Ranking Locations for Facility Selection Based on Potential Influences

Jin Huang[†1] Zeyi Wen[‡2] Mukaddim Pathan[#3] Kerry Taylor[#4] Yuan Xue[‡5] Rui Zhang[‡6]
[†]South China University of Technology, China
[#]Commonwealth Scientific and Industrial Research Organization (CSIRO), Australia
[‡]University of Melbourne Australia
[1]jin.h@iojin.com, [2]wenzeyi@gmail.com,[3,4]{Mukaddim.Pathan, Kerry.Taylor}@csiro.au,[5,6]{yuaxue,rui}@csse.unimelb.edu.au

***Abstract*—We study the following location ranking problem. Given a set of existing facilities and a set of customers, we rank a set of candidate locations based on their potential influences, where the potential influence of a candidate location is defined as the number of customers the candidate location can attract. This problem is important in deciding where to establish a new facility. We formulate the problem as the *potential influence location ranking query* and analyze its basic properties. The analysis shows that the query is computationally expensive and efficient query processing techniques are needed. We propose a *nearest location circle* algorithm and a *Voronoi diagram* based algorithm to process the query. Experiments on both real and synthetic datasets show that the proposed algorithms are effective and efficient.**

***Index Terms*—Decision support system, facility location selection, nearest neighbor query, ranking query**

I. Introduction

Suppose a convenience store chain is to open a new store and would like it to attract as many customers as possible to maximize its profit. Assume that customers are attracted by their respective nearest convenience stores. Then the new store should be open at a place that is nearest to the largest number of customers. Usually, there are a set of candidate locations for the establishment of the new store. We define the number of customers that the new store can attract if it is established at candidate location c as the potential influence of c, and study the problem of ranking the candidate locations based on their potential influences. The ranking result can be fed to further evaluation procedure to decide where to establish the new store. The location ranking problem has many applications since many businesses or governmental organizations that manage large numbers of facilities may face the problem of adding new facilities (e.g., McDonald's may want to add a restaurant to compete with other fast food chains such as KFC).

In general, the above location ranking problem can be described as follows. Given a set of existing facility locations, a set of residential customer locations, and a set of candidate locations for establishing a new facility, the *potential influence location ranking problem* ranks the candidate locations based on their influences. For example, in Fig. 1 where stars denote candidate locations, pentagons denote existing facilities and circles denote customers, $c_1, ..., c_6$ as six candidate locations whose influence values are 2, 2, 4, 3, 2 and 3. Therefore, the ranking order of the candidate locations is c_3, c_4, c_6, c_1, c_2, c_5.

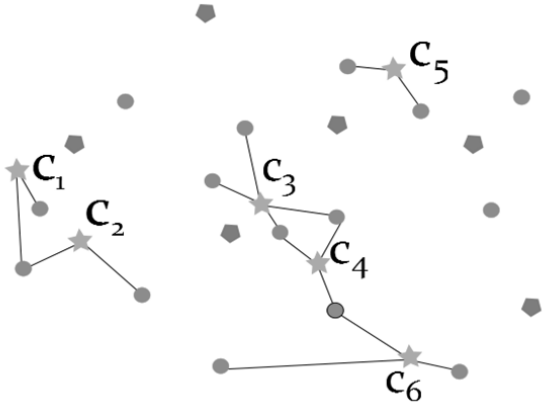

Fig. 1. Example for influence ranking

Note that the potential influence location ranking problem provides a primary result for deciding which location should be used for establishing the new facility. This problem could also be extended to support decisions on non-geographic location decision problems such as product pricing and profile market analysis. For example, when introducing a new model of laptop to the market, a company may want to rank all candidate models based on their potential of attracting customers. This potential can be modeled as the influence where the distance is the distance between the expectations of customers and the offer of the product. Thus, the ranking problem is practical for many decision making related applications in real world and designing efficient algorithms for it is essential.

In this paper, we formulate the above described problem as the *potential influence location ranking query* and propose algorithms to process the query efficiently. Contributions of the paper are as follow:

- We propose a novel location ranking query for deciding where to establish a new facility.
- We propose a nearest location circle algorithm and a Voronoi diagram based algorithm to process the query.
- We perform extensive experiments to validate the proposed algorithms and to show their efficiency.

The rest of the paper is organized as follows. We first

review related works in Section II, then formally define the ranking query in Section III. We introduce a nearest location circle algorithm and a Voronoi diagram based algorithm in Section IV. In section V, experimental results on the proposed algorithms are presented. Finally, we conclude the paper in Section VI.

II. Related Works

Our location ranking query is closely related to the *reverse nearest neighbor (RNN)* query [1], which retrieves the objects perceiving the query object as one of their nearest neighbors. It has many applications in *geographic information systems (GIS)*, military simulation systems, and decision support systems, etc. Thus, this query type is widely studied [2, 3, 4, 5, 6, 7]. Based on whether the query objects and the retrieved objects belong to the same dataset, the RNN query can be classified into the *Monochromatic RNN (MRNN)* query and the *Bichromatic RNN (BRNN)* query. The MRNN query retrieves RNNs of the query object from the dataset that contains the query object. The TPL algorithm [6] is an efficient the MRNN query processing algorithm in terms of both response time and I/O cost. The BRNN query retrieves objects from datasets that do not contain the query object. The TPL algorithm cannot be applied straightforwardly to process the BRNN query since it evolves mutual pruning between the query object and retrieved objects. Korn and Muthukrishnan [1] propose a naive solution to handle the BRNN query by computing the nearest distances between objects in two sets, drawing a circle for each potential reverse nearest object, and then checking which circles enclose the query object. Stanoi et al. [5] process BRNN by introducing a Voronoi diagram based pruning algorithm. Chemma et al. [7] extend this Voronoi diagram based algorithm to find an influence zone on-the-fly for every query object. Objects in the influence zone of a query object form the set of the reverse k nearest neighbors of the query object.

There are also studies of the RNN query under various problem settings. For example, the continuous BRNN query focuses on continuously monitoring the minimum distance relationships [9, 10]. It differs from many other queries such as the continuous intersection join query [8], which finds intersection relations between moving objects. A most recent work on monitoring RNN for moving objects is [11], which gives a detailed review on existing techniques and proposes advanced solutions for problems in both Euclidean space and spatial networks.

Using the cardinality of an object's BRNN set as an indicator of the object's influence, several studies have given results in different facility location problems. In [12] and [13], the authors aim at finding the top-t most influential objects in a given region. Their problem settings differ from ours in that *they select influential locations from existing facilities in a given region, while we rank locations in a candidate location set based on the influence of a new facility if added there*. This difference makes their priority queue and *k-d tree* based approach inapplicable in our problem. Wong et al. [14] propose a nearest location circle based solution to find the optimal region for a new facility to be establish in. This solution cannot solve our problem directly since it finds a region instead of suggesting a optimal candidate location. Ghaemi et al. study the problem in networks. A more recent study [16] proposes two branch and bound algorithms on selecting top-k most influential locations for a new facility. However, instead of *picking up k superior locations, our problem here centers on ranking all candidates based on their potential influences*. Our ranking query is a more general query. Therefore, solutions tuned for top-k selection in that paper do not apply.

III. Preliminaries

We assume a set C of candidate locations, a set F of existing facility locations and a set M of customer locaitons.

Definition 1 (**Bichromatic Reverse Nearest Neighbor**) *Given two location datasets F and M, for a location f ($f \in F$), its **bichromatic reverse nearest neighbors**, denoted as $BRNN(f, F, M)$, is a subset of M containing every location m ($m \in M$) that views f as its nearest neighbor, i.e., $BRNN(f, F, M) = \{m \in M \mid \forall f' \in F \setminus \{f\}, d(f, m) \leq d(f', m)\}$, where $d(f, m)$ is the Euclidean distance between f and m.*

Definition 2 (**Influenced Relationship**) *Given two location datasets F and M, for a location f ($f \in F$) and a location m ($m \in BRNN(f, F, M)$), m is **influenced** by f.*

Definition 3 (**Influence Value**) *Given two location datasets F and M, for a location f ($f \in F$), its **influence value**, denoted as $Inf(f, F, M)$, is the number of its bichromatic reverse nearest neighbors in M, i.e. $Inf(f, F, M) = \mid BRNN(f, F, M) \mid$.*

Definition 4 (**Potential Influence Value**) *Given three location datasets C, F and M, for a location c ($c \in C$), its **potential influence value**, denoted as $Inf_P(c, F, M)$, is the influence value it earns if it is added to F, i.e. $Inf_P(c, F, M) = Inf(c, F \cup \{c\}, M)$.*

Definition 5 (**Potential Influence Location Ranking Query**) *Given three location datasets C, F and M, the **potential influence location ranking query** ranks locations in C according to their potential influence values in a descending order.*

Note that the potential influence value of a candidate location is computed by assuming that the candidate location is added to the existing facility set, which differs from previously studied problems, where influence values of existing facilities are the research focus. Applying conventional algorithms for BRNN queries to compute potential influence values will result in forming facility datasets for each candidate location and is therefore inefficient. To avoid such problem, we propose algorithms that can compute the influence values without forming a facility datasets for every candidate location.

IV. Proposed Solution

In this section, two algorithms for processing the potential influence location ranking query are introduced.

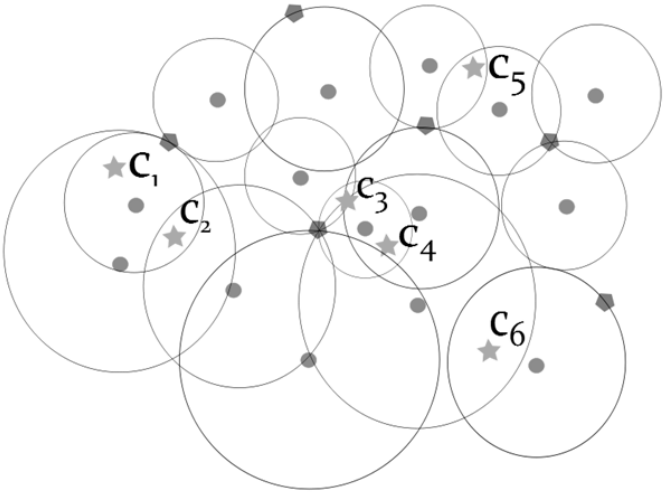

Fig. 2. An example of the NLC Algorithm

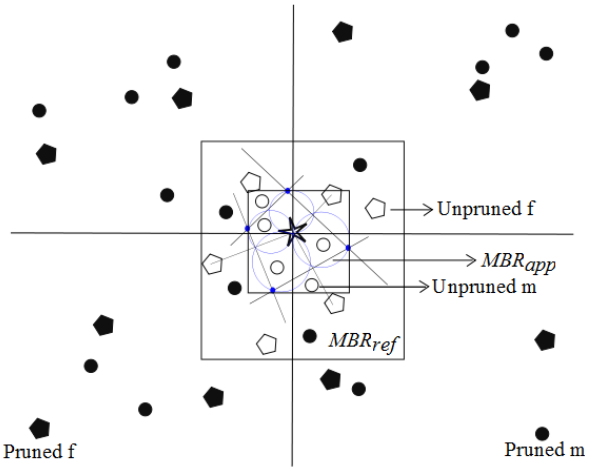

Fig. 3. An example for VAR on computing influence value for one c

A. A Nearest Location Circle (NLC) Algorithm

NLC is based on the observation that a candidate location c can influence a customer location m if and only if the Euclidean distance between c and m, denoted as $d(c, m)$, is no larger than the minimum distance between m and any existing facility f, denoted by $m.dist$.

Theorem 1. *If $d(c, m) \leq m.dist$, then m is influenced by c.*

Proof If $d(c, m) \leq m.dist$, then for any object $s \in F \cup \{c\}$, $d(s, m) \geq d(c, m)$. According to Definitions 1 and 2, m is in $BRNN(c, F \cup \{c\}, M)$. Therefore, m is influenced by c.

Based on Theorem 1, NLC works as follows. First, NLC computes $m.dist$ for every customer location m, then for every candidate location c, NLC checks every customer location m whether $d(c, m) \leq m.dist$. If so, NLC increases the potential influence value of c by 1.

Figure 2 gives an example for the algorithm, where the number of circles enclosing c represents the potential influence value of c. In this example, the potential influence values of c_1, c_2, c_3, c_4, c_5, c_6 are 2, 3, 4, 3, 2 and 2, respectively. Algorithm 1 gives the pseudo-code of NLC, where $c.Inf_P$ denotes the potential influence value of c. Straightforwardly, the algorithm's computational complexity is $O(\mid F \mid\mid M \mid + \mid C \mid\mid M \mid + \mid C \mid \log(\mid C \mid)) = O(\mid F \mid\mid M \mid + \mid C \mid\mid M \mid)$.

Algorithm 1: Nearest Location Circle Algorithm

```
for all m ∈ M do
  for all f ∈ F do
    if m.dist > d(f, m) then
      m.dist ← d(f, m)
for all c ∈ C do
  for all m ∈ M do
    if m.dist ≥ d(c, m) then
      c.Inf_P ++
sort C based on c.Inf_P
```

B. A Voronoi-Approximate-Refinement (VAR) Algorithm

VAR tries to improve the query processing efficiency by reducing the search space for finding candidate location's BRNNs. To compute the influence value of a candidate location c, instead of considering all customer locations in M, VAR only checks the the customer locations that are close to m, since customer locations faraway may be influenced by the existing facilities and thus cannot be influenced by c. The idea is that we use the geometric relationship between c and the existing facilities to prune some customer locations from influence value computation. Furthermore, since only the existing facilities that are close to c can affect which customers are influenced by c, we also use the restricted region on the customer locations to prune faraway existing facilities when finding which existing facilities affect the influence value of c.

The detailed steps of VAR are as follows:

a) For each candidate location c, use c as the origin of a coordinate system whose two axes parallel with the original axes, , and find four nearest existing facilities in the four quadrants.
b) Draw the perpendicular bisectors between c and the found existing facilities. The four intersections between the bisectors are called the Voronoi points.
c) Use the midpoints between c and the Voronoi points as the centers and the distances between them as the diameters to draw four circles. Bound these circles with a minimal bounding rectangle (MBR), denoted as MBR_{app} (cf. Fig. 3).
d) Double the distances between c and the lower left and upper right vertices of MBR_{app} and form a new MBR, denoted as MBR_{ref}.
e) For each existing facility $f \in MBR_{ref}$ and customer $m \in MBR_{app}$, compute $m.dist$ and check whether $d(c, m) \leq m.dist$. If so, increase the influence value $c.Inf_P$ by 1.

f) After (a) to (e) are performed on every candidate location, sort the candidate locations based on the influence values.

The following theorems guarantee the correctness of VAR.

Theorem 2. *If a customer m is influenced by a candidate location c, then $m \in MBR_{app}$.*

Proof If m is influenced by c, then according to Definition 2, $m \in BRNN(c, F \cup \{c\}, M)$. According to the definition of the Voronoi diagram, m is in the Voronoi diagram formed by c and the existing facilities. Since this Voronoi diagram is contained in MBR_{app}, $m \in MBR_{app}$.

Theorem 3. *Given a customer $m \in MBR_{app}$, if an existing facility f influences m, then $f \in MBR_{ref}$.*

Proof Assume that f is outside of MBR_{ref} and f influences $m \in MBR_{app}$. Let the midpoint between c and f be p. Then p lies outside of MBR_{app}. Since f influences $m \in MBR_{app}$, the bisector between c and f intersects with the aforementioned diameters. Let o' be the intersecting point, R be the corresponding circle, and $l_{c,p}$ be the diameter. Points p, c, and o' form a right rectangle with circumscribing circle R' maintaining a diameter $l_{c,o'}$. Since o' is between c and o, R must contain R'. Thus, $p \in MBR_{app}$ which is in contradiction with the assumption. Therefore, the theorem is proved.

According to Theorem 2 and Theorem 3, only the customers in MBR_{app} and the existing facilities in MBR_{ref} are needed by further computation. Since MBRs are supported natively by the R-Tree, we can index M and F with two R-Trees, and then perform range queries to retrieve relevant customers and existing facilities when needed. Furthermore, instead of computing all customers in MBR_{app}, we can additionally perform pruning after retrieving a customer from the R-Tree by checking whether it lies in the approximated Voronoi diagram formed by the aforementioned 4 Voronoi points, since only the customers in this region can be influenced by c. We call the algorithm with this enhancement VAR-VE. An example for VAR on computing the potential influence value for a specific candidate location is shown in Fig. 3. Algorithm 2 gives the pseudo-code of VAR-VE. Simply ignoring line 6 of the algorithm will result in the algorithm of VAR. VAR-VE's computational complexity is between $O(\mid C \mid\mid F \mid)$ and $O(\mid C \mid\mid F \mid\mid M \mid)$, depending on the pruning performance on M and F.

Algorithm 2: Voronoi Approximate Refinement Algorithm

```
Index F and M with R-Trees
for all c ∈ C do
    c as the origin, find nearest f in four quadrants
    build MBR_app and MBR_ref as in Section IV.B
    for all m ∈ MBR_app do
        if m lies in region formed by Voronoi points then
            for all f ∈ MBR_ref do
                if m.dist > d(f, m) then
                    m.dist ← d(f, m)
            if m.dist ≥ d(c, m) then
                c.Inf_P ++
sort C based on c.Inf_P
```

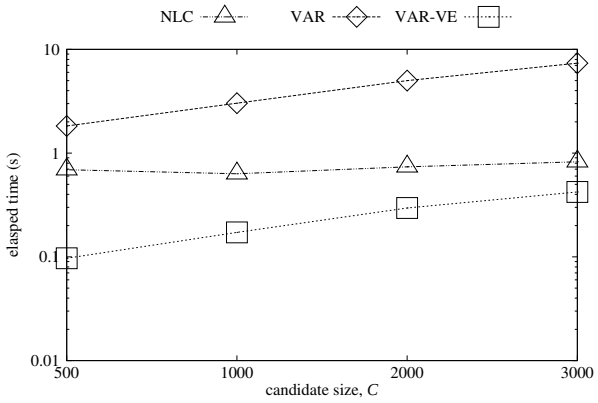

Fig. 4. CPU time versus. Varying C, F=6000, M=24000

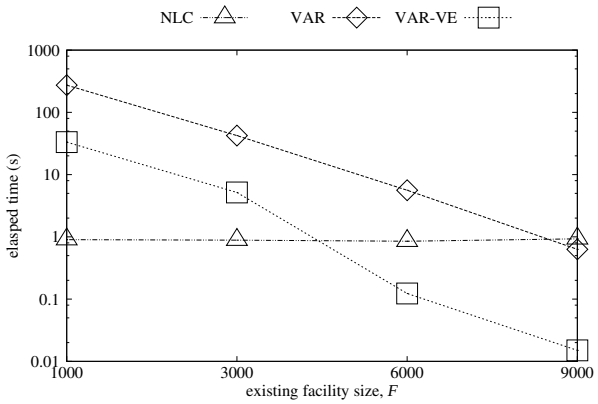

Fig. 5. CPU time versus. Varying F, C=100, M=6000

V. Experiments

We empirically study the performance of the proposed algorithms using real datasets and synthetic datasets with Uniform and Gaussian distributions.

A. Experimental Settings

The experiments are conducted on a personal computer with a 2.1GHz CPU and 2.0 GB main memory. Implementations are in Visual Studio 2008 using C++. Real datasets are locations in US and Mexico retrieved from the R-Tree Portal [17]. Locations in synthetic datasets are generated in a domain of 2048×2048. We use both Uniform and Gaussian distributed synthetic datasets. For Gaussian distribution, the parameters used are $\mu = 0$ and $\delta^2 = 256$. To confirm the effectiveness of the enhancement on VAR of pruning customer locations using approximated Voronoi diagrams, we test both VAR (without line 6 in Algorithm 2) and VAR-VE.

B. Experiments on Real Datasets

Fig. 4 to 6 present the results on real datasets. In all these figures, VAR-VE outperforms VAR by several orders of magnitude, which verifies the effectiveness of the enhancement. In what follows, we will focus on comparing NLC with VAR-VE. In Fig. 4, VAR-VE beats NLC by 5 to

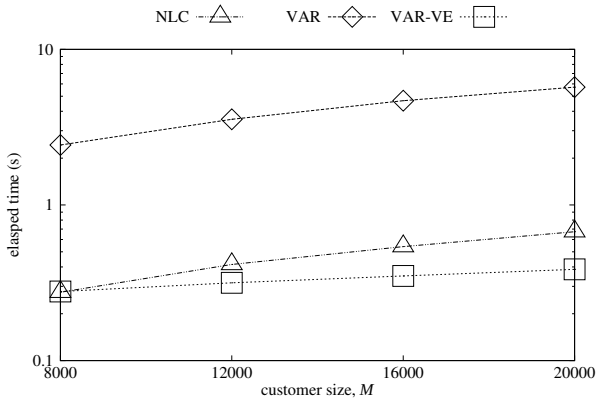

Fig. 6. CPU time versus. Varying F, C=100, M=6000

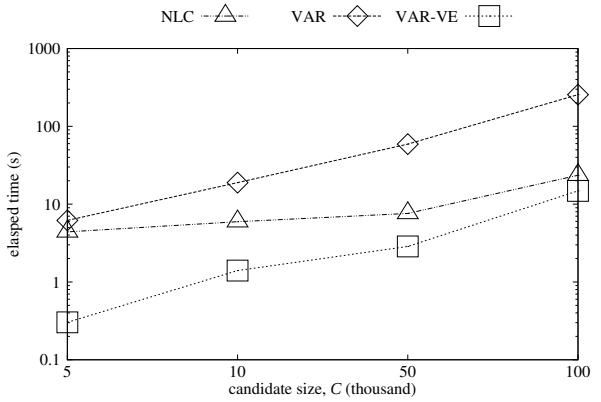

Fig. 7. CPU time versus. Varying C, Uniform, F=10K, M=100K

10 times. Also, the performance of NLC varies little when C grows. This is because NLC maintains a complexity of $O(\mid F \mid\mid M \mid + \mid C \mid\mid M \mid)$, and in scenario corresponding to Fig. 4, C is constantly smaller than F, leaving $\mid F \mid\mid M \mid$ as the major bottleneck.

In Fig. 5, it is observed that NLC beats VAR-VE when F is relatively small, while it loses its advantages to VAR-VE as F grows. Since VAR-VE relies on F to prune irrelevant existing facilities and customers from further computations, it is expected that the larger the F, the smaller the distances between each candidate location and its nearest existing facilities in the 4 quadrants, and the smaller MBR_{app} and MBR_{ref}. This promises better pruning efficiency and therefore enables VAR-VE to improve its complexity from $O(\mid C \mid\mid F \mid\mid M \mid)$ to $O(\mid C \mid\mid F \mid)$. Fig. 6 shows that the performance of both NLC and VAR-VE degrades as M becomes larger, while VAR-VE constantly outperforms NLC.

C. Experiments on Synthetic Datasets

The experimental results on Uniform datasets are presented in Fig. 7 to 9, and the results on Gaussian datasets are shown in Fig. 10 to 12.

As shown in these figures, all algorithms' performance degrades as datasets grow. VAR-VE again beats VAR with

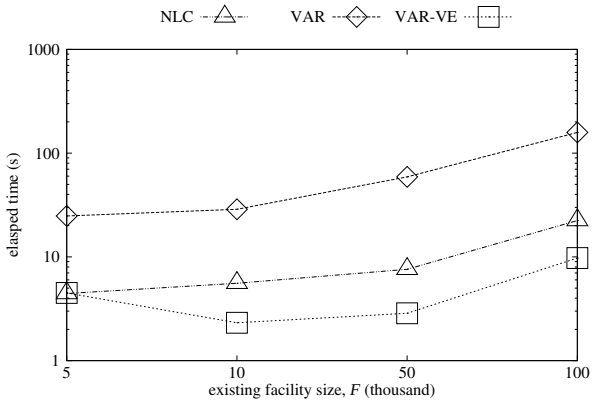

Fig. 8. CPU time versus. Varying F, Uniform, C=10K, M=100K

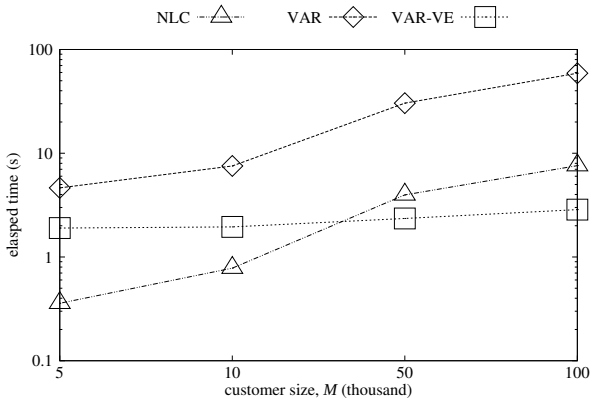

Fig. 9. CPU time versus. Varying M,Uniform, C=10K, F=10K

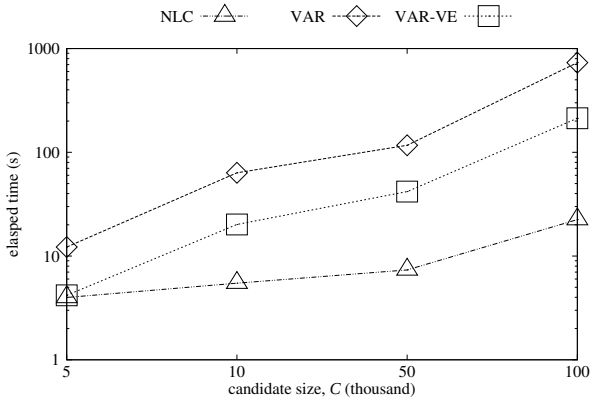

Fig. 10. CPU time versus. Varying C, Gaussian, F=10K, M=100K

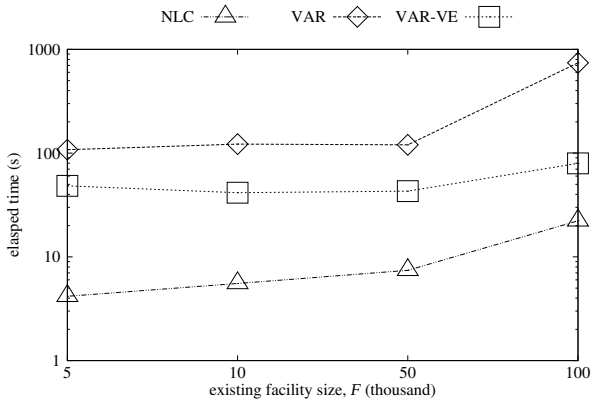

Fig. 11. CPU time versus. Varying F, Gaussian, C=10K, M=100K

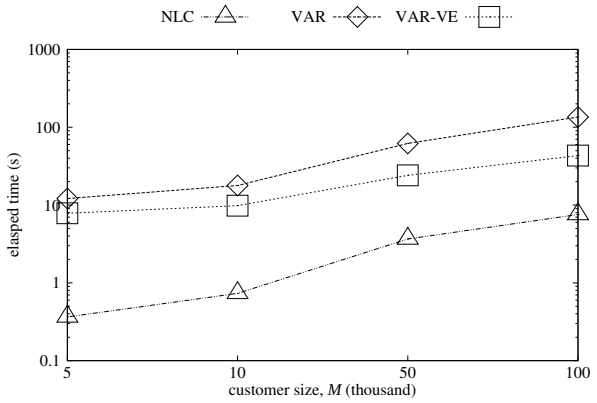

Fig. 12. CPU time versus. Varying M,Gaussian, C=10K, F=10K

significant advantages.

In Fig. 8, as F becomes larger, the cost of VAR-VE first decreases and then increases again. This trend is explained as follows. As F grows, the pruning becomes more effective, which reduces the complexity from $O(\mid C \mid\mid F \mid\mid M \mid)$ to $O(\mid C \mid\mid F \mid)$. But when F becomes huge, the growth of the dataset size overweights the benefits it brings on computational complexity. In Fig. 9, the performance of VAR-VE stays almost static as M grows, which confirms the effectiveness of pruning by using Voronoi-Approximate-Refinement and Voronoi-pruning techniques. Consider Fig. 8 and Fig. 11. NLC spends almost the same time on processing datasets of the same sizes, while the performance of both VAR and VAR-VE becomes uncompetitive in Gaussian datasets (Fig.10 to 12). This can be explained by the fact that VAR-VE relies on range queries on R-Trees to prune unnecessary records from computation, which means the cardinality of the range query results influences the pruning performance. In Gaussian distributed datasets, regions near the center of the distribution may contain larger numbers of records, which affects the efficiency of the Voronoi pruning process.

In summary, NLC is preferable when datasets are huge or with Gaussian distribution (Fig. 10 to Fig. 12) since its performance is almost irrelevant to dataset distribution, while in other scenarios, (Fig. 4 to Fig. 9) VAR-VE is superior in terms of CPU time. This enables us to choose a better solution when facing different real-life application scenarios. Moreover, we observe that no proposed algorithm achieves global superiority. Thus there is considerable room of improvement for further studies.

VI. Conclusions and Future Work

In this paper we studied the potential influence location ranking query for new facility establishment. The study made the contributions of formulating the query, proposing two methods to process the query, and performing experiments to validate the effectiveness of the proposed methods and to show the algorithm efficiency. As for future work, we will study ranking the candidate locations to support the establishment of multiple new facilities at the same time, and extending the query to other problem settings such as considering movements of the customers or using network distance in distance computation.

Acknowledgment

This work is supported in part by the research funding of the Commonwealth Scientific and Industrial Research Organization (CSIRO), towards Zeyi Wen's PhD study.

References

[1] F. Korn and S. Muthukrishnan, "Influence sets based on reverse nearest neighbor queries," *ACM SIGMOD Record*, vol. 29, Jun. 2000.
[2] M.L. Yiu and N. Mamoulis, "Reverse nearest neighbors search in ad hoc subspaces," *TKDE*, vol.19(3), 2005, pp. 643-654.
[3] W. Wu, F. Yang, C. Chan, K. Tan, "FINCH: Evaluating Reverse k-Nearest-Neighbor Queries on Location Data," in *VLDB*, 2008.
[4] M.A. Cheema, X. Lin, W. Wang, W. Zhang, J. Pei, "Probabilistic Reverse Nearest neighbor Queries on Uncertain Data," *IEEE TKDE*, vol.22(4), 2010, pp. 550-564.
[5] Stanoi, M. Riedewald, D. Agrawal, and A.E. Abbadi, "Discovery of influence sets in frequently updated database," in *VLDB*, 2001
[6] Y. Tao, D. Papadias, X. Lian, and X. Xiao, "Multidimensional reverse kNN search," *The VLDB Journal*, vol. 16, Dec. 2005.
[7] M.A. Cheema, X. Lin, W. Zhang, and Y. Zhang, "Influence Zone : Efficiently Processing Reverse k Nearest Neighbors Queries," in *ICDE*, 2011.
[8] R. Zhang, D. Lin, K. Ramamohanarao, E. Bertino, "Continuous Intersection Joins Over Moving Objects," in *ICDE*, 2008.
[9] T. Xia, D. Zhang, "Continuous Reverse Nearest Neighbor Monitoring, in *ICDE*, 2006.
[10] J. M. Kang, M. F. Mokbel, S. Shekhar, T. Xia, D. Zhang, "Continuous Evaluation of Monochromatic and Bichromatic Reverse Nearest Neighbors," in *ICDE*, 2007.
[11] M. A. Cheema, W. Zhang, X. Lin, Y. Zhang, X. Li, "Continuous Reverse Nearest Neighbors Queries in Euclidean Space and in Spatial Networks", *The VLDB Journal*, 2011.
[12] T. Xia, D. Zhang, E. Kanoulas, and Y. Du, "On computing top-t most influential spatial sites," in *VLDB*, 2005.
[13] Y. Du, D. Zhang, and T. Xia, "The optimal-location query," in *SSTD*, 2005.
[14] R.C.-wing Wong, M.T. Ozsu, P.S. Yu, A.W.-chee Fu, and L. Liu, "Efficient method for maximizing bichromatic reverse nearest neighbor," *PVLDB*, 2009.
[15] P. Ghaemi, K. Shahabi, J.P. Wilson, and F. Banaei-Kashani, "Optimal network location queries," In *ACM GIS*, 2010.
[16] J. Huang, Z. Wen, J. Qi, R. Zhang, J. Chen, H. Zhen, "Top-k Most Influential Location Selection," in *ACM CIKM*, 2011.
[17] R-Tree portal, http://www.rtreeportal.