LETTER *Special Section on Data Engineering and Information Management*

ℓ-Close Range Friends Query on Social Grid Index*

Changbeom SHIM†, Wooil KIM†, Wan HEO†, Sungmin YI††, *Nonmembers*, *and* Yon Dohn CHUNG†a), *Member*

SUMMARY The development of smart devices has led to the growth of Location-Based Social Networking Services (LBSNSs). In this paper, we introduce an *ℓ-Close Range Friends* query that finds all ℓ-hop friends of a user within a specified range. We also propose a query processing method on *Social Grid Index* (*SGI*). Using real datasets, the performance of our method is evaluated.

key words: *spatial database, location-based service, geo-social networks, close range friends query*

1. Introduction

The proliferation of location-aware smart devices allows people to post their position to Location-Based Social Networking Services (LBSNSs) such as *Foursquare, Facebook, Meetup*, and *Instagram*. Although plenty of studies for the services have progressed, some additional studies are needed to cover several specific situations. For example (See Fig. 1), a user Jack holds a party at a point p and wants to invite his friends or the friends of his friends within a reasonable range (circle). Among four persons in the range, Mia and Lucas can be invited because they are connected with Jack by friendship lines. In this situation, if we use existing methods (e.g., [1]), it is difficult to know whether they are friends of the user or acquaintances of the user's friends. In turn, a new type of geo-social query processing method, which considers spatial and social network data at the same time, is needed. In this paper, we define an *ℓ-Close Range Friends* (*ℓ-CRF*) query, and propose an efficient *ℓ-CRF* query processing method.

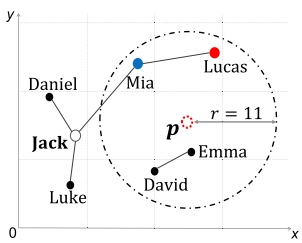


Fig. 1 An example of 2-*CRF* query

Manuscript received June 23, 2016.
Manuscript revised October 7, 2016.
Manuscript publicized January 17, 2017.

†The authors are with the Department of Computer Science and Engineering, Korea University, Seoul, Korea.
††The author is with the Department of IT Convergence, Korea University, Seoul, Korea.

*This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) (No. NRF-2014R1A2A1A11053657).

a) E-mail: ydchung@korea.ac.kr (Corresponding author)

DOI: 10.1587/transinf.2016DAL0003

2. Problem Statement

Let $G=\{V, E\}$ be an undirected graph, where each vertex $v \in V$ corresponds to data object and each edge $e \in E$ represents a friendship of the data objects, i.e., if there is an edge $e(v_i, v_j)$, it signifies that v_i and v_j are friends. All data objects have their own position p_v in Euclidean space, and a *spatial distance* $d(p_{v_i}, p_{v_j})$ indicates a distance between two positions. A *social distance* $s(v_i, v_j)$ of two data objects v_i and v_j represents the minimum number of edges (hop) between two data objects in the graph.

Definition 1 (***ℓ-CRF* query).** Given a query user v, a spatial radius r, a friendship degree ℓ and a point p_q, *ℓ-CRF* query finds a result set R, which consists of all ℓ-hop friends of v within r from p_q. R is defined as follows:

$$R = \{v' \mid v' \neq v \wedge s(v, v') \leq \ell \wedge d(p_q, p_{v'}) \leq r\}$$

3. The Proposed Method

3.1 Naïve Search Algorithm

We present two kinds of naïve approaches for processing *ℓ-CRF* query where each method consists of two major steps:

- Naïve1: Finding all data objects in a query range. → Checking whether the data objects are ℓ-hop friends of the query user or not.
- Naïve2: Finding all ℓ-hop friends of a query user. → Checking whether the ℓ-hop friends are within the query range or not.

We adopt a grid-index structure [2] for processing the range query, and the shortest path (distance) algorithm in [3] for checking the ℓ-hop friends of the query user.

Although two naïve algorithms can find the result of *ℓ-CRF* query, these methods cause some inefficiencies - that is, when there are a great number of data objects in the query range, or a large number of ℓ-hop friends of the query user. To tackle the drawbacks, an efficient method is necessary.

3.2 The Social Grid Index

Social Grid Index. For an efficient *ℓ-CRF* query processing, we suggest a *Social Grid Index* (*SGI*), which is a grid-index structure including friend cells of each cell.

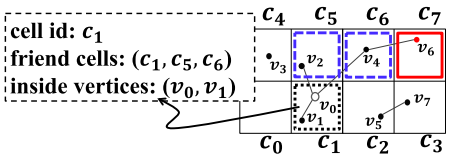


Fig. 2 An example of friend cell

Algorithm 1: *ℓ-CRF* Query Processing

Procedure: *ℓ-CRF* query processing on $SGI(v, r, \ell, p_q)$
1 initialize a set $\mathcal{R} = \emptyset$, a heap $\mathcal{H} = \emptyset$, two lists $\mathcal{FC} = \emptyset, \mathcal{F} = \emptyset$;
2 $c_v \leftarrow$ a cell that includes a query user v;
3 $\mathcal{FC} \leftarrow$ all ℓ-hop friend cells of c_v;
4 **for** *each cell* $c \in \mathcal{FC}$ **do**
5 **if** $mindist(c, p_q) \leq r$ **then** // $mindist(c, p_q)$ is the minimum distance between c and p_q
6 insert c into $\mathcal{H}$;
7 $\mathcal{F} \leftarrow$ all ℓ-hop friends of v;
8 **while** $\mathcal{H}$ *is not empty* **do**
9 $c' \leftarrow$ pop the next entry of $\mathcal{H}$;
10 **for** *each data object* v'' *in* c' **do**
11 **if** $v'' \in \mathcal{F} \wedge d(p_{v''}, p_q) \leq r$ **then**
12 insert v'' into $\mathcal{R}$;
13 **return** $\mathcal{R}$;

Definition 2 (**Friend cell**). Let C be a set of grid cells and $c \in C$ be a cell. Suppose that v_i and v_j are friends, and v_i and v_j are in c_i and c_j, respectively. Then c_i and c_j are friend cells.

For example, in Fig. 2, a cell c_1 has three friend cells c_1, c_5 and c_6, because v_0 in c_1 has three friends v_1 in c_1, v_2 in c_5 and v_4 in c_6.

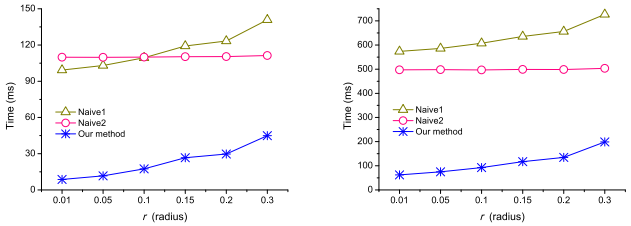
Query Processing on *SGI*. The main idea of our method is to minimize the number of visited cells. When *ℓ-CRF* query is given, the procedure of our method is as follows:

1. Finding ℓ-hop friend cells of a query cell c_v (includes a query user v) in the given range, and ℓ-hop friends of v. The ℓ-hop friend cell is decided based on the Definition 2. For example (See Fig. 2), 2-hop friend cells of a c_1 are (c_1, c_5, c_6, c_7).
2. Checking whether each data object in the cells (found at the first step) is the ℓ-hop friend of v or not, and the distance between the position of each data object and query point p_q is less than r, at once.

All identified data objects are returned as the result of the query. The details are presented in Algorithm 1.

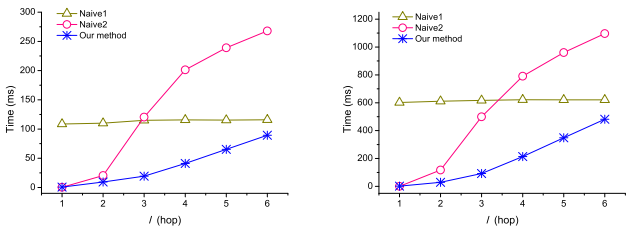
4. Performance Evaluation

Setup and Dataset. All experiments were implemented by a Java on an Intel Core i3 with 3.4GHz CPU and 16GB RAM. For efficiency evaluation, we used two different real datasets [4]. A dataset *Brightkite* consists of 58,228 vertices, 214,078 edges and 7.4 average degree. Another dataset *Gowalla* has 196,591 vertices, 956,327 edges and 9.7 average degree. The entire space is switched over a square 1 unit on a side. We also set several parameters. The query range r is from 0.01 to 0.3 (default: 0.1), the friendship degree (hop) ℓ is from 1 to 6 (default: 3) and the number of grid cells is 256×256.



(a) Brightkite (b) Gowalla

Fig. 3 Execution time vs. radius r



(a) Brightkite (b) Gowalla

Fig. 4 Execution time vs. friendship degree ℓ

Efficiency Evaluation. Figure 3 shows some results varying the radius r. Although Naïve1 and our method show similar trends, our method is superior to Naïve1. The reason is that our method visits only ℓ-hop friend cells within the query range. Naïve2 maintains consistent results for each radius, because the number of data objects to be checked is fixed regardless of the radius.

Figure 4 shows the effects of the varying friendship degree ℓ. The friendship degree has a weak influence on Naïve1, because the number of data objects to be checked is always same. The costs of Naïve2 and our method increase with ℓ, but our method mostly outperforms Naïve2.

5. Conclusion

We have introduced a new *ℓ-CRF* query, as a fundamental geo-social query and the efficient query processing method on *SGI*. Experiments, using two real datasets, showed that our method is efficient. As future work, we plan to study the other types of geo-social queries.

References

[1] A. Guttman, R-trees: a dynamic index structure for spatial searching, ACM, 1984.
[2] M.F. Mokbel, X. Xiong, and W.G. Aref, "Sina: Scalable incremental processing of continuous queries in spatio-temporal databases," Proceedings of the 2004 ACM SIGMOD, pp.623–634, ACM, 2004.
[3] E.W. Dijkstra, "A note on two problems in connexion with graphs," Numerische mathematik, vol.1, no.1, pp.269–271, 1959.
[4] E. Cho, S.A. Myers, and J. Leskovec, "Friendship and mobility: user movement in location-based social networks," Proceedings of the 17th ACM SIGKDD, pp.1082–1090, ACM, 2011.