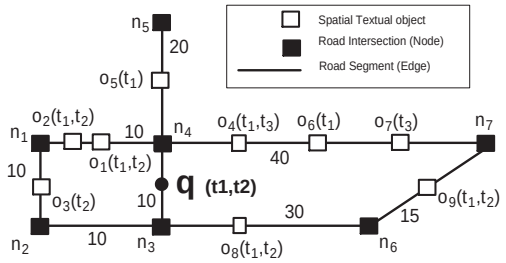


Figure 2: Example of Diversified SK Search

EXAMPLE 2. *Fig. 2 illustrates an example of the road network $\mathcal{G}$ and a set of spatio-textual objects $\mathcal{O}$. There are 7 nodes, 8 edges and 9 spatio-textual objects where the keywords of each object are listed (e.g., object o_1 contains keywords t_1 and t_2) where the vocabulary $\mathcal{V}$ is $\{t_1, t_2, t_3\}$. We also label the distance (length) of each edge (e.g., $d(n_1, n_4) = 10$). For presentation simplicity, we use the distance as the weight of each edge, i.e., $w(n_1, n_4) = d(n_1, n_4) = 10$. In the example, we have $\delta(q, o_1) = 10$, $\delta(q, o_2) = 12$, $\delta(q, o_8) = 15$, $\delta(o_1, o_2) = 2$, $\delta(o_1, o_8) = 25$, and $\delta(o_2, o_8) = 27$.*

SK Query on Road Networks. Below is a formal definition of the spatial keyword (SK) query on road networks.

[1]we have $w(n_1, p) = w(n_1, n_2) \times \frac{d(n_1, p)}{d(n_1, n_2)}$ where $d(n_1, n_2)$ represents the distance (length) between two end-node n_1 and n_2, and $d(n_1, p)$ corresponds to the distance (length) between node n_1 and the point p along the edge (n_1, n_2).

DEFINITION 1 (SK QUERY). *Given a road network $\mathcal{G}$, a set $\mathcal{O}$ of spatio-textual objects, a query point q which is also a spatio-textual object, and a network distace δ_{max}, a spatial keyword query retrieves objects each of which contains **all** query keywords of q and is within network distance δ_{max} from q; that is, we retrieve objects $o \in \mathcal{O}$ with $\delta(o, q) \leq \delta_{max}$ and $q.T \subseteq o.T$. We use $SK(\mathcal{O}, q, \delta_{max})$ to record the objects retrieved by the above SK query.*

For presentation simplicity, we use *spatial keyword constraint* to denote the distance constraint and keyword constraint. We say that an object satisfies the spatial keyword constraint if it is within network distance of δ_{max} from q and contains all query keywords.

EXAMPLE 3. *In Fig. 2, given a query q with $q.T = \{t_1, t_2\}$ and $\delta_{max} = 20$, we have $SK(\mathcal{O}, q, \delta_{max}) = \{o_1, o_2, o_8\}$. Note that, although o_9 contains both t_1 and t_2, it is excluded because of the network distance constraint. Meanwhile, objects o_3, o_4 and o_5 are eliminated due to the query keyword constraint although they are within network distance 20 from q.*

Bi-criteria objective function (f). Given a subset $\mathcal{S}$ of objects with $|\mathcal{S}| = k$, we use the popular *max-sum diversification* function [12] as the bi-criteria objective function, denoted by f, where the **relevance** of $\mathcal{S}$ ($Rel(\mathcal{S})$) is measured by the *network distances* of the objects to the query and the **diversity** of $\mathcal{S}$ ($Div(\mathcal{S})$) is captured by their pairwise *network distances*.

$$\begin{aligned} f(S) &= \lambda \times Rel(\mathcal{S}) + (1 - \lambda) \times Div(\mathcal{S}) \quad (2) \\ &= \frac{\lambda}{k} \sum_{u \in S} (1 - \frac{\delta(u, q)}{\delta_{max}}) + \frac{1 - \lambda}{k(k-1)\delta_{max}} \sum_{u, v \in S} \delta(u, v) \end{aligned}$$

Here, λ ($0 \leq \lambda \leq 1$) is a parameter specifying the trade-off between the relevance and the diversity. We assume the larger values are preferred in this paper. So the relevance score of an object u, denoted by $Rel(u)$, is measured by $1 - \frac{\delta(u,q)}{\delta_{max}}$, and $Rel(S) = \sum_{u \in S} Rel(u)$. Observe that there are k objects and $\frac{k(k-1)}{2}$ pairs of objects in $\mathcal{S}$, together with the fact that $\delta(u, q) \leq \delta_{max}$ and $\delta(u, v) \leq 2\delta_{max}$ for objects in the results of SK query, we have $0 \leq Rel(\mathcal{S}) \leq 1$ and $0 \leq Div(\mathcal{S}) \leq 1$ where the larger $Rel(\mathcal{S})$ ($Div(\mathcal{S})$) value represents the higher relevance (diversity) of $\mathcal{S}$. As shown in [13], the results are usually expected to be diverse enough without sacrificing relevance for $0.5 \leq \lambda \leq 0.9$.

Problem Statement. In this paper, we investigate the problem of diversified spatial keyword search on road networks. Given a road network $\mathcal{G}$, a set $\mathcal{O}$ of *spatio-textual* objects, a query object q, a distance δ_{max}, a bi-criteria objective function f, and a natural number k we aim to find a set of objects $\mathcal{S} \subseteq SK(\mathcal{O}, q, \delta_{max})$ such that $|\mathcal{S}| = k$ and $f(\mathcal{S})$ is maximized. In this paper, ties are broken arbitrarily.

EXAMPLE 4. *In Fig. 2, given a diversified SK query q with $q.T = \{t_1, t_2\}$, $\delta_{max} = 20$, $k = 2$ and $\lambda = 0.6$, objects $\{o_1, o_8\}$ will be retrieved. More specifically, since there are three objects satisfying spatial keyword constraint in Example 3, we aim to choose a set $\mathcal{S}$ with $|\mathcal{S}| = 2$ such that $f(\mathcal{S})$ is maximized. There are three possible solutions where $\mathcal{S}_1 = \{o_1, o_2\}$, $\mathcal{S}_2 = \{o_1, o_8\}$ and $\mathcal{S}_3 = \{o_2, o_8\}$. When $\lambda = 0.6$, we have $f(\mathcal{S}_1) = 0.29$, $f(\mathcal{S}_2) = 0.475$, and $f(\mathcal{S}_3) = 0.465$. Therefore, we have $\mathcal{S} = \{o_1, o_8\}$. $\mathcal{S}$ turns to $\{o_1, o_2\}$ when $\lambda = 0.9$, which puts a high priority to the network distances of the objects from q.*

2.2 Disk-based Road Network Presentation

We adopt the popular connectivity-clustered access method (CCAM) [18] to represent the road network $\mathcal{G}$ which effectively organizes the adjacent lists of the road nodes so that we can take advantage of the access locality and reduce the I/O costs during the query processing on road networks. Nodes of the road network are sorted by their Z-ordering according to the spatial locations. Moreover, in [18] the network is partitioned into groups by recursively applying a two-way-partition method until the nodes' adjacent lists of each group can be fit into one page. For an adjacent list of a node n_i, the information of edges are stored including end-nodes, distance, and weight. We also build a network R-tree [16] to organize the minimal bounding rectangles of the edges of the network. For a given object, we may identify its corresponding edges by utilizing the MBRs of the edges in a branch-and-bound fashion.

2.3 Greedy Algorithm for Diversification

We define a *diversification distance* between two objects u and v, denoted by $\theta(u, v)$, against the road network $\mathcal{G}$ as follows.

$$\begin{aligned}\theta(u,v) &= \lambda(2 - \frac{\delta(u,q)}{\delta_{max}} - \frac{\delta(v,q)}{\delta_{max}}) \\ &\quad + \frac{1-\lambda}{\delta_{max}}\delta(u,v) \end{aligned} \quad (3)$$

where $\theta(u, v)$ records the *relevance* and the *diversity* for a pair of objects u and v in $\mathcal{S}$. Since we have

$$\begin{aligned}\frac{\lambda}{k}\sum_{u\in\mathcal{S}}(1 - \frac{\delta(u,q)}{\delta_{max}}) &= \frac{\lambda}{k(k-1)}(k-1)\sum_{u\in\mathcal{S}}(1 - \frac{\delta(u,q)}{\delta_{max}}) \\ &= \frac{\lambda}{k(k-1)}\sum_{u,v\in\mathcal{S}}(1 - \frac{\delta(u,q)}{\delta_{max}} + 1 - \frac{\delta(v,q)}{\delta_{max}})\end{aligned}$$

Then the bi-criteria objective function f in Equation 2 can be rewritten as follows.

$$f(S) = \frac{1}{k(k-1)}\sum_{u,v\in\mathcal{S}}\theta(u,v) \quad (4)$$

where $f(S)$ is the average pairwise *diversification distances* of the objects in $\mathcal{S}$.

Algorithm 1: **Diversified SK query ($\mathcal{P}$, k)**

Input : $\mathcal{P}$: a set of objects satisfying spatial keyword constraint
Output : $\mathcal{S}$: a set of diversified objects with size k

1 $\mathcal{S} := \emptyset$;
2 **for** $i := 1$ to $\lfloor \frac{k}{2} \rfloor$ **do**
3 Find pair $u, v \in \mathcal{P}$ with **argmax**$_{u,v\in\mathcal{P}}$ $\theta(u, v)$;
4 $\mathcal{S} := \mathcal{S} \cup \{u, v\}$; $\mathcal{P} := \mathcal{P} \setminus \{u, v\}$;
5 add an arbitrary object $u \in \mathcal{P}$ to $\mathcal{S}$ **If** k is odd ;
6 **return** $\mathcal{S}$

As shown in [12], the problem of finding maximal $f(\mathcal{S})$ in Equation 4 is NP-hard and there is a greedy algorithm which provides 2-approximation of the optimal solution. Algorithm 1 illustrates the detail of the greedy algorithm where we assume the objects retrieved by the SK query and their pairwise *diversification distances* are readily available. In each iteration (Line 2-4), a pair of objects u and v with the longest *diversification distance* will be chosen, and they will not be considered in the following computation. Note that we randomly pick one more object from the remaining objects if k is odd.

3. SK SEARCH ON ROAD NETWORKS

To support the diversified SK search, we need to develop efficient algorithm to retrieve objects which satisfy the spatial keyword constraint (i.e., keyword constraint and *network distance* constraint). Section 3.1 proposes a signature-based inverted index structure to organize objects such that many non-promising objects can be pruned at a cheap cost. Section 3.2 presents an efficient SK search algorithm based on the network expansion. Section 3.3 improves the effectiveness of our indexing technique by partitioning objects on the same edge.

3.1 Signature-Based Inverted Index

A large number of irrelevant objects may be loaded if we simply store objects together with their corresponding edges in the CCAM structure introduced in Section 2.2. Therefore, in addition to the CCAM structure which effectively captures the topology of the road network for the network expansion, it is desirable to utilize other indexing structure to organize the *spatio-textual* objects. In this paper, we adopt the popular inverted indexing technique to organize objects. Specifically, for each keyword t, objects containing t are kept with their corresponding edges which are maintained by a B^+ tree where the key of an edge is the Z-ordering code of its center point. We also keep the offset distance of the object to the reference end-node of the edge. Recall that we can derive $w(n_1, o)$ and $w(n_2, o)$ for an object o based on $d(n_1, n_2)$, $d(n_1, o)$ and $w(n_1, n_2)$ if o lies on the edge (n_1, n_2).

A nice property of the inverted indexing structure is that, for a given query q, only the objects containing at least one query keyword will be involved in the search. Nevertheless, many objects which do not contain **all** query keywords may also be loaded. This may seriously deteriorate the performance especially when the number of the query keywords is not small. Therefore, we further improve the I/O efficiency by building signatures of the edges and then exploit the **and** semantics of the keyword constraint. Intuitively, we can use $I(e, t)$ to represent the signature of an edge e where $I(e, t) = 1$ indicates that there is at least one object with keyword t lying on edge e, and $I(e, t) = 0$ otherwise. Clearly, we don't need to explore an edge e if there is a query keyword t with $I(e, t) = 0$, and hence reduce the I/O cost because the size of the signatures of the edges are usually much smaller than the inverted index file and can be easily fit into the main memory. For presentation simplicity, we use a bitmap to represent the signature of a keyword where a bit is used to keep the signature of an edge in the road network $\mathcal{G}$. In our implementation, we do not build signature for a keyword t if its inverted file can be fit into one data page. Moreover, we recursively divide the edges by KD-tree partion method based on the center points of the edges, and each leaf node corresponds to the signature of an edge. Then the signature size of a keyword can be significantly reduced by compacting the tree node if all of its descendant nodes share the same signature value.

Given a set T of query keywords and an edge e, Algorithm 2 illustrates how to utilize the signature-based inverted index to load objects which lie on the edge e and each of which contains all keywords in T. Clearly, none of the objects will be retrieved if we have $I(e, t) = 0$ for any keyword $t \in T$ (Line 1-3). Let $\mathcal{R}$ denote the objects satisfying the

Algorithm 2: **LoadObjects(** e, T**)**

```
   Input  : e : an edge, T : query keywords
   Output : R : objects satisfying query keyword constraint
1  for each query keyword t ∈ T do
2      if I(e,t) = 0 then
3          return ∅
4  R_i ← objects lying on e with t_i ∈ T;
5  return ⋂_{i=1}^{|T|} R_i
```

query keyword constraint, we have $\mathcal{R} = \bigcap_{i=1}^{|T|} \mathcal{R}_i$ where $\mathcal{R}_i$ represents the objects containing the i-th keyword, denoted by t_i, of T. The dominant cost of Algorithm 2 is the loading of the objects from the inverted files, our performance analysis and empirical evaluation demonstrate that the signature technique can significantly reduce the I/O costs.

3.2 SK Search Algorithm

Since we aim to support the general road network in which various cost models (e.g., distance and travel time) may be used, we adapt the incremental network expansion (INE) algorithm in [16] to incrementally access objects because INE algorithm does not rely on specific restrictions (e.g., Euclidean distance restriction) or pre-computation (e.g., voronoi diagram and shortcut) of the road networks. Observe that the *network distance* is calculated from scratch for each individual object encountered in [16], we integrate the Dijkstra's algorithm [8] with the INE so that *network distances* of the objects are calculated in an accumulative way during the network expansion.

The spatial keyword based pruning is applied in the sense that only objects within the search region are accessed during the network expansion (i.e., *network distance* constraint) and many non-promising objects are pruned by taking advantage of the signature-based inverted indexing technique (i.e., keyword constraint).

Algorithm 3 illustrates the implementation details of the spatial keyword search on road networks. For presentation simplicity, we assume the query point q starts from a node n in Line 2. A min-priority-queue $\mathcal{Q}$ is employed to keep nodes accessed during the expansion where the key of a node n is denoted by $\delta(n)$ where $\delta(n) = \infty$ if the node n has not been visited. In this paper, we have $\delta(n) = \delta(q, n)$ if a node n is marked (Line 2 and 7). Similarly, we use $\delta(o)$ to compute $\delta(q, o)$, and we have $\delta(o) = \delta(q, o)$ if an object o is marked (Line 22 and 25). In Algorithm 3, nodes are accessed in non-decreasing order of their *network distances* from q. Line 6 updates δ_T which is the lower bound of the *network distance* for any unmarked node. The expansion terminates when $\delta_T > \delta_{max}$, which implies that $\delta(q, n_x) > \delta_{max}$ for any unmarked node n_x. For each node n_i in the adjacent list of the node n, Line 10 updates $\delta(n_i)$ if n_i is not marked, and the objects on edge (n, n_i), which satisfy the keyword constraint, will be loaded if n_i is visited at the first time (Line 12), followed by the computation of *network distance* based on $\delta(n)$ (Line 15). If the node n_i is already marked, Line 18-22 may update the *network distance* of the objects and set them marked since both end-nodes (n and n_i) of the edge are marked. At the end of Algorithm 3, objects with *network distance* longer than δ_{max} are pruned from $\mathcal{R}$.

Correctness. The correctness of the *network distance* computation is immediate for the nodes as the computation follows the Dijkstra's algorithm [8]. Suppose o is an object lying on edge (n_1, n_2), according to Equation 1, we can come up with correct $\delta(q, o)$ if n_1 and n_2 are marked. Due

Algorithm 3: **SK Search(**q, δ_{max}**)**

```
   Input  : q : query object, δ_max : maximal network
            distance
   Output : R : objects satisfying SK query condition
1  R := ∅; Q = ∅; δ_T := 0 ;
2  n ← the node where q is located, and n is marked;
3  Push n into Q with δ(n) = 0 ;
4  while Q ≠ ∅ do
5      n ← Q.dequeue() ;
6      δ_T := δ(n); Terminate While Loop if δ_T > δ_max ;
7      Mark the node n;
8      for each unmarked node n_i in adjacent list of n do
9          if δ(n_i) > δ(n) + w(n, n_i) then
10             δ(n_i) := δ(n) + w(n, n_i) ;
11         if n_i is unvisited then
12             F := LoadObjects( (n, n_i) , q.T );
13             if F ≠ ∅ then
14                 for each object o ∈ F do
15                     d(o) := d(n) + w(n, o);
16                 R := R ∪ F;
17             Push n_i into Q;
18     for each marked node n_i in adjacent list of n do
19         for each object o lying on edge (n_i, n) do
20             if δ(o) > δ(n) + w(n, o) then
21                 δ(o) := δ(n) + w(n, o);
22             Mark the object o;
23 for each object o ∈ R do
24     R := R \ o if δ(o) > δ_max;
25     Mark o ∈ R if o is unmarked ;
26 return R
```

to the monotonic property of δ_T in Algorithm 3, we have $\delta(q, o) > \delta_{max}$ if n_1 and n_2 are not marked since $\delta(q, o) \geq min(\delta(q, n_1), \delta(q, n_2))$. In the case only one node (say n_1) is marked, i.e., $\delta(o) = \delta(n_1) + w(n_1, o)$, we have $\delta(q, o) = \delta(o)$ if $\delta(o) \leq \delta_T$ since $\delta(q, n_2) \geq \delta_T$. Consequently, all objects satisfying the keyword constraint and *network distance* constraint are correctly retrieved in Algorithm 3.

Performance Analysis. The main cost of Algorithm 3 consists of two parts: road network traverse ($C_{\mathcal{G}}$) and the loading of objects ($C_{\mathcal{O}}$). Let l_n and l_e denote the number of nodes and edges accessed during the network expansion, then we have $C_{\mathcal{G}} = l_n \log(l_n) + l_e + I_n$ where I_n is the I/O latency to load the adjacent lists which is $l_n \times\ t_{io}$ in the worst case assuming the delay of each I/O is t_{io}. Similarly, we have $C_{\mathcal{O}} = \rho \times l_e \times |q.T| \times t_{io}$ in the worst case where ρ is the percentage of edges which load objects from the inverted files. Note that we have $\rho = 1$ if the signature technique is not employed. As demonstrated in the empirical study, $C_{\mathcal{O}}$ is the dominant cost for Algorithm 3 and hence it is critical to minimize ρ by utilizing signature technique. Below, we analyze the expected number of objects loaded regarding different object indexing techniques discussed in Section 3.1.

Assume there are on average m objects lying on each edge and s keywords for each object which are randomly chosen from the vocabulary $\mathcal{V}$. Let C_1, C_2 and C_3 denote the number of objects loaded in Algorithm 3 when the objects are organized by CCAM structure, inverted index and signature-based inverted index respectively. It is immediate that $C_1 = l_e \times m$ since we need to load all objects lying on each edge for the keyword constraint test. Recall that l_e represents the number of edges accessed at Line 12 of Algorithm 3. As the expected number of objects lying on an edge with the keyword t is $\frac{m \times s}{|\mathcal{V}|}$, we have $C_2 = l_e \times \frac{m \times s}{|\mathcal{V}|} \times |q.T|$. With probability $p_s = 1 - (1 - \frac{s}{|\mathcal{V}|})^m$, the signature

of the edge is set to 1 for each keyword. Therefore, an edge will pass the signature test with probability $p_s^{|q.T|}$. Then we have $C_3 = l_e \times p_s^{|q.T|} \times \frac{m \times s}{|\mathcal{V}|} \times |q.T|$. According to the above analysis, the signature-based inverted indexing technique is expected to achieve better performance compared with other two alternatives.

3.3 Enhancement of the Signature Technique

In this subsection, we enhance the effectiveness of the signature technique by partitioning the objects on the same edge. We first introduce the motivation of the method, then a dynamic programming based method is proposed.

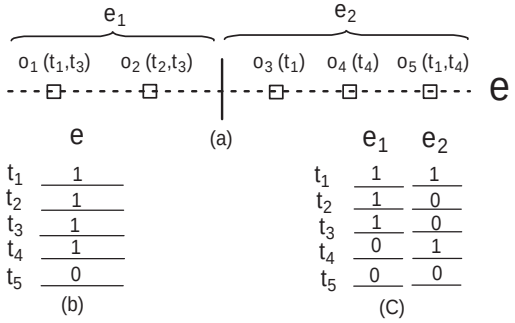


Figure 3: Example of Edge Partition

Motivation. As shown in Fig. 3(a), suppose we have five objects $o_1(t_1, t_3)$, $o_2(t_2, t_3)$, $o_3(t_1)$, $o_4(t_4)$, and o_5 (t_1, t_4) lying on an edge e and the vocabulary $\mathcal{V} = \{t_1, t_2, t_3, t_4, t_5\}$. The signatures of five keywords are shown in Fig. 3(b) where $I(e, t_1) = 1$, $I(e, t_2) = 1$, $I(e, t_3) = 1$, $I(e, t_4) = 1$, and $I(e, t_5) = 0$ respectively. Given a query q with $q.T = \{t_2, t_4\}$, all objects will be loaded if e is accessed in Algorithm 3 since $I(e, t_2) = 1$ and $I(e, t_4) = 1$. However, none of the objects contains both t_2 and t_4. We say this is a *false hit* if an edge passes the signature test of a query but does not return any object satisfying the keyword constraint. Similarly, a query q with $q.T = \{t_3, t_4\}$ may result in a *false hit* as well. Note that it is a *true hit* regarding $q.T = \{t_1, t_3\}$ since the object o_1 contains both t_1 and t_3. And $q.T = \{t_1, t_5\}$ is also not a *false hit* since it fails the signature test. Intuitively, we can partition the edge e into two virtual edges e_1 and e_2 as shown in Fig. 3(a), and the signature of e can be refined as shown in Fig. 3(c). Then we can avoid loading objects resulted from the false hit when $q.T = \{t_2, t_4\}$ since it fails the signature tests for both e_1 and e_2.

In this paper, a partition P consists of a set of *virtual edges* resulting from a number of cuts against an edge and each virtual edge covers a set of objects along the edge. In Fig. 3(a), we have $P = \{e_1, e_2\}$. There are $\binom{m-1}{b}$ possible partitions if there are m objects on edge e and $b+1$ virtual edges (i.e., b cuts). In this subsection, we propose a dynamic programming based technique to partition objects lying on an edge for a given number of cuts allowed such that the number of objects loaded due to *false hits* can be minimized. Specifically, for a given SK query q, $\xi(q, e)$ denotes the *false hit cost* of e (i.e., the number of objects loaded due to the *false hits* of the edge e). Similarly, we can define the *false hit cost* of a partition P, denoted by $\xi(q, P)$, where

$$\xi(q, P) = \sum_{e' \in P} \xi(q, e'). \quad (5)$$

Note that $\xi(q, e) = 0$ if i) e is not visited, ii) e fails the signature test, or iii) it is a true hit (i.e., find an object satisfying the keyword constraint). Regarding the example in Fig. 3, assume that $\mathcal{Q} = \{q_1, q_2, q_3\}$ where $q_1.T = \{t_1, t_3\}$, $q_2.T = \{t_2, t_4\}$ and $q_3.T = \{t_1, t_2\}$, we have $\xi(q_1, e) = 0$, $\xi(q_2, e) = 5$, and $\xi(q_3, e) = 5$. Similarly we have $\xi(q_1, P) = \xi(q_1, e_1) + \xi(q_1, e_2) = 0$, $\xi(q_2, P) = 0$, and $\xi(q_3, P) = 2 + 0 = 2$. Let $\xi(\mathcal{Q}, P)$ denote the *false hit cost* of the partition P and $Pr(q)$ denote the probability that a query $q \in \mathcal{Q}$ is issued, we have

$$\xi(\mathcal{Q}, P) = \sum_{q \in Q} \xi(q, P) \times Pr(q) \quad (6)$$

Algorithm. Suppose an edge e contains m objects which are indexed by their visiting order along the edge (as shown in Fig. 3(a)). Given a number c, we aim to find a partition of e with c cuts which has the minimal *false hit cost*. The key idea is to use the dynamic programming technique to find optimal solution based on the local optimal solutions with less number of cuts. Specifically, we use $\mathcal{P}(i, j, c)$ to denote all possible partitions each of which divide objects between o_i and o_j (inclusive) into $c+1$ virtual edges. Let $P^*(i, j, c)$ denote a partition in $\mathcal{P}(i, j, c)$ with the minimal cost $P_s^*(i, j, c)$. Note that we use P_s to denote the cost of a partition P. Initially, we have

$$P_s^*(i, j, 0) = \xi(\mathcal{Q}, P(i, j, 0)) \quad (7)$$

where $P(i, j, 0)$ is the partition with one virtual edge (i.e., $c = 0$) which contains all objects between o_i to o_j.

The key observation is that, in order to compute $P^*(i, j, c)$, we assume that one of the cut is located at the k-th object with $i \leq k < j$. Let $\mathcal{Q}(i, j, k, c)$ denote the partitions in $\mathcal{P}(i, j, c)$ where one of the cuts is exactly located at the k-th object. We use $Q^*(i, j, k, c)$ to represent the partition in $\mathcal{Q}(i, j, k, c)$ with the lowest cost, denoted by $Q_s^*(i, j, k, c)$. Then, we can come up with $Q^*(i, j, k, c)$ by enumerating all possible combinations of the partitions regarding two sides of the k-th object. Specifically we have

$$Q_s^*(i, j, k, c) = \min_{0 \leq v \leq c-1} \{P_s^*(i, k, v) + P_s^*(k+1, j, c-v-1)\} \quad (8)$$

Note that we simply set $P_s^*(i, j, c) = \infty$ if there are no enough cutting positions, i.e., $j - i < c$.

By exhausting all possible fixed cutting postions, we have

$$P_s^*(i, j, c) = \min_{i \leq k \leq j-1} \{Q_s^*(i, j, k, c)\} \quad (9)$$

Algorithm 4 illustrates the outline of the dynamic programming based technique to identify a partition with c cuts such that its *false hit cost* is minimized. Line 1-2 compute the cost for each possible simple partition (i.e., partition with one virtual edge) according to Equation 7. Line 3-5 iteratively compute the optimal partitions with k cuts $(1 \leq k < c)$ for all possible partitions based on Equation 8 and 9. Then we come up with the final solution by identifying $P^*(1, m, c)$ based on the above intermediate solutions (Line 6).

Lines 3-5 contribute the dominant cost of Algorithm 4, which is $O(c^2 m^3)$. This is cost-prohibitive in practice, and hence we resort to the greedy heuristic in our implementation. Specifically, starting from the whole edge, i.e., a 0-cut partition which covers all objects, at each iteration, we find a cutting position j $(1 \leq j \leq m-1)$ such that the cost of the refined partition is minimized. In each iteration, it takes $O(m \times s_t)$ time to set up the signatures where s_t denotes the

Algorithm 4: **Partition**(e, c)

Input : e : edge for partition, c : the number of cuts
Output : $P^*(e)$: the optimal partition of e
1 **for** $1 \leq i \leq j \leq m$ **do**
2 Compute $P_s^*(i, j, 0)$ based on Equation 7;
3 **for** $k = 1$ to $c - 1$ **do**
4 **for** $1 \leq i < j \leq m$ **do**
5 Compute $P_s^*(i, j, k)$ based on Equation 8 and 9 ;
6 Compute $P_s^*(1, m, c)$ based on Equation 9 ;
7 **return** $P^*(1, m, c)$

average number of keywords appeared on the edge e. And the partition cost calculation takes $O(m \times |\mathcal{Q}| \times q_t)$ where q_t is the average number of the query keywords. Therefore, the total cost of the greedy algorithm is $O(c \times m \times (s_t + |\mathcal{Q}| \times q_t))$ where c is the number of cuts (partitions).

REMARK 1. *When the query log $\mathcal{Q}$ is not available, we can generate it on the fly based on the assumption that the high frequent keyword is more likely to appear as a query keyword.*

4. DIVERSIFIED SK SEARCH

In this section, we present a diversified SK search on road networks. Section 4.1 introduces the motivation of our diversified SK search algorithm. Section 4.2 presents an efficient algorithm to continuously maintain a *diversification distance* threshold for the pruning purpose. Section 4.3 develops an incremental diversified SK search algorithm.

4.1 Motivation

A straightforward implementation of the diversified SK query is to first retrieve a set $\mathcal{R}$ of objects based on Algorithm 3, i.e., objects satisfying spatial keyword constraint, and then feed them to Algorithm 1 (Section 2.3). Since we do not rely on pre-computation of the road network or some specific restrictions, it may be expensive to compute pair-wise *diversification distances* for objects in $\mathcal{R}$ when $|\mathcal{R}|$ is large. Similarly, the I/O cost of loading objects grows against $|\mathcal{R}|$ in Algorithm 3. This motivates us to develop effective pruning technique so that a significant number of objects may be pruned from $\mathcal{R}$ at a cheap cost.

For presentation simplicity, we assume k is an **even** number in this section. Let $\mathcal{CP}(\mathcal{R})$ denote the *core pairs* of the objects in $\mathcal{R}$ which are the $\frac{k}{2}$ pairs of objects chosen in Algorithm 1; that is, the top $\frac{k}{2}$ pairs of objects with the longest *diversification distances* where each object can contribute to at most one pair. The objects involved, denoted by $\mathcal{CO}$, are *core objects* which correspond to the k diversified objects in the candidate objects. In this paper, we aim to incrementally process the objects so that some objects can be safely pruned if they have no chance to be chosen as *core objects*. Based on the fact that any unvisited object in Algorithm 3 is not within *network distance* δ_T, we can employ a min-priority queue to maintain objects marked and output them in a non-decreasing order of their *network distances* from q.

We use θ_T to record the shortest *diversification distance* in $\mathcal{CP}$ for objects seen so far. It is shown in Section 4.2 that θ_T grows monotonically. This implies that we can safely prune an object o from the diversified search if there is no another object $o' \in \mathcal{R}$ so that $\theta(o, o') \geq \theta_T$. Another important issue is how to efficiently maintain θ_T as it requires us to incrementally compute *core pairs* for objects seen so far. It is expensive to re-compute $\mathcal{CP}$ from the scratch (i.e., invoke Algorithm 1) against the arrival of each new object. Section 4.2 presents an efficient algorithm to incrementally maintain θ_T. In Section 4.3, we propose an efficient incremental diversified SK search algorithm based on the pruning technique.

4.2 Incrementally Maintain the Threshold θ_T

In this subsection, we introduce how to incrementally maintain θ_T. Below, we first introduce some notations and a lemma.

For presentation simplicity, we assume the *diversification distances* are distinct values and *core pairs* in $\mathcal{CP}(\mathcal{R})$ are decreasingly ordered by their *diversification distances*, where $\rho_i(\mathcal{R})$ denotes the objects in the i-th *core pair*. Regarding the example of Fig. 4, we have $\rho_1(\mathcal{R}) = \{o_1, o_2\}$ and $\rho_2(\mathcal{R}) = \{o_3, o_4\}$. We say a new arriving object o is *dominated* by a *core object* $o' \in \mathcal{CO}(\mathcal{R})$ if $\theta(o, o') < \theta(o', o_y)$ where (o', o_y) is a *core pair*. For instance, in Fig. 4 o_4 dominates o if $\theta(o, o_4) < \theta(o_3, o_4)$. We use $\mathcal{R}_i$ to denote the set of first i objects arrived. Following lemma implies that we do not need to consider a pair (o, o') for the update of *core pairs* when o arrives if o is *dominated* by o'.

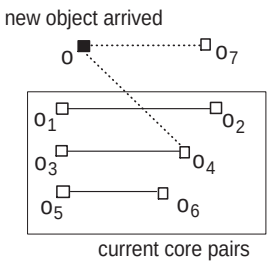


Figure 4: Update $\mathcal{CP}$

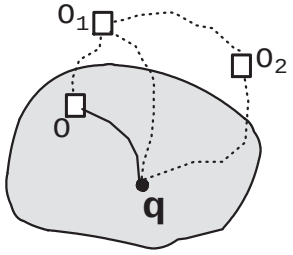


Figure 5: Example of Pruning

LEMMA 1. *Let o be the $(i+1)$-th arrived object, (o, o') cannot become a core pair if o is dominated by a core object $o' \in \mathcal{CO}(\mathcal{R}_i)$.*

PROOF. The proof is by contradiction. Suppose (o', o_y) is on the j-th *core pair* of $\mathcal{R}_i$, i.e., $o' \in \rho_j(\mathcal{R}_i)$, we have $\rho_y(\mathcal{R}_i) = \rho_y(\mathcal{R}_{i+1})$ for any $1 \leq y < j$ if (o, o') becomes a *core pair* in $\mathcal{R}_{i+1}$. This is because the distance of the y-th pair is longer than $\theta(o, o')$ and o cannot contribute to the y-th pair for any $1 \leq y < j$. Then the object o_y is not chosen for the first $j-1$ *core pairs*, this is conflict with the fact that (o, o') is a *core pairs* of $\mathcal{R}_{i+1}$ while $\theta(o, o') < \theta(o', o_y)$. □

Algorithm 5 illustrates how to efficiently update *core pairs* against the arrival of a new object, where $\mathcal{CP}$ and $\mathcal{CO}$ denote current *core pairs* and *core objects* respectively. We assume there are at least k objects (i.e., $|\mathcal{CP}| = \frac{k}{2}$) before the new object o arrives. For the object o, we use $\phi(o) = \{o_y\}$ to denote objects where $\theta(o, o_y) > \theta_T$ and o_y does not *dominate* o (Line 2). In Line 3, o' represents the object in $\phi(o)$ with the longest distance to o. Then we have the following three cases:

i) $\phi(o) = \emptyset$ (Line 15).
ii) o' is *not* a *core object* (Line 6-8).
iii) o' is a *core object* (Line 10-13).

In the first case, the algorithm immediately terminates. In the second case, Line 7 removes the $\frac{k}{2}$-th *core pair* from $\mathcal{CP}$ and then add a new *core pair* (o, o'). For instance, we have $\mathcal{CP} = \{(o_1, o_2), (o, o_7), (o_3, o_4)\}$ if $o' = o_7$ in the example of Fig. 4. Then θ_T is updated and the algorithm terminates. In the third case, we use (o, o') to replace the *core pair* (o', o_y),

Algorithm 5: **Update Core Pairs and θ_T**

Input : o: the arriving new object
Output : Updated *core pairs* and the distance threshold θ_T

```
1  while true do
2      φ(o) ← {o_y | θ(o, o_y) > θ_T and o_y does not dominate o };
3      o' ← the furtherest object in φ(o) w.r.t o;
4      if φ(o) ≠ ∅ then
5          if o' ∉ core objects CO then
6              (o_1, o_2) ← the (k/2)-th core pair in CP ;
7              CP := CP \ (o_1, o_2); CP := CP ∪ (o, o');
8              Update θ_T and Terminate the algorithm;
9          else
10             o_y ← the object o_y where (o', o_y) is a core pair;
11             CP := CP \ (o', o_y); CP := CP ∪ (o, o');
12             Update θ_T;
13             o ← o_y;
14     else
15         Terminate the algorithm;
```

where (o', o_y) is a *core pair* before the arrival of o. θ_T is updated and we repeat the *while loop* (Line 1) by treating o_y as the new arriving object o which may contribute to *core pairs* again.

Time Complexity. Suppose there are at most n objects in $\mathcal{R}$, it takes $O(n)$ time to choose o'. In the algorithm correctness analysis below, we show that the *while loop* can repeat **at most** $\frac{k}{2}$ times. Consequently, Algorithm 5 takes $O(kn)$ time in the worst case.

Algorithm Correctness. The correctness of Algorithm 5 is immediate for case i). In the second case (Lines 6-8), suppose there are j *core pairs* with distances longer than $\theta(o, o')$, they will remain unchanged because their corresponding objects cannot contribute to any *new core pair* with the object o according to the selection criteria of o'; that is, o cannot affect the first j *core pairs* in $\mathcal{CP}$. Then (o, o') will serve as the $(j+1)$-th *core pair* in $\mathcal{CP}$. This implies that o will not affect the selection of *core pairs* after the $(j+1)$-th pair since o' is not a *core object* before the arrival of o, and hence we only need to remove the previous $\frac{k}{2}$-th *core pair*. Regarding the third case (Lines 10-13), we use (o', o_y) to denote the j-th *core pair* before the arrival of o. With the same rationale, o cannot affect any *core pair* ranked before (o, o_y), and (o', o_y) will replace (o, o') as the j-th *core pair*. Based on the facts that (o', o_y) is the j-th *core pair* before the arrival of o, and o_y is kicked out from *core objects* by o, o_y cannot contribute to the first j *core pairs* in the future. For instance, if (o_3, o_4) is replaced by (o, o_4) in the example of Fig. 4, o_3 cannot appear in the first two *core pairs* anymore. This implies that the *while loop* of Algorithm 5 can repeat at most $\frac{k}{2}$ times.

Monotonic Property of θ_T. Since θ_T will be updated when the $\frac{k}{2}$-th *core pair* is replaced in the case ii) or the case iii) of Algorithm 5 by another pair with longer distance, the monotonic property of θ_T is immediate as stated in the following theorem.

THEOREM 1. *The diversification distance threshold θ_T grows monotonically against the arrival of the objects.*

4.3 Incremental Diversified SK Search

In this subsection, we introduce the pruning technique based on the monotonic property of θ_T, as well as the details of the incremental diversified SK search algorithm.

Diversity Based Pruning Technique. Fig. 5 illustrates an motivating example of the diversity based pruning technique. We use γ to denote the maximal *network distance* among the objects visited so far, e.g., objects in the shaded area. For any unvisited object o_1, we have $\delta(q, o_1) \geq \gamma$ since objects arrive according to the increasing order of their *network distances*. Moreover, we have $\delta(o_1, o_2) \leq 2 \times \delta_{max}$ for two unvisited objects o_1 and o_2 since $\delta(q, o_1) \leq \delta_{max}$ and $\delta(q, o_2) \leq \delta_{max}$. According to Equation 3, we can come up with the upper bound of $\theta(o_1, o_2)$, denoted by $\bar{\theta}_u$, based on γ and δ_{max}. Similarly, we can derive the upper bound of $\theta(o, o_1)$ between a visited object o and an unvisited object o_1, denoted by $\bar{\theta}_u(o)$, based on the fact that $\delta(o, o_1) \geq \delta(q, o) + \gamma$ for any visited object o. Clearly, we can safely prune all unvisited objects if $\bar{\theta}_u \leq \theta_T$ (i.e., upper bound for any two unvisited objects) and $\bar{\theta}_u(o) < \theta_T$ for any visited object o (i.e., upper bound between o and any unvisited object); that is, none of the unvisited objects can become a *core object* in the diversified search. With the same rationale, a visited object o can also be eliminated from future computation if $\bar{\theta}_u(o) < \theta_T$ and $\theta(o, o') < \theta_T$ for any other visited object o'.

Algorithm 6: **Diversified SK Search(q, k, δ_{max})**

Input : q : query object, k : number of objects requested
δ_{max} : maximal *network distance*
Output : $\mathcal{S}$: diversified k objects regarding SK query

```
1  Compute CP and θ_T on the first k arrived objects;
2  for each o arrived in order (from Algorithm 3) do
3      Update core pairs CP and threshold θ_T (Algorithm 5);
4      γ ← δ(q, o) ;
5      o_1, o_2 ← two objects with δ(q, o_1) := δ(q, o_2) := γ
6      and δ(o_1, o_2) := 2 × δ_max;
7      if θ(o_1, o_2) < θ_T then
8          F := true ;
9          for each object o_x visited so far do
10             Let δ(o_x, o_1) := 2 × γ ;
11             if θ(o_x, o_1) > θ_T then
12                 F := false; Goto Line 2 ;
13             else if θ(o_x, o') < θ_T for any visited object o'
               then
14                 Remove o_x from future computation;
15         if F = true then
16             Terminate the network expansion of Algorithm 3;
17 return CP
```

Algorithm. Algorithm 6 illustrates the details of the incremental diversified SK search. Line 1 initializes *core pairs* and θ_T based on the first k arrived objects in Algorithm 3. Then Lines 2-16 incrementally maintain *core pairs* and θ against the arriving objects (i.e., objects incrementally output from Algorithm 3). Recall that Algorithm 3 can incrementally output the objects satisfying spatial keyword constraint in increasing order of their *network distances* to q. For each new arriving object o, we can calculate the pair-wise *diversification distances* between o and other visited objects. Note that we may invoke Algorithm 3 to conduct *network distances* computation which terminates when all pair-wise *network distances* are calculated between o and other visited objects. Based on the diversity based pruning technique, we terminate the network expansion if all unvisited objects cannot contribute to the diversified k results (Line 16). Finally, Line 17 returns the objects in *core pairs* for the diversified SK search.

In each iteration of Algorithm 6, it takes $O(n_v \log(n_v) + n_e + n_o)$ time to compute the *network distances* for each incoming object against existing objects where n_v and n_e de-

note the number of nodes and edges accessed in the road network respectively, and n_o is the number of objects accessed so far. Line 3 takes $O(n_o \times k)$ time to maintain *core pairs* and θ_T. The pruning procedure (Line 4-16) takes $O(n_o)$ time. Consequently, the total time cost of Algorithm 6 is $O(n_o \times (n_v \log(n_v) + n_e + n_o \times k))$. This implies that it is essential to reduce n_o, i.e., the number of objects accessed in the diversified SK search.

5. PERFORMANCE EVALUATION

In this section, we present results of a comprehensive performance study to evaluate the efficiency and scalability of the proposed techniques in the paper. In our implementation, the road network is organized by CCAM structure as discussed in 2.2. We evaluate the effectiveness of the following indexing techniques for *spatio-textual* objects in the spatial keyword search (Algorithm 3).

- **IR** Inverted R-tree [23], which is a natural extension of the spatial object indexing method in [16] [2].
- **IF** The inverted indexing technique described in Section 3.1. Note that the same indexing structure is employed in [17] for a different problem.
- **SIF** The signature-based inverted indexing technique proposed in Section 3.1.
- **SIF-P** Enhanced SIF by partition technique proposed in Section 3.3. It is reported in our initial experiments that the greedy approach is up to two orders of magnitude faster than the dynamic programming based approach while they achieve similar performance in terms of I/O costs reduced. Therefore, in the experiment we use the greedy partition approach with the maximal cuts 3.

We also evaluate the efficiency of the diversified spatial keyword search algorithm where the *spatio-textual* objects are organized by SIF indexing structure.

- **SEQ** A straightforward implementation of the diversified spatial keyword search algorithm discussed in Section 4.1. Specifically, we first retrieve all objects satisfying spatial keyword constraint (Algorithm 3), and then feed the results into the greedy Algorithm 1 in Section 2.3.
- **COM** The incremental diversified spatial keyword search algorithm proposed in Section 4 (Algorithm 6).

Datasets. Performance of various algorithms is evaluated on both real and synthetic datasets. The following four datasets are deployed in the experiments. Road Network of **NA** is obtained from the North America Road Network(`http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm`) with 175,812 nodes and 179,178 road segments. The *spatio-textual* objects are obtained from the US Board on Geographic Names(`http://geonames.usgs.gov`) in which each object is associated with a geographic location and a short text description. Similarly, we generate dataset **SF** by obtaining the spatial locations from corresponding spatial datasets from Rtree-Portal (`http://www.rtreeportal.org`) and randomly geo-tagging these objects with user-generated textual contents from 20 Newsgroups (`http://people.csail.mit.edu/jrennie/20Newsgroups`). The road network of *SF* is obtained from San Francisco Road Network (`http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm`) where there are 174,955 nodes and 223,000 edges. The dataset **TW** contains 11.5 millions tweets with geo-locations [14] from May 2012 to August 2012, and the road network is obtained from San Francisco Bay Area (`http://www.dis.uniroma1.it/challenge9/download.shtml`) with 321,270 nodes and 800,172 edges. To investigate the scalability of the algorithms, we also include the synthetic dataset **SYN**. The locations of the objects are randomly chosen from *SF* dataset, and their corresponding keywords are obtained from a vocabulary whose term frequencies follow the *zipf* distribution where the parameter z varies from 0.9 to 1.3 with default value 1.1. By default, the number of objects (n_o), the vocabulary size (n_v) and the number of keywords per object (n_t) are set to 1 million, 100 thousands and 15 respectively. Note that we move an object to its closest road segment if it does not lie on any edge in the road network. In the experiments, the locations of all datasets (spatio-textual objects and road networks) are scaled to the 2-dimensional space $[0,\ 10000]^2$. Table 2 summaries the important statistics of four datasets.

Property	SYN	NA	TW	SF
# objects	$1M$	$2.2M$	$11.5M$	$2.25M$
vocabulary size	$100K$	$208K$	$1.6M$	$81K$
avg. # keywords	15	6.8	10.8	26
# nodes.	$17K$	$179K$	$321K$	$175K$
# edges.	$223K$	$179K$	$800K$	$223K$

Table 2: Dataset Statistics

Workload. A workload for the SK query and diversified SK query consists of 500 queries. The average query response time, the average number of disk accesses and the average number of candidate objects are employed to evaluate the performance of the algorithms. The query locations are randomly selected from the locations of the underlying objects. On the other hand, the likelihood of a keyword t being chosen as query keyword is $\frac{freq(t)}{\sum_{t_i \in \mathcal{V}} freq(t_i)}$ where $freq(t)$ is the term frequency of t in the dataset. The number of query keywords (l) varies from 1 to 4 with default value 3. By default, the maximal search distance (δ_{max}) is set to $500 \times l$. In addition, the number of results (k) grows from 5 to 20 and λ varies from 0.5 to 0.9 for diversified spatial keyword search. By default, k and λ is set to **10** and **0.8** respectively.

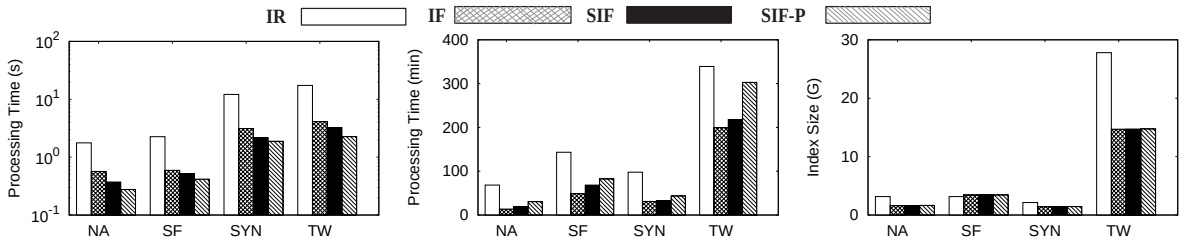
All algorithms in the experiments are implemented in Java and important data structures (e.g., R-tree and B^+ tree) are obtained from the source code package released by [21]. Experiments are run on a PC with Intel Xeon 2.40GHz dual CPU and 4G memory running Debian Linux. For the construction of SIF-P, we only consider the edges whose number of objects ranked at the top 10% compared with other edges, and the number of cuts is set to 3. We use an LRU memory buffer whose size is set to 2% of the network dataset size. All index structures are disk resident, and the page size is fixed to 4096 bytes.

5.1 Evaluating SK Search

In this subsection, we evaluate the performance of spatial keyword search against four indexing structures: IR, IF, SIF and SIF-P.

Evaluation on different datasets. We investigate the query response time, index construction time and index size of four indexing structures against four datasets *NA*, *SF*, *SYN* and *TW* where other parameters are set to default values. Fig. 6(a) reports the response time of four algorithms. The performance of IR is nearly 4 times slower than other three indexing techniques. This is because the construction of IR is independent to the underlying road network struc-

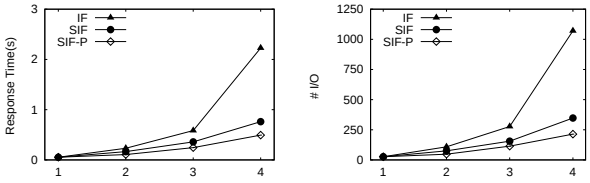
[2]We slightly change the *find entity* function in [16] to efficiently support spatial keyword search for the inverted R-tree index. Instead of searching entities on a global R-tree which organize all objects, we only need to explore the inverted R-trees related to the query keywords.



(a) Response Time (b) Index Construction Time (c) Index Size

Figure 6: SK search on Diff. Datasets

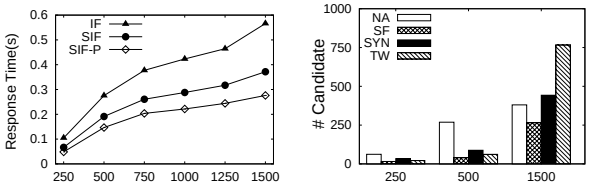
ture, and it is cost expensive to check the objects lying on an edge. Therefore, we exclude IR from the following performance evaluations. IF greatly improves the performance by taking advantage of the connections between the edges and their corresponding objects in the inverted index. SIF and SIF-P can further significantly reduce the response time by utilizing the signature technique. As expected, it is reported that the refined signature in SIF-P can achieve better performance. As shown in Fig. 6(b), compared with IF and SIF, SIF-P takes the longest time for the index construction as the partition of the edges may be time consuming. Due to the compactness of the signatures, Fig. 6(c) shows that SIF and SIF-P only take slightly more space to keep the signatures compared with IF which only keeps the inverted files.



(a) Time (b) I/O cost

Figure 7: Diff. number of query keywords (l)

Effect of the number of query keywords (l). We evaluate the effect of the number of query keywords in Fig. 7 on dataset *NA* where l varies from 1 to 4. Fig. 7 reports that the performance of three algorithms in terms of the query response time and the number of I/O accesses degrades against the growth of l. As expected, SIF significantly outperforms IF since the signature-based index can reduce the number of I/O invoked by *false hits*. SIF-P achieves the better performance compared with SIF because the advanced partition technique can further improve the effectiveness of the signature technique.



(a) Time (b) #Candidates

Figure 8: Diff. search range (δ_{max})

Effect of the search range (δ_{max}). Fig. 8(a) illustrates the query response time of the algorithms as a function of δ_{max} on dataset *NA*. It is shown that IF is much more sensitive to the growth of δ_{max} in comparison with SIF and SIF-P. This is because the number of *false hits* grows quickly against δ_{max} and IF cannot avoid the unnecessary I/O accesses incurred by the *false hits*. As expected, Fig. 8(b) shows that the number of candidate objects on four datasets increases where δ_{max} grows from 250 to $1,500$.

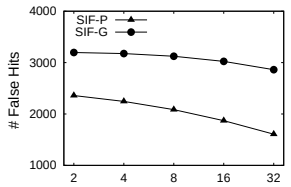


Figure 9: Diff. cuts

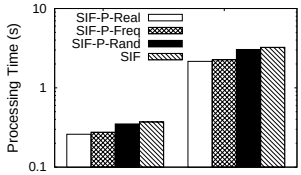


Figure 10: Diff. logs

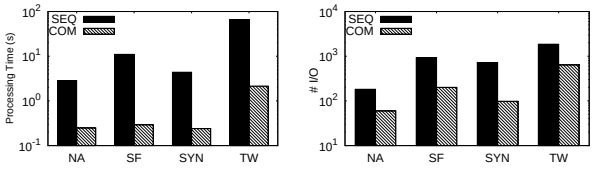
Evaluation on space cost-effectiveness. To evaluate the space cost-effectiveness of the advanced signature-based indexing technique (SIF-P), Fig. 9 reports the number of *false hits* on *SF* where the number of maximal cuts allowed grows from 2 to 32. As expected, the performance of SIF-P improves when the number of maximal cuts (i.e., available index space) grows. Meanwhile, we also evaluate another simple and intuitive approach, namely group-based indexing technique (SIF-G). Specifically, besides the individual terms, we also build the signature file and inverted list for the combinations of the frequent terms. For instance, two frequent terms t_1 and t_2 together can serve as a new term, and only the edges containing an object with both terms t_1 and t_2 are kept in its corresponding inverted list. For each number of maximal cut in Fig. 9, we choose the top x most frequent terms so that the space occupied by their pairwise combined terms is 10 times larger than the signature file of SIF-P index structure. For instance, the size of the signature file of SIF-P index is 53M when the number of maximal cuts goes to 32. For the corresponding SIF-G, we consider the pairwise combination of the top 25 most frequent terms which leads to 530M extra index space compared with SIF technique. As shown in Fig. 9, the advanced signature-based indexing technique is more space cost-effective because the size of the signature files is much smaller than that of inverted files.

Evaluation on query logs. In Section 3.3, we use the query log to construct the advanced signature-based index structure. Intuitively, we can achieve the best performance if the query log is directly obtained from the query load. In Fig. 10, we demonstrate to what extent the performance of the advanced signature-based technique is affected by the keywords distribution of the query log. Four indexing techniques are evaluated in the experiment on two real datasets, *NA* and *TW* respectively. SIF-P-Real, SIF-P-Freq and SIF-P-Rand are advanced signature-based indexing structures constructed based on different query logs. Specifically, the query load is used as query log in SIF-P-Real. For SIF-P-Freq, we generate query log based on the frequency of the keywords on each edge which is the default setting in the experiment. While the query log is generated by randomly choosing keywords on each edge for SIF-P-Rand. As expected, Fig. 10 shows that SIF-P-Real achieves the best performance since we use the query load to build the index. SIF-P-Freq has the similar performance because we assume that a query keyword is chosen based on its corresponding

object keywords frequency in our experiments. The performance of SIF-P-Rand degrades since the keywords distribution of the query log used is quite different to that of the query load. Nevertheless, it still outperforms the simple signature-based technique (SIF).

5.2 Evaluating Diversified SK Search

In this subsection, we evaluate the efficiency of two algorithms SEQ and COM against various factors which may potentially affect the performance of the algorithms.



(a) Time (b) I/O

Figure 11: Diversified SK search on Diff. Datasets

Evaluation on different datasets. We investigate the query response time and the number of I/O accesses for SEQ and COM against four datasets *NA*, *SF*, *SYN* and *TW* where other parameters are set to default values. In Fig. 11, COM demonstrates superior performance in terms of response time and I/O costs. This is because that SEQ retrieves *all* objects which satisfy the spatial keyword constraint while COM can take advantage of the diversity based pruning technique to eliminate non-promising objects during the network expansion.

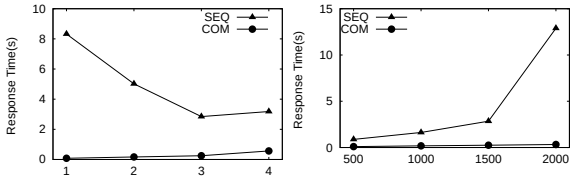


Figure 12: Varying l **Figure 13: Varying δ_{max}**

Effect of the number of query keywords (l). Fig. 12 reports the performance of two algorithms when l varies from 1 to 4. On one hand, the larger l will lead to the smaller chance for each object to meet the keyword constraint. On the other hand, the search region increases with l in our experiment setting. This implies that more objects are involved in the computation when l grows. It is observed that the performance of SEQ fluctuates against the growth of l while the performance of COM consistently degrades. Nevertheless, COM significantly outperforms SEQ because a large number of non-promising objects are pruned in COM.

Effect of the search region (δ_{max}). We evaluate the impact of the search range in Fig. 13, where dataset *NA* is used. It is shown that COM significantly outperforms SEQ under all settings, especially when the search range is larger. This is because SEQ needs to load *all* objects satisfying the spatial keyword constraint, but COM can enjoy benefits from the diversity based pruning technique.

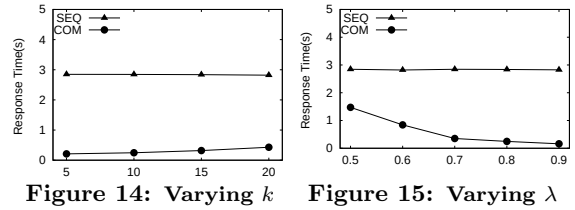


Figure 14: Varying k **Figure 15: Varying λ**

Effect of the k and λ. Fig. 14 and Fig. 15 report the query response time of the algorithms as a function of k and λ against the dataset *NA*. The performance of SEQ is not sensitive to the growth of k and λ because the number of candidate objects remains the same with the growth of k and λ and the dominant cost of SEQ is to retrieve all candidate objects. On the other hand, COM becomes less efficient because a larger k will lead to a smaller diversification distance threshold θ_T, i.e., lower pruning capability. Similarly, a larger λ value indicates the higher priority to the relevance (closeness) aspect, which may result in an early termination of the network expansion.

Effect of the term frequency skewness (z). In Fig. 16 (a), we evaluate the impact of the term frequency skewness on the algorithms, where z varies from 0.9 to 1.3. As the query keyword is selected based on the term frequencies, the number of objects satisfying keyword constraint in the search region will increase when z increases, and hence the performances of both algorithms degrade when z is large. Nevertheless, the performance of COM is more scalable.

Effect of the number of objects (n_o). Fig. 16 (b), shows the response time for SEQ and COM as we vary the object number from 0.5M to 2M. As expected, the growing of the number of objects leads to more candidate objects in the region, which deteriorates the performance of SEQ and COM. Nevertheless, compared with SEQ, the growth of COM is less significant.

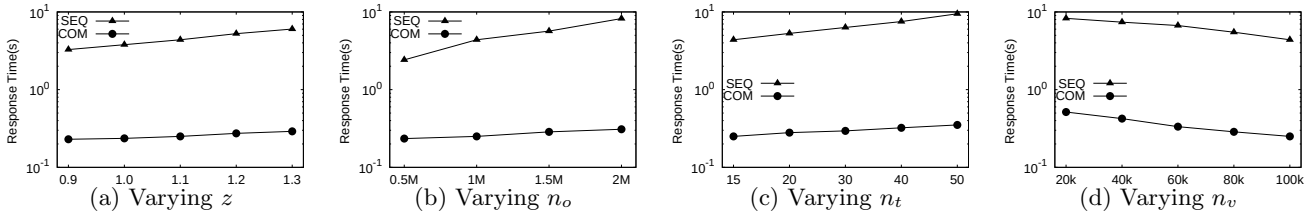
Effect of the average number keywords per object (n_t). This experiment evaluates the performance when we vary n_t. As expected, Fig. 16 (c) shows that the performance of both algorithms degrade with the growth of n_t because more objects are likely to satisfy the keyword constraint for larger n_t value.

Effect of the vocabulary (n_v). This experiment is to study the effect of the number of the vocabulary. As discussed in Section 3.2, a larger vocabulary size ($|\mathcal{V}|$) leads to a smaller number of objects which satisfy the keyword constraint, and hence results in the better performance. It is reported in Fig. 16 (d) that the performance of both algorithms improve when the vocabulary size grows from 20k to 100k.

6. RELATED WORK

To the best of our knowledge, there is no existing work on the problem of diversified spatial keyword search on road network. Below, we introduce two important categories of related work.

Spatial Keyword Search. Spatial keyword search has been intensively studied in recent years due to the massive number of *spatio-textual* objects rapidly accumulated in many important applications (See [6, 2] for a comprehensive survey). One of the most important queries is the *boolean spatial keyword search* which aims to find a set of *spatio-textual* objects based on the spatial proximity (e.g., k nearest neighbor search and range search) and query keywords constraint (i.e., each object must contain all query keywords). Many efficient indexing techniques have been proposed such as inverted R-tree [23], information retrieval R-tree [11] and inverted linear quadtree [22]. Although we also study the problem of boolean spatial keyword search, these indexing techniques cannot effectively support query processing on road network since they are constructed in Euclidean space and are independent to the underlying road network. Besides the boolean spatial keyword search, there are many important variants in the literature with different focuses such as the ranked spatial keyword query (e.g., [7,



(a) Varying z (b) Varying n_o (c) Varying n_t (d) Varying n_v

Figure 16: Effect of term frequency skewness (z), #objects (n_o), #keywords per object (n_t) and vocabulary size (n_v)

21]) and collective spatial keyword search [15].

Regarding the spatial keyword search on road network, Rocha-Junior *et al.* investigate the ranked keyword ranking on road network [17]. The problem of keyword-aware travel route search is investigated to find optimal route which covers a set of query keywords along the route (e.g.,[3]). Nevertheless, these problems are inherently different from the diversified spatial keyword search on road networks.

Results Diversification. In recent years, results diversification has been widely used to enhance the research results (e.g., [4, 9, 10]) with different focuses (e.g., content, novelty and coverage). The Maximal Marginal Relevance (MMR) model is first studied by Carbonell and Goldstein [4] for text retrieval and summarization. Particularly, MMR aims to find a results by maximizing the query relevance and also minimizing the similarity between objects in the results. The max-sum diversification function used in our study can be regarded as a natural extension of MMR. In [12], Gollapudi and Sharma develop an axiomatic approach to characterize and design diversified systems in which the max-sum diversification function is mapped to the facility dispution problem. Regarding the **spatial diversity**, there are many research work (e.g., [20, 19, 13, 5]) which aim to find spatially diversified results so that objects are well spread in the region of interest. Recently, Kucuktunc *et al.* [13] investigate the diversified k neareast neighbor problem base on the angular similarity. In [5], Catallo *et al.* investigate diversified search over objects embedded in a low dimensional space over a bounded region. Techniques in the above work are developed in Euclidean space and hence they are not suitable to the problem studied in the paper.

7. CONCLUSIONS AND FUTURE WORK

This is the first work to study the problem of diversified spatial keyword search on road networks, which has a wide spectrum of applications. To facilitate the diversified spatial keyword search, we propose a signature-based inverted indexing mechanism on road network as well as an incremental diversified search algorithm to reduce the computational cost. Moreover, effective spatial keyword pruning and diversity pruning techniques are developed in the paper to eliminate non-promising objects. Our comprehensive experiments convincingly demonstrate the efficiency of our techniques. As a possible future work, we will investigate how to extend our signagure-based indexing techniques and pruning techniques to the problem of diversified ranked spatial keyword query on road networks.

Acknowledgments. Ying Zhang is supported by ARC DE140100679 and DP130103245. Wenjie Zhang is supported by ARC DE120102144 and DP120104168. Xuemin Lin is supported by NSFC61232006, NSFC61021004, ARC DP120 104168 and DP110102937. Muhammad Aamir Cheema is supported by ARC DE130101002 and DP130103405.

8. REFERENCES

[1] A. Angel and N. Koudas. Efficient diversity-aware search. In *SIGMOD Conference*, 2011.

[2] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu. Spatial keyword querying. In *ER*, 2012.

[3] X. Cao, L. Chen, G. Cong, and X. Xiao. Keyword-aware optimal route search. *PVLDB*, 5(11), 2012.

[4] J. G. Carbonell and J. Goldstein. The use of mmr, diversity-based reranking for reordering documents and producing summaries. In *SIGIR*, 1998.

[5] I. Catallo, E. Ciceri, P. Fraternali, D. Martinenghi, and M. Tagliasacchi. Top-k diversity queries over bounded regions. *ACM Trans. Database Syst.*, 38(2), 2013.

[6] L. Chen, G. Cong, C. S. Jensen, and D. Wu. Spatial keyword query processing: An experimental evaluation. *PVLDB*, 2013.

[7] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1), 2009.

[8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms (second edition)*. 2001.

[9] M. Drosou and E. Pitoura. Disc diversity: result diversification based on dissimilarity and coverage. *PVLDB*, 6(1), 2012.

[10] M. Drosou and E. Pitoura. Dynamic diversification of continuous data. In *EDBT*, 2012.

[11] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, 2008.

[12] S. Gollapudi and A. Sharma. An axiomatic approach for result diversification. In *WWW*, 2009.

[13] O. Kucuktunc and H. Ferhatosmanoglu. Diverse nearest neighbors browsing for multidimensional data. *IEEE Trans. Knowl. Data Eng.*, 25(3), 2013.

[14] G. Li, Y. Wang, T. Wang, and J. Feng. Location-aware publish/subscribe. In *KDD*, 2013.

[15] C. Long, R. C.-W. Wong, K. Wang, and A. W.-C. Fu. Collective spatial keyword queries: a distance owner-driven approach. In *SIGMOD*, 2013.

[16] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *VLDB*, pages 802–813, 2003.

[17] J. B. Rocha-Junior and K. Nørvåg. Top-k spatial keyword queries on road networks. In *EDBT*, 2012.

[18] S. Shekhar and D.-R. Liu. Ccam: A connectivity-clustered access method for networks and network computations. *IEEE Trans. Knowl. Data Eng.*, 9(1), 1997.

[19] J. Tang and M. Sanderson. Spatial diversity, do users appreciate it? In *GIR*, 2010.

[20] M. J. van Kreveld, I. Reinbacher, A. Arampatzis, and R. van Zwol. Multi-dimensional scattered ranking methods for geographic information retrieval. *GeoInformatica*, 9(1), 2005.

[21] D. Wu, G. Cong, and C. S. Jensen. A framework for efficient spatial web object retrieval. *VLDB J.*, 2012.

[22] C. Zhang, Y. Zhang, W. Zhang, and X. Lin. Inverted linear quadtree: Efficient top k spatial keyword search. In *ICDE*, 2013.

[23] Y. Zhou, X. Xie, C. Wang, Y. Gong, and W.-Y. Ma. Hybrid index structures for location-based web search. In *CIKM*, 2005.