Group Nearest Compact POI Set Queries in Road Networks

Sen Zhao
Emory University
Atlanta, GA, USA
zhaoseneasy@outlook.com

Li Xiong
Emory University
Atlanta, GA, USA
lxiong@emory.edu

***Abstract*—Identifying a set of points of interest (POIs) is an important problem that finds applications in Location-Based Services (LBS). In this paper, we study a new spatial keyword query motivated by the scenario where a group of users staying at different places wishes to find a compact set of POIs (such as a restaurant and two museums) that is close to all users. We define the problem of group nearest compact POI set (GNCS) query in road networks and show that this problem is NP-hard. To solve the problem, we design query processing algorithms including a first feasible result search algorithm based on the perspective of each individual user, and an exact algorithm with optimizations based on the heuristic of first minimizing the aggregate distance between the POI set and the user group. Extensive performance studies using two real datasets confirm the efficiency and accuracy of our proposed algorithms.**

***Keywords*-group, POI set query, road networks, approximation algorithms**

I. Introduction

Spatial keyword queries are receiving increasing attention due to the prevalence of Location-Based Services (LBS) such as Google Maps[1] and Foursquare[2]. Such queries exploit both location and textual descriptions of spatial objects. We can classify existing queries in the literature into several main categories with respect to the query user (single user vs. a user group) and query object (single object vs. an object set) (shown in Table I). At the same time, Location-Based Social Networks (LBSN) such as Facebook[3], Google+[4], and Meetup[5] enable a group of friends to remain connected from virtually anywhere at any time via location-aware mobile devices. New location-based services need to consider the social aspect and target not only individual users but also user groups.

Most of the spatial keyword queries optimize the distance between the query object(s) and the query user(s) given the keyword requirement. In this paper, we study a spatial keyword query that returns a set of POIs for a user group with additional consideration of the compactness of the POI set motivated by the following scenario. A group of friends, living at different places, may want to find a set of POIs containing a few museums and a restaurant to have a relaxing weekend together. The set of POIs should ideally satisfy the following: (1) it satisfies the users' requirement, e.g. a few museums and a restaurant; (2) it is close to all users in the user group, such that none of them need to travel a long distance; (3) the POI set is compact in terms of number of POIs and the distances between the POIs such that the users can visit all of them with minimal travel.

Motivated by this, we introduce a new type of query called **g**roup **n**earest **c**ompact POI **s**et (GNCS) query for a user group who wishes to find nearest compact POI set. Specifically, a GNCS query is defined over a road network with POIs. The input of the query consists of four parameters: 1) a road network graph, 2) a group of user locations, 3) a set of query keywords, and 4) the required number of POIs for each keyword. We note that a POI may cover multiple different keywords (e.g. a dinner theater), however, it can only cover a single keyword once and users may wish to have more than one POI for the same keyword (e.g. a few different museums). The query returns a POI set that satisfies users' required keywords, and minimizes the cost for the users which consists of two components: 1) the network distance between the POI set and the user group, and 2) the compactness of the POI set determined by both the maximum network distance between the POIs (diameter of the POI set) and the number of POIs in the POI set.

Table I highlights the novelty of our proposed query in comparison to different types of queries in the literature. Existing queries that also aim to find a POI (object) set for a user group [6] [16] mainly optimize the distance between the POI set and the user group. In contrast, GNCS queries emphasize the compactness of the POI set regarding both the distances between the POIs and the number of POIs in order to better address the social query needs, i.e. the users can visit all of the POIs with minimal travel. In addition, GNCS also generalizes the keyword requirements, i.e. a keyword may need to be covered by one or more POIs. Because of the consideration of the compactness of the POI set, in particular, the number of POIs, existing approaches that optimize the distance only will not yield optimal or efficient solutions.

We show that the GNCS query problem is NP-hard. For answering the GNCS query, a brute-force approach based on

[1] www.google.com/maps
[2] https://foursquare.com/
[3] www.facebook.com
[4] https://plus.google.com/
[5] www.meetup.com

Table I: A Summary of Different Types of Queries

query user \ query object	single object	object set	
		distance only	distance & compactness
single user	top-k nearest keyword search, e.g. [5] [7] [18] [8] [15] [10]	spatial group keyword queries, e.g., [3] [12] [2] [4] [21]	
user group	group nearest neighbor queries, e.g., [13] [14] [19] [11]	group nearest group queries, e.g., [6] [16]	GNCS (proposed)

enumerating all POI sets is computationally expensive, and a simple greedy approach would not guarantee the optimality. Existing approaches for optimizing distances only without considering compactness of the POI set are not directly applicable or will not yield optimal or efficient solutions. To address above challenges, based on the features of road network graph, we present efficient algorithms that exploit properties related to both the distance and number of POIs.

Contributions. The key contributions of this paper are summarized as follows:

1) We define a new type of spatial keyword query, group nearest compact POI set (GNCS) query, to address the social needs of a user group who wishes to find a nearest and compact POI set satisfying a set of keywords. We show that this problem is NP-hard.
2) We propose a set of pruning and refining rules for pruning the search space and refining the current candidate optimal solution efficiently by exploiting the distances in both Euclidean space and road network, and present efficient basic query operations.
3) We present two efficient algorithms that exploit properties related to both distance between the POIs and users and number of POIs to solve the GNCS query. They are: 1) a first feasible result search (FFS) algorithm based on the perspective of each individual user which can be used to significantly prune the search space, and 2) an exact algorithm with optimization (EAO) based on the heuristic of first minimizing the aggregate distance cost.
4) Extensive experimental results on two real datasets show that the proposed algorithms are efficient and scalable with excellent utility.

Organization. The rest of paper is organized as follows. Section II formalizes the GNCS query problem. Section III proposes our query processing algorithms. Section IV reports the empirical studies. Section V discusses the related work and Section VI concludes this paper. For all the lemmas and theorems we present in Section II and III, we omit the proofs due to space limitations and refer readers to our technical report [22] for full proofs.

II. PROBLEM STATEMENT

In this section, we first present the definitions of road network graph and distance metrics. Then, we define the problem of group nearest compact POI set (GNCS) query. Finally, we show the hardness of this GNCS problem.

Definition 1: **Road Network Graph.** A road network with POIs is modeled as a road network graph $\mathcal{R} = (V, E, D_S)$, where V is a set of nodes and each $v \in V$ represents an intersection node in the road network or a POI node with keywords $v.T$, E is a set of weighted edges and each $e \in E$ represents a road segment with a weight $e.w$ in the road network, D_S is the two-dimensional Euclidean space in which each node in V is located in.

Definition 2: **Euclidean/Network Distance between a Pair of Nodes.** Given a road network graph $\mathcal{R} = (V, E, D_S)$, for two nodes v_i located at (x_i, y_i) and v_j located at (x_j, y_j) in Euclidean space D_S, the **Euclidean distance** between them is

$$Dist_S(v_i, v_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}.$$

The **network distance** between v_i and v_j is defined as the shortest distance between v_i and v_j in network graph. Assume that $(e_1, e_2, \cdots, e_n)$ is the shortest route between v_i and v_j, the network distance between them is

$$Dist(v_i, v_j) = \sum_{k=1}^{n} e_k.w.$$

Lemma 1: (Pruning Rule) The Euclidean distance computed can be used as a lower bound of the network distance, i.e., $Dist(v_i, v_j) \geq Dist_S(v_i, v_j)$.

Definition 3: **Cost of a POI Set.** Given a road network graph $\mathcal{R} = (V, E, D_S)$, for a user group $\mathcal{U}$ and a POI set $\mathcal{S}$, the cost of $\mathcal{S}$, denoted by $cost(\mathcal{S})$, is computed as the linear combination of the distance between $\mathcal{U}$ and $\mathcal{S}$ and the compactness of $\mathcal{S}$. That is,

$$cost(\mathcal{S}) = \alpha \cdot Dist(\mathcal{U}, \mathcal{S}) + (1 - \alpha) \cdot |\mathcal{S}| \cdot Diameter(\mathcal{S}) \quad (1)$$

where $Dist(\mathcal{U}, \mathcal{S}) = \min\{Dist(\mathcal{U}, \int_i) \mid \int_i \in \mathcal{S}\}$, $Dist_S(\mathcal{U}, \int) = \max\{Dist_S(u_i, \int) \mid u_i \in \mathcal{U}\}$, $Diameter(\mathcal{S}) = \max\{Dist(\int_i, \int_j) \mid \int_i, \int_j \in \mathcal{S}, \int_i \neq \int_j\}$, and $\alpha \in [0, 1]$ is a user specified parameter to trade off the aggregate distance against the compactness of the POI set.

Problem Definition. Based on these definitions, we formally define the group nearest POI set (GNCS) query as follows:

Definition 4: **GNCS Query.** The GNCS query $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$ aims to find a POI set $\mathcal{S}$ in $\mathcal{R}$ (if such POI set exists) such that

$$\mathcal{S} = \arg_{\mathcal{S}} \min cost(\mathcal{S}),$$

subject to

$$\Psi \subseteq \bigcup_{\int \in \mathcal{S}} \int.T, \quad (2)$$

$$\forall_{\psi \in \Psi} K.\psi \leq \sum_{\int \in \mathcal{S}, \psi \subseteq \int.T} 1, \quad (3)$$

where $\mathcal{U}$ is the group of user-location nodes, Ψ is the set of required keywords, $K.\psi \in K$ is the required number of POIs for keyword $\psi \in \Psi$, and $\mathcal{R}$ is a road network graph.

In order to present the rest of this paper clearly, the parameter α in the cost function is omitted. However, when α is enabled, the proposed algorithms remain applicable.

Lemma 2: (Upper and lower bound of POI set size) For a GNCS query $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$, given a feasible POI set $\mathcal{S}$, the upper bound of the number of POIs in $\mathcal{S}$ is computed as $K_u = \sum_{\psi \in \Psi} K.\psi$, and the lower bound is computed as $K_l = \max_{\psi \in \Psi} K.\psi$.

The GNCS query problem is NP-hard and can be reduced from the problem in [16] by omitting the number of POIs $|S|$ in the cost function. It can be then further proved by a reduction from 3-satisfiability (3-SAT) [17]. The proof is similar to [16] and thus omitted.

III. ALGORITHMS FOR GNCS QUERY

For GNCS query, a brute-force algorithm is to enumerate all feasible POI sets that satisfy the users' requirements (Ψ, K), and then search the POI set with the smallest cost as the final result. The time complexity of the brute-force algorithm is $O(\prod_{\psi \in \Psi} \binom{|V_\psi|}{K.\psi})$, where $|V_\psi|$ denotes the number of nodes with keyword $\psi \in \Psi$ in $\mathcal{R}$, which leads to a prohibitive cost in practice. Hence, in this section, we propose efficient algorithms for answering the GNCS query. The algorithms are based on two indexes we built in the Euclidean space and network graph and a set of pruning rules in the two spaces respectively (please see [22]). First, we propose an algorithm FFS, which can search a first feasible POI set efficiently but gives a $\{4 \cdot K_u + 3\}$-factor approximation. Then, we present an exact algorithm with optimizations EAO that utilizes pruning based on properties related to both the distance between the POI set and the user group and the POI set size.

A. Basic Query Operations

We first present three basic query operations which can be used as building blocks for our GNCS query.

Operation 1: Pair Distance. Dist$(v_s, v_t, \mathcal{R})$ computes the network distance between a pair of nodes v_s and v_t in the road network graph $\mathcal{R}$. We adopt A* algorithm [20] to compute the exact network distance between a pair of nodes.

Operation 2: Range of Grid Cells. RG$(\mathcal{U}, \Psi, \triangle, \mathcal{R})$ returns the range of grid cells that should be considered for nodes covering any keyword in Ψ with network distances to $\mathcal{U}$ no more than $\triangle$ in $\mathcal{R}$.

Operation 3: Nearest Keyword Nodes. NK$(v, \psi, k, \mathcal{R})$ returns the top-k nearest nodes to a node v in the road network graph $\mathcal{R}$ containing keyword ψ. The general idea is to first obtain k nodes containing the keyword as the first feasible result and then refine the solution by pruning the search space based on grid distance, Euclidean distance and network distance sequentially.

Due to space limitation, the details of above basic query operations are shown in our technical report [22].

B. First Feasible POI Set Search Algorithm

We now propose an efficient first feasible POI set search algorithm FFS. The idea is to construct a feasible POI set based on each user in the user group separately and then select the best set among them. The details are shown in Algorithm 1. In particular, We first initialize two node sets $\mathcal{S}_{FF}$ and $\hat{\mathcal{S}}$, to store the current best POI set and the current candidate feasible result, and two variable C_{FF} and $C_{\hat{\mathcal{S}}}$ to store the costs of $\mathcal{S}_{FF}$ and $\hat{\mathcal{S}}$ respectively (lines 1-4). The constructions of the feasible sets are shown in lines 5-13. For each user node $u \in \mathcal{U}$, we first search the nearest node v_n to u with any keyword $\psi' \in \Psi$, then search other nearest POI nodes to v_n, to construct a feasible POI set (by calling Operation 3). After that, we can obtain $|\mathcal{U}|$ feasible POI sets. Finally, we take the POI set with the best cost as the first feasible result. The following example illustrates how this technique works.

Algorithm 1: FFS$(\mathcal{U}, \Psi, k, \mathcal{R})$

Input: $\mathcal{U}, \Psi, k, \mathcal{R}$.
Output: $\mathcal{S}_{FF}$: the first feasible POI set.

1 $\mathcal{S}_{FF} \leftarrow \emptyset$;
2 $C_{FF} \leftarrow +\infty$;
3 $\hat{\mathcal{S}} \leftarrow \emptyset$;
4 $C_{\hat{\mathcal{S}}} \leftarrow +\infty$;
5 **for** *each user-location node* $u \in \mathcal{U}$ **do**
6 search the nearest node v_n with any $\psi' \in \Psi$ to u;
7 search top-$\{K.\psi' - 1\}$ nearest node with ψ' and top-$\{K.\psi\}$ nearest node with other keyword $\psi \in \Psi$ to v_n; // by Calling Operation 3
8 put these POI nodes into $\hat{\mathcal{S}}$;
9 $C_{\hat{\mathcal{S}}} \leftarrow cost(\hat{\mathcal{S}})$;
10 **if** $C_{\hat{\mathcal{S}}} < C_{FF}$ **then**
11 $\mathcal{S}_{FF} \leftarrow \hat{\mathcal{S}}$;
12 $C_{FF} \leftarrow C_{\hat{\mathcal{S}}}$;
13 return $\mathcal{S}_{FF}$;

Complexity. Since the time complexity of Operation 3 is $O(|V_c|)$, the time complexity of FFS is $O(|\mathcal{U}| \cdot |\Psi| \cdot |V_c|)$.

Theoretical Analysis.

Theorem 1: FFS gives an $\{4 \cdot K_u + 3\}$-factor approximation for the GNCS query.

C. Optimized Exact Algorithm

Since the time complexity of the brute-force algorithm is prohibitive, we now present an exact algorithm with optimizations (EAO). We can use the FFS algorithm above to obtain the first feasible POI set and then prune the search space. The basic idea is to: 1) find the meeting node (aggregator node) which determines the distance between the POI set and the user group, 2) find the diameter nodes which determine the diameter and the compactness of the POI set, and 3) find the remaining nodes of the POI set. The pruning utilizes both the distances between an aggregator node and

the user group, and the size of a POI set. Recall that K_u and K_l are the upper and lower bound of the POI set size as defined in Lemma 2.

Algorithm 2: EAO($\mathcal{U}$, Ψ, K, $\mathcal{R}$)

Input: $\mathcal{U}$, Ψ, K, $\mathcal{R}$.
Output: $\mathcal{S}_{opt}$: the optimal POI set.

$\mathcal{S}_{opt} \leftarrow FFS(\mathcal{U}, \Psi, K, \mathcal{R})$; // Obtain the current best result by Calling Algorithm 1
$C_{opt} \leftarrow cost(\mathcal{S}_{opt})$;
$\hat{\mathcal{S}} \leftarrow \emptyset$;
$C_{\hat{\mathcal{S}}} \leftarrow +\infty$;
Initialize two grid cell queues Q_a and Q_r; // all grid cells in Q_a are sorted in ascending order of their grid distances to $\mathcal{U}$; all grid cells in Q_r are sorted in ascending order of their grid distances to aggregator node v_a
Initialize a grid cell set G_r;
$Q_a \leftarrow RG(\mathcal{U}, \Psi, C_{opt}, \mathcal{R})$; // Pruned by Lemma 3
$flag_a = 0$;
while Q_a *is not empty* **do**
  select the nearest grid cell g_a in Q_a;
  $Q_a \leftarrow Q_a - \{g_a\}$;
  for *each node* $v_a \in g_a$ *with any keyword in* Ψ *sorted in ascending order of its Euclidean distance to* $\mathcal{U}$ **do**
    if $Dist_S(v_a, \mathcal{U}) \geq C_{opt}$ **then** // Pruned by Lemma 1 and 4
      break;
    else
      if $uDist(v_a, \mathcal{U}) < C_{opt}$ **then** // Refined by Lemma 11 and 4
        $flag_a \leftarrow 1$;
      else
        if $lDist(v_a, \mathcal{U}) \geq C_{opt}$ **then** // Pruned by Lemma 11 and 4
          continue;
        else
          compute $Dist(v_a, \mathcal{U})$;
          if $Dist(v_a, \mathcal{U}) < C_{opt}$ **then**
            $flag_a \leftarrow 1$;
      if $flag == 1$ **then**
        $\hat{\mathcal{S}}$ takes v_a as the aggregator node;
        $Q_r \leftarrow RG(\{v_a\}, \Psi, \frac{C_{opt} - Dist(U, v_a)}{\max(|\hat{\mathcal{S}}|, K_l)}, \mathcal{R})$; // Pruned by Lemma 5
        for *each pair of nodes* v_{r_1} *and* v_{r_2} *with rest keyword which can meet the constraints in Lemma 6 in ascending order of* $Dist(v_{r_1}, v_{r_2})$ *in* Q_r **do** // Refined/Pruned by Lemma 1, 11, 12 and 6
          $G_r \leftarrow \bigcap_{v \in \{v_a, v_{r_1}, v_{r_2}\}} RG(v, \Psi, Dist(v_{r_1}, v_{r_2}), \mathcal{R})$; // Pruned by Lemma 7
          if *the remaining nodes which cover the rest keyword and satisfy the constraints in Lemma 8 can be found in* G_r **then** // Refined/Pruned by Lemma 1, 11, 12 and 8
            put v_{r_1}, v_{r_2} and remaining nodes into $\hat{\mathcal{S}}$;
            if $cost(\hat{\mathcal{S}}) < cost(\mathcal{S}_{opt})$ **then**
              $\mathcal{S}_{opt} \leftarrow \hat{\mathcal{S}}$;
              $C_{opt} \leftarrow C_{\hat{\mathcal{S}}}$;
              $Q_a \leftarrow Q_a \cap RG(\mathcal{U}, \Psi, C_{opt}, \mathcal{R})$;
            $\hat{\mathcal{S}} \leftarrow \emptyset$;
            $flag_a \leftarrow 0$;
            break;
        $\hat{\mathcal{S}} \leftarrow \emptyset$;
        $flag_a \leftarrow 0$;
return $\mathcal{S}_{opt}$;

Aggregator node search. Given a GNCS query $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$, let $\mathcal{S}_c$ be a candidate POI set. The lower bound of $cost(\mathcal{S}_c)$ can be determined by the aggregate distance (which is a part of the cost function). The aggregate distance can be determined by the node in $\mathcal{S}_c$ with the minimal network distance to $\mathcal{U}$, which we refer to as *aggregator node*. Hence, to avoid constructing some unnecessary candidate POI sets, we can prune the search space for aggregator nodes using the following rules.

Lemma 3: (Pruning Rule) Given a GNCS query $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$, assume that $\mathcal{S}_{opt}$ is a current best result and $r_{opt} = cost(\mathcal{S}_{opt})$. Then, if there exists a feasible POI set with cost smaller than r_{opt}, the search space for the aggregator node of that POI set is limited to RG($\mathcal{U}, \Psi, r_{opt}, \mathcal{R}$).

Lemma 4: (Pruning Rule) Given a GNCS query $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$, assume that $\mathcal{S}_{opt}$ is the current best POI set and $r_{opt} = cost(\mathcal{S}_{opt})$. Then, if there exists a candidate POI set whose aggregator node has a larger cost than r_{opt} to $\mathcal{U}$, we can prune it immediately.

The above Lemma 3 limits the "most" grid cells we need to consider during the aggregator node search. Thus, when we obtain a current best POI set $\mathcal{S}_{opt}$, we only need to consider the candidate results whose aggregator node is inside RG($\mathcal{U}, \Psi, cost(\mathcal{S}_{opt}), \mathcal{R}$).

Diameter node search. After we have found the aggregator node v_a, we can determine the aggregate distance (i.e., $Dist(\mathcal{U}, v_a)$) of the current intermediate result $\hat{\mathcal{S}}$. Based on the cost of the current best result $\mathcal{S}_{opt}$, if $\hat{\mathcal{S}}$ has a better cost than $\mathcal{S}_{opt}$, it must have a diameter $< \frac{cost(\mathcal{S}_{opt}) - Dist(\mathcal{U}, v_a)}{\max(|\hat{\mathcal{S}}|, K_l)}$, where $|\hat{\mathcal{S}}|$ is the current number of POIs in $\hat{\mathcal{S}}$. It is obvious that the diameter of a candidate result can be determined by two nodes which are farthest to each other, which we refer to as *diameter nodes*. Based on above observations, we can prune the search space for the diameter nodes using the following rules.

Lemma 5: (Pruning Rule) Given a GNCS query $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$, assume $\mathcal{S}_{opt}$ is the current best POI set and $r_{opt} = cost(\mathcal{S}_{opt})$. Then, if there exists a better result with aggregator node v_a, the largest grid cell space for diameter nodes search is limited to RG($v_a, \Psi, \frac{r_{opt} - Dist(\mathcal{U}, v_a)}{\max(|\hat{\mathcal{S}}|, K_l)}, \mathcal{R}$).

Lemma 6: (Pruning Rule) Given a GNCS query $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$, assume that $\mathcal{S}_{opt}$ is the current best POI set and $r_{opt} = cost(\mathcal{S}_{opt})$. If there exists a better POI set $\hat{\mathcal{S}}$ with aggregator node v_a, the diameter nodes v_{r_1} and v_{r_2} of $\hat{\mathcal{S}}$ must satisfy three constraints: (1) $Dist(\mathcal{U}, v_{r_1}) \geq Dist(\mathcal{U}, v_a)$ and $Dist(\mathcal{U}, v_{r_2}) \geq Dist(\mathcal{U}, v_a)$; (2) $Dist(v_a, v_{r_1}) < \frac{r_{opt} - Dist(\mathcal{U}, v_a)}{\max(|\hat{\mathcal{S}}|, K_l)}$ and $Dist(v_a, v_{r_2}) < \frac{r_{opt} - Dist(\mathcal{U}, v_a)}{\max(|\hat{\mathcal{S}}|, K_l)}$; (3) $Dist(v_{r_1}, v_{r_2}) < \frac{r_{opt} - Dist(\mathcal{U}, v_a)}{\max(|\hat{\mathcal{S}}|, K_l)}$.

Remaining node search. After the aggregator node and the diameter nodes of a possible better POI set have been found, we can search for the remaining nodes with the required keyword to construct this result. In the intermediate result, v_a is the aggregator node which determines the distance, v_{r_1} and v_{r_2} are the two diameter nodes which determines the diameter. Hence the search space for the remaining nodes can be pruned by the following rules.

Lemma 7: (Pruning Rule) Given a GNCS query $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$, if there exists a better POI set with aggregator node v_a and diameter nodes v_{r_1} and v_{r_2}, the grid cell search space for the remaining nodes of the POI set is limited to $\bigcap_{v \in \{v_a, v_{r_1}, v_{r_2}\}} \text{RG}(v, \Psi, Dist(v_{r_1}, v_{r_2}), \mathcal{R})$.

Lemma 8: (Pruning Rule) Given $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$, if there exists a better POI set $\mathcal{S}'$ with aggregator node v_a and diameter nodes v_{r_1} and v_{r_2}, each node v' in the remaining nodes of $\mathcal{S}'$ must satisfy two constraints: (1) $Dist(\mathcal{U}, v') \geq Dist(\mathcal{U}, v_a)$; (2) $\bigwedge_{v \in \mathcal{S}'} Dist(v', v) \leq Dist(v_{r_1}, v_{r_2})$.

Due to space limitation, the details of Lemma 11 used for refining and Lemma 12 used for pruning are shown in our technical report [22].

The complete algorithm. Based on above pruning rules, we propose a complete search strategy for the GNCS query which is shown in Algorithm 2. Due to space limitation, the details of this search strategy are shown in our technical report [22].

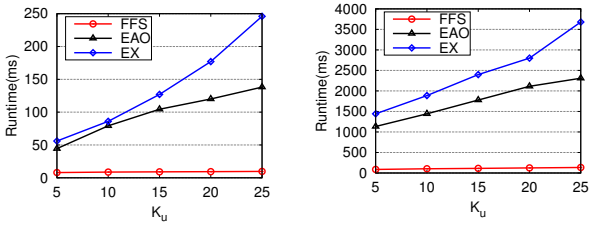
Complexity. In the worst case, the time complexity of EAO is as high as that of brute-force algorithm. However, the pruning based optimization techniques (such as Lemma 4 and Lemma 6) adopted in EAO can significantly prune the search space.

Correctness.

Theorem 2: EAO returns a feasible POI set with the smallest cost for the GNCS query.

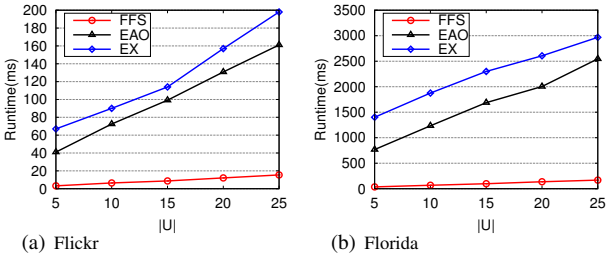
IV. EXPERIMENTAL STUDY

In this section, we present a comprehensive experimental evaluation. First, we introduce the dataset and experiment settings. Then, we present the results evaluating the efficiency and accuracy of proposed algorithms.



(a) Flickr (b) Florida

Figure 1: Runtime (total number of required keywords K_u)



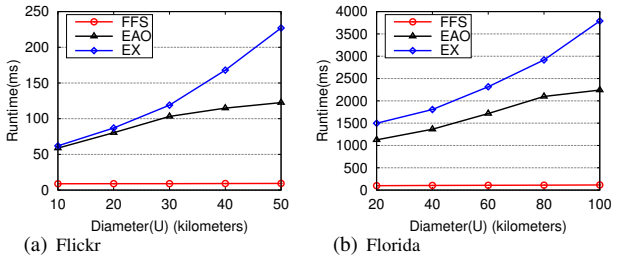
(a) Flickr (b) Florida

Figure 2: Runtime (the number of user-location nodes $|\mathcal{U}|$)

A. Experimental Settings

Datasets. We use two datasets, Flickr generated from Flickr[6] and Florida generated from the Florida road network[7] following the work [1]. Due to space limitation, the details of above two datasets are shown in our technical report [22].

Queries. For the GNCS query $Q = \langle \mathcal{U}, \Psi, K, \mathcal{R} \rangle$ used in our experimental study, the user-location nodes $\mathcal{U}$ are randomly selected in the road network $\mathcal{R}$. To evaluate the impact of each parameter, we vary the number of users in the user group $|U|$ (5, 10, 15, 20 and 25, default 16), the diameter of the user group $Diameter(U)$ in kilometers (10, 20, 30, 40, 50 with default 25 for Flicker dataset; and 20, 40, 60, 80, 100 with default 50 for Florida dataset), and the total number of required keywords K_u (5, 10, 12, 15, 20 and 25, default 12).



(a) Flickr (b) Florida

Figure 3: Runtime (the Diameter of user-location nodes)

Algorithms. We study the performance of our proposed algorithms: the first feasible POI set search algorithm FFS (Algorithm 1) and the exact algorithm EAO (Algorithm 2). To make a comparison, we simply revise the exact algorithm EXA in [16] based on our cost function and employ our three basic query operations, and we refer to it as EX. All algorithms were implemented in C and run on Intel(R) Core(TM) i7-6700K CPU@ 4.00GHz with 64 GB RAM. The parameter α in the cost function is set as 0.5.

B. Experimental Results

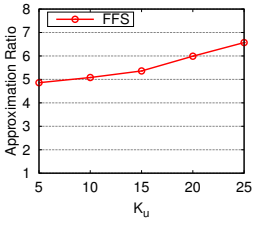


Figure 4: Appro. Ratio (K_u)

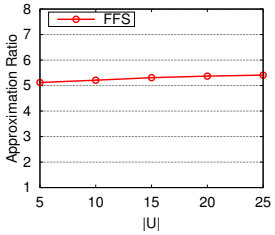


Figure 5: Appro. Ratio $|\mathcal{U}|$)

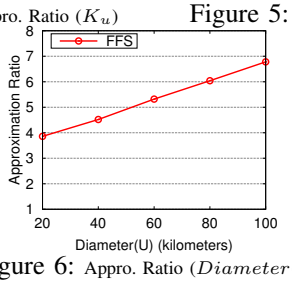


Figure 6: Appro. Ratio ($Diameter(\mathcal{U})$)

1) Efficiency and Impact of Query Parameters: We now evaluate the efficiency of the proposed algorithms with different query parameters including: 1) the number of user-location nodes $|\mathcal{U}|$, 2) the total number of required

[6]https://www.flickr.com/

[7]http://www.dis.uniroma1.it/challenge9/download.shtml

keywords K_u, and 3) the diameter of user-location nodes $Diameter(\mathcal{U})$. The average runtime over five runs is reported on Flickr and Florida datasets respectively.

From Figure 1, 2, 3, we observe that our proposed algorithms scale well with these three input parameters. We also observe that the exact algorithm EAO achieves fairly good efficiency thanks to the pruning/refining techniques.

2) Approximation Ratio of the First Feasible POI Set Search Algorithm: In this set of experiments, we report the average approximation ratio of the FFS algorithm over the query sets compared to the exact algorithm EAO. The Florida dataset is employed for the evaluation.

From Figure 4, 5 and 6, we observe that the approximation ratio of FFS scales well with all the parameters.

V. RELATED WORK

Keyword search on spatial database has been intensively studied in recent years. Table I shows a categorization of the main related work. Specifically, *top-k nearest keyword search* has been studied based on DIR-tree [5], IR^2-tree [7] and W-IR-Tree [18] respectively, which focuses on single object query for single user. The *collective spatial keyword query* problem has been solved well by [3], [12] and [2], which focuses on an object set query for single user.

As the Euclidean distance can be an inaccurate approximation of the road network distance, keyword search on graphs or networks has drawn increasing interest. Specifically, *top-k nearest keyword queries* have been studied based on bi-level index structure [8], distance oracle and shortest-path tree [15], and FS and FBS [10], which focus on single object query for single user. The *length-constrained maximum-sum region query* [4] and *popularity-aware collective keyword query* [21] have been studied well, which focus on an object set query for single user. In summary, these works are not directly applicable to our problem due to the consideration of a single or no query user.

Existing works also considered user group in spatial keyword queries. The *group nearest neighbor queries* which focus on single object for a user group has been solved well based on the cost function of SUM-distance [13] [14] [19] and MAXIMUM-distance [11] respectively. *Group trip planning queries* focus on a route query for a user group [9] [23] and aim to schedule optimal routes for a group of users based on their requirements. Most relevant to our work are those that also aim to find a POI set for a user group [6] [16]. The main difference is that GNCS queries emphasize the compactness of the POI set. Our search strategies specifically exploited the bound on POI set size (such as Lemma 2, 5, and 6) in addition to pruning strategies in both Euclidean and road network space to answer GNCS query efficiently.

VI. CONCLUSIONS

In this paper, we studied the problem of group nearest compact POI set query (GNCS). We showed this problem is NP-hard and presented efficient algorithms to solve this problem. Extensive experimental results on two real datasets demonstrated the efficiency and accuracy of the proposed algorithms. In our future work, we plan to propose enhanced approxmiation algorithms for better accuracy and efficiency tradeoff.

REFERENCES

[1] X. Cao, L. Chen, G. Cong, and X. Xiao. Keyword-aware optimal route search. *PVLDB*, 5(11):1136–1147, 2012.

[2] X. Cao, G. Cong, T. Guo, C. S. Jensen, and B. C. Ooi. Efficient processing of spatial group keyword queries. *ACM Trans. Database Syst.*, 40(2):13, 2015.

[3] X. Cao, G. Cong, C. S. Jensen, and B. C. Ooi. Collective spatial keyword querying. In *SIGMOD*, pages 373–384, 2011.

[4] X. Cao, G. Cong, C. S. Jensen, and M. L. Yiu. Retrieving regions of interest for user exploration. *PVLDB*, 7(9):733–744, 2014.

[5] G. Cong, C. S. Jensen, and D. Wu. Efficient retrieval of the top-k most relevant spatial web objects. *PVLDB*, 2(1):337–348, 2009.

[6] K. Deng, S. W. Sadiq, X. Zhou, H. Xu, G. P. C. Fung, and Y. Lu. On group nearest group query processing. *IEEE Trans. Knowl. Data Eng.*, 24(2):295–308, 2012.

[7] I. D. Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *ICDE*, pages 656–665, 2008.

[8] H. He, H. Wang, J. Yang, and P. S. Yu. BLINKS: ranked keyword searches on graphs. In *SIGMOD*, pages 305–316, 2007.

[9] R. Jahan, T. Hashem, and S. Barua. Group trip scheduling (GTS) queries in spatial databases. In *EDBT*, pages 390–401, 2017.

[10] M. Jiang, A. W. Fu, and R. C. Wong. Exact top-k nearest keyword search in large networks. In *SIGMOD*, pages 393–404, 2015.

[11] F. Li, B. Yao, and P. Kumar. Group enclosing queries. *IEEE Trans. Knowl. Data Eng.*, 23(10):1526–1540, 2011.

[12] C. Long, R. C. Wong, K. Wang, and A. W. Fu. Collective spatial keyword queries: a distance owner-driven approach. In *SIGMOD*, pages 689–700, 2013.

[13] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis. Group nearest neighbor queries. In *ICDE*, pages 301–312, 2004.

[14] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui. Aggregate nearest neighbor queries in spatial databases. *ACM Trans. Database Syst.*, 30(2):529–576, 2005.

[15] M. Qiao, L. Qin, H. Cheng, J. X. Yu, and W. Tian. Top-k nearest keyword search on large graphs. *PVLDB*, 6(10):901–912, 2013.

[16] S. Su, S. Zhao, X. Cheng, R. Bi, X. Cao, and J. Wang. Group-based collective keyword querying in road networks. *Inf. Process. Lett.*, 118:83–90, 2017.

[17] C. A. Tovey. A simplified np-complete satisfiability problem. *Discrete Applied Mathematics*, 8(1):85–89, 1984.

[18] D. Wu, M. L. Yiu, G. Cong, and C. S. Jensen. Joint top-k spatial keyword query processing. *IEEE Trans. Knowl. Data Eng.*, 24(10):1889–1903, 2012.

[19] M. L. Yiu, N. Mamoulis, and D. Papadias. Aggregate nearest neighbor queries in road networks. *IEEE Trans. Knowl. Data Eng.*, 17(6):820–833, 2005.

[20] W. Zeng and R. L. Church. Finding shortest paths on real road networks: the case for A. *International Journal of Geographical Information Science*, 23(4):531–543, 2009.

[21] S. Zhao, X. Cheng, S. Su, and K. Shuang. Popularity-aware collective keyword queries in road networks. *GeoInformatica*, 21(3):485–518, 2017.

[22] S. Zhao and L. xiong. Technical report. http://www.mathcs.emory.edu/aims/pub/gncs18.pdf.

[23] S. Zhao, L. Zhao, S. Su, X. Cheng, and L. Xiong. Group-based keyword-aware route querying in road networks. *Inf. Sci.*, 450:343–360, 2018.