Top-k Most Influential Location Selection

Jin Huang
South China Univ. of Tech.
Guangzhou, China
jin.h@iojin.com

Zeyi Wen
University of Melbourne
Melbourne, Australia
wenzeyi@gmail.com

Jianzhong Qi
University of Melbourne
Melbourne, Australia
jiqi@csse.unimelb.edu.au

Rui Zhang
University of Melbourne
Melbourne, Australia
rui@csse.unimelb.edu.au

Jian Chen
South China Univ. of Tech.
Guangzhou, China
ellachen@scut.edu.cn

Zhen He
La Trobe University
Bundoora, Australia
z.he@latrobe.edu.au

ABSTRACT

We propose and study a new type of facility location selection query, the *top-k most influential location selection query*. Given a set M of customers and a set F of existing facilities, this query finds k locations from a set C of candidate locations with the largest influence values, where the influence of a candidate location c ($c \in C$) is defined as the number of customers in M who are the reverse nearest neighbors of c. We first present a naive algorithm to process the query. However, the algorithm is computationally expensive and not scalable to large datasets. This motivates us to explore more efficient solutions. We propose two branch and bound algorithms, the *Estimation Expanding Pruning (EEP)* algorithm and the *Bounding Influence Pruning (BIP)* algorithm. These algorithms exploit various geometric properties to prune the search space, and thus achieve much better performance than that of the naive algorithm. Specifically, the EEP algorithm estimates the distances to the nearest existing facilities for the customers and the numbers of influenced customers for the candidate locations, and then gradually refines the estimation until the answer set is found, during which distance metric based pruning techniques are used to improve the refinement efficiency. BIP only estimates the numbers of influenced customers for the candidate locations. But it uses the existing facilities to limit the space for searching the influenced customers and achieve a better estimation, which results in an even more efficient algorithm. Extensive experiments conducted on both real and synthetic datasets validate the efficiency of the algorithms.

Categories and Subject Descriptors

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

General Terms

Algorithms, Performance, Theory

Keywords

Location Selection, Top-k Query, Reverse Nearest Neighbors

CIKM'11, October 24–28, 2011, Glasgow, Scotland, UK.

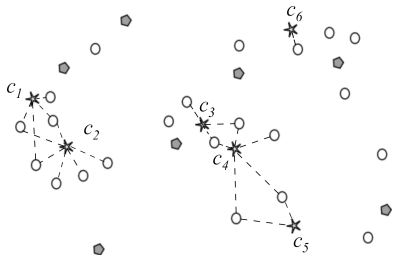


Figure 1: The top-k most influential location selection problem

1 Introduction

A common problem many businesses and organizations share is finding a suitable place for the establishment of a new facility. For example, McDonald's may want to add a new restaurant to compete with other restaurants; a wireless service provider may want to add a hotspot to a densely populated area to improve the quality of service. In most cases, there is a set of candidate locations to select from (e.g., real estate agents have databases of places for lease or sale). Then, an important business decision (for McDonald's or a wireless service provider) is to select a location that attracts as many customers as possible. Sometimes, one may want to first have a number (k) of such locations and then make decisions with further considerations. In this paper, we investigate this problem of finding the top-k candidate locations that attract the largest numbers of customers, where k is a user input parameter. These k candidate locations can then be fed to further decision making procedures for selecting an overall best location for the new facility. Here, we assume a customer is attracted by her nearest facility and the business has the knowledge of customer distribution from surveys or past sales records.

In urban development simulation, very often urban planners need to simulate the above facility location selection procedure, so that the influence of establishing a new public facility or business center on the residents can be observed. Figure. 1 gives an example, where circles, pentagons and stars denote customers, existing facilities and candidate locations (labeled as $c_1, ..., c_6$), respectively. First, for every candidate location c_i, we compute the number of customers c_i can attract. For example, if a new facility is established at c_1, then it will attract 4 customers. Similarly, we can compute the numbers for c_2, c_3, c_4, c_5 and c_6, which are 6, 3, 5, 2 and 1, respectively. Suppose now we want to have the top-3 candidate locations that attract the largest numbers of customers, i.e., $k = 3$. Then, c_2, c_4 and c_1 are returned as the answer set.

The above location selection problem aims at maximizing the "influence" of the candidate locations, where the influence of a can-

didate location c denotes the number of customers whose respective nearest facility will be c if a new facility is established at c. This problem is frequently raised in applications like urban development simulation. Thus, we formulate it as the top-k most influential location selection query. We propose algorithms to process this query efficiently and make the following contributions.

- We formulate the top-k most influential location selection query.
- We analyze the properties of the query and propose pruning techniques to reduce the search space for processing the query.
- Based on the proposed pruning techniques, we propose two algorithms, namely, the Estimation Expanding Pruning algorithm (EEP) and the Bounding Influence Pruning (BIP) algorithm, to efficiently process the query.
- We perform an extensive experimental study. The results confirm the effectiveness of the proposed pruning techniques and the efficiency of the proposed algorithms.

The rest of the paper is organized as follows. § 2 reviews previous studies on related topics. § 3 defines the problem and presents the naive algorithm. § 4 and 5 describe our EEP algorithm and BIP algorithm, respectively. § 6 presents the experimental results and § 7 concludes the paper.

2 Related Work

The studied problem is based on the concept of *reverse nearest neighbor (RNN)* query [10], which finds for a query object o a set of objects, who perceive o as their nearest neighbors. Since it was proposed, various techniques [14, 16, 1] have been proposed to process this type of query and its variants under different settings. For example, a most relevant RNN variant, the *reverse k nearest neighbor (RkNN)* query, extends the answer set to include objects who perceive the query object as among their k nearest neighbors. Wu et al. [16] study the RkNN query on continuously moving objects, which correlates two sets of moving objects according to their closeness. The continuous join query on moving objects [18] also correlates multiple sets, but it focuses on intersection of objects with a time constraining technique rather than closeness. While these studies work well for processing a single R(k)NN query, they are not designed to compute R(k)NNs for large amount of objects at the same time, which is one of the key difficulties in our study.

Regarding to facility location problems, the most relevant study [8] ranks the candidate locations according to their influence values. However, the proposed solution cannot find the top-k locations and then early terminates the algorithm. Therefore, it does not solve our problem, while our proposed algorithms can be used to rank the locations by setting k to the number of candidate locations. Other studies on facility location problems have quite different problem settings. Cabello et al. [2] propose a facility location problem based on the MAXCOV optimization criterion, which is to find regions in the data space to maximize the numbers of RNNs for the points in these regions. The study gives a theoretical analysis, but no efficient algorithm is presented. Chen et al. [4] study this problem further and propose an efficient solution for finding the nearest facilities. Xia et al. [17] propose the top-t most influential sites problem, which finds the top-t most influential existing sites within a given region Q. Unlike our study, they do not consider any candidate locations. Du et al. [5] propose to find a point from a continuous candidate region that can maximize the influence value. They use L_1 as the distance metric and have a strong assumption that all the roads are either horizontal or vertical. We consider L_2 distance, which is a more general problem setting. More importantly, *we consider a candidate location set instead of a candidate region. This is a more practical problem setting because in many real applications, we can only choose from some candidate locations (e.g. a McDonald's restaurant can only be opened at a place for lease or sale rather than anywhere in a region).* Cheema et al. [3] propose to find an influence zone for a query location c, where customers inside this zone form exactly the RkNN query result of c. They compute *Voronoi cells* on the fly for the query location to obtain its RkNNs. Similarly, Wong et al. [15] propose to find the optimal region in which all query objects are of the maximum influence. Compared with this problem, our problem focuses on the numbers of RNNs of the candidate locations instead of specific locations of the RNNs. These problems use different settings from ours. Therefore, their solutions do not apply.

3 Preliminaries

3.1 Problem Definition

We assume three object sets, namely, a set of customers M, a set of existing facilities F and a set of candidate locations C. All the data objects (customers, existing facilities or candidate locations) are represented by points in an Euclidean space. We may refer to a data object as a data point or simply as a point. The distance between two points p_1 and p_2 is denoted as $d(p_1, p_2)$.

Given a customer $m \in M$, we denote her nearest existing facility as $nf(m)$, and call the distance between m and $nf(m)$, $d(m, nf(m))$, the *nearest facility distance (NFD)* of m. We say that customer m is attracted by a candidate location $c \in C$ if the distance between m and c is less than the nearest facility distance of m, i.e., $d(m, c) < d(m, nf(m))$. In this case, if we add a facility at c, it will become the new nearest facility of m.

A candidate location $c \in C$ may attract multiple customers. The number of these attracted customers defines the *influence* of c, $inf(c)$. Formally, $inf(c) = |\{m \in M | d(m, c) < d(m, nf(m))\}|$.

The top-k most influential location selection query is to find k candidate locations that attract the largest numbers of customers.

DEFINITION 1. *Top-k Most Influential Location Selection (TILS) Query: Given a constant k, a set of points M as customers, a set of points F as existing facilities and a set of points C as candidate locations, the top-k most influential location selection query finds a set $C' \subset C$ with k points, so that $\forall c_i \in C', c_j \in C \setminus C' : inf(c_i) \geq inf(c_j)$.*

We call the subset C' the *top-k most influential location set* and denote it as TOPINF(K,M,F,C)[1]. Next, we present a naive algorithm to find this subset.

3.2 A Naive Algorithm

A *naive algorithm (NA)* to process the TILS query is to first compute the nearest facility distance for every $m \in M$, which is done by scanning F to find m's nearest existing facility. Then, for every candidate location $c \in C$, we scan the customer set M. If the distance between c and a customer m, $d(c, m)$, is less than the nearest facility distance of m, i.e., c attracts m, then we increment the influence value of c. After we have scanned every pair of c and m, we obtain the influence value for every c. We then sort the candidate locations according to their influence values, and output the first k candidate locations in the sorted result as the query answer set.

The NA algorithm is inefficient in that it has to scan every pair of customer and existing facility to find customers' nearest existing facilities, and every pair of candidate location and customer to compute candidate locations' influence values. This motivate us

[1]Note that there may be ties in the influence values. To simplify our discussion, we always return the first k candidate locations found that have the largest influence values.

to explore effective pruning techniques to reduce the search space. We assume that the datasets are maintained in spatial indexes [2] and propose two algorithms utilizing different geometric properties to achieve effective pruning. In what follows, we present the overall ideas of these two algorithms. Technical details are omitted due to space limit and they are given in a technical report [9].

4 Estimation Expanding Pruning

In this section, we propose an algorithm that estimates the distances to the nearest existing facilities for the customers and the numbers of influenced customers for the candidate locations, and gradually refines these estimations to obtain k candidate locations with the largest influence values. We use two R-trees and an R-tree variant to index the three datasets. The **estimations** and the refinement are performed during a traversal on the trees (**expanding** tree nodes), where the *importance* value is introduced to help achieve a best first tree traversal while branch and bound techniques are introduced to **prune** the search space. We call this algorithm the *Estimation Expanding Pruning (EEP)* algorithm.

EEP indexes the set of existing facilities F and the set of candidate locations C with two R-trees tr_F and tr_C, respectively. It indexes the set of customers M with an R-tree variant called the *aggregate R-tree (aR-tree)* [11] tr_M, where a tree node stores the number of data objects that are enclosed by its MBR in addition to what is stored in a regular R-tree node. These three trees are traversed simultaneously. The traversal is managed using three auxiliary lists L_M, L_F and L_C, which store objects (either nodes or data objects) from tr_M, tr_F and tr_C, respectively. For an object O in an auxiliary list, we define its *importance* value with its estimated distance to its nearest existing facility (if O is an object in the customer tree tr_M) or estimated number of influenced customers (if O is an object in the existing facility tree tr_F or the candidate location tree tr_C). We use the importance value to determine which objects should be accessed first, and which objects should be pruned. Every object in these lists stores some *influence relation information* that indicates its position relation with the objects in other lists and will be used for importance value computation. Initially, each of these three lists contains only the root node of the corresponding tree. Then, these three lists are accessed repeatedly in the order of L_M, L_F and L_C.

Every time a list L is accessed, EEP computes the importance value $imp(O)$ for every object O contained in L. The object in L with the largest importance value is accessed first. If the accessed object O is a node, the following 5 steps are performed: (i) O's child objects (either nodes or data objects) use O's influence relation information to compute their own influence relation information according to their positions; (ii) EEP prunes a child object O' of O if the influence relation information of O' indicates that the data objects enclosed in O' will not influence or be influenced by the data objects enclosed in the objects of other lists; (iii) if O is a node in tr_C, we compute an influence value upper bound for each of its child objects O', denoted by $maxinf(O')$, which is an upper bound of the influence value for all the candidate locations enclosed by O'. We prune O' if $maxinf(O') < inf_{pr}$, where inf_{pr} is an influence threshold value used for pruning objects in tr_C from further consideration. (iv) the unpruned child objects are used to update the influence relation information of the objects in other lists and further prune some of the objects. (v) EEP removes O and inserts the unpruned child objects of O into L. If the accessed object O is a candidate location $c \in C$, EEP computes the maximum and minimum possible influence values for c, denoted by $maxinf(c)$ and $mininf(c)$, and compare $mininf(c)$ with the largest $maxinf$ value, inf_{mx}, for all the objects in L_C. If $mininf(c) \geq inf_{mx}$, then c is put in the query answer set TOPINF(K,M,F,C) and the pruning influence value inf_{pr} is updated to $mininf(c)$. If O is an existing facility or a customer, there is no computation required for processing O. However, we cannot simply remove O because there are objects in other lists that are related to O. Therefore, EEP skips it and continues to access objects in other lists. The algorithm terminates when TOPINF(K,M,F,C) is filled with k candidate locations and no other candidate location can have larger influence values than those obtained so far.

[2]We assume the R-tree [6] (or its variant), although our algorithms apply to any hierarchical index.

5 Bounding Influence Pruning

This section proposes a second strategy to estimate and refine the influence values for candidate locations. This strategy employs the concept of *influencing region*, which is a region computed from the existing facilities near a candidate location c to enclose all the customers who are influenced by c. The number of customers in this region forms an upper bound for c's influence value. By gradually refining the influencing region (**pruning** customers), we reduce c's **influence** value upper **bound** until we get c's exact influence value.

We present a branch and bound algorithm called the *Bounding Influence Pruning (BIP)* algorithm that utilizes the above strategy to process the TILS query. Like the EEP algorithm, the BIP algorithm also indexes the set of candidate locations C and the set of existing facilities F with two R-trees tr_C and tr_F. It indexes the set of customers M with an aR-tree tr_M. BIP also takes a best first approach. The algorithm uses a priority queue Q_C to perform a best first traversal on tr_C, where each queue element is a node N_C from tr_C, associated with a set of *relevant F nodes* $N_C.R_F$ from tr_F for influencing region computation, a set of relevant M nodes $N_C.R_M$ from tr_M for influence value estimation, and an estimated upper bound for the influence values of the candidate locations enclosed by the MBR of N_C as N_C's priority. To simplify the discussion, we call this upper bound the influence value upper bound of N_C. Initially, Q_C only contains the root node of tr_C, denoted as $root_C$, with $root_C.R_F = \{root_F\}$ and $root_C.R_M = \{root_M\}$, where $root_F$ and $root_M$ denote the root nodes of tr_F and tr_M.

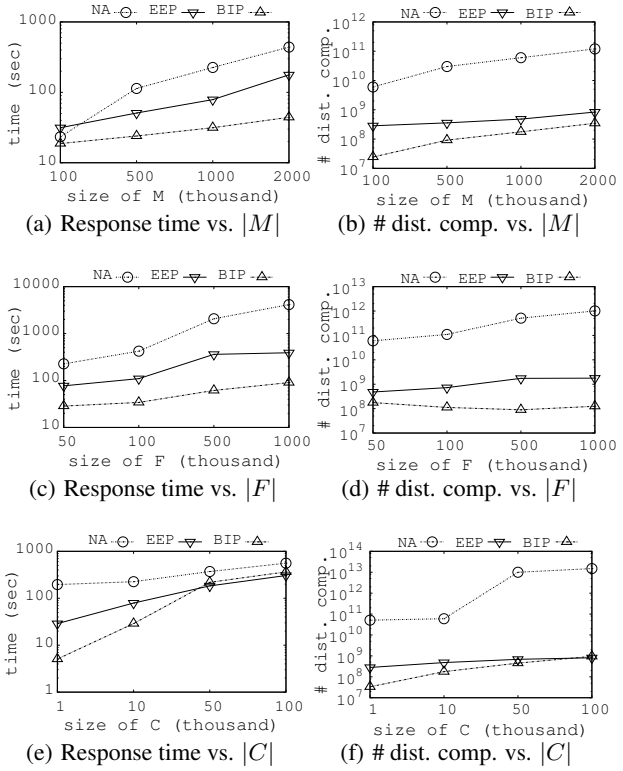
Every time Q_C is accessed, the first node N_C in Q_C is popped out. If N_C is a non-leaf node, the algorithm (i) retrieves N_C's child nodes, (ii) constructs their own sets of relevant nodes according to $N_C.R_F$ and $N_C.R_M$, (iii) computes their influencing regions and influence value upper bounds, (iv) removes a child node if its influence value upper bound is less than the smallest influence value of the candidate locations in TOPINF(K,M,F,C), inf_{pr}, and (v) pushes the unpruned child nodes into Q_C. If N_C is a leaf node, the algorithm (i) retrieves the candidate locations indexed in N_C, (ii) computes their own influencing regions and exact influence values using $N_C.R_F$ and $N_C.R_M$, (iii) inserts a candidate location into TOPINF(K,M,F,C) if its influence value is larger than inf_{pr}, and (v) updates inf_{pr}. The algorithm stops when $Q_C = \emptyset$.

6 Performance Study

6.1 Experimental Settings

All experiments were conducted on a PC with a 2GHz CPU and 2GB RAM. R-trees and aR-trees are used to index the datasets. With practical cardinalities of the three datasets, the total data size is less than 100MB. Given the current computer memory size, it is reasonable to assume that all the datasets reside in the memory and our performance measurement focuses on the total response time. We also measure the number of distance metric computations of the algorithms, which is a good indicator of their pruning power.

We conduct experiments on both synthetic and real datasets. Syn-



(a) Response time vs. $|M|$ (b) # dist. comp. vs. $|M|$

(c) Response time vs. $|F|$ (d) # dist. comp. vs. $|F|$

(e) Response time vs. $|C|$ (f) # dist. comp. vs. $|C|$

Figure 2: Effect of dataset cardinality on the total response time

thetic datasets are generated in a space domain of 1000×1000. The dataset cardinalities range from 1,000 to 2,000,000. To simulate real-life scenarios, where residents and facilities' distributions are skewed while candidate locations' distribution is relatively uniform, we generate M and F with Zipfian distribution with $\alpha = 1.2$, and C with uniform distribution. To verify the effect of the value of k, we use values ranging from 10 to 5,000. As previous studies [13, 7, 19] on main memory databases show, the tree node size of main memory index has a significant impact on the index performance. Therefore, we vary the node size ranging from 1K to 4K to study the effect of node size. By default, we use $k = 10$ and 2K as the tree node size. The real dataset we use is the North East dataset [12]. which contains 123,593 postal addresses representing three metropolitan areas of the USA. We uniformly sample from this dataset to generate M, F and C, with dataset cardinalities ranging from 500 to 100,000.

6.2 Effect of Dataset Cardinality

Figure 2 shows the results of the experiments varying the dataset cardinalities on the synthetic datasets. When varying the cardinality of one dataset, we set the cardinalities of the other two datasets to their default values. The default values used are $|M|$ = 1M, $|F|$=50K and $|C|$ = 10K.

From the figure, we can see that both the EEP and BIP algorithms consistently outperform the NA algorithm in terms of the total response time and the number of distance metric computations. This is because of the pruning techniques used by EEP and BIP to reduce the search space. We can also make other observations as follows. (i) With the increase in the customer set cardinality, EEP and BIP keep relatively stable performance, while NA's performance deteriorates drastically (please note the logarithmic scale in the figures). The reason is that NA sequentially scans all the datasets, and the dataset cardinality directly affect the number of distance computations. In contrast, EEP and BIP use pruning techniques to keep relatively small search spaces and thus achieve much better performance. (ii) With the increase in the number of existing facilities, the number of distance metric computations of BIP becomes smaller. This is because BIP uses the set of F to prune some of the nodes in tr_F as well as tr_M. More existing facilities means larger search space can be pruned and BIP achieves better performance.

We have also performed similar experiments on real datasets, experiments on the effect of tree node size, and experiments on the effect of the value of k. The comparative performance of the different methods are very similar to the above experiments. We omit presenting the results due to space limit.

7 Conclusions

We formulated the top-k most influential location selection query and conducted a comprehensive study on processing this query. We first analyzed the basic properties of this query type and proposed a naive algorithm (NA) to process the query. However, the NA algorithm is inefficient due to repeated scanning on datasets. Motivated by this, we explored geometric properties of spatial data objects, and proposed techniques to prune the search space. This resulted in two algorithms, the EEP algorithm and the BIP algorithm. Experimental results show that the proposed pruning techniques are effective and the proposed algorithms outperform the naive algorithm significantly. In most cases, BIP performs better than EEP.

8 Acknowledgments

We thank Google for funding a travel award that enabled Jianzhong Qi to attend the conference. This work is supported by the Australian Research Council's Discovery funding scheme (project number DP0880250). Zeyi Wen is supported by the Commonwealth Scientific and Industrial Research Organization (CSIRO).

9 References

[1] E. Achtert, H.-P. Kriegel, P. Kröger, M. Renz, and A. Züfle. Reverse k-nearest neighbor search in dynamic and general metric databases. In *EDBT*, 2009.

[2] S. Cabello, J. M. D, S. Langerman, and C. Seara. Reverse Facility Location Problems. In *CCCG*, 2006.

[3] M. A. Cheema, X. Lin, W. Zhang, and Y. Zhang. Influence Zone : Efficiently Processing Reverse k Nearest Neighbors Queries. In *ICDE*, 2011.

[4] Y. Chen and J. M. Patel. Efficient Evaluation of All-Nearest-Neighbor Queries. In *ICDE*, 2007.

[5] Y. Du, D. Zhang, and T. Xia. The Optimal-Location Query. *Advances in Spatial and Temporal Databases*, 3633:163–180, 2005.

[6] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD*, 1984.

[7] R. A. Hankins and J. M. Patel. Effect of node size on the performance of cache-conscious b+-trees. In *SIGMETRICS*, 2003.

[8] J. Huang, Z. Wen, M. Pathan, K. Taylor, and R. Zhang. Ranking Locations for Facility Selection based on Potential Influences. In *the 37th Annual Conference of the IEEE Industrial Electronics Society*, 2011.

[9] J. Huang, Z. Wen, J. Qi, R. Zhang, J. Chen, and Z. He. Top-k most influential location selection. *http://ww2.cs.mu.oz.au/~zeyiw/TR_TkMIL.pdf*.

[10] F. Korn and S. Muthukrishnan. Influence Sets Based on Reverse Nearest Neighbor Queries. *ACM SIGMOD Record*, 29(2):201–212, June 2000.

[11] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao. Efficient OLAP Operations in Spatial Data Warehouses. *Advances in Spatial and Temporal Databases*, 2121:443–459, 2001.

[12] R.-T. Portal. http://www.rtreeportal.org/.

[13] J. Rao and K. A. Ross. Cache conscious indexing for decision-support in main memory. In *VLDB*, 1999.

[14] I. Stanoi, M. Riedewald, D. Agrawal, and A. E. Abbadi. Discovery of Influence Sets in Frequently Updated Database. In *VLDB*, 2001.

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

[16] W. Wu, F. Yang, C. Y. Chan, and K.-L. Tan. Continuous Reverse k-Nearest-Neighbor Monitoring. In *Proc. of MDM*. Ieee, Apr. 2008.

[17] T. Xia, D. Zhang, E. Kanoulas, and Y. Du. On Computing Top-t Most Influential Spatial Sites. In *VLDB*, 2005.

[18] R. Zhang, D. Lin, K. Ramamohanarao, and E. Bertino. Continuous Intersection Joins Over Moving Objects. In *ICDE*, 2008.

[19] R. Zhang and M. Stradling. The hv-tree: a memory hierarchy aware version index. *Proc. VLDB Endow.*, 3:397–408, September 2010.