Parameterized Spatial Query Processing based on Social Probabilistic Clustering

Liang Tang[1] Haiquan Chen[2] Wei-Shinn Ku[1] Min-Te Sun[3]
[1]Dept. of Computer Science and Software Engineering, Auburn University, USA
[2]Dept. of Mathematics and Computer Science, Valdosta State University, USA
[3]Dept. of Computer Science and Information Engineering, National Central University, Taiwan

ABSTRACT

In this paper, we propose two parameterized frameworks, namely the Uniform Watchtower (UW) framework and the Hot zone-based Watchtower (HW) framework, for the evaluation of spatial queries on large road networks. The motivation of this research is twofold: (1) how to answer spatial queries efficiently on large road networks with massive POI data and (2) how to take advantage of social data in spatial query processing. In UW, the network traversal terminates once it acquires the Point of Interest (POI) distance information stored in watchtowers. In HW, by observing that users' movements often exhibit strong spatial patterns, we employ probabilistic clustering to model mobile user check-in data as a mixture of 2-dimensional Gaussian distributions to identify hot zones so that watchtowers can be deployed discriminatorily. Our analyses verify the superiority of HW over UW in terms of query response time.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Application—*spatial databases and GIS*

General Terms

Algorithms

Keywords

Spatial query, Road networks

1. INTRODUCTION

Due to recent advances in wireless communication technology, mobile devices (e.g., smart phones, tablets, etc.) with Internet access and positioning chips are significantly increasing in popularity. Moreover, many vendors provide various map and navigation services (e.g., Google Maps and Bing Maps). As a result, we are witnessing the fast growth of location-based services (LBS), which allow mobile users to issue spatial queries from their mobile devices based on user-specified locations in a ubiquitous manner. Among all spatial queries, the k nearest neighbor (kNN) query [11, 8] is one of the most important building blocks used to realize LBS with more complex queries [9, 1, 12]. Therefore, how to evaluate kNN queries has received considerable attention from both industry and academia in the past decade. One of the major challenges is how to provide efficient evaluation of kNN queries for large road networks with massive Points of Interest (POIs), especially when the desired POIs are far away from the query point. Two of the well-known solutions that attempt to address this challenge are ROAD [7] and Islands [6]. However, the significant overhead limits their applications in practice. As listed in Table 1, ROAD requires about one hour for index setup for the North America road network while Islands needs more than 30 GB for index storage. Consequently, neither of them scale well towards large road networks.

Approach	Setup time	Storage cost
ROAD	$\approx$ 60 mins	$\approx$ 35 MB
Islands	$\approx$ 243 mins	> 30 GB

Table 1: The overheads of ROAD and Islands on the North America road network (179,178 edges and 175,812 nodes) with 100,000 random synthetic POIs.

In this paper, we propose two watchtower-based frameworks, namely the Uniform Watchtower framework and the Hot zone-based Watchtower framework, to speed up spatial query processing on large road networks. What distinguishes our work from other research efforts is that in our proposed frameworks (1) watchtowers are deployed on road networks as distance signatures in a tunable granularity without the need to maintain any tree-based structure for network traversal and (2) we incorporate mobile users' movement information (i.e., geo-social check-in data) into the construction of watchtowers by using probabilistic clustering. The contributions of this work are as follows:

- We propose the Uniform Watchtower (UW) framework to deploy watchtowers in a parameter-tunable manner for efficient evaluation of spatial queries.
- In order to further elevate query efficiency, we provide the Hot zone-based Watchtower (HW) framework by incorporating mobile users' movement information into the construction of watchtowers. Based on users' check-in data collected from popular geo-social web sites, we deploy watchtowers in hot zones and non-hot zones discriminatorily by using probabilistic clustering.

The rest of this paper is organized as follows. The Uniform Watchtower framework is introduced in Section 2. In Section 3, we describe the Hot zone-based Watchtower framework. The evaluation of kNN queries under our proposed frameworks is presented

SIGSPATIAL'14, November 04 - 07 2014, Dallas/Fort Worth, TX, USA

http://dx.doi.org/10.1145/2666310.2666428.

in Section 4. In Section 5, we discuss the performance difference between the two proposed frameworks. Section 6 concludes the paper and provides future research directions.

2. THE UNIFORM WATCHTOWER FRAMEWORK

The fundamental idea of using watchtowers for spatial query processing is to distribute and store the distance information (distance signature) of each POI on road networks so that the POI lookup can terminate once it encounters enough watchtowers to answer the query. For any POI o_i, we use watchtowers to store the distance information from this current watchtower to o_i. With watchtowers, a query can be answered efficiently by checking only a few watchtowers close to the query point. In this section, we elaborate on how to distribute watchtowers to road networks in an approximately uniform fashion, which involves two steps, (1) anchor point deployment and (2) watchtower setup. Table 2 summarizes the notations in this paper.

2.1 Anchor Point Deployment

Given a road network $G = (V, E)$, we distribute anchor points over G based on the following two rules. (1) For any node with a degree greater than 2 or equal to 1, we create an anchor point on it. (2) For two adjacent anchor points i and j obtained in (1), if their network distance is greater than λ, we add an anchor point every λ distance starting from i (or j) along the adjacent road segments so that the distance of two adjacent anchor points is always less than λ. Consequently, in G, any query point is able to find at least one anchor point within the distance of $\lambda/2$, where λ is a tunable parameter set by applications or users. Take Figure 1(a) as an example. We first establish anchor points at n_1, n_2, n_3, and n_6, because their degrees are greater than 2. If λ is set to be 14 miles, for any segment longer than 14 miles, anchor points are deployed every 14 miles along the segment from one end.

2.2 Watchtowers Construction

The naive way to set up watchtowers is to treat each anchor point, which we obtained in the above subsection, as a watchtower to store distance information of POIs. However, this design is neither practical nor efficient because it requires a prohibitively huge amount of storage space. Here, we propose a parameter-tunable approach to establish watchtowers. Specifically, for every l adjacent anchor points, we set up a watchtower.

The following describes in detail how we distribute watchtowers. Given a POI o_i, a Dijkstra-based expansion from o_i is launched. The expansion searches the whole graph and sets up a watchtower for every l anchor points. When an anchor point is selected as watchtower t_j for o_i, a distance tuple $(o_i, |SP(o_i, t_j)|)$ is added to this watchtower, where o_i is the POI's ID and $|SP(o_i, t_j)|$ is the shortest distance from o_i to watchtower t_j. We repeat the above expansion process for every POI until we set up watchtowers for all POIs over the entire graph. Note that there could be many such distance tuples inserted into a watchtower because an anchor point might be selected as a watchtower by more than one POI. If more than one tuple is added to a watchtower (i.e., this watchtower is shared by more than one POI), we sort all the distance tuples by their distance $|SP(o_i, t_j)|$ in ascending order. Take Figure 1(b) as an example. The distance between two adjacent anchor points is 1 and l is set to 3, i.e., only one anchor point serves as the watchtower for every three adjacent anchor points. As a result, the distance information for o_1, o_2, and o_3 are distributed along the network. In watchtower t_2, three distance tuples are stored, indicating that the distances from t_2 to o_1, o_2 and o_3 are 6, 2.5, and 6, respectively.

To save the storage cost, we actually do not need to store all distance tuples because there usually exists an upper bound of k (denoted as b_k) for kNN queries. If $b_k = 10$, only the top-10 distance tuples need to be stored in each watchtower because those tuples are sufficient to answer the query. Therefore, the storage complexity of our watchtower-based approach can be estimated as $O(b_k \frac{\sum w_e}{\lambda})$, where $\sum w_e$ is the sum of the distances of all the edges in the road network.

Symbol	Meaning
t	A watchtower
λ	The maximum distance between two adjacent anchor points
$SP(.,.)$	The shortest path between two points
$\|SP(.,.)\|$	The distance of a shortest path
$\mathbb{SP}(.,.)$	The possible shortest path between two points
$\|\mathbb{SP}(.,.)\|$	The distance of $\mathbb{SP}(.,.)$
$Dijkstra(u, w)$	Dijkstra's algorithm-based search of up to distance w from node u
l	The number of anchor points between two adjacent watchtowers (the selecting parameter)
r_d	The maximum network distance for a hot zone

Table 2: Symbolic notations.

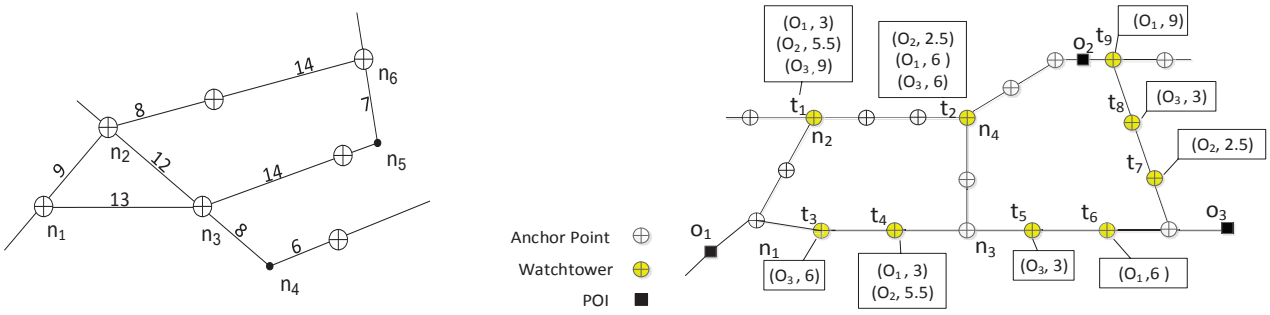
3. THE HOT ZONE-BASED WATCHTOWER FRAMEWORK

Many papers on social networks, such as [2, 10], reveal the fact of "Location Sparsity", which means that most users who subscribe to location-based services move only within *limited areas*, i.e., people's movement often exhibits a strong spatial pattern. This idea inspires us to refine the Uniform Watchtower (UW) framework into the Hot zone-based Watchtower (HW) framework. In this design, we first collect users' movement data from a popular geo-social service, Gowalla, and then cluster those data to derive hot zones of people's movements (i.e., the geographic areas where people are most likely to appear and launch spatial queries). The clustering process yields a number of hot zones. Afterwards, based on the hot zones, we build watchtowers discriminatorily for hot zones and non-hot zones over road networks.

3.1 Probabilistic Clustering on Social Data

People's movement information can be obtained from popular geo-social services, e.g., Gowalla and Foursquare. Here, we elaborate on how to apply Model-based Clustering (Mclust) [5, 4] to analyze the structure of user check-in data to identify hot zones. We do not adopt the k-means (where k stands for the number of clusters) algorithm because we have no prior knowledge of the number of clusters. We employ Mclust, where different clustering models are compared and the expectation-maximization (EM) algorithm [3] is used to maximize the likelihood. In our study, Mclust is used as an efficient algorithm to estimate k, the number of clusters. Given the upper bound of the number of clusters, denoted as K^+, Mclust iteratively decides whether a cluster should be split further into smaller clusters by calculating the Bayesian Information Criterion. The details of this splitting process are illustrated in [5]. When Mclust is executed, k increases gradually until it is stable. The k value is then returned and reported as the number of clusters found by Mclust.

In our approach, the check-in information is modeled as a mixture of 2-dimensional gaussian distributions $\mathcal{N}(\mu_i, \sum_i)$, where $\mu_i = (\mu_i.x, \mu_i.y)$ is the center of the guassian distribution and $\sum_i$ is the covariance matrix. Since there is no clear evidence for "non-symmetric" distributions on X and Y dimensions in geo-social



(a) Anchor point deployment.

(b) Watchtower construction.

Figure 1: Anchor point deployment and watchtower construction on example road networks.

networks [10], $\sum_i$ can be assumed to be diagonal and isotropic. The probabilistic density function of a cluster $C_i(\mu_i, \sigma_i)$, returned by Mclust, for a particular location L can be represented by Equation 1.

$$f(L.x, L.y) = \frac{1}{2\pi\sigma_i^2} e^{\frac{(\mu_i.x - L.x)^2}{-2\sigma_i^2} + \frac{(\mu_i.y - L.y)^2}{-2\sigma_i^2}} \quad (1)$$

We cluster users' check-in locations using Mclust to determine the hot zones for a road network. We treat each cluster as a hot zone. Specifically, a hot zone is the region centering at point μ_i for cluster C_i with a radius of r_d, which is a network distance threshold predefined for all hot zones. Figure 2 displays the clustering results of the check-ins for California road networks (where 8,523 check-ins were sampled from Gowalla[1]). In Figure 2, we varied K^+ from 5, 10, to 20, leading to distinct numbers of resulting clusters. Each cluster, highlighted with a distinct color, corresponds to a hot zone.

3.2 Hot Zone-based Watchtower Construction

To build a Hot zone-based Watchtower framework, we deploy all watchtowers in such a manner that watchtowers in hot zones are constructed densely to speed up query processing while watchtowers in non-hot zones are distributed sparsely to save storage space. This design leads to a significantly higher query performance without the need of more storage space. Here, we illustrate the difference between the Uniform Watchtower framework and the Hot zone-based Watchtower framework. In the former, for each POI, we deploy watchtowers uniformly on the road network. In the latter, we analyze and leverage social data with the objective of allocating watchtowers to more important (i.e., populated) areas where queries are more likely to be launched.

The following describes the index construction process. For each cluster C_i, we obtain the set V_{ci} ($V_{ci} \subset V$), where for any node v in V_{ci}, the network distance between μ_i and v must be less than or equal to r_d. As a result, we can obtain the graph $G^{'} = (V^{'}, E^{'})$, where $V^{'}$ is the union of V_{ci} for all the clusters, and $E^{'}$ is a subset of E including all the edges of which the two end points are in $V^{'}$. Next, we launch a Dijkstra's algorithm-based expansion from each POI and check if the currently visited edge e is in $G^{'}$ or not. If e is in $G^{'}$, we set up watchtowers on e with higher density (by using a relatively smaller l). Otherwise, if e is not in $G^{'}$, we deploy watchtowers on e sparsely (by applying a relatively larger l).

4. SPATIAL QUERY PROCESSING

The evaluation of kNN queries is based on the bidirectional Dijkstra's algorithm. Because the distance between two adjacent anchor points is at most λ, the distance of two adjacent watchtowers must be smaller than or equal to $l\lambda$ (recall that we set up watchtowers every l anchor points). Therefore, for any POI o_i, $Dijkstra(q, l\lambda)$ is able to find at least one distance tuple for o_i in the shortest path from q to o_i, which can be denoted as $SP(o_i, q)$.

At each watchtower, backward search results (POI distance information) are stored. A priority queue Q_o of size k is maintained as well. Because $Dijkstra(q, l\lambda)$ will return *at least two* watchtowers for any POI, we calculate the top k tuples for each watchtower and decide if Q_o needs to be updated. The updating rule is based on computing the shortest distance from q to o_i, which can be represented by:

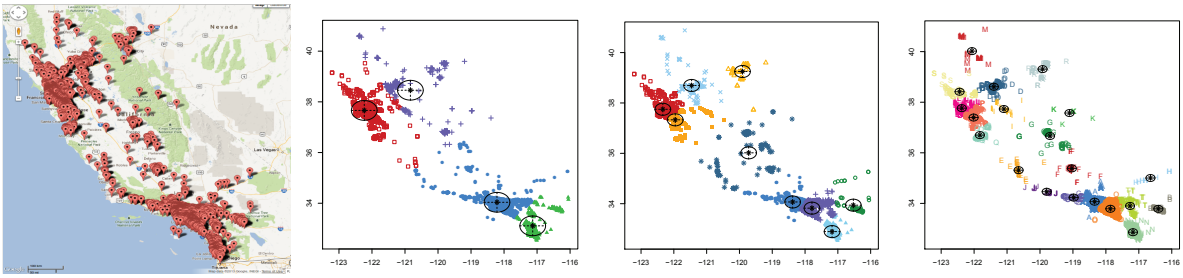
$$\begin{aligned} |SP(o_i, q)| &= min\{|\mathbb{SP}(o_i, q)|\} \\ &= min\{|SP(o_i, t)| + |SP(t, q)||t \in Dijkstra(q, l\lambda)\} \end{aligned} \quad (2)$$

In Equation 2, $|SP(o_i, t)|$ denotes the distance between POI o_i and watchtower t stored for the o_i tuple, and $|SP(t, q)|$ represents the distance from the query point q to watchtower t. The sum of the two items corresponds to a possible path $\mathbb{SP}(o_i, q)$ between POI o_i and q while $|\mathbb{SP}(o_i, q)|$ is their network distance. Assume that from some watchtower t, we retrieve the top k POI tuples and one of them is o_i. Now we need to decide if the tuple of o_i needs to be inserted into Q_o. First, we check if $|\mathbb{SP}(o_i, q)|$ is less than the maximum distance in Q_o. If it is true, we add o_i into Q_o in the case that o_i is not in Q_o and update the shortest distance accordingly. We update Q_o for every POI tuple in t until the search is expanded to $l\lambda$ distance. Because the search $Dijkstra(q, l\lambda)$ is able to return at least two watchtowers of each POI, one of which must be in $SP(o_i, q)$, we can derive the distance $|SP(o_i, q)|$ for every POI. In other words, the search needs to expand up to only the distance of $l\lambda$. The top k POIs in Q_o are the result of the kNN query.

It is worth noting that some POIs might be within the distance of $l\lambda$ from the query point q. In this case, the Dijkstra search actually might not have to expand as far as $l\lambda$ to answer a kNN query. Therefore, for those POIs, we also insert their distance information into the priority queue Q_o during the search. The search stops once there is enough POI distance information to answer the query. This leads to the early termination of the search.

To answer a kNN query, in the worst case, $Dijkstra(q, l\lambda)$ is needed. The time complexity includes the Dijkstra search cost and the priority queue update cost. $O(VlogV)$ is the complexity of the Dijkstra search while the maximum number of watchtowers encountered by $Dijkstra(q, l\lambda)$ can be estimated as $\frac{\sum w'}{\lambda}$, where $\sum w'$ is the sum of the distances of all the edges traversed by $Dijkstra(q, l\lambda)$. If we keep only the top b_k POI distance tuples in each watchtower, $O(\frac{b_k \sum w'}{\lambda})$ is the cost of the priority queue

[1]http://snap.stanford.edu/data/loc-gowalla.html



(a) Check-ins (b) $K^+ = 5$ (c) $K^+ = 10$ (d) $K^+ = 20$

Figure 2: Mclust with various K^+ values.

update. Therefore, the time complexity of our proposed algorithm is $O(V'logV' + \frac{\sum w'}{\lambda} b_k)$, where V' is the number of the vertices traversed by $Dijkstra(q, l\lambda)$.

5. PERFORMANCE COMPARISON OF UW AND HW

In this section, we discuss the superiority of HW over UW in terms of query response time. Without loss of generality, we assume the road network G is a 2D Manhattan network in a square area consisting of only horizontal and vertical edges. Suppose that l_* is the selecting parameter in the UW framework while l_h and l_{nh} ($l_h < l_{nh}$) represent the selecting parameters in hot zones and non-hot zones in the HW framework. For a kNN query, if the query point is in a hot zone, in the worst case, we have to search $l_h\lambda$ to answer the query. On the contrary, if the query point is outside any hot zone, in the worst case, the search for $l_{nh}\lambda$ distance is needed to get the result. Let W_G be the sum of the weights of all the edges in G. Assuming that α is the ratio of W_G to the sum of the weights of edges in all hot zones, $\frac{W_G}{\alpha}$ is the sum of the weights of edges in all hot zones while $W_G - \frac{W_G}{\alpha}$ is the sum of the weights of edges in all non-hot zones. Because the index size is proportional to the number of watchtowers, assuming that UW and HW occupy the same amount of storage space, we can obtain Equation 3

$$\begin{cases} \frac{W_G}{l_*} = \frac{\frac{W_G}{\alpha}}{l_h} + \frac{W_G - \frac{W_G}{\alpha}}{l_{nh}} \\ l_{nh} = \beta l_h \end{cases} \tag{3}$$

where β is the ratio of l_{nh} to l_h. Based on Equation 3, the ratio of l_* to l_h, denoted as H, can be calculated as Equation 4.

$$H = \frac{l_*}{l_h} = \frac{\alpha\beta}{\alpha + \beta - 1} \tag{4}$$

As discussed before, the time complexity of our watchtower-based query processing algorithm is $O(V'logV' + \frac{\sum w'}{\lambda} b_k)$, where V' is the number of the vertices traversed by $Dijkstra(q, l\lambda)$. To answer a query issued at point q, in the worst case, UW requires $Dijkstra(q, l_*\lambda)$ while HW needs $Dijkstra(q, l_h\lambda)$. Therefore, if the query point is in a hot zone, HW is able to reduce the query response time by a factor of H^2, compared with UW. According to Equation 4, the performance gain of HW over UW can be approximated using Equation 5.

$$\begin{aligned} speedup &= (\frac{\alpha\beta}{\alpha + \beta - 1})^2 > (\frac{\alpha\beta}{\alpha + \beta - 1 + |\beta - \alpha| + 1})^2 \\ &= \frac{(\alpha\beta)^2}{4(MAX\{\alpha, \beta\})^2} = \frac{(MIN\{\alpha, \beta\})^2}{4} \end{aligned} \tag{5}$$

6. CONCLUSION

In this paper, we introduce two watchtower-based parameter-tunable frameworks for efficient spatial query processing on large road networks. To elevate query performance, mobile users' check-in data are taken into account during the construction of watchtowers on road networks by discovering populated areas (hot zones) using the probabilistic clustering algorithm. In addition, we compare the performance difference between the two proposed frameworks by theoretical analysis. In the future, we plan to extend our solution to support other spatial query types.

Acknowledgements

This research has been funded in part by the National Science Foundation grant CNS-0917137 and the Faculty Scholarship award from Valdosta State University.

7. REFERENCES

[1] H. Chen, W.-S. Ku, M.-T. Sun, and R. Zimmermann. The partial sequenced route query with traveling rules in road networks. *GeoInformatica*, 15(3):541–569, 2011.

[2] Z. Cheng, J. Caverlee, and K. Lee. You are where you tweet: a content-based approach to geo-locating twitter users. In *CIKM*, pages 759–768, 2010.

[3] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, pages 1–38, 1977.

[4] C. Fraley and A. E. Raftery. How many clusters? Which clustering method? Answers via model-based cluster analysis. *Comput. J.*, 41(8):578–588, 1998.

[5] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002.

[6] X. Huang, C. S. Jensen, and S. Saltenis. The Islands Approach to Nearest Neighbor Querying in Spatial Networks. In *SSTD*, pages 73–90, 2005.

[7] K. C. K. Lee, W.-C. Lee, and B. Zheng. Fast object search on road networks. In *EDBT*, pages 1018–1029, 2009.

[8] K. C. K. Lee, W.-C. Lee, B. Zheng, and Y. Tian. ROAD: A new spatial object search framework for road networks. *IEEE Trans. Knowl. Data Eng.*, 24(3):547–560, 2012.

[9] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On Trip Planning Queries in Spatial Databases. In *SSTD*, pages 273–290, 2005.

[10] R. Li, S. Wang, H. Deng, R. Wang, and K. C.-C. Chang. Towards social user profiling: unified and discriminative influence model for inferring home locations. In *KDD*, pages 1023–1031, 2012.

[11] H. Samet, J. Sankaranarayanan, and H. Alborzi. Scalable network distance browsing in spatial databases. In *SIGMOD Conference*, pages 43–54, 2008.

[12] J. Zhang, W.-S. Ku, X. Jiang, X. Qin, and Y.-L. Hsueh. Evaluation of Spatial Keyword Queries with Partial Result Support on Spatial Networks. In *MDM*, pages 279–282, 2013.