Efficient Processing of Relevant Nearest-Neighbor Queries

CHRISTODOULOS EFSTATHIADES, National Technical University of Athens
ALEXANDROS EFENTAKIS, Research Center “Athena”
DIETER PFOSER, George Mason University

Novel Web technologies and resulting applications have led to a participatory data ecosystem that, when utilized properly, will lead to more rewarding services. In this work, we investigate the case of Location-Based Services, specifically how to improve the typical location-based Point-of-Interest (POI) request processed as a *k*-Nearest-Neighbor query. This work introduces Links-of-Interest (LOI) between POIs as a means to increase the relevance and overall result quality of such queries. By analyzing user-contributed content in the form of travel blogs, we establish the overall popularity of an LOI, that is, how frequently the respective POI pair was visited and is mentioned in the same context. Our contribution is a query-processing method for so-called *k*-Relevant Nearest Neighbor (*k*-RNN) queries that considers spatial proximity in combination with LOI information to retrieve close-by and relevant (as judged by the crowd) POIs. Our method is based on intelligently combining indices for spatial data (a spatial grid) and for relevance data (a graph) during query processing. Using landmarks as a means to prune the search space in the Relevance Graph, we improve the proposed methods. Using in addition A*-directed search, the query performance can be further improved. An experimental evaluation using real and synthetic data establishes that our approach efficiently solves the *k*-RNN problem.

Categories and Subject Descriptors: H.2.8 [**Information systems**]: Spatial-Temporal Systems

General Terms: Algorithms, Performance

Additional Key Words and Phrases: Nearest-neighbor queries, geospatial crowdsourcing, text mining, context

ACM Reference Format:
Christodoulos Efstathiades, Alexandros Efentakis, and Dieter Pfoser. 2016. Efficient processing of relevant nearest-neighbor queries. ACM Trans. Spatial Algorithms Syst. 2, 3, Article 9 (September 2016), 28 pages.
DOI: http://dx.doi.org/10.1145/2934675

1. INTRODUCTION

Location-Based Services have been at the forefront of mobile computing as they provide an answer to the simple question as to what is around us. A lot of effort has been dedicated to improving such services, typically by improving the selectivity of each request. Rating sites augment Points-of-Interest (POIs) with quality criteria. Preferences add further user-specific parameters to a request. Context limits the available information based on situational choices. However, what has not been captured yet are

This work has been supported by the NGA NURI grant HM02101410004 and the European Union Seventh Framework Programme - Marie Curie Actions, Initial Training Network GEOCROWD (www.geocrowd.eu) under grant agreement No. FP7-PEOPLE-2010-ITN-264994.
Authors’ addresses: C. Efstathiades, Department of Computer Science and Engineering, School of Sciences, European University Cyprus, 6 Diogenes Street, Engomi, Nicosia, P.O. Box 22006, 1516 Nicosia, Cyprus; email: c.efstathiades@euc.ac.cy; A. Efentakis, Institute for the Management of Information Systems, Research Center “Athena”, Artemidos 6 & Epidavrou, Marousi 15125, Greece; email: efentakis@imis.athena-innovation.gr; D. Pfoser, Department of Geography and Geoinformation Science, George Mason University, Exploratory Hall, Rm 2203, 4400 University Drive, MS 6C3, Fairfax, VA, 22032; email: dpfoser@gmu.edu.

DOI: http://dx.doi.org/10.1145/2934675

user experiences *per se*, that is, assessing what people want in terms of what people in the same situation have done in the past. Our objective is to provide location-based services—specifically, relevant k-Nearest Neighbor (k-NN) searches—based on the crowdsourced choices and experiences that other users have had in the past. The basic premise of a so-called k Relevant Nearest Neighbor (k-RNN) query is to combine *spatial proximity* with *cognitive proximity*, or relevance, as observed in previous user behavior. We introduce the concept of a Link-of-Interest (LOI) between two POIs to express respective *relevance*, that is, "find related nearest POIs to my location." In this first approach, relevance is inferred by observing pairs of POIs that are frequently mentioned in the same context. We discover this co-occurrence of POI pairs in travel blogs based on textual proximity, for example, same paragraph. The more frequently POIs co-occur, the stronger and the more relevant is the LOI existing between them. Capturing POIs and LOIs in combination with k-RNN queries can be used to rediscover travel patterns, that is, links between objects that frequently appear in other people's itineraries. Since not all objects are of equal type, for example, restaurants, museums, bus stops, shops, hotels, and so on, and this is in contrast to classical LBS, k-RNN queries will allow us to discover semantic chains of objects; thus, they provide the basis for an actual trip-planning software. Eventually, such semantic chains will be linked to user profiles that can then be used to generate customized travel guides.

While existing work addresses spatio-textual search, that is, introducing a spatial aspect to (Web) search, thus enabling it to index and retrieve documents according to their geographic context, to the best of our knowledge, the problem of combining user experience (expressed as relevance) with spatial proximity in the form of k-RNN search has only been addressed in Efstathiades and Pfoser [2013]. Here, relevance information is represented by means of a graph in which POIs represent nodes and LOIs are links. The spatial aspect of the data (POI locations) is captured using a spatial grid. The overall challenge in this work is how to combine these two data structures (graph and grid) to efficiently process k-RNN queries. Initially, two basic methods are presented. *GR-Sync* (GR = Grid/Graph) expands the two indexes separately, but synchronizes their search at certain steps. *GR-Link* uses a tighter integration in that spatial search results are seeded to the graph search to minimize costly (and most often unnecessary) expansions. Focusing on the more efficient *GR-Link* method, our additional contribution will be optimizing the search in the Relevance Graph by using landmark-based estimation. This approach belongs to the family of approximate shortest-path methods, since the distances between pairs of vertices can be estimated based on precomputed distances to a fixed set of landmark nodes. We use those estimates to benefit our graph traversal by excluding nodes that, based on the approximate distances, are not worth exploring (filter step). The resulting method, *GR-Link-LM*, also utilizes bidirected BFS to improve performance. Using the information that we gain on lower-bound distance approximations, we introduce *GR-Link-LM**, which additionally uses A*-search to further prune the search space. Experimental evaluation shows that the use of landmark-based estimation significantly improves the performance of k-RNN query processing.

The outline of the remainder of this work is as follows. Section 2 discusses related work. The basic concepts and data structures for processing k-RNN queries are introduced in Section 3. Sections 4 and 5 introduce the various k-RNN query processing methods and respective optimizations. The results of the experimental evaluation are presented in Section 6. Section 7 presents conclusions and directions for future research.

2. RELATED WORK

To the best of our knowledge, the exact problem of combining user experience (expressed as relevance) with spatial proximity has not been studied in literature. A related research topic is spatio-textual search. According to Vaid et al. [2005], studies in this area try to make web search geographically aware, thus enabling it to index and retrieve documents according to their geographic context. Current research focuses on combining spatial and textual indexes and proposes hybrid methods to support geographical awareness. Current approaches are dealing with merging R trees, regular grids, and space-filling curves with a textual index such as inverted files or signature files. One of the first works in spatio-textual indexing was conducted in the context of the SPIRIT search engine [Vaid et al. 2005]. SPIRIT facilitates the use of regular grids as spatial indexes and inverted files for the indexing of documents. Following this idea, Zhou et al. [2005] present hybrid indexing approaches comprised of R* trees [Beckmann et al. 1990] for spatial indexing and inverted files for text indexing. Several approaches following the combination of R* trees and inverted files followed [Zhou et al. 2005; Cong et al. 2009; Rocha-Junior et al. 2011]. Chen et al. [2006] use space-filling curves for spatial indexing, whereas in De Felipe et al. [2008], R trees are combined with signature files in the internal nodes of the tree, so that a combined index is created. The algorithm in Hjaltason and Samet [1999] is used for incremental nearest-neighbor queries. Cong et al. [2009] propose the IR-tree index, creating an inverted file for each node of an R tree, again using the algorithm in Hjaltason and Samet [1999]. Here, linear interpolation is used [Martins et al. 2005] in order for the spatial distance to be combined with the textual distance and therefore produce a combined score. In our work, we use a similar approach when combining the relevance score with spatial distance. A very related approach to Cong et al. [2009] is presented in Li et al. [2011]. Recently, Rocha-Junior et al. [2011] proposed the hybrid index S2I, which uses aR trees [Papadias et al. 2001]. The authors claim to outperform all other proposed approaches in this problem domain.

Another type of spatio-textual query is spatial-group keyword queries. These aim at finding groups of spatio-textual objects that collectively satisfy a number of given keywords, while minimizing the collective distances between points in the group and the given query point. Zhang et al. [2009] and Zhang et al. [2010] introduce the bR*-tree structure in order to solve the *mCK* query. This query searches for the group of spatio-textual objects that collectively satisfy *m* given keywords while the maximum distance between the objects remains minimal. Cao et al. [2011] introduces two spatial-group keyword queries. Both retrieve a group of objects that cover a given set of keywords; the first requires that the total distance between the objects in the group and the query point is minimized, while the second seeks to minimize the sum of the maximum distance between a point in the group and the query point and the maximum distance between any objects in the group. Long et al. [2013] revisit the spatial-group keyword query with the maximum sum cost and propose a variation based on minimizing the maximal distance between any two objects of the group or any object of the group and the query point.

Our work differs significantly in that we do not consider textual similarity, but use the co-occurrence of POIs in text as an indicator for relevance. This relevance information (and no actual textual information) is then combined with the spatial aspect of the POIs to retrieve relevant nearest neighbors.

To the best of our knowledge, the most related work that combines user experiences and spatial data is Cao et al. [2010], which introduces the notion of prestige to denote the textual relevance between nearby to the query point objects. It also uses a graph, but the way it is constructed is based on the spatial distance between the objects as opposed to our method of using relevance. The prestige from a given query point is propagated through the graph, information that is later used to extend the IR tree [Cong et al. 2009]. Compared to our work, our index does not include a factor of randomness in the relevance score calculation, as is the case with PageRank-based algorithms. In general, the notion of relevance in Cao et al. [2010] is based on different measures that are query dependent (textual relevance), a fact that is not being considered in our work. Our query considers only the location of a query point. Relevance is only derived from the collected dataset.

Our work is also related to the area of geo-social query processing. Geo-social networks are graphs that consist of users (nodes) and friendships with other users (edges). The nodes of the graph have spatial properties (location). A first work towards geo-social query processing [Armenatzoglou et al. 2013] introduces a general framework to query a geo-social network. The social and geographical aspect are considered separately; as a result, the answers to their proposed queries are not ranked based on spatial and social dimensions simultaneously. In combining spatial and social proximity, Mouratidis et al. [2015] compute top-*k* users based on a minimized weight function combining Euclidean distance and distance in a social graph. Somewhat similar to this work, spatial and graph distance are combined in a scoring function. However, the actual graph is different in that we use a weighted graph. Related to weights is also how the data is updated. While new POIs (nodes) can be added to the graph as well, the bulk of the updates will come from newly discovered relevance information, which requires frequently updating the weights of the graphs. Further, our scoring function combines edge weights with graph proximity, which makes it considerably harder to add such relative graph distance information to (the nodes of) a spatial index such as proposed in the work of Mouratidis et al. [2015]. Li et al. [2015] propose a query that combines social proximity, text similarity, and time freshness, building a respective 3D index to incorporate these 3 dimensions. Although the techniques that they use in pruning the social graph in order to compute social proximity are similar to the landmark methods that we use, no spatial criterion is being used during the search, that is, they do not consider the spatial distance between users in a social network. In addition, they consider in-memory datasets, while we assume disk-resident data.

The problem of combining relevance and spatial distance is closely related to the problem of computing top-*k* queries that are based on different subsystems, such as those studied by Fagin [1999] and Fagin et al. [2003]. The difference in our approach is that we have a specific focus on spatial data and on how to combine the result sets. Also, in our final *k*-RNN list, we have the exact scores, compared to the approximate scores of the NRA algorithm.

In the context of efficient *k*-NN query processing, Nutanong et al. [2008] propose the V*-Diagram to compute moving *k*-NN queries, that is, assessing the *k*-NNs of a moving query point. This approach provides incremental evaluation of the *k*-NN set, but differs significantly from our problem, since we do not take moving objects into consideration. Koudas et al. [2004] investigate approximate *k*-NN queries for data streams and propose the DISC technique to answer the e-approximate *k*-NN problem (e*k*-NN) using space-filling curves. *e* refers to an error by which the result set is bounded. This work differs from ours, since we do not take into consideration either data streams or approximate results. Sun et al. [2015] propose the *k-nearest neighbor temporal aggregate query* (*k*-NNTA), which computes the nearest POIs to a query point based on spatial distance and the number of check-ins for a query point in a given time interval. A weighted sum is used to compute a combined ranking score, which is similar to our work, but used in a different problem setting.

In this work, relevance is computed by counting the co-occurrences of POIs in the same paragraphs of texts. In spite of the simplicity of this metric, recent results show that POIs co-occur in documents when they are spatially close, have similar properties, or interact with each other [Liu et al. 2014]. On a similar note, Skoumas et al. [2013] show that textual narratives can be used to derive spatial relationships between POI pairs. In their analysis, they go even further and use this textual information as a means to perform qualitative positioning, that is, discovering the location of POIs using only spatial relationships mentioned in texts.

In our work, we use *landmarks* as a means to optimize k-RNN processing. Landmarks in relation to shortest-path problems were introduced in Goldberg and Harrelson [2005]. In this work, a small set of vertices called landmarks is chosen and, for each vertex, the authors precompute distances to and from every landmark. The resulting method is the seminal ALT algorithm. Potamias et al. [2009] significantly augmented the concept of landmarks in various ways. To the best of our knowledge, they were the first to claim that the upper bounds obtained by the landmark preprocessing phase are also important. They expand the usage of landmarks to undirected unweighted graphs (similar to our work). As a result, their preprocessing phase is faster and easier, as it suffices to calculate distances from landmarks to all other vertices. Although there were many important works utilizing the concept of landmarks in various types of graphs, such as Tretyakov et al. [2011] and Gubichev et al. [2010], our work is heavily influenced and expands the research results of Goldberg and Harrelson [2005] and Potamias et al. [2009]. We use undirected graphs (similar to Potamias et al. [2009]) and utilize lower bounds obtained by the preprocessing phase of landmarks in order to exclude neighbors that cannot possibly belong to the relevant neighbors set (see Section 5). Moreover, we use a variant of the ALT algorithm of Goldberg and Harrelson [2005] to accelerate graph traversal from each spatial seed toward the query point. Results of Section 6 clearly demonstrate the correctness and efficiency of our choices.

Finally, the work presented in this article is based on Efstathiades and Pfoser [2013], which presented the basic k-RNN query processing methods. This work introduces landmarks, the use of bidirected BFS graph traversal, and A* search as a means to optimize k-RNN processing. These additions also resulted in an extended and revised experimentation section.

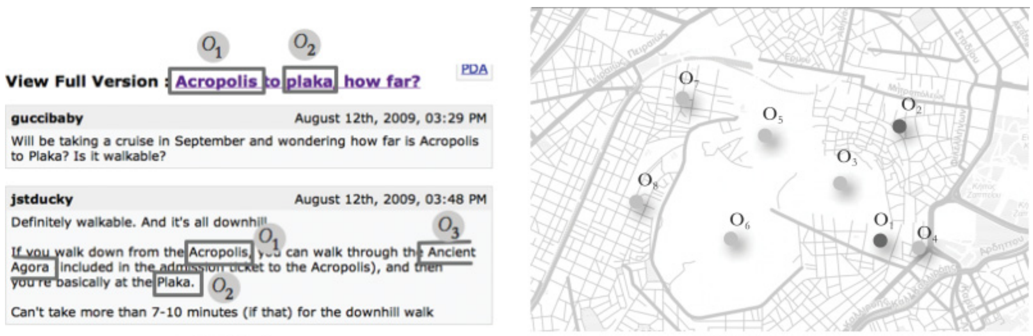
3. DATA AND *K*-RNN QUERIES

With the proliferation of the Internet as the primary medium for data publishing and information exchange, we have seen an explosion in the amount of online content available on the Web. In addition to professionally produced material being offered free on the Internet, the public has also been encouraged to make its content available online to everyone as User-Generated Content (UGC). In the following, we describe the data we want to utilize and how to exploit it in the context of k-RNN queries.

3.1. Data

Web-based services and tools can provide means for users through attentional (e.g., geo-wikis, geocoding photos) or unattentional efforts (e.g., routes from their daily commutes) to create vast amounts of data concerning the real world that contain significant amounts of information ("crowdsourcing"). The simplest possible means to generate content is by means of *text* when (micro)blogging. Any type of text content may contain geospatial data, such as the mentioning of POIs, but also data characterizing the relationship between two POIs, e.g., eating at Eleni's Tavern after visiting the Acropolis in Athens, Greece.

In this work, we try to discover collections of POIs in texts and to use k-RNN queries as a means to navigate this forest of interesting locations to retrieve not only nearby but also relevant objects. Consider the example of Figure 1(a). In the specific text snippet of a travel blog, three distinct spatial objects are mentioned: $O_1 = \{\text{Acropolis}\}$, $O_2 = \{\text{Plaka}\}$, and $O_3 = \{\text{Ancient Agora}\}$. We introduce the concept of *Link-of-Interest* (LOI) as a means to express relevance between two POIs. Assuming that $O = \{O_1, \dots, O_n\}$ is the set of all discovered POIs, then a text paragraph $P_x \subset O$ contains a set of POIs. It also holds that $O = \bigcup P$. We state that there exists an LOI $L_{i,j}$ between two POIs O_i and O_j if both POIs are mentioned in the same text paragraph P_x. The set of all LOIs


View Full Version : Acropolis to plaka how far? PDA

guccibaby August 12th, 2009, 03:29 PM

Will be taking a cruise in September and wondering how far is Acropolis to Plaka? Is it walkable?

jstducky August 12th, 2009, 03:48 PM

Definitely walkable. And it's all downhill

If you walk down from the Acropolis, you can walk through the Ancient Agora (included in the admission ticket to the Acropolis), and then you're basically at the Plaka.

Can't take more than 7-10 minutes (if that) for the downhill walk

(a) POIs in text (b) POIs on a map

Fig. 1. “Close by” points of interest.

is defined as follows:

$$L = \{L(O_i, O_j) | \exists P_x \in P : O_i \in P_x \wedge O_j \in P_x\}. \quad (1)$$

It is apparent that the definition of relevance is a very simple one, that is, co-occurrence in a text paragraph. In future work, we intend to exploit the entire document structure as well as use sentiment information. However, any of these considerations will only affect the way we compute relevance and not the presented techniques for computing the specific type of query.

In the example in Figure 1, when considering O_1 as the query POI, we can see that it has a stronger relevance to O_2 than to O_3 and O_4, since O_1 and O_2 co-occur more often in the same paragraph of the given text snippet. Although O_4 is spatially closer to O_1, the higher relevance of O_2 makes it the immediate relevant neighbor of O_1, that is, when considering both, the spatial distance and relevance. O_3 is spatially closer to O_1, but less relevant; thus, it is overall more “distant.”

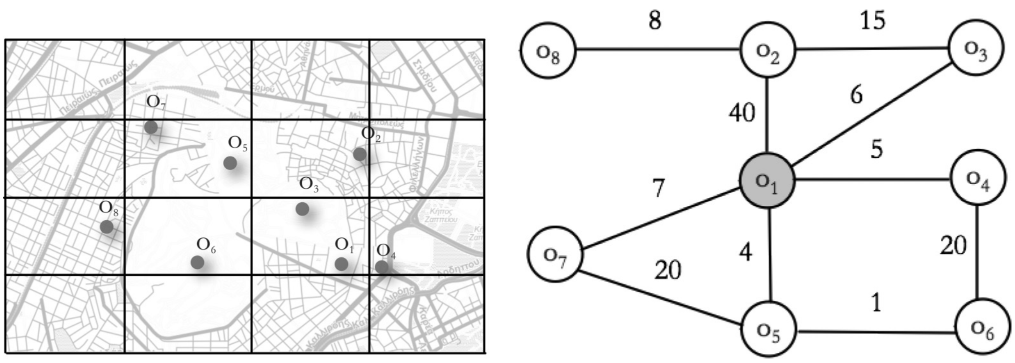
As we will see in the following section, *k*-RNN queries have the ambition to rediscover close-by POIs that have been visited by people in similar situations before. Here, we combine the relevance information extracted from travel blog narratives with spatial proximity. Relevance can be considered an additional filter to navigate the forest of POIs in, for example, a tourist context.

3.2. *k*-RNN Queries

Combining relevance information with spatial distance will allow us to provide better query results, that is, *POIs that are close by and relevant*.

Let D be a spatial database that contains spatial objects O and LOIs L. Each spatial object is defined as $O_i = (O_i.id, O_i.loc)$ and each LOI as $L_k = (O_i.id, O_j.id, r)$, $O.id$ is a unique object identifier, $O.loc$ captures the object location in two-dimensional space, and r provides the relevance (score) of an LOI existing between two POIs O_i and O_j. The relevance r is computed as the total number of occurrences of an LOI, that is, in the entire text corpus. In other words, r measures how often two POIs co-occur in paragraphs of the entire text corpus. The intuition is that the more frequently they are mentioned together, the more important is the LOI existing between them.

We introduce the *k*-RNN query as follows. Given a query point that is represented as a POI, find the *k*-relevant nearest neighbors by taking into account both spatial proximity and relevance. An example is given in Figure 2 for a 1-RNN neighbor. The edge weights denote the relevance between the POIs, for example, the weight of the edge linking O_2 and O_8 is 8. This means that the two objects co-occur eight times in the same paragraph when considering all documents of the available corpus. A



(a) Map (b) Relevance graph

Fig. 2. Relevance graph and spatial “map” data.

combined k-RNN score assesses the spatio-relevance distance between the two points, that is, the smaller the distance (score), the better.

Equation (2) provides a means to compute a combined k-RNN score s_{rnn} that considers both spatial distance and relevance.

$$s_{rnn} = \frac{\alpha * s_r(Q, O)}{s_r.max} + \frac{(1-\alpha) * s_d(Q, O)}{s_d.max} \tag{2}$$

Q is the query point, O is the spatial object for which we compute the score with respect to Q, s_r is a relevance score and can be any kind of metric, $s_r.max$ is the maximum possible relevance score, s_d is the Euclidean distance between Q and O, and $s_d.max$ is the maximum distance and depends on the query space. The parameter $0 \leq \alpha \leq 1$ is used to denote the importance of each distance function (relevance or spatial) and can be tuned according to the user's needs.

The following sections give detailed explanations as to the calculation of the relevance score and the computation of k-RNN queries.

3.3. Access Methods

Given a query and trying to define an efficient method for solving it in a data-management context, access methods are used to speed up processing. The following discussion mentions some methods that, either on their own or by combining them, efficiently solve the k-RNN query-processing problem. When considering efficiency, we will also argue for the simplicity of a method, as the more complex a proposed access method is, the bigger is the challenge in implementing it in a given data management infrastructure.

Following this approach, we try to utilize spatial indexing methods with graph data structures. A regular *spatial grid* is used for indexing the locations of our POIs. In the *spatial grid*, the geographical coverage of the earth is divided into a set of equally sized grid cells. Each cell corresponds to a disk block storing the POI of the respective grid cell. Each cell is identified by a unique id, which is computed as a hash value of the cell's geographic coordinates. We maintain a hash table with key-value pairs for each grid cell, for which the key is the grid cell's id and the mapped value is a pointer to the actual block on the disk. In this way, we can locate the corresponding page on the disk in $O(1)$ time. This is also a reason for using a regular grid instead of other types of access methods.

We are using a grid instead of a more efficient spatial access method since it is easier to integrate it with a graph search. Using an R tree, for example, we would have to keep respective graph distance information with the nodes of the spatial index. As we will see later, this presents a challenge, as the *relative position* of nodes in the relevance graph (hops) affects the graph distance (see the recursive formula to compute the relevance score of Equation (3)). In addition, with newly discovered LOI information, the weights of the graph frequently change, thus requiring frequent updates to the index structure.

As we will show, k-NN queries can be processed incrementally by "radiating out" from the query point (discussed later), which is a data structure that serves as a simple and elegant way of showing how a spatial index can be combined with others to index, for example, space + relevance. Should a node reach its maximum capacity (fanout), an overflow node is added.

The *relevance graph* is defined as a graph $G(V, E)$, where V is the set of vertices that correspond to the POIs found in the set of documents, and E is the set of edges that correspond to LOIs between the POIs. The edge weights denote the relevance score r between a pair of vertices. Both indices are constructed incrementally. We consider that the data to be inserted concern one document/description at a time. Therefore, the input to the insert and update procedures are POIs and LOIs.

Regular Grid Update. The *spatial grid* consists of a set of cells, each cell corresponding to an index node and, thus, a disk page. Since we aim for having a one-to-one correspondence between nodes and pages, should a node reach its maximum capacity, an overflow node is added. The grid size is static and access is implemented using hashing. Thus, updates to the spatial grid are made in $O(1)$ time.

Relevance Graph Update. Updating the relevance graph involves introducing new vertices (new POIs), edges and edge weights (both based on new relevance). The relevance graph is stored by means of an adjacency list. Therefore, with the input of a list of points, as well as the identified LOIs, if a POI has not been previously added to the graph, we add it and add the possible relations to other POIs. Edges can be added in $O(1)$ time and the updates in $O(|E|)$ time.

4. *K*-RNN QUERY PROCESSING

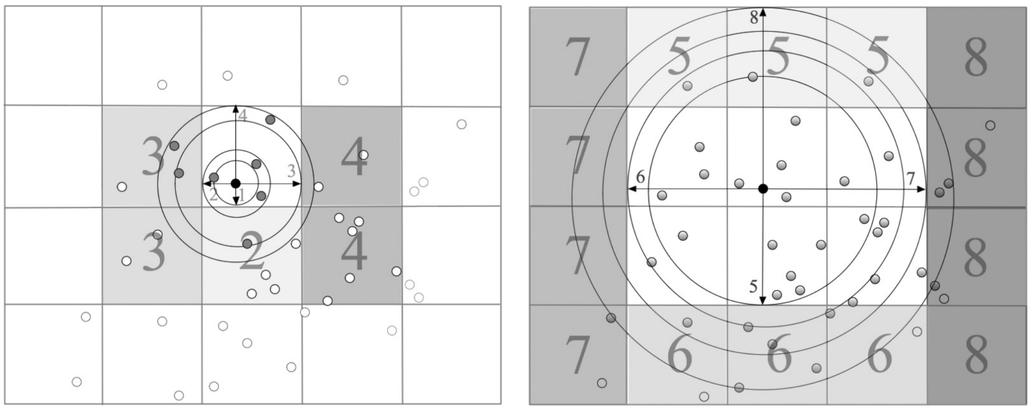
The major contribution of this work is how to combine the spatial grid and the relevance graph so as to efficiently support the processing of k-RNN queries. The following sections present various approaches to this integration, starting with the simple *GR-Sync* method.

4.1. Index Synchronization

In a first, basic query-processing method, termed *GR-Sync* (derived from Grid/Graph synchronization), we combine the results of the two separate indices to answer k-RNN queries. *GR-Sync* consists of two separate methods for identifying the k-RNN candidates in the spatial grid and in the relevance graph, respectively. The intuition behind this approach is an intermixed, stepwise execution of the search utilizing both indexes. After each step of the so-called *expansion process* in both data structures, the results (current list of respective k-NN neighbor candidates) are combined. The search stops when the neighbors found are guaranteed to be the k-RNN.

4.1.1. Spatial Search. The *spatial expansion* algorithm uses a spatial grid, as illustrated by Figure 2(a). For each inserted POI, we store four distances to the respective sides of the cell. Figure 3 shows an example of how the algorithm behaves for eight expansions beginning from a query POI. The algorithm first locates the grid cell of the query point.

Given that this algorithm tries to establish the relevance and proximity between POIs ("Where to go next?"), the query point is recruited from the set of POIs. If necessary, any user location can be mapped to the closest POI location.



(a) Steps 1-4 (b) Steps 5-8

Fig. 3. Spatial expansion.

The objective of the spatial expansion is to discover the nearest-neighbor POIs of the query point. We do so by retrieving close-by POIs in neighboring cells in a step-wise fashion. As we will see, this results in a *snail-like expansion*. The order in which the cells are retrieved depends on the location of the query point within its containing cell. Consider the example of Figure 3(a). The query point is closest to the bottom side of the cell (arrow labeled "1"). To guarantee that all POIs have been examined that are within distance 1, only the cell of the query point needs to be loaded (indicated also by the circle around the query point of radius "1"). However, for the case of distance "2," the points of the bottom cell (labeled 2) also need to be retrieved. This snail-like expansion next retrieves two cells labeled "3," then two cells labeled "4." While we also retrieve points that are farther away than d_4, we are also certain that we have not missed any candidates.

The points that are not within the maximum distance are added to a *retrievedPoints* list, whereas the points that are within the maximum distance are added to the k-NN list based on Euclidean distance. With increasing distance, the search expands to neighboring cells, as shown in the example of Figure 3(b). As expected, the larger the distance from the query point, the more points are retrieved in each step, for example, 4 cells in Step 8.

4.1.2. Computing the Relevance Score. To retrieve the *relevance score* s_r, we have to examine the relevance graph. Our method orients itself on the breadth-first graph traversal. It starts with the query point and in a first step examines all adjacent nodes in the relevance graph. Subsequent steps examine all neighbors of the initially visited nodes, and so on. To compute a relevance score, we use the following recursive formula that computes the score of a node k based on the score of its predecessor or parent node p. For the initial expansion $p = q$,

$$s_r(k) = 1 - \frac{w_{p,k}}{\sum_{i=1}^{N} w_{p,i}} + s_r(p) + s_h(k) \tag{3}$$

$$s_h(k) = s_h(p) + h(k), \tag{4}$$

where s_r is the *relevance score* for node k, $w_{p,k}$ is the weight from the parent node p to k, N is the number of the parent's one-hop neighbors, the number of nodes that are expanded during the same step, $s_r(p)$ is the parent's relevance score, and $s_h(k)$ is a score derived from $h(k)$, which is the number of hops needed to reach k from q. $s_h(k)$ is based

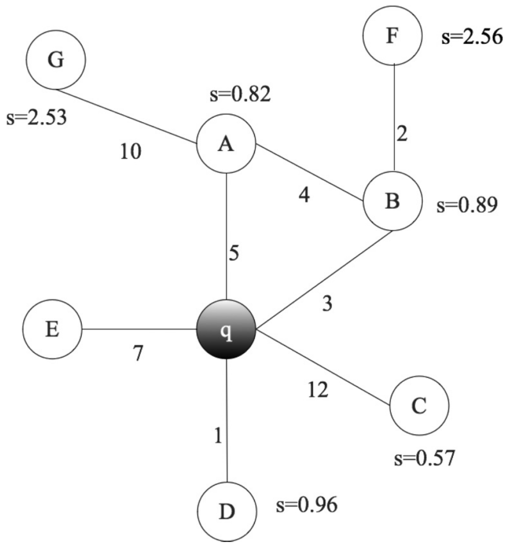


Fig. 4. Graph expansion and relevance score.

on the respective score of the parent node and increases with the distance of k from d. This also ensures that any node l with $h(l) > h(k)$ will have a higher relevance score than node k, that is, $s_r(l) > s_r(k) : h(l) > h(k)$. The recursive relevance score computation emphasizes that even the "worst" one-hop neighbor is preferable to the "best" two-hop neighbor. In terms of semantics, one-hop neighbors are POIs that co-occur at least once in a paragraph. Two-hop neighbors are POIs that co-occur with another POI only in a transitive, "friend-of-a-friend" fashion. As such, the more distant two POIs are in the relevance graph, the less obvious is a relevance relationship existing between them. As such, both the relevance score and the spatial score are a penalizing score, as they reflect distance; that is, the higher s_r, the less relevant a node k with respect to q.

What follows are some relevance-score computation examples. *Example 1*: Figure 4 depicts an example of the calculation of the relevance score for the neighbors of a query point. First, the sum of the weights of the edges that connect the query point to its neighbors is calculated. In this example, $\Sigma(0) = 28$. To compute each one-hop neighbor's relevance score s_r, we need to get the intermediate score of each neighbor node based on the edge weight, that is, $1 - (w_{p,k}/\sum_{i=1}^{N} w_{p,i})$. For example, node B has an intermediate score of $1 - 3/28 = 0.89$. Applying the rest of Equation (3), with $s_r(p) = 0$, since $p = q$ and $h = 0$ with the hop count starting at 0, $s_r(B) = 0.89$. Continuing the expansion, that is, retrieving the two-hop neighbors, for example, for node F, $s_r(F) = (1 - 2/6) + 0.89 + 1 = 2.56$ (the edge that connects the neighbor to its parent is not taken into consideration). It should be noted that, for computing the score of a node, we consider the shortest path as far as the number of hops from the query point is considered. Additionally, should a node be reachable by the same number of hops through multiple nodes, we consider the lowest scoring node as a parent node.

Example -: Using Figure 2 as a running example, we seek the 1-RNN neighbor of query point O_1. To compute the score of O_2 with respect to O_1, first, a spatial search is initiated on the spatial grid, and the spatial distance between O_1 and O_2 is found to be $s_d(O_1, O_2) = 350m$. Expanding O_1 on the relevance graph, we use Equation (3) to calculate the relevance score of O_2, which is 0.32. O_2 is 1 hop away from O_1, thus $s_h(O_2) = 0$, and its parent's relevance score is $s_h(O_1) = 0$ (by default). With $\alpha = 0.5$ (equal preference for relevance and spatial distance), $s_r.max = 2$, and $s_d.max = 600m$,

Table I. Calculation of Relevance Score Based on Figure 2 Example

O	$s_h(k)$	$s_r(k)$	$s_d(Q, O)$	s_{rnn}
O_2	0	0.32	350	**0.37**
O_3	0	0.89	200	0.39
O_4	0	0.96	100	0.40
O_5	0	0.93	480	1.03
O_6	1.96	1.96	475	1.28
O_7	0	0.88	550	1.13
O_8	1.32	1.32	600	1.33

Note: O_2 is the 1-RNN of O_1 with combined score $S_{rnn} = 0.37$.

the combined score between O_1 and O_2 is $s_{rnn} = 0.37$ (calculated using Equation (2)). Computing the s_{rnn} scores for all other objects with respect to O_2, as shown in Table I, we can see that indeed O_2 is the 1-RNN of O_1.

4.1.3. Synchronizing Expansion. To compute the k-RNN score, the *spatial and the relevance graph search need to be synchronized* and their scores combined. The following approach computes combined scores and evaluates the status of the search at *fixed intervals* determined by the number of expansions performed in each index. The spatial grid utilizes a snail-like expansion and retrieval of cells surrounding the query point; the relevance graph uses a BFS-like expansion of increasing distance to the query point. After each expansion step, the two lists contain the closest in terms of spatial and relevance-score neighbors, respectively. To synchronize the two expansions, we define a respective rate of expansion steps. The expansion ratio χ determines the ratio of spatial to relevance-graph expansions. χ will typically be in the range of 4 to 16, as spatial expansions are considerably cheaper (read accesses). Each search maintains a list of expanded POIs and their respective score. After each expansion cycle (considering the expansion ratio), the two lists are checked and the common POIs, that is, appearing in both lists, are identified. Using Equation (2), their combined score s_{rnn} is computed. All POIs with such a score are added to a queue sorted by s_{rnn} and essentially containing all top k-RNN neighbors that have been identified at this point, that is, POIs that have been examined in both data structures.

To define a termination criterion, we have to guarantee that all k-RNN POIs have been found, that is, further search will not reveal any POIs with a better score than the ones already identified. Each search keeps an *open list* for each set of respective POIs found so far to record the corresponding k-RNN score. If a POI is found in only one index, to compute its score, we use a *best-case estimate for the missing score*, that is, that it will be found during the next expansion. For example, assuming that a POI was retrieved in the spatial search, but not yet in the relevance search, the relevance score will be the lowest possible score after the next expansion (the relevance score is dominated by number of hops = expansion steps). Similarly, should the spatial score be missing, we assume the best case, that is, that the POI will be discovered during the next round of expansion with a distance from the query point just beyond the current search distance. As the searches progress, the scores of the POIs, thus the open lists, will be updated based on the current number of hops and search distance. The *GR-Sync* algorithm is shown in Algorithm 1. *SpatialExpansion*$(q, k, (n \times \chi), SG, sg)$ takes as input the query point q, the number of sought neighbors k, the current expansion radius $(n \times \chi)$, the spatial grid SG, and the *sg* list that contains the discovered points so far by the spatial-expansion algorithm (see Section 4.1.1). *RelevanceExpansion*(q, k, n, RG, rg) takes as input the query point q, the number of sought neighbors k, the relevance graph step count n, the relevance graph RG and the *rg* list that contains the discovered points so far by the relevance-expansion algorithm (see Section 4.1.2). *GR-Sync* uses s^d_{rnn} and s^r_{rnn}

ALGORITHM 1: GR-Sync k-RNN Algorithm

```
   Input: Query POI $q$, Number of Neighbors $k$, Relevance Graph $RG$, Spatial Grid $SG$,
          Discovered POIs in $RG$: $rg$, Discovered POIs in $SG$: $sg$, $k$-RNN result list $rnn$,
          Relevance Graph step count $n$, Expansion ratio for Spatial Grid $\chi$.
   Output: Complete $k$-RNN result list $rnn$.
1  $complete$ = false;
2  while $\neg$ $complete$ do
3      $SpatialExpansion(q, k, (n \times \chi), SG, sg)$;
4      $RelevanceExpansion(q, k, n, RG, rg)$;
5      $rnn = sg \cap rg$ NN results with complete score;
6      $s^r_{rnn} = MinScore(rg)$ Min. predicted score, Relevance Graph;
7      $s^d_{rnn} = MinScore(sg)$ Min. predicted score, Spatial Grid;
8      $s'_{rnn} = Min(s^d_{rnn}, s^r_{rnn})$ Min. expected score;
9      $s^*_r = MaxScore(rnn)$ Max. score in current result list;
10     If the max $k^{th}$ RNN distance found so far is less than the min predicted distance, the
       algorithm completes;
11     $complete = (|rnn| \geq k \wedge s^*_r < s'_{rnn})$;
12 end
```

as the minimum predicted combined k-RNN scores for nodes with no valid spatial or relevance score, respectively. If $s^*_{rnn} = min(s^d_{rnn}, s^r_{rnn})$, the minimum score that still could be found, is less than $max(s_{rnn})$, that is, the maximum score of identified rnn candidates, it means that the POIs in the (top-k) result list are guaranteed to be the top k-RNN neighbors (Line 11). This condition can be used as the termination criterion.

4.2. Index Linking

The k-RNN query-processing algorithm presented so far does not actually combine the two indices (spatial and relevance), but only evaluates the results at times with the expansion steps used as a means of synchronization. An inherent problem with this method is the search in the relevance graph, which, after the first hops, becomes very costly. Big expansions in the relevance graph (empirically observed for hops > 3) retrieve a lot of data, thus incur disk activity. We have devised a variation of our query processing technique that manages to *bound the relevance graph expansion*. This so-called *GR-Link* method, termed as such since it combines the two indexes (grid/graph linking), we use the results of the spatial search as seed elements for the search in the relevance graph. The intuition is that many 1-hop expansions (spatial search seeds) are "cheaper" than a single n-hop expansion of the query point. POIs retrieved by the spatial search (and not discovered yet in the relevance search) are expanded in the relevance graph (see Figure 5). The intuition is that the graph searches of (i) the seeded POIs and (ii) the query point will meet eventually, thus we can compute a POI's relevance score. Without *seeding the search*, we would have to wait until the query point expansion reaches a POI. The simplified example of Figure 5 should illustrate that the POIs expanded are much fewer than the POIs that would have been retrieved if a second expansion step from the query point would have been performed.

The *GR-Link* approach also allows us to better bound the estimates of missing scores. Placing a POI on the relevance graph, we know for sure that if their expansions (POI and query point) do not meet, the base for their score computation is at least the sum of the two expansions plus one hop (since they did not meet). Therefore, the predicted s_r score for such a point is larger than what it would have been in the nonhybrid case.

The pseudo-code for this method is shown in Algorithm 2. The difference from the *GR-Sync* algorithm is the statements of Lines 4 to 8, which are seed POIs for the relevance search.

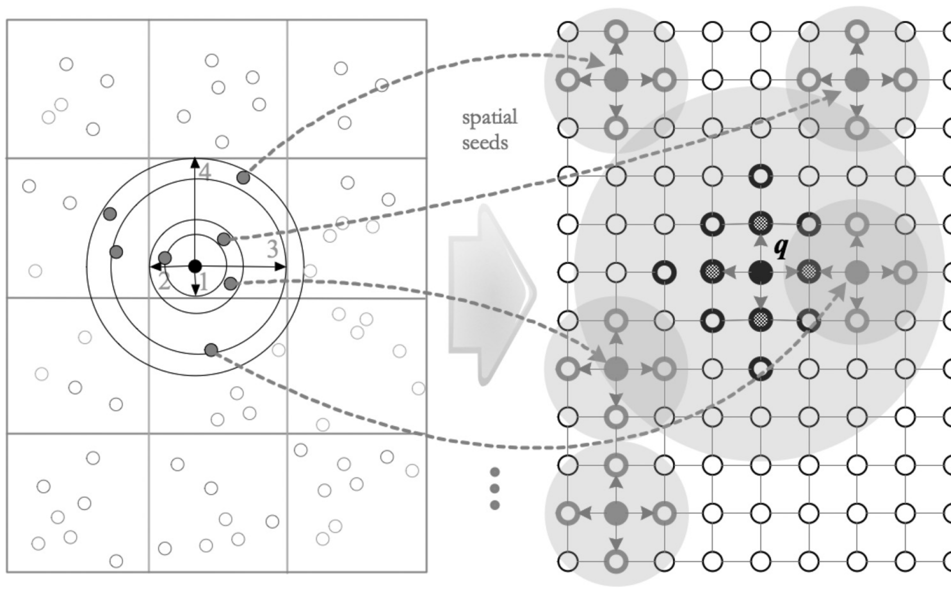


Fig. 5. *GR-Link* method using grid POIs in graph search.

ALGORITHM 2: *GR-Link* *k*-RNN Algorithm

```
Input: Query POI $q$, Number of Neighbors $k$, Relevance Graph $RG$, Spatial Grid $SG$
       Discovered POIs in $RG$: $rg$, Discovered POIs in $RG$: spatial seeds: $rg'$, Discovered
       POIs in $SG$: $sg$, $k$-RNN result list $rnn$, Relevance Graph step count $n$, Expansion
       ratio Spatial Grid $\chi$.
Output: Complete $k$-RNN result list $rnn$.
1  $complete$ = false;
2  while $\neg$ $complete$ do
3      $SpatialExpansion(q, k, (n \times \chi), SG, sg)$;
4      for $each\ poi \in sg$ do
5          $RelevanceExpansion(poi, 1, RG, rg')$;
6      end
7      $RelevanceExpansion(q, n, RG, rg)$;
8      $rg = Connect(rg, rg')$ Combine graph expansions;
9      $rnn = sg \cap rg$ NN results with complete score;
10     $s^r_{rnn} = MinScore(rg)$ Min. predicted score, Relevance Graph;
11     $s^d_{rnn} = MinScore(sg)$ Min. predicted score, Spatial Grid;
12     $s'_{rnn} = Min(s^d_{rnn}, s^r_{rnn})$ Min. expected score;
13     $s^*_r = MaxScore(rnn)$ Max. score in current result list;
14     If max $k^{th}$ RNN distance found so far is less than min predicted distance, the algorithm
       completes;
15     complete = $(|rnn| \geq k \wedge s^*_r < s'_{rnn})$;
16 end
```

As the experimental section will show, the intuition of choosing many small relevance-graph expansions over one large expansion pays off and the *GR-Link* method shows superior performance in terms of IO when compared to the *GR-Sync* solution.

5. *K*-RNN QUERY OPTIMIZATION

Having introduced the k-RNN problem and several processing algorithms, in the following, we introduce two novel methods that provide improved query processing by optimizing the search in the relevance graph. The methods involve *landmark-based distance estimation* and the *ALT* algorithm for additional pruning of the search space.

5.1. Landmarks

Landmark-based distance estimation in graphs is a method that involves the preselection of a set of landmark nodes and computing the distances of all graph vertices from those landmarks. Given two arbitrary graph vertices, the distance between them can be estimated based on the respective distances from landmarks, as follows. Given a set $S \subseteq V$ of landmarks and distances $d(L_i, v)$, $d(v, L_i)$ for all nodes $v \in V$ and landmarks $L_i \in S$, the following triangle inequalities hold: $d(u, v) + d(v, L_i) \geq d(u, L_i)$ and $d(L_i, u) + d(u, v) \geq d(L_i, v)$. Therefore, the function $max_{L_i} max\{d(u, L_i) - d(v, L_i), d(L_i, v) - d(L_i, u)\}$ provides a lower bound for the distance $d(u, v)$.

For the specific case of k-RNN queries, we will utilize landmarks for the relevance-graph search to estimate the distance of seeded POIs to the query point, and thus minimize the necessary expansions in the relevance graph. The necessary preprocessing stage is divided into two phases: the landmark selection process and the computation of distances from landmarks to all other graph vertices (for undirected graphs).

As far as the landmark selection process is concerned, many alternative strategies have been suggested in Goldberg and Harrelson [2005], Goldberg and Werneck [2005], and Potamias et al. [2009]. As Delling and Wagner [2007] suggest, no technique picks landmarks that universally yield the smallest search space for random queries (although some perform better than others). Therefore, in our work, we utilize the simplest strategy, which is the *farthest* landmark selection strategy introduced in Goldberg and Harrelson [2005]. In a nutshell, the algorithm randomly selects a starting vertex and performs a Dijkstra search. Upon termination, the whole graph has been expanded, producing shortest paths between the start and all other vertices of the graph. Choosing now the most distant vertex, it becomes the first landmark. Continue the search by finding at each step a new vertex that is farthest away from the current set of landmarks. Stop when n landmarks have been added to the set of landmarks L. The intuition behind this procedure is to choose landmarks that are at the periphery of the graph.

We also implemented a custom simplified alternative strategy, to have a clear view of the impact of the landmark selection process on our results. Our new method, referred to hereafter as the *partitioning/highest-degree* method initially partitions the relevance graph. For each partition, the node of the highest degree is added to the landmarks set. To partition the relevance graph, we used METIS [Karypis and Kumar 1998] a free, well-known graph-partitioning tool, used often in the context of shortest-path computation [Köhler et al. 2006; Maue et al. 2010; Efentakis et al. 2011, 2012]. Thanks to METIS partitioning, our partitioning/highest-degree method guarantees a uniform distribution of landmarks throughout the relevance graph. Picking the highest-degree node for each partition is also a natural choice since, as Potamias et al. [2009] suggest, "the more connected a node is, the higher the chance that it participates in many shortest paths." Following the selection of landmarks L, we compute the distance from the vertices of the graph to the landmarks. As such, these landmark distances will provide us with a *relevance score s_r estimate* for each spatial seed.

To simplify the approach, we consider the *relevance graph G* to be unweighted and undirected. This is a simplification over the approach presented so far, since the focus

ALGORITHM 3: *GR-Link-LM* k-RNN Algorithm

```
Input: Query POI $q$, Number of Neighbors $k$, Relevance Graph $RG$, Spatial Grid $SG$
       Discovered POIs in $RG$: $rg$, Discovered POIs in $RG$: spatial seeds: $rg'$, Discovered
       POIs in $SG$: $sg$, $k$-RNN result list $rnn$, Relevance Graph step count $n$, Expansion
       ratio Spatial Grid $\chi$.
Output: Complete $k$-RNN result list $rnn$.
1  $complete$ = false;
2  while $\neg$ $complete$ do
3      $SpatialExpansion(q, k, (n \times \chi), SG, sg)$;
4      for $each\ poi \in sg$ do
5          $LandmarkExpansion(poi, q, RG, L, D, rg')$;
6      end
7      $RelevanceExpansion(q, n, RG, rg) rg = Connect(rg, rg')$ Combine graph expansions;
8      $rnn = sg \cap rg$ NN results with complete score;
9      $s^r_{rnn} = MinScore(rg)$ Min. predicted score, Relevance Graph;
10     $s^d_{rnn} = MinScore(sg)$ Min. predicted score, Spatial Grid;
11     $s'_{rnn} = Min(s^d_{rnn}, s^r_{rnn})$ Min. expected score;
12     $s^*_r = MaxScore(rnn)$ Max. score in current result list;
13     If the max $k^{th}$ RNN distance found so far is less than min predicted distance, the
       algorithm completes;
14     complete = $(|rnn| \geq k \wedge s^*_r < s'_{rnn})$;
15 end
```

is on the efficiency of the graph traversal and not on the relevance calculation *per se*. Another modification in relation to the graph dataset is that now the graph needs to be fully connected, since landmark-based approaches do not apply to unconnected graphs.

Following landmark selection, it suffices to precompute the graph (hop) distances from each landmark to all graph nodes by running a breadth-first search from each landmark. This information is captured by a two-dimensional distance matrix D of size $N * M$, where N is the total number of vertices and M the number of landmarks used. Therefore, D_{ij} refers to the hop distance from landmark j to vertex i.

The upper- and lower-bound distances to landmarks can now be used as estimates for the true distance between any two graph vertices. Given a set of landmarks L and two vertices u and v whose distance we need to determine for undirected, unweighted graphs, the following equation holds [Potamias et al. 2009]:

$$max_{L_i} |d(L_i, v) - d(L_i, u)| \leq d(u, v) \leq min_{L_i} |d(L_i, v) + d(L_i, u)|. \tag{5}$$

Therefore, a good approximation for the lower bound of the graph distance $d(u, q)$ between any vertex u and the query point q in the relevance graph is as follows:

$$d(u, q) \geq max_{L_i} |d(L_i, u) - d(L_i, q)|. \tag{6}$$

Having a way to approximate the distance between a vertex and a query point in the relevance graph, we can prune the graph search space based on this approximation. In Algorithm 3, the *GR-Link-LM* algorithm is shown, which is an expansion of the *GR-Link* algorithm for the computation of the k-RNN query.

The algorithm works just like the *GR-Link* approach, with the difference that, for the expansion of the spatial seeds, the routine *LandmarkExpansion* (Line 5) is used. More specifically, after a number of χ expansions are made in the spatial index, a set rg' of POIs has been discovered. These POIs serve as spatial seeds in the relevance graph search (see *GR-Link* approach). At this point, instead of initiating a “blind” BFS traversal of the spatial seeds, we use the *LandmarkExpansion* algorithm in order to facilitate the use of landmark lower bounds, and therefore improve the performance of

ALGORITHM 4: Landmark-Based Expansion Algorithm

```
Input: The spatial seed to be expanded $s$, the query POI $q$, Relevance Graph $RG$, Set of
       landmarks $L$, Distance matrix from landmarks to all nodes $D$.
Output: Discovered POIs in $RG$ - spatial seeds: $rg'$
1  Chosen landmark $L_i = -1$;
2  Lower-bound $d_{approx} = -1$;
3  $L_i = selectLandmark(q, s, L, D)$;
4  $d_{approx} = computeLowerBound(L_i, q, s, D)$;
5  if $\neg worthExpanding(s, L, d_{approx})$ then
6      return;
7  end
8  while $\neg$ completedHopExpansion() do
9      $BFSexpansion(s.BFSqueue, s, q)$;
10 end
```

the graph traversal. Algorithm 4 shows the algorithm for the expansion of the spatial seeds.

Let s be the spatial seed node that is to be expanded in the relevance graph G, q is the query POI, and L_i is a landmark node. For the spatial seed s, we calculate the lower-bound $max_{L_i}|d(L_i, s) - d(L_i, q)|$ using Equation (6). If the calculated lower bound for s is greater than the actual graph distances of the current k-RNN results, then the spatial seed s could be safely ignored, that is, it is too *far away* from the query point in the relevance graph to ever become a viable solution. With this simple but crucial optimization, we efficiently prune the spatial seeds that need to be expanded, thus significantly lower the number of I/O operations. This, of course, results in improved performance (see Section 6), while still guaranteeing the optimality of results.

An example of an k-RNN query with landmarks is shown in Figure 6. In comparison to Algorithm 2, the spatial search portion is omitted. The example shows five spatial seeds. Before each of the seeds is expanded, its lower-bound distance to the query point is calculated based on the landmark information. In this example, two spatial seeds are not expanded further since their landmark distance is above the threshold for them to become viable k-RNN candidates. The remaining three seeds are expanded. This example illustrates that, by eliminating nodes, the expanded graph portion becomes smaller. It is expected that the *GR-Link-LM* method thus shows improved performance.

5.2. ALT Algorithm Optimization

The presented k-RNN method combines *central* BFS traversal from the query point q with the individual BFS traversals originating from the seed points. In this section, we further optimize this process by using the lower bounds of the landmark preprocessing phase to exclude spatial seeds that cannot belong to the k-RNN set. *GR-Link-LM* traverses the graph from the seed points by using plain BFS in an undirected fashion. This is also illustrated by the gray circles in Figure 6. In what follows, we augment the seed expansions by using a unidirectional version of the ALT algorithm of Goldberg and Harrelson [2005]. The ALT algorithm is a well-known goal direction, shortest-path technique that combines A* search [Hart et al. 1968] with the lower bounds provided by the landmark preprocessing phase. Combining the *central* BFS traversal from the query point q and the individual unidirectional ALT algorithms from the seed points requires several additional enhancements. For that purpose, we adapt several core concepts from Goldberg and Harrelson [2005]:

—For each unidirectional ALT algorithm originating from a seed point, we must maintain the shortest path length μ seen so far between the seed and the query point.

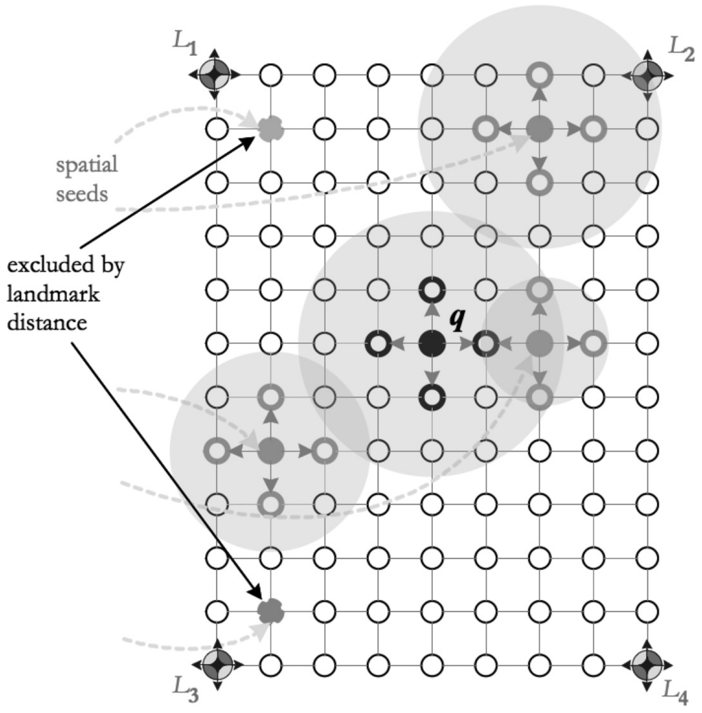


Fig. 6. *GR-Link-LM*method using landmark nodes on the graph periphery.

Initially, for each such search, μ is set to infinity. When an edge (u, w) is scanned by the ALT algorithm and w has already been scanned in the BFS traversal from the query point, we know that the shortest paths for $s - u$ and $w - q$ are of lengths $d_s(u)$ and $d_q(w)$, respectively. If $\mu \geq d_s(u) + l(u, w) + d_q(w)$, we have found a shorter path than those seen before, so we update μ accordingly.

—The unidirectional ALT algorithm cannot terminate as soon as it scans a node already scanned by the *central* BFS traversal from the query point. Instead, we can safely abort the search only provided the ALT algorithm is about to scan a vertex v with $k(v) = d_s(u) + max_{L_i} |d(L_i, u) - d(L_i, q)| \geq \mu$.

Another essential difference between the *GR-Link* method and the current proposal is the way that we alternate between the two opposing searches (the central BFS traversal from the query point and searches originating from the seeds). For the *GR-Link* method, the alternation strategy was measured in hops, that is, we performed one-hop BFS expansion from the query point and one-hop BFS expansion from each seed. Although we still use a one-hop BFS expansion from the query point, the ALT search from the seeds can no longer be measured in hops. Instead, we keep track of the number of nodes extracted from the priority queue per seed search. Once the set number of nodes has been taken off the priority queue of the ALT search for a particular seed, we then proceed to the next seed. Once all seeds perform the necessary number of node extractions from their priority queues, we perform another one-hop BFS expansion from the query point and proceed in this iterative fashion. Algorithm 5 presents the details.

An example of a k-RNN query using the ALT optimization is shown in Figure 7. The decision as to whether or not the spatial seed, or the subsequently discovered nodes,

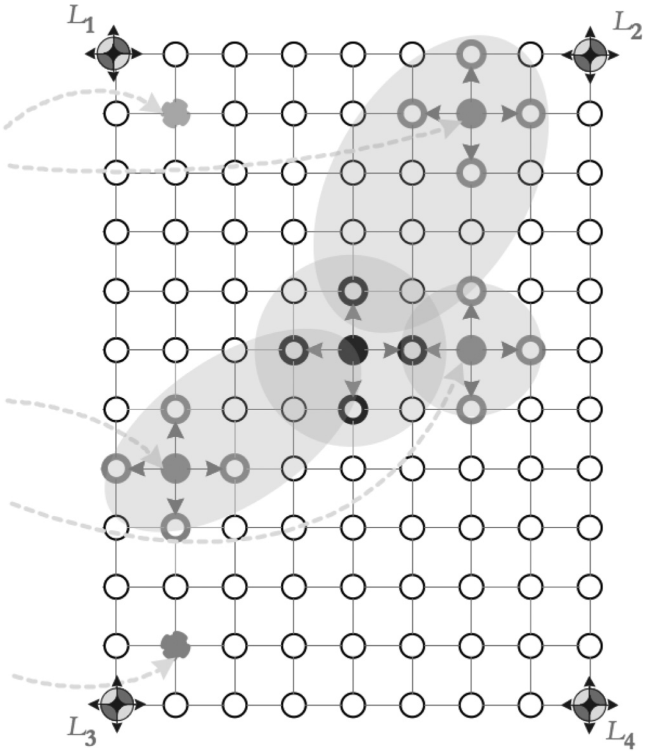


Fig. 7. *GR-Link-LM* * method: ALT algorithm significantly optimizes the relevance-graph search.

ALGORITHM 5: The Unidirectional ALT Algorithm Optimization

```
Input: The spatial seed to be expanded $s$, the query POI $q$, Relevance Graph $RG$, set of
       landmarks $L$, distance matrix from landmarks to all nodes $D$, $ALT$ expanded nodes
       $\sigma$.
Output: Discovered POIs in $RG$ - spatial seeds: $rg'$.
1  Chosen landmark $L_i = -1$;
2  Lower bound $d_{approx} = -1$;
3  $L_i = selectLandmark(q, s, L, D)$;
4  $d_{approx} = computeLowerBound(L_i, q, s, D)$;
5  if $\neg worthExpanding(s, L, d_{approx})$ then
6      return;
7  end
8  while $\sigma \neq 0$ do
9      $A * expansion(s.ALTqueue, s, q)$;
10     $\sigma = \sigma - 1$;
11 end
```

will be further expanded is taken based on the landmark lower bounds. Additionally, since the *ALT* algorithm uses the landmark-based lower bounds, we observe that nodes closer to the query point q are expanded first, allowing the algorithm to find a common node with the query point BFS expansion sooner, and therefore calculate the spatial seed's relevance score way before all nodes will be expanded.

The intuition behind the *GR-Link-LM** approach is shown in Figure 8. The spatial seed s, using the *ALT* algorithm, expands toward the query POI q. In this way, it will

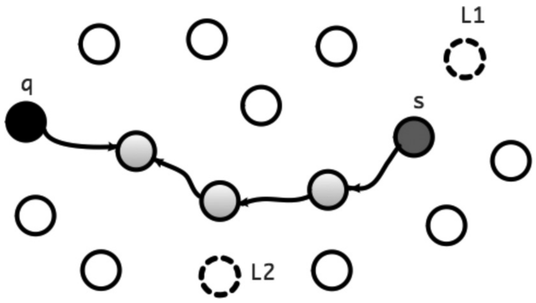


Fig. 8. Intuition for A*/ALT search: Prune the graph space and go faster from *s* toward the query POI's *q* expansion.

retrieve a smaller number of relevance-graph nodes until it reaches *q*'s central BFS expansion. The overall impact of this approach is visualized by the ellipses, rather than the circles, of the *GR-Link-LM* case in Figure 7. Using this directed search, a smaller portion of the relevance graph is expanded, overall resulting in improved performance.

The following section will examine the performance of the proposed algorithms with respect to existing approaches.

6. EXPERIMENTAL EVALUATION

We assess the efficiency of the proposed *k*-RNN query processing methods in a performance study measuring accessed data (disk I/O operations) and CPU time. The experiments use real and synthetic datasets. Overall, we compare five methods: (i) the *GR-Sync* method (naïve method), (ii) the *GR-Link* method, (iii) the *GR-Link-LM* method, (iv) the *GR-Link-LM** method and (v) a hypothetic ideal method (explanation provided later). What we expect from the experiments is to see a well-performing *GR-Link* method as well as a boost in performance from the *GR-Link-LM* and *GR-Link-LM** methods, which come close to the performance of the ideal method.

6.1. Data

Actual POI + LOI data were collected from a corpus of 120k documents from UGC found in travel blog sites[1]. The texts were preprocessed to collect the necessary information for our index: (i) identification of POIs, (ii) geocoding of POIs (spatial position), and (iii) location within the document (paragraph id and offset).

The procedure of generating the realistic dataset used in this work follows simple heuristic rules derived from the characteristics of the real-world data. Essentially, the data-generation approach is based on hotspots that resemble cities, related POIs, then LOIs between the POIs. We generate center points (cities, attractors) in an area of 30×30 degrees (approximately $3300 \times 3300km$) using a uniform distribution. For each center point, using a normal distribution, we generate neighboring points (POIs). We then generate relationships (LOIs) between POIs, using again a normal distribution, that is, the neighbors that are spatially closer to a POI in question are more likely to be linked to it than the ones farther away (Tobler's spatial bias). On average, we generate 15 LOIs per POI (maximum = 50). Our synthetic dataset consists of approximately 810k POIs that generate a relevance graph with a total of 6 million edges (LOIs). The synthetic data is considerably larger than the relevance graph derived from travel blogs, thus should produce more conclusive results. Figure 9 visualizes the major nodes and a closeup view of the links between nodes.

[1]www.travelblog.com, www.traveljournal.com, www.travelpod.com.

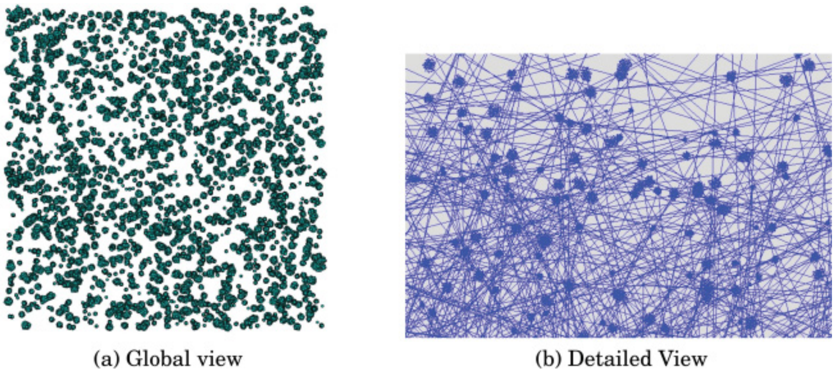


Fig. 9. Synthetic dataset.

6.2. Experimental Setup

We evaluate the various query-processing methods not only by means of different datasets, but also under a varying set of parameters, including (i) the number of k and (ii) the preference parameter α emphasizing either spatial distance or relevance. The performance is assessed in terms of number of page accesses (I/O) performed to retrieve the data from the disk for each query as well as CPU time. In each case, we performed 100 queries and computed the average I/O and CPU time shown in the respective charts. The 100 query points were randomly picked from the POIs in the dataset. An important aspect in the experiments is the spatial grid used to index the data. We use a regular grid with a spacing of 0.02 degrees (approximately $2km$) in longitude and latitude. The synthetic dataset consists of approximately 810k POIs covering an area of 30 degrees longitude and latitude, respectively, an extent somewhat comparable to Europe. The 810k POIs are grouped into 160k cells amounting to an average space utilization of 5, but not exceeding 30. Keep in mind that we consider only occupied cells in our index.

The real dataset contains 120k points that are scattered all over the globe. Here, 80k cells contain at least one POI. This amounts to an average space utilization of < 2, with few cells containing more than 5 POIs. Essentially, we used this dataset as a template for synthetic data generation. Still, we wanted to present the results of the performance study to see whether the performance trends with respect to the indexing methods persist.

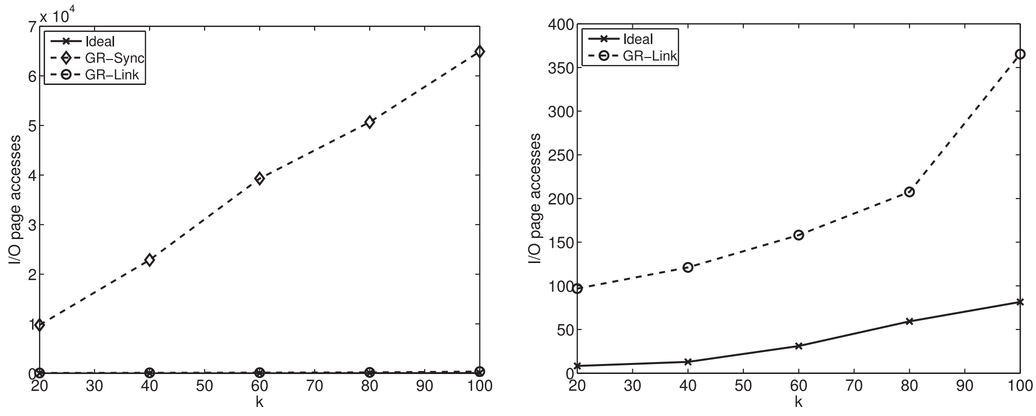
The expansion ratio χ was set to 12, that is, for each expansion in the relevance graph, 12 expansions in the spatial grid are performed. Unless mentioned otherwise, the parameter α used in Equation (2) to compute the k-RNN score s_{rnn} is fixed to 0.5, that is, considering the spatial and relevance score equally important.

All experiments were performed using a computer with an Intel Core i5 2400 CPU and 8GB DDR3 RAM, running Ubuntu Linux 11.10. The index and algorithms were implemented in Java.

6.3. Basic *k*-RNN Query Performance

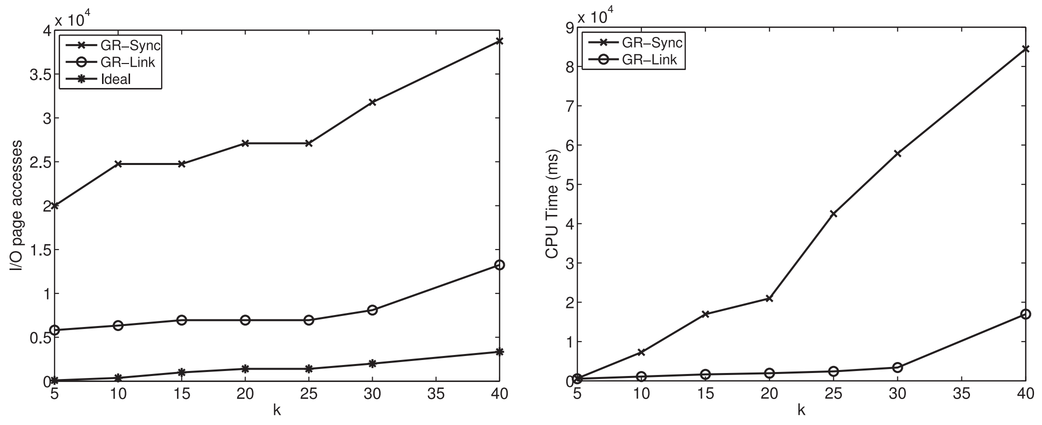
In our experimental setup, we first vary the number of sought nearest neighbors k to see how scalable our algorithms are. α is set to 0.5, considering the spatial and relevance score equally important.

The first experiment should verify the performance advantage of the *GR-Link* approach over the naïve *GR-Sync* algorithm. Figure 10(a) shows that *GR-Link* outperforms *GR-Sync* by an order of magnitude. This is due to the fact that *GR-Sync* needs



(a) *GR-Sync* vs. *GR-Link* vs. Ideal (b) *GR-Link* vs. Ideal

Fig. 10. Naïve, hybrid and ideal index I/O performance.



(a) I/O performance (b) CPU Performance

Fig. 11. Real dataset: varying *k*.

to search large parts of, both the spatial grid and the relevance graph in order to guarantee the result. Therefore, when POIs are found in one of the two indices, it needs to keep searching the other to find the combined *k*-RNN score. This causes the algorithm to retrieve too many (irrelevant) pages. On the other hand, the *GR-Link* method uses the results of the spatial search to limit the expansions in the relevance graph.

While a comparison to the naïve *GR-Sync* approach does not pose a challenge, the next experiment relates the performance of *GR-Link* to an "ideal" approach (see Figure 10(b)). The ideal approach simulates a relevance graph search that terminates as soon as the *k*-RNN POIs are found, that is, we have *a priori* knowledge of the results and are expanding the graph only until the result is "discovered." As such, this method is unrealistic, but serves as a lower bound for the performance of the hybrid index. Figure 10(b) shows that the *GR-Link* method examines more data than the ideal approach (albeit little data overall). Still, in terms of comparison to this baseline approach, the hybrid index's performance is close to that of the ideal method.

Using a real-world dataset (see Figure 11), the number of page accesses as well as CPU time when compared to the experiments with the synthetic dataset appear to be

Table II. Overhead for the Number of Landmarks

Number of Landmarks	Overhead (MB)
1	3.08
2	6.16
4	12.33
8	24.66
12	36.96
16	49.28

orders of magnitude greater due to a sparse dataset. Overall, however, this experiment shows the same trends observed for synthetic data.

6.4. Tuning Landmark-Based Optimizations

The naïve *GR-Sync* approach performs orders of magnitude worse than *GR-Link*. We thus excluded it from the following experiments, which establish an optimal setting for (i) the number of landmark nodes N, (ii) the choice of a landmark selection algorithm, (iii) the number of expanded nodes during the *ALT* traversal for each expanded spatial seed, and (iv) the number of $k-$RNNs.

An important parameter is the number of *ALT* expansions σ per spatial seed for all numbers of sought neighbors k. As explained in Section 5.2, for every step of the main query point BFS expansion, for each of the spatial seeds, the *ALT* algorithm expands a fixed number of σ nodes in the graph search. This is not the case for BFS since there we can ensure that each expansion step is, in fact, a one-hop expansion. We also perform experiments varying the number N of landmarks, as this affects the performance of the *GR-Link-LM* and *GR-Link-LM** algorithms. Both proposed landmark selection algorithms, that is, using the *farthest* as well as the *partitioning / highest-degree* landmarks, are assessed.

The disk space overhead for the *GR-Link-LM* method includes the distance table D from the landmark nodes to all other nodes in the graph. The overhead for each number of landmarks is shown in Table II. The size of the database containing the data needed to build the indices (POI and LOI information) is 74.96MB.

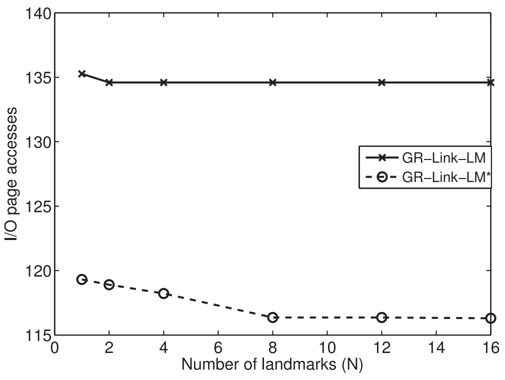
What follows are experiments relating to the various parameters affecting the k-RNN query performance.

6.4.1. Varying Landmarks. Seeking $k = 60$ results, Figure 12 shows that the *GR-Link-LM* and *GR-Link-LM** algorithms perform best with $N = 8$. Choosing more landmarks provides a better lower-bound estimate (to points to "triangulate" to). However, increasing the number of landmarks to $N > 8$ does not provide tighter distance bounds, thus no further performance gain.

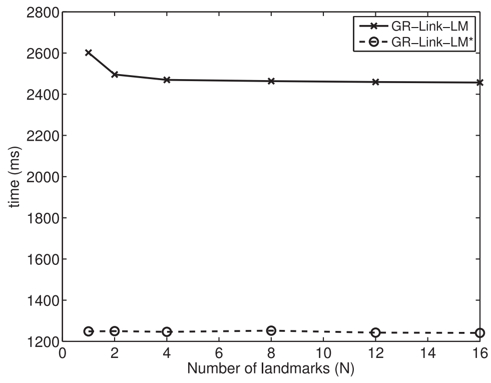
Using a different landmark selection algorithm, the *partitioning / highest-degree* method, Figure 13 shows a performance similar to that of the *farthest* landmark selection method, that is, reaching $N = 8$, we have the best performance.

The *farthest* method precomputes the distances between the landmarks and all other nodes in the graph in one step. The *partitioning / highest-degree* method requires extra preprocessing steps (i) to partition the graph (METIS) and then (ii) to select the high-degree nodes from each of the partitions. Therefore, the preprocessing cost using the *farthest* landmark selection method is lowest. Since both methods have comparable performance, we chose the *farthest* method for subsequent experimentation. Also, the number of landmark nodes will be fixed to $N = 8$.

6.4.2. ALT Optimization. The *GR-Link-LM** algorithm uses an *ALT* approach for the expansion of the spatial seeds (A* combined with landmarks). Using stepwise expansion, we need to determine the number of the *ALT* expanded nodes σ in each *GR-Link-LM**

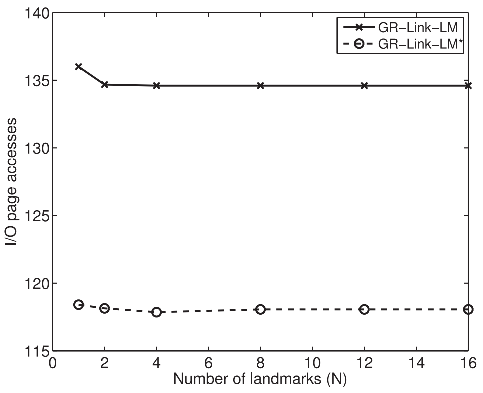


(a) I/O Performance

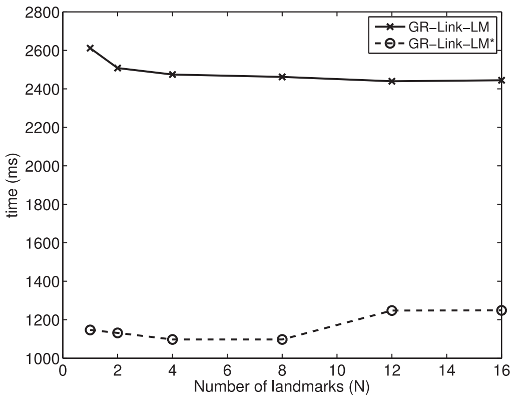


(b) CPU Performance

Fig. 12. Varying the number of landmark nodes N, I/O, and CPU performance. Landmarks are selected using the *farthest* method ($k = 60$).



(a) I/O Performance



(b) CPU Performance

Fig. 13. Varying the number of landmark nodes N, I/O, and CPU performance. Landmarks are selected using the *partitioning*/*highest-degree* method ($k = 60$).

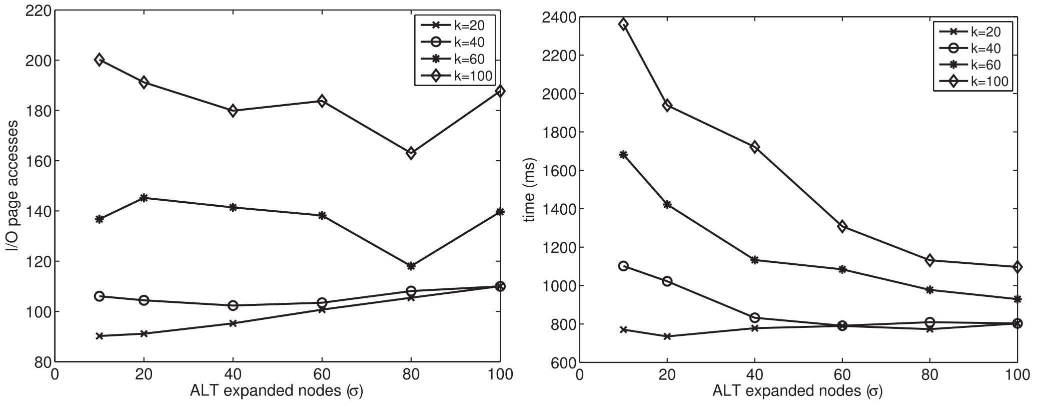
step for each spatial seed. Figure 14 shows the performance for varying σ and k in terms of I/O disk page accesses.

For smaller k, the optimal σ is lower, whereas for larger k, the optimal σ increases. The best choice seems to be $\sigma = 80$. For smaller ks, the main query point expansion reaches in a few steps the spatial seeds (not a lot of data to look for). In this case, performing extra node expansions for the spatial seeds incurs additional disk page accesses without any benefit (the solution was found already). However, for larger ks, in which case the *GR-Link-LM** algorithm needs to search larger portions of the graph to guarantee the top k-RNN, expanding more nodes at each *ALT* step results in fewer node accesses. The graph search is directed toward the query point expansion with a larger "momentum."

6.5. Query Performance for Varying *k*

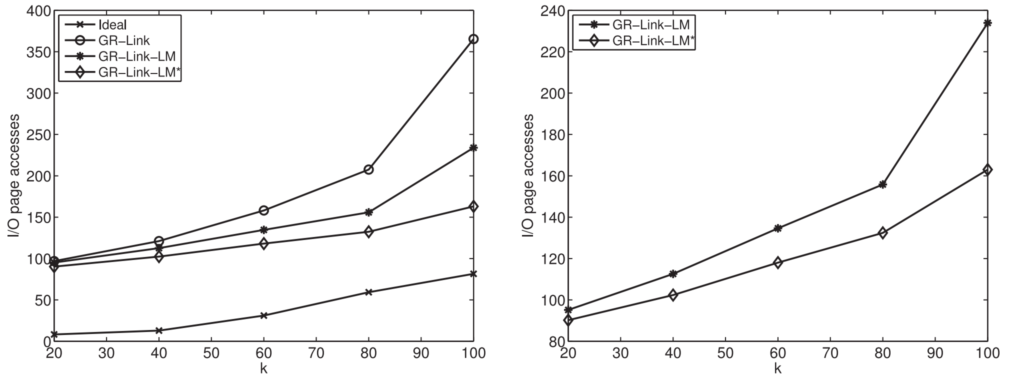
Having determined the optimal σ values, the core experiments will assess how the k-RNN processing methods perform under varying k values.

Figure 15(a) presents a comparison of all of the methods discussed in this article. Figure 15(b) focuses specifically on the respective performance of the *GR-Link-LM* and



(a) I/O Performance (b) CPU Performance

Fig. 14. Varying the number of expanded *ALT* nodes σ, I/O, and CPU performance. Different *k* values show different optimal σ values.



(a) *GR-Link* vs. LM vs. LM* vs. Ideal (b) *GR-Link-LM* vs. *GR-Link-LM**

Fig. 15. *GR-Link*, *GR-Link-LM*, *GR-Link-LM** and ideal index I/O performance.

*GR-Link-LM** methods. As mentioned, the benchmark "ideal approach" simulates a relevance-graph search that terminates as soon as the *k*-RNN POIs are found, that is, we have *a priori* knowledge of the results and are expanding the graph only until "discovered." As we can see, the *GR-Link-LM* and *GR-Link-LM** methods outperform the *GR-Link* method, both in terms of I/O page accesses and CPU time. Also, while not quite reaching the performance of the ideal method, especially the *GR-Link-LM** method not only reduces the gap considerably, but also shows the same I/O and CPU cost *k* growth behavior as the ideal method.

Focusing on the respective performance of the two new methods, for large *k*, *GR-Link-LM** performs considerably better than the simple, breadth-first approach of *GR-Link-LM*. The *GR-Link* algorithm, as expected, performs worse than our proposed *GR-Link-LM* and *GR-Link-LM** algorithms. This shows that the exclusion of expanded nodes based on our landmark-based method benefits the proposed algorithms and shows a considerable performance gain over the simple "blind" bidirectional BFS approach. In addition, the performance of *GR-Link-LM** is improved over the *GR-Link-LM* approach. The *ALT* algorithm used by *GR-Link-LM** for the spatial seed

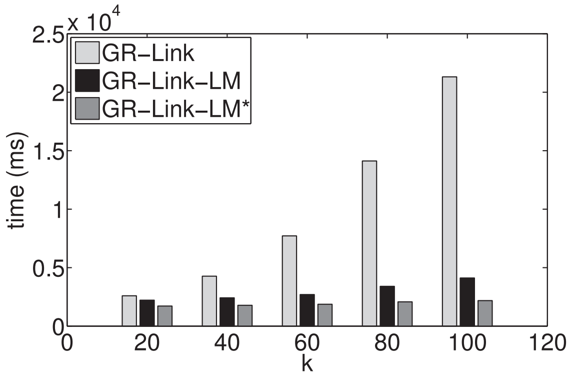
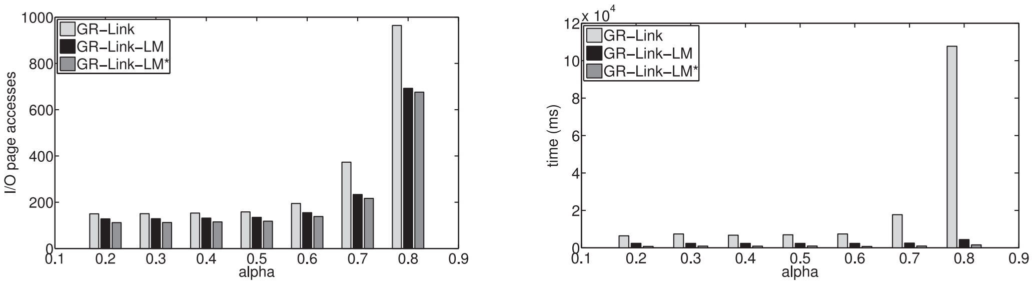


Fig. 16. CPU performance, varying k.



(a) I/O Performance (b) CPU Performance

Fig. 17. Varying preference parameter α.

expansions proves to work as expected, directing the search toward the query point, therefore saving the algorithm from expanding unused graph nodes.

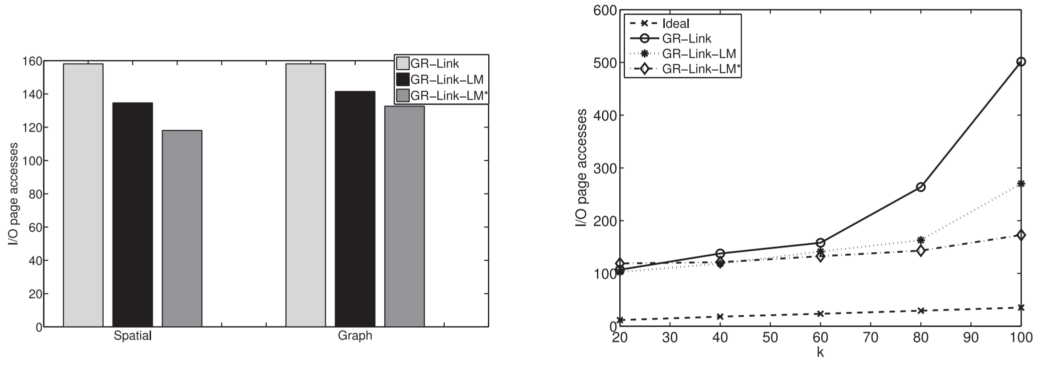
Figure 16 shows the runtimes of the algorithms. The results are in line with the I/O-based experiments and the respective performance advantage is even more evident.

6.6. Assessing the Effect of α

The preference parameter α balancing the effect of spatial distance versus relevance on the query result is not only a critical factor for the quality of the result, but also affects query-processing cost. With $\alpha < 0.5$, it favors the spatial score and with $\alpha > 0.5$, it favors the relevance score. Figure 17 shows that, with an emphasis on Relevance ($\alpha > 0.5$), the cost increases due to a more expensive relevance-graph expansion. Also, algorithms with an optimized graph search increase their performance advantage with increasing α. The results are analogous when measuring CPU time.

6.7. Spatial Grid versus Graph Partitioning

All POI data is kept on disk. The assumption so far is that each populated spatial grid cell corresponds to a disk block. This approach supposedly has the disadvantage of considering only the spatial characteristics without taking into account that the nearest-neighbor search explores the relevance graph as well. It is expected that the farther the points are spatially distributed, the more blocks need to be retrieved, thus increasing the I/O cost. Our approach in extending the current disk layout is now to group POIs based on relevance-graph proximity. This is achieved by partitioning the relevance graph using the METIS graph partitioning tool [Karypis and Kumar 1998].



(a) k=60 (b) Graph partitioning, varying k

Fig. 18. Partitioning: spatial grid versus relevance graph.

However, the grid-based partitioning is essential for spatial search and the underlying expansion mechanism. Using graph-based partitioning, we mapped the spatial grid cells to graph partitions. Now, when the spatial search requests a specific grid cell, the respective one or more graph partitions are fetched.

Ultimately, Figure 18(a) shows that graph-partitioning does not provide a performance advantage. What is interesting is that the performance advantage of the ideal method when compared to *GR-Link* is diminished in this case. An explanation here is that the expansion solely relies on the graph; thus, a respective partitioning would provide some (respective) advantage for this method. Overall, however, the ideal method performs best in the case of the spatial grid, which is evident when comparing Figures 10(b) and 18(b).

6.8. Summary

The experiments show that, by optimizing graph search, the resulting algorithms, *GR-Link-LM* and especially *GR-Link-LM** manage to improve the performance of the *GR-Link* method considerably. The two proposed algorithms also perform very well in comparison to an "ideal" method.

7. CONCLUSIONS

The purpose of this work was to introduce k-RNN queries and to create efficient methods for processing them, a version of the NN problem that also considers the relevance between query points. Relevance is introduced as a means to navigate a "forest" of POIs by retracing the steps of other people in similar situations, that is, where did people typically go next. Relevance, in our case, is defined as the co-occurrence of POIs in texts (LOIs). While a rather simplistic measure, it is adequate for defining and evaluating the proposed approach. To solve the k-RNN problem, we define query-processing methods that rely on a spatial grid and a relevance graph to capture the spatial and relevance aspect of the data, respectively. Landmarks are introduced as a means to improve the k-RNN query performance by optimizing the costly search on the relevance graph. *GR-Link-LM** the best performing algorithm, uses the *ALT* algorithm, which combines landmarks and A* search to optimize the graph search. The proposed methods show a significant performance improvement; their performance comes close to that of a hypothetical, "ideal" method.

Directions for future work include optimizing the spatial aspect of k-RNN query processing. The challenge is to create an index that more tightly integrates the processing of k-RNN queries. More efficient spatial access methods could be used, and our goal is to make the relevance graph information part of the index. The specific data and relevance score computation turn this goal into quite a challenge. Additional work will focus on an empirical "relevance" assessment of the query results involving real-world data collected from the Web. Extracting POIs and LOIs from Web documents requires an accurate and powerful geoparsing/geocoding algorithm. Testing Google Places and other named entity recognition (NER) tools has yielded some promising initial results.

REFERENCES

Nikos Armenatzoglou, Stavros Papadopoulos, and Dimitris Papadias. 2013. A general framework for geo-social query processing. *Proceedings of the VLDB Conference* 6, 10, 913–924.

Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. 1990. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the SIGMOD Conference.* 322–331.

Xin Cao, Gao Cong, and Christian S. Jensen. 2010. Retrieving top-k prestige-based relevant spatial web objects. *Proceedings of the VLDB Conference* 3, 1–2, 373–384.

Xin Cao, Gao Cong, Christian S. Jensen, and Beng Chin Ooi. 2011. Collective spatial keyword querying. In *SIGMOD*.

Yen-Yu Chen, Torsten Suel, and Alexander Markowetz. 2006. Efficient query processing in geographic web search engines. In *Proceedings of the SIGMOD Conference.* 277–288.

Gao Cong, Christian S. Jensen, and Dingming Wu. 2009. Efficient retrieval of the top-k most relevant spatial web objects. *Proceedings of the VLDB Conference* 2, 1, 337–348.

Ian De Felipe, Vagelis Hristidis, and Naphtali Rishe. 2008. Keyword search on spatial databases. In *Proceedings of the 24th ICDE Conference.* 656–665.

Daniel Delling and Dorothea Wagner. 2007. Landmark-based routing in dynamic graphs. In *Proceedings of the 6th International Conference on Experimental Algorithms (WEA'07)*. Springer, Berlin, 52–65.

Alexandros Efentakis, Dieter Pfoser, and Agnès Voisard. 2011. Efficient data management in support of shortest-path computation. In *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Computational Transportation Science (CTS'11)*. ACM, New York, NY, 28–33.

Alexandros Efentakis, Dimitris Theodorakis, and Dieter Pfoser. 2012. Crowdsourcing computing resources for shortest-path computation. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems (SIGSPATIAL'12)*. ACM, New York, NY, 434–437.

Christodoulos Efstathiades and Dieter Pfoser. 2013. User-contributed relevance and nearest neighbor queries. In *Proceedings of the 13th SSTD Symposium.* 312–329.

Ronald Fagin. 1999. Combining fuzzy information from multiple systems. *Journal of Computer and System Sciences* 58, 1, 83–99.

Ronald Fagin, Amnon Lotem, and Moni Naor. 2003. Optimal aggregation algorithms for middleware. *Journal of Computer and System Sciences* 66, 4, 614–656.

Andrew V. Goldberg and Chris Harrelson. 2005. Computing the shortest path: A search meets graph theory. In *Proceedings of the 16th SODA Conference.* 156–165.

Andrew V. Goldberg and Renato F. Werneck. 2005. Computing point-to-point shortest paths from external memory. In *Proceedings of the 7th Workshop on Algorithm Engineering and Experiments (ALENEX'05)*. 26–40.

Andrey Gubichev, Srikanta Bedathur, Stephan Seufert, and Gerhard Weikum. 2010. Fast and accurate estimation of shortest paths in large graphs. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM'10)*. ACM, New York, NY, 499–508.

Peter Hart, Nils Nilsson, and Bertram Raphael. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics* 4, 100–107.

Gísli R. Hjaltason and Hanan Samet. 1999. Distance browsing in spatial databases. *ACM Transactions on Database Systems* 24, 2, 265–318.

George Karypis and Vipin Kumar. 1998. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on Scientific Computing* 20, 1, 359–392.

Ekkehard Köhler, Rolf H. Möhring, and Heiko Schilling. 2006. Fast point-to-point shortest path computations with arc-flags. In *IN: 9th Dimacs Implementation Challenge*.

Nick Koudas, Beng Chin Ooi, Kian-Lee Tan, and Rui Zhang. 2004. Approximate NN queries on streams with guaranteed error/performance bounds. In *Proceedings of the 30th International Conference on Very Large Data Bases - Volume 30 (VLDB'04)*. VLDB Endowment, 804–815. http://dl.acm.org/citation.cfm?id=1316689.1316759

Yuchen Li, Zhifeng Bao, Guoliang Li, and Kian-Lee Tan. 2015. Real time personalized search on social networks. In *IEEE 31st International Conference on Data Engineering (ICDE'15)*. IEEE, 639–650.

Zhisheng Li, Ken C. K. Lee, Baihua Zheng, Wang-Chien Lee, Dik Lee, and Xufa Wang. 2011. IR-tree: An efficient index for geographic document search. *IEEE Transactions on Knowledge and Data Engineering* 23, 4, 585–599.

Yu Liu, Fahui Wang, Chaogui Kang, Yong Gao, and Yongmei Lu. 2014. Analyzing relatedness by toponym co-occurrences on web pages. *Transactions in GIS* 18, 1, 89–107.

Cheng Long, Raymond Chi-Wing Wong, Ke Wang, and Ada Wai-Chee Fu. 2013. Collective spatial keyword queries: A distance owner-driven approach. In *SIGMOD*.

Bruno Martins, Mário J. Silva, and Leonardo Andrade. 2005. Indexing and ranking in Geo-IR systems. In *Proceedings of the Geographic Information Retrieval Workshop*. 31–34.

Jens Maue, Peter Sanders, and Domagoj Matijevic. 2010. Goal-directed shortest-path queries using precomputed cluster distances. *Journal of Experimental Algorithmics* 14, Article 2, 1.07 pages.

K. Mouratidis, Jing Li, Yu Tang, and N. Mamoulis. 2015. Joint search by social and spatial proximity. *IEEE Transactions on Knowledge and Data Engineering* 27, 3, 781–793.

Sarana Nutanong, Rui Zhang, Egemen Tanin, and Lars Kulik. 2008. The V*-diagram: A query-dependent approach to moving KNN queries. *Proceedings of the VLDB Endowment* 1, 1, 1095–1106. DOI:http://dx.doi.org/10.14778/1453856.1453973

Dimitris Papadias, Panos Kalnis, Jun Zhang, and Yufei Tao. 2001. Efficient OLAP operations in spatial data warehouses. In *Proceedings of the 7th SSTD Symposium.* Springer, 443–459.

Michalis Potamias, Francesco Bonchi, Carlos Castillo, and Aristides Gionis. 2009. Fast shortest path distance estimation in large networks. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM'09)*. ACM, New York, NY, 867–876.

João B. Rocha-Junior, Orestis Gkorgkas, Simon Jonassen, and Kjetil Nørvåg. 2011. Efficient processing of top-k spatial keyword queries. In *Proceedings of the 12th SSTD Symposium.* 205–222.

Georgios Skoumas, Dieter Pfoser, and Anastasios Kyrillidis. 2013. On quantifying qualitative geospatial data: A probabilistic approach. In *Proceedings of the 2nd ACM SIGSPATIAL International GEOCROWD Workshop*. 71–78.

Yu Sun, Jianzhong Qi, Yu Zheng, and Rui Zhang. 2015. K-nearest neighbor temporal aggregate queries. In *Proceedings of the 18th International EDBT Conference.* 493–504.

Konstantin Tretyakov, Abel Armas-Cervantes, Luciano García-Bañuelos, Jaak Vilo, and Marlon Dumas. 2011. Fast fully dynamic landmark-based estimation of shortest path distances in very large graphs. In *Proceedings of the 20th CIKM Conference.* 1785–1794.

Subodh Vaid, Christopher B. Jones, Hideo Joho, and Mark S. 2005. Spatio-textual indexing for geographical search on the web. In *Proceedings of the 5th SSTD Symposium.* 218–235.

Dongxiang Zhang, Yeow Meng Chee, Anirban Mondal, Anthony K. H. Tung, and Masaru Kitsuregawa. 2009. Keyword search in spatial databases: Towards searching by document. In *ICDE*.

Dongxiang Zhang, Beng Chin Ooi, and Anthony K. H. Tung. 2010. Locating mapped resources in Web 2.0. In *ICDE*.

Yinghua Zhou, Xing Xie, Chuang Wang, Yuchang Gong, and Wei-Ying Ma. 2005. Hybrid index structures for location-based web search. In *Proceedings of the 14th CIKM Conference.* 155–162.

Received January 2015; revised February 2016; accepted May 2016