Circle of Friend Query in Geo-Social Networks*

Weimo Liu[1], Weiwei Sun[1,**], Chunan Chen[1], Yan Huang[2], Yinan Jing[1], and Kunjie Chen[1]

[1] School of Computer Science, Fudan University
Shanghai 201203, China
{liuweimo,wwsun,ynjing,chenkunjie}@fudan.edu.cn, huangyan@unt.edu
[2] Department of Computer Science and Engineering, University of North Texas
Denton, TX 76203-5017, USA
chenchunan@fudan.edu.cn

Abstract. Location-Based Services (LBSs) are becoming more social and Social Networks (SNs) are increasingly including location components. Geo-Social Networks are bridging the gap between virtual and physical social networks. In this paper, we propose a new type of query called *Circle of Friend Query* (*CoFQ*) to allow finding a group of friends in a Geo-Social network whose members are *close* to each other both socially and geographically. More specifically, the members in the group have tight social relationships with each other and they are constrained in a small region in the geospatial space as measured by a "diameter" that integrates the two aspects. We prove that algorithms for finding the *Circle of Friends* (*CoF*) of size k is NP-hard and then propose an ε-approximate solution. The proposed ε-approximate algorithm is guaranteed to produce a group of friends with diameter within ε of the optimal solution. The performance of our algorithm is tested on the real dataset from Foursquare. The experimental results show that our algorithm is efficient and scalable: the ε-approximate algorithm runs in polynomial time and retrieves around 95% of the optimal answers for small ε.

Keywords: Circle of friend query, geo-social networks.

1 Introduction

The wide availability of wireless communication and positioning enabled mobile devices allows people to access Location-Based Services (LBSs) whenever and wherever they want. On the other hand, Social Networks (SNs) are penetrating people's daily lives. For example, by January 2011, Facebook has more than 600 million active users. Half of these users login everyday and more than 150 million active users access Facebook through mobile devices. LBS and SN are increasingly integrated together into Geo-Social Networks (GSNs). In GSNs, e.g. Foursquare, users form a social network, can communicate, and share their

* This research is supported in part by the National Natural Science Foundation of China (NSFC) under grant 61073001.
** Corresponding author.

S.-g. Lee et al. (Eds.): DASFAA 2012, Part II, LNCS 7239, pp. 126–137, 2012.

locations. Recently many SNs are adding support of locating friends nearby. In fact, there is a trend of increasingly bridging the gap between virtual and physical worlds. Aligning with this trend, this paper proposes a new query type called Circle of Friend Query (*CoFQ*) in a GSN. A *k-CoFQ* allows a user q with location $q.l$ to identify k "circle of friends" in "spatial proximity". Here "circle of friends" refers to the group of $k+1$ users including q with small pairwise social distances in a social network. And "spatial proximity" is to constrain the diameter of the point set formed by the locations of the $k+1$ users. The *k-CoFQ* is the integration of social and physical world distances. This query allows a circle of friends with the following two properties to be found: (1) The friends are mutual which is ensured by the small pairwise social distance; (2) The friends are in spatial proximity.There are many challenges in answering *k-CoFQ* in a large GSN. The three major ones are:

1. There are two types of distances in GSNs: social network distance and spatial network distance. The integration of these two requires careful balance of the different metrics;
2. The *k-CoFQ* requires to find a subgraph with the smallest diameter. The NP-hard max-clique problem can be reduced to the *k-CoFQ* problem which makes *k-CoFQ* NP-hard and computational expensive. With spatial proximity considered, it is even more challenging to design a scalable algorithm;
3. It is known that obtaining real social network dataset for validation is very difficult in social network research especially in academic settings.

The paper addresses these challenges and makes the following contributions:

1. We formulate a new and useful query in GSNs, namely, *k-CoFQ*. This query enables many applications in a GSN. Examples include group sports, social deal, friend gathering, and community service. To bridge the gap between virtual and physical world, this paper uses a normalization method with a empirical parameter to combine the two metric values.
2. We prove that the *k-CoFQ* is NP-hard. We propose an ε-approximate algorithm to *k-CoFQ*. We identify an upper-bound and lower-bound when searching k-circle of friends in a social network. This allows the the ε-approximate algorithm prunes most of the search space. For the candidate groups remained, the ε-approximate algorithm narrows the range of the optimal group's diameter gradually by checking the candidates in a binary search fashion, tightening the bounds along the way until the distance between bounds is less than ε. The suboptimal result is guaranteed to be with in ε of the optimal solution.
3. We algebraically analyze the cost of the algorithms using small world assumption [16]. We test the performance of our algorithm on real dataset from Foursquare. Experimental results demonstrate efficiency and scalability of the algorithms proposed.

The rest of this paper is organized as follows: Section 2 overviews the research on GSNs. Section 3 defines the circle of friend query and proves it is NP-hard.

Section 4 proposes an ε-approximate algorithm in social networks. Section 5 introduces the ε-approximate algorithm in GSNs. Experimental results are provided in section 6. In section 7, we conclude our work and discuss the future work.

2 Related Work

The research on location-based social networks is attracting much attention recently. A main activity is to mine user's location and social network data to retrieve the relationships between the locations and the users. It has been shown that users with short-distance links are often geographically close [15][14]. C. Hung, et al. [8] store users' trajectory profiles by using a probabilistic suffix tree structure. They formulate distance of profiles and uses clustering algorithms to identify communities. M. Lee, et al. [11] focus on the semantic information of location. Their paper proposes a structure that organizes the location information into categories to compute the similarity between users. M. Ye, et al. [18] recommend locations to users based-on their social networks. Their paper proves that there is a co-relationship between friends' ratings on the same location and discusses how to build a location recommendation system based on the co-relationship. The above works attempt to discover some useful Geo-Social information from the history information. In this paper we propose a new kind of query which is very useful in real life applications such as activity planning.

C. Chow, et al. [3] present GeoSocialDB which supports location-based social query operators such as location-based news feed, location-based news ranking, and location-based recommendation. Y. Doytsher, et al. [4] propose a socio-spatial graph model and introduce a query language formed by a set of socio-spatial operators. The socio-spatial operators mentioned in this paper are different from ours. The circle of friend query in our paper aims to find a group based on the social relationship and the location of users.

D. Yang, et al. [17] discuss the problem of finding a group of given number of members from a social network. The group should satisfy the hop and unfamiliar member constraints and make the sum of closeness between the query and each member of the group smallest. Our work aims to find a group with smallest diameter and considers both social network distance and geo-distance.

Finding k points in Euclidean space with minimum diameter or minimum sum of all distance has been studied. It has been shown that polynomial time algorithm is available [1]. However, the problem of finding a k-vertex group with the smallest diameter in a graph has not been studied yet.

Another query to find a group in spatial databases is proposed recently called collective spatial keyword query [2]. It is to find a group of objects that cover all the keywords in the query and minimize the maximal distance between the objects and the sum of distances from the objects to the query location.

In the area of spatial databases, the problem of group nearest neighbor query, also named aggregate nearest neighbor query, is studied in [13][19]. Given a group of k members, the group nearest neighbor query is to find an aggregate point to

minimize the total distance from all the members to the aggregate point. This differs from our work because we aim to find a group to minimize the diameter of a group containing a given point. Furthermore, we consider not only spatial distance but also social network distance.

3 Problem Definition

A social network is modeled as an undirected weighted graph $G(V, E)$ with vertex set V and edge set E. For $u, v \in V$, $(u, v) \in E$ if u and v are friends, and the weight of (u, v), denoted as $w(u, v)$, is defined by their direct interactions such as message exchange or co-authorship. For example, in the case of co-authorship $w(u, v)$ can be defined as $1/\Sigma_{i=1}^{n}\frac{1}{x_i}$, where n represents the number of papers u and v co-write, x_i is the number of the i^{th} paper's authors [12]. The closeness $closeness(u, v)$ between u and v is defined as[10][9]:

$$closeness(u,v) = \begin{cases} w(u,v), & \text{if } (u,v) \in E \\ \text{the accumulative weights of the} \\ \quad \text{shortest path between } u \text{ and } v \text{ in } G, & \text{otherwise} \end{cases}$$

In a GSN, every vertex $v \in V$ is geo-tagged and associated with a location. The geographical distance between u and v is denoted as $dist(u, v)$. This distance can be the common distance measures such as Euclidean or network distances. We use Euclidean distance in this paper.

3.1 Circle of Friend Query (*CoFQ*) in Social Networks

Definition 1 (diameter). *Given a subset of vertices* S *in a graph* G*(*V*,* E*), the* diameter *of* S*, denoted as* $dia(S)$*, is the distance between the farthest pair in* S*:*

$$dia(S) = max_{a,b \in S} closeness(a, b). \tag{1}$$

Problem Statement [*k-circle of friend query in social networks*]. Given a query point q in a social network $G(V, E)$ and the size k, the *k-circle of friend query* ($CoFQ(q, k)$) is to find a set $V' \subseteq V$, satisfying that:

$$q \in V'; \tag{2}$$

$$| V' | = k + 1; \tag{3}$$

$$dia(V') \text{ is the smallest;} \tag{4}$$

The set V' is called the *k-circle of friends* of q, denoted as $CoF(q, k)$.

3.2 Geo-Social Circle of Friend Query (*gCoFQ*)

We denote the diameters of a vertex set S with respect to geometric distance and social closeness as $dia_{geo}(S)$ and $dia_{social}(S)$:

$$dia_{geo}(S) = max_{a,b \in S} dist(a, b), \tag{5}$$

$$dia_{social}(S) = max_{a,b \in S} closeness(a, b). \tag{6}$$

The Ranking Function. Because we want the $dia_{geo}(S)$ and $dia_{social}(S)$ to be small, the combining function of $dia_{geo}(S)$ and $dia_{social}(S)$ needs to be monotonic with respect to both. The linear combination of the two factors is a reasonable method:

$$dist_{gs}(u,v) = \lambda \frac{dist(u,v)}{dia_{geo}(V)} + (1-\lambda)\frac{closeness(u,v)}{dia_{social}(V)}. \tag{7}$$

And the ranking function is

$$f(S) = dia_{gs}(S) = max_{a,b \in S} dist_{gs}(a,b). \tag{8}$$

Here, λ varies according to the users' different demands. Even for the same user, the demand may change at different moment. For example, if a user wants to invite some friends to play football, the geo-distance weights more in the function since the friends nearby are convenient to come together. However if the user wants to hold a party, the closeness is more important. The user probably invites a very close friend although the friend lives a little far. A user can pick up a reasonable λ based on her demand possibly through taking suggestion from the system who learns the value from similar queries.

Problem Statement [*Geo-Social Circle of Friend Query*]. Given a query point q in a GSN $G(V, E)$ and the size k, the *Geo-Social Circle of Friend Query* ($gCoFQ(q, k)$) is to find a set $V' \subseteq V$, satisfying that:

$$q \in V'; \tag{9}$$

$$| V' | = k + 1; \tag{10}$$

$$dia_{gs}(V') \text{ is the smallest;} \tag{11}$$

and the set V' is called the Geo-Social circle of friends of q, denoted as $gCoF(q, k)$.

3.3 NP-Hard Proof

The max-clique problem is to find the largest complete subgraph in an unweighted undirected graph and is NP-Hard. We reduce the max-clique problem to *CoFQ* problem.

Theorem 1. *Finding* CoF(q, k) *is NP-hard.*

Proof. Assume that the circle of friend problem's running time is T. Suppose that all the edges' weights of a weighted undirected graph G are one, we iterate all the vertices in G and let k be 1, 2, ..., n ($n = |G|$). We can get different k_0 in the following condition: if G is a complete graph, $k_0 = n$; if G is not a complete graph, there exists q that makes the *CoF*'s diameter equals to 1 when $k{=}k_0$ and for any $k{>}k_0$ there is no q makes *CoF* problem has a solution that satisfies that the diameter equals to 1. The *CoF* problem's solution for $k{=}k_0$ is the max-clique problem's solution. The algorithm complexity is $O(k{*}n){*}T$, so MAX-CLIQUE $\leq_P$CIRCLE OF FRIEND. So *CoFQ* is also an NP-hard problem.

Theorem 2. *Finding the Geo-Social Circle of Friends is NP-hard.*

Proof. Assume that the running time of *gCoFQ* is *T*. Then set the value of λ as 0. We have

$$dist_{gs}(u,v) = \lambda \frac{dist(u,v)}{dia_{geo}(V)} + (1-\lambda)\frac{closeness(u,v)}{dia_{social}(V)} = \frac{closeness(u,v)}{dia_{social}(V)} \quad (12)$$

Since $dia_{social}(V)$ is constant, the *gCoFQ* is equivalent to *CoFQ*. CIRCLE OF FRIEND $\leq_P$GEO-SOCIAL CIRCLE OF FRIEND. So *gCoFQ* is NP-hard.

4 Algorithm for Circle of Friend Query (*CoFQ*)

In this section, by exploring the upper bound and lower bound of the diameter, we propose an ε-approximate solution to *CoFQ*. We then optimize the algorithm and analyze its complexity.

4.1 Find the Upper Bound and the Lower Bound

The circle of friend result is different from the kNN of the query point. However, the upper bound and the lower bound of the diameter of the circle of friends can be computed based on the kNN result:

Lemma 1 (Upper bound property). *Let* N *be the union of* q *and the set of* q*'s* k*NNs,* D_{max} = dia*(*N*),* M = {u | u ∈ V *and* closeness*(*q, u*)* ≤ D_{max}}*, then* CoF*(*q, k*) is a subset of* M*.*

Proof omitted due to space constraint.

Lemma 2 (Lower bound property). *Let* r *be the distance between* q *and its* k^{th} *NN, then we have* dia*(*CoF*(*q, k*))* ≥ r.

Proof omitted due to space constraint.

Based on Lemma 1, a very straightforward solution to the *CoFQ* is to compute the diameter of every possible group in M and find the one with the smallest diameter. This method needs $\binom{|M|-1}{k}$ times of diameter computation, once for each possible group. The computation time is very expensive.

4.2 ε-approximate Algorithm

We introduce our first algorithm to *CoFQ*. The main idea of this algorithm is to process a binary search on the $\binom{|M|-1}{k}$ groups generated by *M*. Firstly, we check if there is a group with diameter smaller than 1/2 D_{max}. If such a group exits, then the diameter of $CoF(q, k)$ falls in the range $[1/4D_{max}, 1/2D_{max}]$. Then we continue the binary search in the subspace until we get an ε-approximate result. By ε-approximate result, we mean that we are sure that the optimal result $dia(CoF(q, k))$ falls in the range $[\delta^-,\delta^+]$ and $\delta^+ - \delta^- < \varepsilon$. Lemma 3 shows that with a tight upper bound and special access order, the binary-search-based algorithm will prune most of the groups and avoid scanning the whole space.

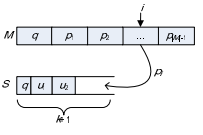

Fig. 1. prune in the ε-approximate algorithm

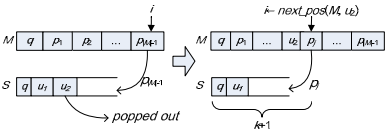

Fig. 2. pop in the ε-approximate algorithm

Lemma 3. *Suppose that we are currently checking if there is a group with diameter smaller than D_{cur}. If there are two points* u, v *in* M *satisfying that* closeness*(*u, v*) > D_{cur}, then any group containing* u *and* v *can be discarded.*

Proof omitted due to space constraint.

Based on Lemma 3, we develop an ε-approximate algorithm. Suppose that the points in M is stored in a list and q is the first element of the list, as shown in Fig. 1. We use a stack S to store the currently checked group. S is initialized as q. Then we add the elements in M to S in sequence. Before adding a element p to S, we check whether or not the diameter of S will become larger than D_{cur} after p is added. Based on Lemma 3, we just have to check the distance between p and the elements in S. For example in Fig. 1, S currently has three elements: q, u_1 and u_2. Before adding p_i to S, we check that if the distance between p_i and q(or u_1, u_2) is larger than D_{cur}. If the above condition is satisfied, p_i is pruned for the current set in S. Otherwise p_i is added to S.

If for the elements in S, adding any other element will lead to a diameter larger than D_{cur}, the top element of S is popped out. In Fig. 2, u_2 is popped out and the element p_j after u_2 in M is the next element to be added to S.

When the size of S becomes k+1, we output the group generated by the elements in S as the currently suboptimal result, denoted as CoF_{cur}. Then we check if the difference between CoF_{cur} and the optimal result is smaller than ε. If so, CoF_{cur} is output as the suboptimal result. Otherwise, the upper bound d_{UP} of $dia(CoF(q, k))$ is set to D_{cur} and we continue to check if there is a group with diameter smaller than 1/2 (D_{cur} + d_{LB}). If all the groups' diameter is larger than D_{cur}, then we continue to check if there is a group with diameter smaller than 1/2 (D_{cur} + d_{UP}).

4.3 Optimized ε-approximate Algorithm

As mentioned above, we use an $|M|\times|M|$ matrix to store the distance between every pair of M in order to compute the diameter efficiently. However, as the value of M becomes larger, both computation time and space cost will increase rapidly. In this section, we propose an optimized algorithm by cutting down the size of the distance matrix. The optimization is based on the following observation.

Observation 1. Suppose that there exists a group whose diameter is smaller than D_{cur}, for a element u in M, if $closeness(u, q) > D_{cur}$, then u will not

appear in the suboptimal result. So the columns or rows containing u will not be used in our binary search process. As shown in Fig. 3, only the sub matrix at the top left corner is useful to our ε-approximate algorithm. We call it "useful space".

Observation 1 can be proved by Lemma 3. As Fig. 3 shows, at the first step of the binary search, we don't have to compute the distance matrix of all the elements in M. Instead, we only compute the *useful space* and check if there is a group whose diameter is smaller than D_{cur}. If such a group doesn't exist, we would have to extend the *useful space* according to the new D_{cur}. The check procedure based on the distance matrix is the same as ε-approximate Algorithm. The only difference is the search space. In ε-approximate Algorithm, the search space is all the elements whose distance to q is smaller than $dia(N)$, while in our second algorithm, the search space depends on D_{cur} and may be extended in the following steps.

5 Algorithm for Geo-Social Circle of Friend Query (*gCoFQ*)

In this section, we explore the problem in **Geo**-Social network environment where both social distance and geometric distance are considered. Firstly, we propose a kNN algorithm in GSNs based on the ranking function. Then a binary search based algorithm to *gCoFQ* is proposed.

The ε-approximate algorithm described in section 4.2 can be applied to *gCoFQ*, too. The first step is to find the kNNs of q ordered by the value of the ranking function. This a multi-objective top-k problem (i.e., the top-k objects are ranked according to the social network distance and spatial proximity). To deal this problem, we propose a Geo-Social kNN algorithm, which is inspired by the NRA algorithm in [5].

5.1 The Geo-Social kNN Algorithm

Definition 2 (Geo-Social kNN query). *Given a query point* q *in a GSN* G*(*V, E*) and the size* k*, the Geo-Social* k*NN query (*GSkNN*(*q, k*)) is to find a set* V *'*∈ V *with* | *V′* |= *k, and for any point* $u \in V'$ *and* $v \notin V'$*,*

$$dist_{gs}(q, u) < dist_{gs}(q, v). \tag{13}$$

The main idea of our algorithm is to search the kNN in the social network and space separately and synchronously. Fig. 4 shows the framework of the algorithm. The kNNs in social networks are searched on the graph $G(V, E)$ (see section 4). While the NNs in geo-space are searched using the R-Tree index [6]. The incremental NN search algorithm [7] are used in this paper.

Like NRA, we append the NNs retrieved in the social network or geo-social space with two attributes: W and B (short for *worst* and *best* score). W means the upper bound of the ranking function and B means the lower bound. We use

a priority queue Q to store the NNs. The elements in Q are sorted by W in ascending order, if W is the same, ties are broken using B, the lower B wins.

Notice that every time a nearest neighbor in social NN_{social} is retrieved, the Euclidean distance can be computed as well using the coordinates of NN_{social}. So we set the value of $NN_{social}.W$ and $NN_{social}.B$ as the exact Geo-Social distance between q and NN_{social}. Meanwhile, when a nearest neighbor in geo-space NN_{geo} is retrieve, the social network distance is unknown. So we compute the value of $NN_{geo}.W$ and $NN_{geo}.B$ with the lower and upper bound of the social network distance. The social network upper bound is set as ∞, while the lower bound $closeness_{LB}$ is initialized as 0 and increases when a new NN is found. For geo-distance, a lower bound $distance_{LB}$ is also maintained. When the lower bound of the ranking function $f(distance_{LB}, closeness_{LB})$ is larger than the last element's in Q, the algorithm ends and the first k elements in Q are output as the top-k result.

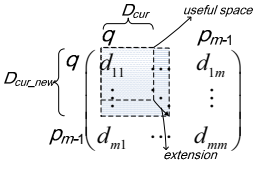

Fig. 3. Useful space in the matrix

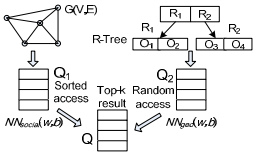

Fig. 4. The Geo-Social kNN algorithm

5.2 The Algorithm for *gCoFQ*

Recall the upper bound property in Lemma 1, the kNN result can be used to prune most of the search space in the social network. This property holds for the *gCoFQ* as well. Suppose that q and its kNNs are stored in N. We compute $dia_{gs}(N)$ and use it as the upper bound. Then we process a range query on the social network with the distance bound of $dia_{gs}(N)$. For every point u lying in this range, we compute $dist_{gs}(u, q)$ using the coordinate of u. If $dist_{gs}(u, q)$ is larger than $dia_{gs}(N)$, u is pruned. Otherwise, u is added to the set M, sorted by $dia_{gs}(N)$. Then we process a binary search on M to get the ε-approximate result, as described in section 4.2.

6 Experiment

We used a real social network from Foursquare, which is a world famous location-based social networking application. The graph consists of 20,550 nodes and 586,424 edges in our experiments. The relation between users and their check-in locations with geographic were crawled from Foursquare by using the open API. Because the API requests' number per hour is limited by the server, we only achieved a subset of the total social network. All experiments were carried out on a Genuine 1.8GHZ CPU desktop computer with 3GB memory and our implementation source code of the algorithms aforementioned were written in C++.

We conducted several experiments for *CoFQ* and *gCoFQ* to: 1) compare the performance of the baseline, the ε-approximate and the optimized ε-approximate algorithm with the increase of k; 2) evaluate the effect of ε on the correctness and the response time.

6.1 *CoFQ* on Social Networks

The Effect of *k*. In this section, we discuss the effect of k when all algorithms use both upper bound aforementioned as the prune condition. As Fig. 5 shows, the ε-approximate and the optimized ε-approximate algorithm significantly outperform the baseline for all value of k. The gap between them even increases considerably as k increase. As our analysis above, the CPU time of the ε-approximate and the optimized ε-approximate algorithm grows in a polynomial way while the Baseline increases in an exponential way due to its huge amount of computation of diameter for every possible group. The Baseline can hardly return any results when k is bigger than 7 in our experiments, which will be considered unacceptable because in real applications such as *groupon*, k is often a relative large number. We also compared the ε-approximate and the optimized ε-approximate algorithm in a relative large extent of k, as is shows in Fig. 6. The optimized ε-approximate algorithm's CPU time still keeps in a low level but the ε-approximate's becomes bigger with increasing k. This is because the ε-approximate use a $|M|\times|M|$ matrix to store the distance between every pair of M. When k becomes larger which will result in the increase of M, the computation time of the distance matrix will increase dramatically. But for the optimized ε-approximate algorithm, only part of the distance matrix (useful space) will be computed, thus the CPU time greatly reduced.

The Effect of ε. Finally, we explore the effect of ε on the correctness and the response time of our algorithm. We used our two ε-approximate algorithms to process 1,000 *CoF* queries whose query points are randomly chosen in the social network. k is set as 6 and ε ranges from 0.001 to 0.1. For each of the queries, we compute the actually *CoF* using the baseline algorithm. We then compare ε-approximate *CoF* with the actual *CoF* and record the number of the correct answers. We define the correct rate $R_{correct}$ and the approximate rate R_{appro} as follows, and the effect of R_{appro} on $R_{correct}$ is shown in Fig. 7.

$$R_{correct} = \frac{num(correctanswers)}{num(totalqueries)} \times 100\% \tag{14}$$

$$R_{appro} = \frac{\varepsilon}{dia(G)} \times 1000‰ \tag{15}$$

6.2 Geo-Social *CoF* Query

The Effect of *k*. Fig. 8 measures the effect of the k (ranging from 3 to 6) on the running time for *gCoFQ*. When k equals to three, three methods make little difference in CPU time. As k increase, the optimized ε-approximate's CPU time still keep in a low level due to its effective exploring way despite the addition

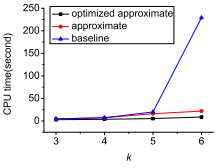

Fig. 5. Small k on the CPU time

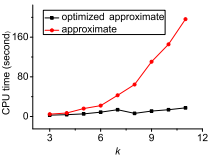

Fig. 6. Large k on the CPU time

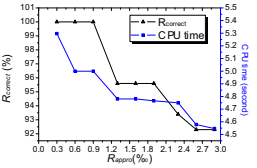

Fig. 7. The effect of ε

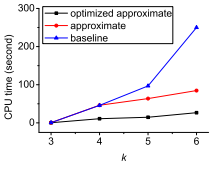

Fig. 8. Small k on the CPU time

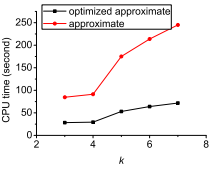

Fig. 9. Large k on the CPU time

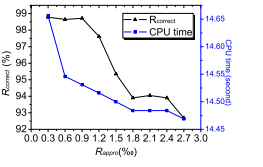

Fig. 10. The effect of ε

computation on the spatial distance. On the other hand, baseline's performance degrades very fast. Again, we still get no result in a reasonable time using baseline, since the method exhausts all possible combinations of the candidate set. And the ε-approximate performs similar as before with using addition 50 seconds mostly because the algorithm need to get the Euclidean space distance.

Fig. 9 illustrates the effect of k on two approximate algorithms. The optimized ε-approximate's CPU time increases in a different way as shown in Fig. 6. With k increase, the method takes more time to calculate the spatial distance for every candidates. The same holds for the ε-approximate algorithm.

The Effect of ε. Fig. 10 shows the effect of ε on the CPU time and the correctness. The approximate ratio is redefined as

$$R_{appro} = \frac{\varepsilon}{dia_{gs}(G)} \times 1000‰. \tag{16}$$

With small ε, most of the results retrieved by the ε-approximate algorithm are the same as the optimal solution. For example, with $R_{appro} = 0.3‰$, all the results got are correct. When R_{appro} gets bigger, the response time becomes less, and the correctness ratio decreases.

7 Conclusion

In this paper, we define a new query in GSNs, namely, the *Geo-Social Circle of Friend Query*. Given a query point in a GSN, the *gCoFQ* is to obtain a group including q and the members are close to each other with respect to Euclidean distance and social relationship closeness. We proved that the problem is NP-hard. After exploring the upper bound and lower bound property of the *gCoF*, we proposed two ε-approximate algorithms to find a suboptimal solution. We tested the performance of our algorithms on real dataset with different parameters. The

response time was reduced from exponential to polynomial. By experimental results, we showed that with $\varepsilon < 0.01$, the ε-approximate algorithms yielded 95% of the correct answers.

An interesting direction for future work is to process Geo-Social queries based on the trajectories of the mobile users. The main challenge is how to calculate the geo-distance between users based on the history of the locations, not only the current locations.

References

1. Aggarwal, A., Imai, H., Katoh, N., Suri, S.: Finding k points with minimum diameter and related problems. Journal of Algorithms 12(1), 38–56 (1991)
2. Cao, X., Cong, G., Jensen, C.S., Ooi, B.C.: Collective spatial keyword querying. In: SIGMOD, pp. 373–384. ACM (2011)
3. Chow, C.Y., Bao, J., Mokbel, M.F.: Towards location-based social networking services. In: LBSN, pp. 31–38. ACM (2010)
4. Doytsher, Y., Galon, B., Kanza, Y.: Querying geo-social data by bridging spatial networks and social networks. In: LBSN, pp. 39–46. ACM (2010)
5. Fagin, R., Lotem, A., Naor, M.: Optimal aggregation algorithms for middleware. In: PODS, pp. 102–113. ACM (2001)
6. Guttman, A.: R-trees: a dynamic index structure for spatial searching, vol. 14. ACM (1984)
7. Hjaltason, G.R., Samet, H.: Distance browsing in spatial databases. TODS 24(2), 265–318 (1999)
8. Hung, C.C., Chang, C.W., Peng, W.C.: Mining trajectory profiles for discovering user communities. In: LBSN, pp. 1–8. ACM (2009)
9. Jøsang, A., Gray, E., Kinateder, M.: Simplification and analysis of transitive trust networks. Web Intelligence and Agent Systems 4(2), 139–161 (2006)
10. Jøsang, A., Pope, S.: Semantic constraints for trust transitivity. In: APCCM, vol. 43, pp. 59–68. Australian Computer Society, Inc. (2005)
11. Lee, M.-J., Chung, C.-W.: A User Similarity Calculation Based on the Location for Social Network Services. In: Yu, J.X., Kim, M.H., Unland, R. (eds.) DASFAA 2011, Part I. LNCS, vol. 6587, pp. 38–52. Springer, Heidelberg (2011)
12. Newman, M.E.J.: Scientific collaboration networks. ii. shortest paths, weighted networks, and centrality. Physical Review E 64(1), 016132 (2001)
13. Papadias, D., Shen, Q., Tao, Y., Mouratidis, K.: Group nearest neighbor queries. In: ICDE, p. 301. IEEE Computer Society (2004)
14. Scellato, S., Mascolo, C., Musolesi, M., Latora, V.: Distance matters: Geo-social metrics for online social networks. In: Proceedings of the 3rd Conference on Online Social Networks, p. 8. USENIX Association (2010)
15. Singla, P., Richardson, M.: Yes, there is a correlation:-from social networks to personal behavior on the web. In: WWW, pp. 655–664. ACM (2008)
16. Watts, D., Strogatz, S.: Collective dynamics of small-world networks. Nature 393(6684), 440–442 (1998)
17. Yang, D.N., Chen, Y.L., Lee, W.C., Chen, M.S.: On social-temporal group query with acquaintance constraint. VLDB 4(6), 397–408 (2011)
18. Ye, M., Yin, P., Lee, W.C.: Location recommendation for location-based social networks. In: SIGSPATIAL GIS, pp. 458–461. ACM (2010)
19. Yiu, M.L., Mamoulis, N., Papadias, D.: Aggregate nearest neighbor queries in road networks. TKDE, 820–833 (2005)