Optimal Network Location Queries

Parisa Ghaemi, Kaveh Shahabi, John P. Wilson*, Farnoush Banaei-Kashani

Computer Science Department
University of Southern California
Los Angeles, CA 90089

[ghaemi, kshahabi, jpwilson, banaeika]@usc.edu

ABSTRACT

Given a set *S* of sites and a set *O* of weighted objects, an optimal location query finds the location(s) where introducing a new site maximizes the total weight of the objects that are closer to the new site than to any other site. With such a query, for instance, a franchise corporation (e.g., McDonald's) can find a location to open a new store such that the number of potential store customers (i.e., people living close to the store) is maximized. Optimal location queries are computationally complex to compute and require efficient solutions that scale with large datasets. Previously, two specific approaches have been proposed for efficient computation of optimal location queries. However, they both assume p-norm distance (namely, L_1 and L_2/Euclidean); hence, they are not applicable where sites and objects are located on spatial networks. In this paper, we focus on *optimal network location* (*ONL*) queries, i.e., optimal location queries with which objects and sites reside on a spatial network. We introduce an approach, namely *EONL* (short for *Expansion-based ONL*), which enables efficient computation of ONL queries. Moreover, with an extensive experimental study we verify and compare the efficiency of our proposed approach with real datasets, and we demonstrate the importance of considering network distance (rather than p-norm distance) with ONL queries.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications – *Spatial databases and GIS.*

Keywords

Optimal Location Query, Query Processing in Spatial Network Databases

1. INTRODUCTION

Optimal location queries have been widely used in spatial decision support systems and marketing in recent years. For instance, a city planner might want to know: "What is the optimal location to open a new public library?" The optimal location is the site that would maximize the number of patrons for whom this is the closest library. Optimal location query is formally defined as follows: Given a set *S* of sites and a set *O* of weighted objects the optimal location query computes a location where introducing a new site would maximize the total weight of objects that are closer to the new site than to any other site.

Optimal location queries are computationally complex to answer. The existing work considers L_1 distance metrics or L_2/Euclidean as the measure of distance between objects and sites and proposes efficient solutions in these p-norm metric spaces [6, 3]. However, with many real world applications objects and sites are located on a spatial network (e.g., roads, railways, and rivers), and therefore, the approaches that assume p-norm distance fail to apply. We verify the importance of assuming network distance with ONL queries in Section 4 via experiments, and we show that in 75% of the cases the results of optimal location queries in L_1 and L_2 spaces are totally disjoint from the actual optimal network location, with less than 20% overlapping in the rest of the cases.

In this paper, for the first time we propose a scalable solution for efficient computation of ONL queries, namely *EONL* (short for *Expansion-based ONL*). We argue that the dominating computational complexity with ONL queries is twofold (this also applies to regular optimal location queries). To answer any ONL query, first one has to compute a spatial locality around each object *o* of the given object-dataset such that if *s* is the nearest site to object *o*, any new site *s′* introduced within the locality of *o* will be closer to *o* as compared to the distance between *s* and *o*. Second, one must compute the overlapping among object localities to identify the optimal network location, which is a network segment (or a set of segments) where the localities of a subset of objects with maximum total weight overlap.

Accordingly, with our proposed algorithm, EONL, we focus on reducing the computational complexity of the latter steps in ONL query answering. In particular, with EONL we simply compute the locality of an object by expanding the network around the object until we reach the nearest site *s* to the object. However, during network expansion we identify and record potential overlapping between localities of the objects to avoid redundant computation at the second step; thus, efficient computation of overlapping among object localities at the second step. Our experimental results with real datasets show that EONL takes less than 0.3 minute to compute ONL query for a dataset of 19160 object points.

* John P. Wilson is a Professor of Geography, and Director of the Spatial Sciences Institute at the University of Southern California.

The remainder of this paper is organized as follows. Section 2 formally defines optimal network location queries in spatial network databases. Section 3 introduces our proposed expansion-based solutions for optimal network location queries. In Section 4, we evaluate our proposed solutions via experiments with real data. Section 5 reviews the related work, and Section 6 concludes the paper and discusses directions for future research.

2. PROBLEM FORMALIZATION

In this section, we formalize the problem of optimal network location as a Maximum Overlap Segment (MaxOSN) problem. Assume we have a set *S* of sites (e.g. public schools, libraries, restaurants) in a 2D environment. Also we have a set *O* of objects with a weight *w(o)* for each object *o*. For instance, object *o* might be a residential building/property where *w(o)* represents the number of people living in this building. A MaxOSN query returns a subset of the spatial network (i.e., a segment or collection of segments) where introducing a new site would maximize the total weight of the objects that are closer to the new site than to any other site. We assume both sites and objects are located on a spatial network, e.g., a road network. We model the road network as a graph G (N, E), where *N* is the set of nodes and *E* is the set of edges of the road network. Each edge *e(a ,b)* has a travel cost. In this paper, we assume the cost of each edge *e* is proportional to the distance between the two end points *a* and *b* of *e*. Accordingly, the network distance *dN(a,b)* between any two nodes *a* and *b*, is the travel cost of the path with least cost from *a* to *b*. Fig. 2 shows a road network with 14 nodes and weighted edges, four objects o_1, o_2, o_3, and o_4 with weights 3, 6, 5, and 4, respectively, and three sites s_1, s_2, and s_3.

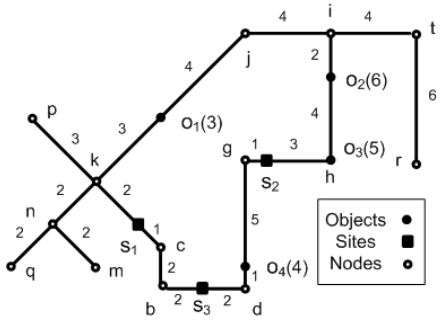


Fig. 2. Road network model

Below, we first define our terminology. Thereafter, we describe the MaxOSN query problem.

DEFINITION 1 (LOCAL NETWORK). *Given an object o, the local network LN(o) of o, is a sub-network expanded at object o that contains all points on the road network with a network distance less than or equal to the network distance between o and its nearest site s.*

In Fig. 3, site s_1 is the nearest site to the object o_1 where $dN(o_1,s_1)$=5. $LN(o_1)$ is identified by expansion, i.e., starting from o_1 we traverse all possible paths up to the network distance equal to 5, and we delimit $LN(o_1)$ by marking the ending points (shown as arrows in Fig. 3). We term this delimitation process *edge marking*. The expanded network consists of a set of *local edges* connecting the associated object to all marked ending points. It is important to note that local edges can fully or partially cover an actual edge of the road network. For instance, the local edges of $LN(o_1)$ are o_1n_2, o_1n_1, o_1n_4 *and* o_1n (shown as bold lines in Fig. 3). Each local edge *e* is also assigned an influence value, denoted by *I(e)*, which is equal to the weight of the corresponding object. For example, all local edges in $LN(o_1)$ have an influence value equal to 3 (i.e., the weight of object o_1).

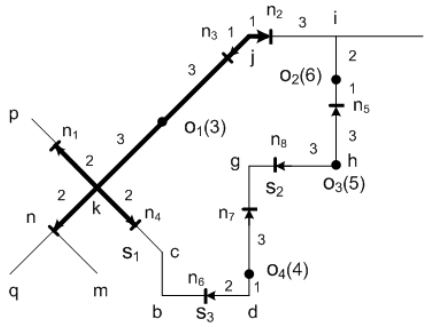


Fig. 3. Local networks

DEFINITION 2 (OVERLAPPING LOCAL NETWORKS). *A local network $LN(o_1)$ overlaps a local network $LN(o_2)$ if there exists a local edge e_1 in $LN(o_1)$ which intersects a local edge e_2 in $LN(o_2)$.*

For instance, in Fig. 3 $LN(o_1)$ overlaps with $LN(o_2)$ since the local edge o_1n_2 in $LN(o_1)$ overlaps with the local edge o_2n_3 in $LN(o_2)$.

DEFINITION 3 (OVERLAP SEGMENT). *Given two overlapping local networks, an overlap segment s is a network segment where the overlapping local edges of the two local networks intersect. The influence value of segment s is equal to $I(e_1)$ +$I(e_2)$, where e_1 and e_2 are the overlapping local edges from the two local networks.*

For example, in Fig. 3 the overlap segment jn_2 is identified by overlapping the local edges o_1n_2 and o_2n_3. Also, its influence value is equal to 9.

DEFINITION 4 (MAXIMUM OVERLAP SEGMENT QUERY (MaxOSN)). *Given a set O of objects, a set S of sites, the MaxOSN query returns optimal network location, i.e., one or more overlap segment(s) which have the maximum influence value (I_0) among all overlap segments.*

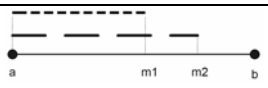
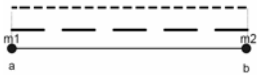
For instance, in the road network illustrated in Fig. 3 the MaxOSN query returns the set of overlap segments $\{o_3n_8, o_3n_5\}$, where each segment has an optimal influence value I_0 =11.

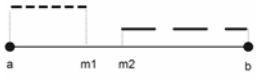
3. Expansion-Based Optimal Network Location (EONL)

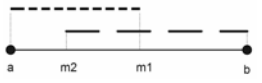
As we mentioned in Section 1, answering an ONL query is a two-phase process. At the first phase, one needs to build the local networks of all objects, whereas at the second phase local networks of the objects are overlapped in order to identify the overlap segment(s) with maximum influence value (i.e., the optimal location/segment). With EONL, we focus on reducing the computational complexity of the second phase. In particular, at the first phase EONL simply uses network expansion to build the local networks. At the second phase, assuming we have *n* objects (and therefore, *n* local networks), one should compute

the overlap between 2^n combinations of local networks. In this case, if (for example) one of the network range-query processing techniques proposed by Papadias et al. [4] is used for overlap computation, the total computational complexity would be in the order of O ($2^{|O|}$ (|N| log |N| + |E|)). Obviously, this approach is not scalable. Instead, with EONL we identify the *potential* optimal segments while expanding local networks at the first phase, and leverage this information at the second phase to efficiently compute the segment(s) with maximum influence value. To be specific, while expanding the local networks at the first phase, for each edge we record all ending points (i.e., the points that mark the border of the local networks of the objects) that lie over the edge. Subsequently, at the second phase we use the information recorded at the first phase to compute a score for each edge, which is equal to the total weight of the objects whose local networks end on the edge. One should observe that a higher score for an edge indicates higher potential of containing an optimal segment. Next, through a refinement process we sort the edges based on their scores in descending order, and starting from the edge with higher score, we use a technique, termed *edge collapsing*, to compute the *actual* overlap segment(s) on each edge. It is important to note that through this refinement process we only have to compute the actual overlap segment(s) for an edge if the score of the edge is more than the influence value of the actual segments computed so far. With our experiments, we observe that EONL only computes the actual overlap segments for a limited subset of the network edges before it identifies the optimal location/segment; hence, effective pruning of the search space for better efficiency.

Table 1. Edge collapsing technique

Case	Overlapping Local Edges	Overlap Segment
1		$S_0 = am_1$ $I_0 = I(e_1) + I(e_2)$
2		$S_0 = ab$ $I_0 = I(e_1) + I(e_2)$

3		If ($I(e_1) > I(e_2)$)
		$S_0 = a\,m_1$; $I_0 = I(e_1)$
		Else
		$S_0 = b\,m_2$; $I_0 = I(e_2)$

4		$S_0 = m_2m_1$ $I_0 = I(e_1) + I(e_2)$

Below, we explain how we implement EONL in more detail; we begin by introducing our edge collapsing technique. Table 1 represents four possible cases by which two local edges e_1 and e_2 might overlap each other. The dashed lines represent local edges e_1 and e_2, the solid line represents the actual edge *ab* of the road network, and m_1 and m_2 are the end points of e_1 and e_2. The third column summarizes how the edge collapsing technique computes the overlap segment (S_0) with the maximum influence value (I_0) in each case. Next, we describe the implementation of EONL as a four-step algorithm:

Step 1 (Expanding local networks): For each object point o, we first expand the local network of object *o*, *LN(o)*, using the Dijkstra algorithm [2]. Then, we mark the ending points of the local networks on the edges. Table 2 shows sample subsets of the marked edges of Fig. 3. Each row of the Marked Edge Table (MET) is an entry in the form of (*e, M, w(e)*) where M is the set of ending points marked on edge *e* and *w(e)* is equal to the sum of influence values of the end points (i.e., the corresponding local edges).

Table 2. Marked Edge Table (MET)

e	M	w(e)
kp	$\{n_1\}$	3
kj	$\{n_3\}$	9
hg	$\{n_8\}$	11

Step 2 (Sorting MET Table): We sort all entries in the MET table in descending order of *w(e)* because of our observation that the optimal solution is mostly derived from the entries with larger *w(e)* values. Then, we apply the edge collapsing technique to the first entry of MET and initialize the S_0 and I_0 values (see Table 1).

Step 3 (Identifying overlap segments): From the set of marked edges in MET table, we identify the overlap segments using the edge collapsing technique. We could apply the edge collapsing technique to all marked edges; however, we do not need to apply this approach for some marked edges if there is another marked edge whose influence value has a smaller value than I_0. While applying edge collapsing to eligible marked edges, we update I_0 to the actual influence value of the marked edge and compute the corresponding overlap segment(s), S_0.

Step 4 (Finding the maximum influence value): When the algorithm terminates, S_0 returns the set of optimal overlap segment(s) with the optimal influence value I_0.

Below, we discuss the computational complexity of our proposed approach. The cost of expanding local networks is *O* (|O| (|N| log |N| +|E|)), constructing the MET table is O (|E|) and sorting the table is O (|E| log |E|). Also, edge collapsing requires *O* (|E| $|O|^2$) time. Thus, the overall running time of EONL approach is *O* (|E| log |E|) + *O* (|O| (|N| log |N| +|E|)) +*O* (|E| $|O|^2$).

4. EXPERIMENTAL EVALUATION

We next describe the experimental setup we used for the experiments, and then present and discuss the experimental results.

4.1 Experimental Setup

For all experiments, we use a spatial network of |N| = 375691 nodes and |E| = 871715 bidirectional edges, representing the LA County road network. We use real dataset for objects and sites. Objects are population data derived from LANDSCAN population data. The centroid of each grid cell is treated as the location of each object and the population within each grid cell as the weight of object. In total we have |O|= 9662 objects. We also deployed four datasets consisting of McDonald's restaurants, hospitals, schools, and all fast food restaurants in LA County (including McDonald's) for the sites (see Table 4).

Table 4. Four real datasets for sites

Datasets	Cardinality	Datasets	Cardinality
McDonald's	328	Schools	2621
Hospitals	308	Fast Foods	19160

4.2 Experimental Results

Below we present the results of the two series of experiments that we ran on the aforementioned datasets.

Accuracy: We first verified that the optimal location query in L_1 and L_2/Euclidean space is not applicable to spatial networks. For this test, we selected four datasets with 20, 40, 60, and 85 object points that were randomly selected from the population data (DS_1-DS_4). All four sets of object points were located on LA county road network. For site points, we selected a subset of McDonald's including seven sites. We applied the L_2 [6] and L_1 [3] distance approaches and identified the optimal location in each case. Then, we performed the EONL algorithms on each dataset and retrieved their corresponding optimal network location. The result of this experiment showed that in 75% of cases (we call it set A) the optimal locations derived by the L_1/L_2 approach did not overlap the optimal network location derived by EONL and when they did overlap, there was < 20% common coverage. From cases included in set A, the average distance between the optimal network location and the optimal location derived from the L_1 and L_2 approaches (<N, L_1>, <N, L_2 >) are similar to the size of the entire area covered by these datasets (see Table 5) and verifies that using the existing L_1 and L_2 approaches for optimal location queries on spatial network databases is not accurate and likely to return irrelevant results.

We also observed that the maximum influence value returned by the optimal network location query is 13% and 12% higher than those returned by the optimal location queries in the L_1 and L_2 approaches, respectively.

Table 5. Average distance of optimal network location and optimal location derived by L_1 and L_2 approaches (Size of the entire area is 6.2 km x 9 km)

Dataset	<N, L_1> (meters)	<N, L_2 > (meters)
DS_1	Overlaps (< 20% coverage)	
DS_2	4998	5305
DS_3	4995	2743
DS_4	6663	6396
Average	5552	4814

Execution Time: In order to evaluate the execution time of our proposed approach, we applied EONL approach to the four sites datasets of Table 4 and we selected the uniformly distributed population data as the fixed object dataset (including all 9662 points). Fig. 8 shows that EONL has lower performance with more skewed dataset, such as Hospitals comparing to uniformly distributed site points (McDonald's, Schools, and Fast Foods). This variability meant that in those parts of the graph with few hospitals, the expansion of the local network traverses a longer path until it hits the nearest site. Also, although the size of Fast Foods restaurants is large, the execution time of EONL is low. This is because the complexity of EONL (O (|E| log |E|) + O (|O| (|N| log |N| +|E|)) +O (|E| $|O|^2$)) is independent of the number of site points, |S|. Finally, it is important to note that the execution of EONL is significantly improved over the naïve optimal network location approach (discussed in section 3), which has an exponential time complexity (O ($2^{|O|}$ (|N| log |N| + |E|))).

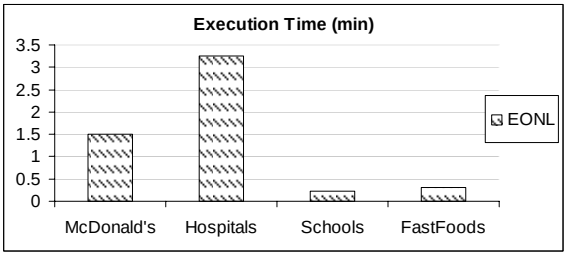


Fig. 8. Execution time of EONL with four different site datasets

5. RELATED WORK

Optimal location queries have been studied by researchers in operations research (OR) and database systems. In OR, most optimal location problems (also called facility location problems) are formulated as *covering problems*. These involve locating *n* sites to cover all or most of the (so-called) demand objects assuming a fixed service distance for sites [5, 1]. While OR-based solutions are effective and address various types of optimal location problems, many of these solutions fail to scale with real datasets that consist of large numbers of sites and objects due to their computational complexity. Accordingly, a number of complementary solutions are proposed by the database community for scalable optimal location query answering. In particular, the solutions provided by Wong et al. [6] and Du et al [3], while efficient, they assume p-norm space ([6] assumes L_2 and [3] assumes L_1), and cannot support optimal location queries on spatial networks. Our proposed solution utilized network distance to address ONL queries.

6. CONCLUSION AND FUTURE WORK

In this study, for the first time we proposed a scalable solution for the problem of optimal location for objects and sites located on spatial networks. We verified and compared the performance of our solution with experimental evaluation with real data.

We plan to study a more complex optimal location problem setting where the sets of sites and/or objects might be located both on and off spatial networks. With this problem, we will investigate and develop hybrid solutions.

7. REFERENCES

[1] Church, R. L., and Revelle, C. 1974. The Maximal Covering Location Problem. *Papers of the Regional Science Association*, 32, 1974.

[2] Dijkstra, E. W. A Note on Two Problems in Connection with Graphs. *Numeriche Mathematik*, 1, 1, 269-271.

[3] Du, Y., Zhang, D., and Xia, T.2005. The Optimal-Location Query. SSTD 2005.

[4] Papadias D., Zhang, J., Mamoulis N., and Tao, Y. 2003. Query Processing in Spatial Network Databases. VLDB 2003.

[5] Toregas, C., Swain, R., Revelle, C., and Bergman, L. 1971. The Location of Emergency Service Facilities. *Operations Research*, 19, 6, 1971.

[6] Wong, R. C., Ozsu, M. T., Yu, P. S., Fu, A. W., and Liu, L 2009. Efficient Method for Maximizing Bichromatic Reverse Nearest Neighbor. VLDB 2009.