Cohesive Group Nearest Neighbor Queries over Road-Social Networks

Fangda Guo[1], Ye Yuan[1*], Guoren Wang[2], Lei Chen[3], Xiang Lian[4], Zimeng Wang[1]
[1]*School of Computer Science and Engineering, Northeastern University, Shenyang, China*
[2]*School of Computer Science and Technology, Beijing Institute of Technology, Beijing, China*
[3]*Department of Computer Science and Engineering, Hong Kong University of Science and Technology, Hong Kong, China*
[4]*Department of Computer Science, Kent State University, Ohio, USA*
[1]{fangda@stumail, yuanye@mail, zimeng@stumail}.neu.edu.cn, [2]wanggr@bit.edu.cn, [3]leichen@cse.ust.hk, [4]xlian@kent.edu

***Abstract*—The group nearest neighbor (GNN) search on a road network G_r, i.e., finding the spatial objects as activity assembly points with the smallest sum of distances to query users on G_r, has been extensively studied; however, previous works have neglected the fact that social relationships among query users, which ensure the maximally favorable atmosphere in the activity, can play an important role in GNN queries. Many real-world applications, such as location-based social networking services, require such queries. In this paper, we study a new problem: a GNN search on a road network that incorporates cohesive social relationships (CGNN). Specifically, both the query users of highest closeness and the corresponding top-j objects are retrieved. One critical challenge is to speed up the computation of CGNN queries over large social and road networks. To address this challenge, we propose a filtering-and-verification framework for efficient query processing. During filtering, we prune substantial unpromising users and objects using social and geographically spatial constraints. During verification, we obtain the object candidates, among which the top j are selected, with respect to the qualified users. Moreover, we further optimize search strategies to improve query performance. Finally, experimental results on real social and road networks significantly demonstrate the efficiency and efficacy of our solutions.**

I. Introduction

With the ever-growing popularity of GPS-enabled mobile devices, many location-based service (LBS) systems (e.g., Google Maps) have been deployed and widely accepted by mobile users, who use them to easily capture and upload their own locations during daily activities. Along with the widespread prevalence of LBSs, recent years have witnessed a massive explosion in location-based social networking (LBSN) applications, such as Yelp, Foursquare and Facebook Places. In all these applications, social network users are always associated with location information (e.g., public places, home/office addresses), which can be shared with their "friends".

It is envisaged that such location information can bridge the gap between the physical world and the virtual world of social networks. The nearest neighbor (NN) search and its variants on road networks are fundamental issues in LBSs due to their importance in a wide spectrum of applications [1]–[3]. Such search capabilities provide social network users with new opportunities for rapidly organizing impromptu offline activities. Below are three typical example query scenarios that arise in the new context of road-social networks.

Q1 To arrange a gathering, Steve wishes to find two highly ranked cafes nearby (e.g., within 6.5 km) at which he and his 3 friends, who preferably know each other well, can meet while minimizing the total distance traveled.

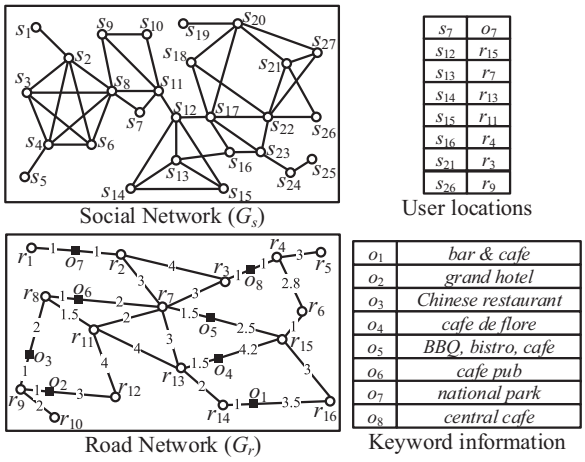


Fig. 1. Example of road-social network.

Q2 A tour company wishes to divide its tourist customers into several groups according to their mutual familiarity. For each group, a mini-coach will be arranged to wait at and depart from a parking location sufficiently close to everyone (e.g., where they can arrive within 30 minutes).

Q3 A company headquarters plans to invite a number (e.g., 30) of its branch management staff, who often have business contacts or have participated in the same projects, and book a hotel ballroom no more than 10 km from the farthest branch at a location that minimizes the total travel cost, to hold a banquet.

As significant and substantial manual coordination is still required, none of the existing methods can solve the above problems. Specifically, the challenges faced in organizing such activities lie in issuing timely invitations that specify both optimal invitees and suitable assembly points in accordance with the closeness of a limited number of candidate attendees and their proximity to corresponding locations in the physical world. Intuitively, when attendee size increases, the organization process becomes more complicated and thereby too tedious to coordinate manually. Thus, it is imperative to develop efficient new techniques to alleviate the effort and support impromptu offline activity planning services. A specific motivating example follows.

Example 1: Fig. 1 illustrates Q1 over a road-social network, which is split into a social layer (G_s) and a road layer (G_r) for clarity. The circles in G_s and G_r represent users and road intersections/end-points, respectively. In G_r, the rectangles denote spatial-textual objects lying on edges, for which keyword

information is listed; the number on each edge represents the travel distance on the corresponding road segment. In G_s, the published user location information provides a mapping specifying the location $r \in G_r$ of each user $s \in G_s$. Suppose that s_{13} in G_s is Steve, whose current location is r_7 in G_r. From Steve's perspective, he may be advised that s_{12}, s_{15}, and s_{16} (at r_{15}, r_{11}, and r_4, respectively) could be invited to o_5 or o_8, both of which meet his requirements of proximity 6.5 km and keyword "cafe", with a minimum total travel distance of 13.8 or 15.8 km, respectively. However, as observed in G_s, s_{12} and s_{15} are both unacquainted with s_{16}; in contrast, s_{12}, s_{14}, and s_{15} (at r_{15}, r_{13}, and r_{11}, respectively) know each other best and thus are the optimal invitees. Accordingly, o_5 and o_6, with minimum total travel distances of 12 and 15.5 km, respectively, are the best selections that should be ultimately recommended to Steve. □

This example motivates us to consider a novel type of query on road-social networks, namely, cohesive group nearest neighbor (CGNN), and to propose efficient processing algorithms. Specifically, given G_s, G_r, the number c of activity attendees (including a query user u_q), and an upper limit ϵ on distance/time , an activity initiator[1] issues a CGNN query that should return c attendees and the top-j objects (i.e., assembly points) from G_r containing keywords w such that the travel cost of each attendee is within ϵ and the total travel cost of all attendees is minimized. The query also incorporates a social constraint on the closeness of the c attendees: each invitee of u_q in G_s should know the others as well as possible.

Challenges. In this paper, we address the following challenges in efficient CGNN query processing. (1) Since the number of attendees is limited to c and their closeness is to be maximized, the social constraint among the attendees is different from the traditional k-core [4], a well-known concept in graph theory, which is the maximal induced subgraph in which every vertex has at least k neighbors. Thus, selecting of the most cohesive set of attendees is nontrivial because of the massive number of possible combinations to be evaluated. (2) Retrieving the total travel cost to a target object in road networks, as opposed to geometric distances in Euclidean space, is more complex since location and accessibility of objects are restricted by the computation of network distance, particularly with increasing c. (3) Both the social and road networks are typically massive. For example, Facebook has 1 billion users, and the USA road network alone has more than 20 million vertices. Thus, it is challenging to efficiently process CGNN queries over typically large road-social networks.

Our Solution and Contributions. A simple strategy for solving a CGNN query is to enumerate all combinations of c–1 invitees of u_q, select the most cohesive one(s), compute the total cost for each object by traversing G_r, and return the top-j objects within ϵ. However, this strategy is obviously infeasible due to the massive costs of enumeration and traversal.

To address the above challenges, we propose a filtering-and-verification framework: (1) to avoid enumeration while rapidly locating comprehensive solutions that consider both the social network topology and the scale of the activity, the most cohesive k-core model is developed to quantify closeness and ensure the maximally favorable atmosphere; (2) an incremental and accumulative strategy is adopted to gradually expand the search space in road networks while significantly pruning unpromising objects; and (3) an effective verification strategy is designed to expedite the extraction of the top-j objects among the candidates. Moreover, a state-of-the-art hierarchical tree structure [5] is adopted to index road networks, allowing candidates to be obtained more quickly. Finally, we optimize the overall query performance in a round-robin fashion. The principal contributions are summarized as follows.

- We formulate a pragmatic query type for road-social networks, i.e., CGNN queries, to identify suitable spatial-textual objects as assembly points for optimal sets of attendees. Such queries can accommodate various types of offline activities of specified scales.
- We develop the first efficient algorithm for finding the most cohesive k-core model to ensure the social closeness of a limited number of attendees.
- Effective pruning and verification strategies for finding the top-j objects are proposed, and optimization techniques are designed to further improve query processing.
- We conduct extensive experiments on real datasets to demonstrate the effectiveness and efficiency of our proposed strategies and algorithms.

The rest of the paper is organized as follows. Section II formulates the CGNN problem. Section III presents our baseline solution. Section IV discusses solution optimization. Section V reports the experimental results. Section VI reviews related work, and Section VII concludes the paper.

II. Problem Definition

In this section, we formally define our CGNN queries over road-social networks. Table I summarizes the mathematical notations used throughout this paper.

A. Preliminaries

Road network. In this paper, a road network is modeled as an undirected weighted graph $G_r = (V_r, E_r)$, where V_r is a set of vertices and $E_r = \{(u, v) | u, v \in V_r \wedge u \neq v\}$ is a set of edges. A vertex $v_r \in V_r$ represents a road intersection or an end of a road, and an edge $e_r = (u, v) \in E_r$ represents a road segment that enables travel between vertices u and v. Each edge (u, v) is associated with a nonnegative weight $w(u, v)$ that represents the cost (e.g., distance or travel time) of a corresponding road segment.

Let p be a spatial point lying on the edge (u, v). The travel cost from vertex u to p, denoted by $w(u, p)$, is assumed to be proportional to the distance (length) between them. For two given points u and v in G_r, we use $dist(u, v)$ to represent the *network distance* (cost) between u and v, which is the sum of the edge weights along the least costly path from u to v. Note that the least costly path corresponds to the shortest path if the edge weight represents the distance.

A spatial-textual object[2] $o \in O$ is described by a spatial point and a set of keywords from a vocabulary, denoted by $o.loc$ and $o.T$, respectively. For simplicity, we assume that objects always lie along the edges (i.e., road segments) of G_r.

[1]To support general offline activity planning, the attendees include the query user and other invitees but not necessarily the activity initiator.

[2]Hereafter, when there is no ambiguity, "spatial-textual object" is abbreviated to "object".

TABLE I
SUMMARY OF NOTATIONS

Notation	Description
c, u_q	number of attendees and query user for an activity
w, ϵ	set of query keywords and a distance/time threshold
j	number of assembly points to select among
$G_s(J), G_s(V_{s^*})$	subgraphs of G_s induced by J and V_{s^*}
$G_s^k, G_s^k(u_q, c)$	k-core of G_s and cohesive k-core(s) of u_q in G_s
$G_s^{k_{max}}(u_q, c)$	$G_s^k(u_q, c)$ with maximum coreness
O	set of spatial-textual objects
$L_s(u_q)$	published location of u_q in the road network
Q	query points, i.e., $L_s(G_s^{k_{max}}(u_q, c))$
$N_{G_s}(v), N_{G_s^k}(v)$	sets of neighbors of v in G_s and G_s^k
$dg_{G_s}(v)$	degree of vertex v in G_s
$w(u, v)$	cost of the road segment between u and v
$dist(u, v)$	network distance between u and v
$dist_{sum}(o, Q)$	total cost from object o to Q
$dist_{max}(o, Q)$	maximum cost from o to any $q_i \in Q$

Suppose that an object o lies on an edge (u, v) with a given cost to each end vertex u and v. A new vertex can be created for o, and (u, v) is then replaced with edges (u, o) and (o, v).

Thus, we can define the aggregate network distance between an object o and a set of locations Q as follows:

$$dist_f(o, Q) = f_{\forall q_i \in Q} dist(o, q_i), \quad (1)$$

where f is an aggregate function that applies to sets of numbers. In this paper, we simultaneously consider two types of aggregate functions: $dist_{sum}(o, Q) = \sum_{\forall q_i \in Q} dist(o, q_i)$ and $dist_{max}(o, Q) = \max_{\forall q_i \in Q} dist(o, q_i)$.

For example, Fig. 1 displays a road network G_r and a textual description of each object. Edge (r_7, r_{15}) has a distance weight of $w(r_7, r_{15}) = 4$. An object o_5 lies on this edge, with $w(r_7, o_5) = 1.5$ and $w(r_{15}, o_5) = 2.5$. The path $r_{11}r_7r_{15}$ is the shortest path from r_{11} to r_{15}, with $dist(r_{11}, r_{15}) = 6$.

Social network. We model a social network as an unweighted and undirected graph $G_s = (V_s, E_s, L_s)$, where V_s is the set of vertices (representing users), $E_s \subseteq V_s \times V_s$ is the set of edges (i.e., social relations), and L_s is the set of mappings defined on V_s such that for each vertex v_s in V_s, $L_s(v_s)$ specifies the attributes of v_s (e.g., name, gender, and location). In our case, $L_s(v_s)$ provides a mapping of each user's location in the road network. Given a vertex v_s, we denote the set of its neighbors, $\{u_s | (u_s, v_s) \in E_s\}$, by $N_{G_s}(v_s)$. The degree of v_s, $|N_{G_s}(v_s)|$, is denoted by $dg_{G_s(v_s)}$. Then, we can formally define the induced subgraph as follows.

Definition 1: (Induced Subgraph). A graph $G_s(V_{s^*}) = (V_{s^*}, E_{s^*}, L_{s^*})$ is called the subgraph of G_s induced by V_{s^*}, where (1) $V_{s^*} \subseteq V_s$ and (2) edge $(u, v) \in E_{s^*}$, iff $u, v \in V_{s^*}$, $(u, v) \in E_s$ and (3) for each $v \in V_{s^*}$, $L_{s^*}(v) = L_s(v)$.

The k-core concept [4], which has been widely used to describe cohesive subgraphs, is formally defined as follows.

Definition 2: (k-Core). Given a graph G_s, an induced subgraph $G_s(J)$ is the k-core of G_s, denoted by G_s^k, iff the following two conditions are true. (1) *k-degree*: $dg_{G_s(J)}(v) \geq k$ for every $v \in J$. (2) *maximality*: For any J' such that $J \subset J' \subseteq V_s$, there exists a $u \in J' \setminus J$ such that $dg_{G_s(J')}(u) < k$.

Note that we have $G_s^{k+1} \subseteq G_s^k$ [6]. A vertex $v \in G_s$ has *coreness* k if it belongs to G_s^k but not to G_s^{k+1}. For any $V \subseteq V_s$, the largest coreness in a graph $G_s(V)$ is called the *coreness* of $G_s(V)$, which is denoted by $cn(G_s(V))$.

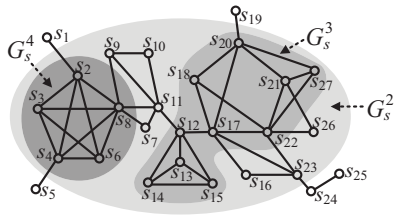


Fig. 2. k-cores of G_s in Fig. 1.

B. Definition of the CGNN Problem

Definition 3: (Cohesive k-Core). Given a constant c and a vertex $u_q \in J \subseteq V_s$ in the social network G_s, the cohesive k-core, denoted by $G_s^k(u_q, c)$, is a connected induced subgraph $G_s(J)$ such that $G_s(J)$ is a k-core and $|J| = c$.

We may conclude that the user group J of $G_s^k(u_q, c)$ consists of $\{u_q, u_1, \ldots, u_{c-1}\}$, whose coreness can be expressed as $cn(G_s(J))$. Among all k-cores $G_s^k(u_q, c)$, any k-core with the maximum coreness is referred to as the most cohesive k-core, denoted by $G_s^{k_{max}}(u_q, c)$; i.e., $\forall k' > k_{max}$, there exists no $G_s^{k'}(u_q, c)$. Note that k_{max} can be up to $c-1$.

Example 2: In Fig. 2, G_s^2, G_s^3 and G_s^4 are annotated in different colors. Suppose that $u_q = s_8$ and $c = 5$; then, the subgraph induced by $\{s_7, s_8, s_9, s_{10}, s_{11}\}$ is a cohesive 2-core. Similarly, another six subgraphs can also be identified as $G_s^2(s_8, 5)$. However, there exists one additional subgraph induced by $\{s_2, s_3, s_4, s_6, s_8\}$ with a coreness of 4 that is the most cohesive k-core, i.e., $G_s^4(s_8, 5)$. □

Definition 4: (Maximum Connected Component). The maximum connected component of graph G_s with regard to a vertex u_q and a constant c, denoted by $C_{max}(u_q, c)$, is the component that contains u_q in the k-core with maximum coreness among all k-cores of G_s such that $|C_{max}(u_q, c)| \geq c$.

Road-social network. A road-social network is a pair of graphs (G_r, G_s), where G_r is a road network and G_s is a social network. Each vertex $u_s \in G_s$ is associated with a vertex v_r or a spatial point p of G_r, indicating that user u_s is currently in location v_r or p, i.e., $L_s(u_s) = v_r$ or p in our case.

For example, Fig. 1 shows a road-social network. The mapping between s_{26} and r_9 in the published user location information indicates that s_{26} is currently at r_9.

Problem statement (CGNN). Given graphs G_s and G_r, an activity initiator issues a query $q = \langle u_q, c, w, \epsilon, j \rangle$, where u_q is the query user in G_s, c is the number of attendees, w is a set of keywords, ϵ is the network distance threshold and j is an integer. A CGNN query retrieves c users and a corresponding set A_q of j objects from set O with the smallest total cost to all locations in set Q over the road-social network such that (1) Q consists of locations in G_r associated with the c users from $G_s^{k_{max}}(u_q, c)$, denoted by $L_s(G_s^{k_{max}}(u_q, c))$, i.e., $\{L_s(u_q), L_s(u_1), \ldots, L_s(u_{c-1})\}$; (2) $A_q \subseteq O^* \subseteq O$ in G_r, $\forall o \in O^*, w \subseteq o.T$, and $|A_q| = j$; and (3) $\forall o \in A_q, \forall o' \in O^* \setminus A_q$, both $dist_{sum}(o, Q) \leq dist_{sum}(o', Q)$ and $dist_{max}(o, Q) \leq \epsilon$.

We regard the locations that comprise the most cohesive k-core for a group of users as the query points Q; e.g., the CGNN query $q = \langle Steve, 4, \text{“}cafe\text{”}, 6.5, 2 \rangle$ maps to Q1. Note that there may be more than one group of c users in $G_s^{k_{max}}(u_q, c)$. As illustrated in Section III-B, the local-search-based solution

Procedure CGNN_Framework{
Input: (G_r, G_s), $q = \langle u_q, c, w, \varepsilon, j\rangle$
Output: $G_s^{k_{max}}(u_q, c)$ and A_q
(1) find the most cohesive k-core of u_q with size c
(2) prune objects with keywords w and network distance ε
(3) verify top-j qualified objects from candidate set S

Fig. 3. Framework for a CGNN query.

returns each group in $G_s^{k_{max}}(u_q, c)$ but the heuristics only select the most promising one that rapidly leads to a solution.

III. Baseline Solution

This section presents our efficient approach for CGNN query answering. In Section III-A, we briefly introduce the framework of our solution. Section III-B shows how to find the most cohesive k-core, and Section III-C presents efficient algorithms for filtering out objects based on network distance restrictions. Section III-D describes new verification techniques for reducing the cardinality of the candidate objects and presents our final CGNN algorithm that integrates these techniques.

A. Framework of Our Approach

For processing CGNN queries, we propose the filtering-and-verification framework shown in Fig. 3. First, we locate and select the most cohesive k-core $G_s^{k_{max}}(u_q, c)$ for u_q. Second, we filter out objects beyond a given network distance threshold ϵ from the locations in G_r (i.e., query points) linked to each user in $G_s^{k_{max}}(u_q, c)$ and keep the remaining objects. Then, we compute the next nearest neighbor (NN) of each query point to obtain the candidate set S of objects until j common objects are found for all query points. Finally, we identify unpromising objects through inference and return the top-j qualified objects. The framework consists of a filtering (i.e., finding the most cohesive k-core in the social network), pruning objects based on keywords and a network distance ϵ on the road network, and verification (i.e., verifying qualified objects in candidate set S).

Note that we initially employ the popular inverted indexing technique to organize the objects. Thus, only objects $O^* \subseteq O$ whose textual descriptions correspond to all user-specified keywords w are retained in the search, and all others are pruned. Loading objects that do not match all query keywords could result in performance degradation, especially when w is not small. In the rest of this paper, we use O^* to denote the objects meeting keywords w.

B. Finding the Most Cohesive k-Core

To determine the most cohesive group relationships between u_q and its correlative neighbors, which may also be interconnected, we propose a local-search-based solution called center expansion (CE). The intuition is that the most cohesive group for a given vertex should be in the vicinity of the vertex. Thus, the entire G_s is not necessarily involved in the search. The local-search-based solution works as follows.

CE leverages the social-distance-based pruning (SD) as described in Section IV-A and the k-core decomposition [7], and treats u_q as a center in $C_{max}(u_q, c)$ for outward diffusion, i.e., a breadth search. In each round of CE, multiple vertices are selected without duplication from among the neighbors of the center, and an unmarked vertex is taken as the new

Algorithm 1: HeuristicsFramework(u_q, c)

Input: u_q: query user; c: size constraint
Output: $G_s^{k_{max}}(u_q, c)$
1 $G'_s \leftarrow$ Social Distance based Pruning(u_q, c);
2 $C_{max}(u_q, c) \leftarrow k$-core of G'_s;
3 $queue$.enqueue(u_q); $subset\ H \leftarrow \emptyset$;
4 **while** $queue \neq \emptyset$ **do**
5 $v \leftarrow queue$.dequeue(); $H \leftarrow H \cup \{v\}$;
6 **if** $|H| = c$ **then**
7 **return** H as $G_s^{k_{max}}(u_q, c)$;
8 **foreach** $(v, w) \in$ *edges in* $C_{max}(u_q, c)$ **do**
9 **if** w is not visited **then**
10 $queue$.enqueue(w);

center. When the size of an expansion reaches c, the subgraph and its coreness are recorded. Once all possible expansions in the current connected component of G_s^k have been completed, CE returns the induced subgraph(s) with coreness k, if any, as $G_s^{k_{max}}(u_q, c)$; otherwise, it expands u_q in the connected component of G_s^{k-1}. CE exploits the graph topology to scale out from the center such that the induced subgraph of each completed expansion is connected and includes u_q.

Although SD may reduce the search space and the time to compute the k-core is linear in the number of edges [6], i.e., $\mathcal{O}(|E_s|)$, the cost is high due to the numerous combinations. The most cohesive k-core must exist in a subgraph of size s containing c users, and we must verify at least C_{c-1}^{s} permutations; thus, the time complexity is $\mathcal{O}(s^c)$. Since the subgraph's coreness cannot be greater than that of $C_{max}(u_q, c)$, which is typically taken to be constant, we can think of s as being closely related to the scale of $|V_s|$, especially when G_s has a high density (e.g., a high average degree). Thus, the lower bound on the complexity of any exact algorithm is $\mathcal{O}(|V_s|^c)$. As shown in Section V, our exact algorithm takes a relatively long time even on a graph with only thousands of vertices. Thus, a time complexity of $\mathcal{O}(|V_s|^c)$ is already beyond reach, and the solution is intractable on big data.

Next, we propose two intelligent lightweight heuristics of constant cost. The basic idea for refining candidate generation is to use a priority queue to select the most promising vertex that will rapidly lead to a solution. Theoretically, Theorem 1 can be used as a prerequisite to support our heuristics.

Theorem 1: For graph G_s and query vertex u_q, given an attendee size c, there must exist $G_s^{k_{max}}(u_q, c) \subseteq C_{max}(u_q, c)$.

Proof: If $\nexists G_s^{k_{max}}(u_q, c) \subseteq C_{max}(u_q, c)$, we can obtain $k_{max} > cn(C_{max}(u_q, c))$, which contradicts Definition 4. □

Intuitively, $G_s^{k_{max}}(u_q, c)$ is a subgraph of the connected component of G_s containing u_q, whose size is no less than c and coreness is the greatest.

Largest increment in coreness (Lc). Selecting the vertex that leads to the largest increment in the coreness measure is a straightforward heuristic since the final goal for $G_s^{k_{max}}(u_q, c)$ is to find a subset H satisfying that $|H| = c$ and $cn(G_s(H))$ is as close as possible to or even equal to $c-1$. In this strategy, the priority $f(v)$ of a vertex v is defined as

$$f(v) = cn(G_s(H \cup \{v\})) - cn(G_s(H)). \tag{2}$$

This approach is a greedy one since only the improvement in $cn(H)$ in the next step is considered. Note that whenever

a vertex is added to H, the coreness of the current H is incremented by at most 1. Hence, this strategy is equivalent to random selection from the vertices adjacent to one of the vertices with the minimal degree in H.

Largest incidence (Li). This selection approach is more intelligent. The priority of a vertex v is defined as

$$f(v) = dg_{G_s(H\cup\{v\})}(v). \quad (3)$$

In this strategy, we select the vertex with the largest number of connections to the current H. This technique yields the fastest increase in the mean degree of $G_s(H)$. In general, the lowest degree of a graph increases as its density grows; consequently, a valid solution H with $cn(G_s(H))$ is expected to be found within finite steps if such a solution exists.

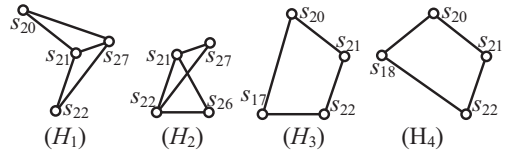


Fig. 4. Local search from s_{21} with $c = 4$.

Example 3: Suppose that s_{21} wishes to find the most cohesive group of 3 invitees in Fig. 2 ($u_q = s_{21}, c = 4$). CE returns $G_s^2(s_{21}, 4)$, consisting of 4 different user groups, as shown in Fig. 4. On the other hand, if we always choose the vertex with the largest number of connections to H, i.e., use the *Lc* or *Li* selection strategy, only $G_s(H_1)$ or $G_s(H_2)$ is found. Because u_q has 4 neighbors, one of which is selected after u_q, e.g., s_{20}, only s_{27} and s_{22} can be successively selected, yielding $G_s(H_1)$, due to the vertex priorities defined in Equations 2 and 3. □

Complexity analysis. Let n' and m' denote the numbers of vertices and edges, respectively, in $G_s(H)$. Generally, *Lc* and *Li* can be implemented in $\mathcal{O}(n' + m'logn')$ time. When a new vertex v is added to the queue, at most $dg_{C_{max}(u_q,c)}(v)$ queue update operations must be executed, that is, at most m' update operations in total. Each queue operation (insert, delete, priority update) generally has a time complexity of $\mathcal{O}(logn')$. Through careful design, *Li* can be implemented in $\mathcal{O}(n'+m')$ time with an expansion cost of $\mathcal{O}(1)$. We maintain a set of lists, each containing vertices with the same $f(v)$. Each time v is added to H, the $f(\cdot)$ value of v's neighbors (except those already in H) increases by 1. We move each influenced neighbor from its original $f(\cdot)$ list to the list for $f(\cdot)+1$. In this way, we can always find one vertex with the maximal $f(\cdot)$ in $\mathcal{O}(1)$ time.

If there is a tie in the maximal $f(\cdot)$ values for both strategies, we select the vertex with the minimal social distance to u_q based on the SD process described in Section IV-A; if there is still a tie, a vertex is selected arbitrarily.

C. Range Filter

To support general road networks with various cost models (e.g., distance or travel time), we adapt the incremental network expansion (INE) algorithm of [3] to incrementally access objects since INE does not rely on specific restrictions (e.g., Euclidean distances [3]) or precomputed road networks (e.g., shortcuts [8] or Voronoi diagrams [9]). In [3], the network distance is calculated from scratch for each object; to alleviate the computational cost, we integrate Dijkstra's algorithm [10] into INE such that the network distances to objects are calculated cumulatively from each query point $q_i \in Q$, i.e., $L_s(G_s^{k_{max}}(u_q, c))$, during network expansion.

Algorithm 2: RangeFilter(Q, ϵ)

Input: Q: query points; ϵ: network distance threshold
Output: R: objects with network distance restriction

```
1  R:=∅; M:=∅; dist_T:=0;
2  foreach query point q_i ∈ Q do
3      u ← the vertex where q_i is; M_i.enqueue(⟨u, 0⟩);
4      while M_i ≠ ∅ do
5          ⟨n, dist⟩ ← M_i.dequeue();
6          dist_T := dist; terminate while if dist_T > ε;
7          Mark the vertex n;
8          foreach unmarked vertex n_i ∈ Adj[n] do
9              Update dist(n_i) if dist(n_i) > dist + w(n, n_i);
10             foreach object o ∈ O* on (n, n_i) do
11                 dist(o) := dist + w(n, o); ⟨o, dist(o)⟩ → R_i;
12             M_i.enqueue(⟨n_i, dist(n_i)⟩);
13         foreach marked n_i ∈ Adj[n] do
14             foreach t ∈ R_i and t.o on (n_i, n) do
15                 t.dist := min{t.dist, dist + w(n, o)};
16     R_i := R_i \ t if ∃t ∈ R_i with t.dist > ε; ∪_{i=1}^{|Q|} R_i → R;
```

For simplicity of presentation, in Algorithm 2, we assume that each query point q_i starts from a vertex[3] u in Line 3. For $q_i \in Q$, a min-priority queue $M_i \in M$ is used to store the vertices accessed during expansion, where $dist(n) = \infty$ if vertex n has not been visited. Here, $dist(n) = dist(q_i, n)$ if a vertex n is marked (Line 7). Similarly, we use $dist(o)$ to compute $dist(q_i, o)$, and $dist(o) = dist(q_i, o)$ if an object o can be visited from both end vertices of its edge (Line 15). In Algorithm 2, vertices are accessed in nondecreasing order of their network distances from q_i. Line 6 updates $dist_T$, which is the lower bound on the network distance for any unmarked vertex. The expansion terminates when $dist_T > \epsilon$, which implies that $dist(q, n_x) > \epsilon$ for any unmarked vertex n_x. For each vertex n_i in the list of vertices adjacent to n, Line 9 updates $dist(n_i)$ if n_i is not marked, and the objects on edge (n, n_i), that satisfy the keyword constraint are loaded if n_i is visited for the first time (Line 10), followed by a network distance computation based on $dist(n)$ (Line 11). If vertex n_i is already marked, Lines 13-15 may update the network distances of the objects since both end vertices (n and n_i) of the edge are marked. Finally, objects with network distances longer than ϵ are pruned from $R_i \in R$. Note that Algorithm 2 can be adapted for directed road networks due to our integration of the INE and Dijkstra algorithm.

Example 4: In Fig. 1, suppose that q_1 is at r_6, with $\epsilon = 4.5$ and $w = \{\text{“}cafe\text{”}\}$. After r_6 is marked, its neighbors r_{15} and r_4 with updated network distances are placed in $M_1 = \{\langle r_{15}, 1\rangle, \langle r_4, 2.8\rangle\}$ since no object is found on either edge (r_6, r_{15}) or (r_6, r_4). Next, $\langle o_5, 3.5\rangle$ and $\langle o_4, 5.2\rangle$ on (r_{15}, r_7) and (r_{15}, r_{13}), derived from the first element in M_1, are discovered and placed in R_1. Then, $M_1 = \{\langle r_4, 2.8\rangle, \langle r_{16}, 4\rangle, \langle r_7, 5\rangle, \langle r_{13}, 6.7\rangle\}$ is updated. Now, r_4 reaches r_3 and r_5 with $\langle o_8, 3.8\rangle$, and r_{16} reaches r_{14} with $\langle o_1, 7.5\rangle$. Since the distance to the next entry $\langle r_3, 4.8\rangle$ in M_1 is larger than ϵ, finally, $R_1 = \{\langle o_5, 3.5\rangle, \langle o_8, 3.8\rangle\}$ with $\langle o_4, 5.2\rangle$ and $\langle o_1, 7.5\rangle$ discarded. □

[3]If q_i is on an edge, we can use the two end vertices of the edge to find nearby objects and merge the answer sets.

Complexity analysis. Let v' and e' denote the numbers of vertices and edges, respectively, accessed during road network expansion; then, Algorithm 2 can be implemented with a time complexity of $\mathcal{O}(c(v'log(v')+e'))$.

D. Verification

This section presents our verification techniques for CGNN queries on road-social networks. Henceforth, the next NN of a query point q_i refers to the object last obtained from R_i as described in Section III-C. The process of continuously obtaining the next NN for a query point is regarded as the expansion of that query point, and the expansion order determines the next query point to be expanded.

1) Obtaining the Candidate Set: The goal of this process is to obtain a set S of CGNN candidates that includes the final results. We assume that there are c query points with respect to Q, and the acquisition of S consists of the following steps.

We obtain the first NN, denoted by $o^1_{q_i}$, from R_i for each $q_i \in Q$, and place each into its own expansion list E_i. We then check whether there are j intersections among E_1 to E_c. If not, we continue to expand the search space for the query point q_i in sequence.

Note that before retrieving the NNs for each q_i, we can simply arrange all corresponding objects in the result set R_i generated by Algorithm 2 in ascending order of the network distance. After $o^k_{q_i}$ is retrieved, we place it at the end of the expansion list E_i, which is actually an ordered list of q_i's NNs, i.e., $E_i = \{o^1_{q_i}, o^2_{q_i}, \ldots, o^k_{q_i}\}$. This phase stops when there are j intersections among E_1 to E_c, i.e., $|\cap^c_{i=1} E_i| = j$, which means that we have identified a set CO of the first j common objects included in the expansions of all q_i. Here, CO is the set of current best CGNN candidates; then, we can compute the total network distance $dist_{sum}(o_j, Q)$ between the j-th common object o_j and the query points Q, denoted by dst_j.

Theorem 2: Once the j-th common object o_j in the expansions of all q_i is identified, i.e., $\cap^c_{i=1} E_i = \{o_1, o_2, \ldots, o_j\}$, the top-$j$ CGNNs are contained in $S = \cup^c_{i=1} E_i$.

Proof: Suppose that o' is one of the CGNNs and that $o' \notin S$. Because $o' \notin S$, $o' \notin E_i$, and E_i is an ordered list of q_i's NNs (i.e., $\forall o \in E_i$, $dist(o, q_i)$ is monotonically nondecreasing), $dist(o', q_i) \geq dist(o, q_i)$ for each $q_i \in Q$. (1) If $dist(o', q_i) > dist(o, q_i)$, we derive $dist_{sum}(o', Q) > dist_{sum}(o, Q)$. However, o_j is already included in S. This indicates that o' cannot be among the top-j CGNNs, which contradicts our assumption. (2) If $dist(o', q_i) = dist(o, q_i)$, $dist(o', q_i)$ is equivalent to the network distance between the last object in E_i and q_i. If the last object in every E_i is o_j, then o_j can represent o'; otherwise, $dist_{sum}(o_j, Q) < dist_{sum}(o', Q)$, such that o' cannot be among the top-j CGNNs. □

Algorithm 3 details the acquisition of the candidate set (Lines 1-9). For each query point q_i, the algorithm finds the first NN and adds it to the expansion list E_i for that query point. In addition, each query point is inserted into a min-heap (H) with weight $dist(q_i, o^1_{q_i})$ (Lines 2-3). Then, we obtain the next NN (e.g., o') of the top element in H and insert it into the corresponding expanded set E_{cur} of the current query point q_{cur}. We also update the weight of q_{cur} in H to $dist(q_{cur}, o')$. These procedures are repeated until there are j common objects in the expansion lists for all query points

Algorithm 3: ObtainCandidates(Q, R)

Input: Q: query points; R: objects of network distances
Output: S: candidate set; CO: common objects; dst_j: total network distances of o_j to Q

1 $H := \emptyset$; $S := \emptyset$; $dst_j := \infty$; $CO := \emptyset$;
2 **foreach** $q_i \in Q$ **do**
3 Get $o^1_{q_i} \in R_i$; $E_i = \{o^1_{q_i}\}$; H.push($\langle q_i, dist(q_i, o^1_{q_i})\rangle$);
4 **while** $|\cap^c_{i=1} E_i| < j$ **do**
5 $e \leftarrow$ Min(H); $q_{cur} := e.q$; get q_{cur}'s next NN $o' \in R_{cur}$;
6 $E_{cur} := E_{cur} \cup \{o'\}$; H.push($\langle q_{cur}, dist(q_{cur}, o')\rangle$);
7 **foreach** $o \in \cap^c_{i=1} E_i$ **do**
8 $CO := CO \cup \langle o, dist_{sum}(o, Q)\rangle$;
9 **return** $S := \cup^c_{i=1} E_i$; $dst_j := dist_{sum}(o_j, Q)$; CO;

(Lines 4-6). Once the set CO of common objects has been constructed, the total network distance dst_j between o_j and Q is returned (Line 9).

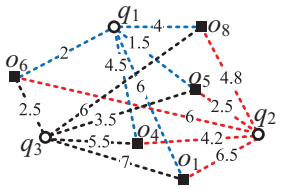


Fig. 5. Outline of network distances.

Example 5: Consider a query $q = \langle Steve, 3, \text{“}cafe\text{”}, 10, 2\rangle$. An outline of the network distances between Q (at r_7, r_{15}, and r_{11}) and the eligible objects is shown in Fig. 5. We now obtain the first NN for each $q_i \in Q$, i.e., $E_1 = \{o_5\}$, $E_2 = \{o_5\}$, $E_3 = \{o_6\}$, and $H = \{\langle q_1, 1.5\rangle, \langle q_2, 2.5\rangle, \langle q_3, 2.5\rangle\}$. The expansion of Q continues until 2 common objects are found. At this point, $E_1 = \{o_5, o_6, o_8, o_4\}$, $E_2 = \{o_5, o_4\}$, $E_3 = \{o_6, o_5, o_4\}$, and $H = \{\langle q_2, 4.2\rangle, \langle q_1, 4.5\rangle, \langle q_3, 5.5\rangle\}$. Thus, $S = \{o_4, o_5, o_6, o_8\}$, and $CO = \{\langle o_5, 7.5\rangle, \langle o_4, 14.2\rangle\}$, where $dst_2 = 14.2$. □

2) Verification: Our objective is to retain qualified objects that will be among the top-j CGNNs from the candidate set S. That is, if there are only j objects in S, then they are the CGNNs we seek. We now present our verification techniques in detail. Before verifying objects from S, we select a query point q_i according to a certain expansion order of the query points Q to retrieve the next NN and then compute $dst = \sum^c_{i=1} dist(q_i, o_i)$, where

$$o_i = \begin{cases} o' & \text{if } o' \in E_i; \\ o^1_{q_i} & \text{otherwise.} \end{cases} \tag{4}$$

If $dst > dst_j$, we calculate the set E'_i for each q_i.

$$E'_i = \begin{cases} \{o | o \in E_i, dist(q_i, o) < dist(q_i, o')\} & \text{if } o' \in E_i; \\ \emptyset & \text{otherwise.} \end{cases} \tag{5}$$

Then, let $S' = \cup^c_{i=1} E'_i$. For all $o \notin S \cap S'$, object o can be discarded from S, as stated in Theorem 3.

Theorem 3: Let o' be the next NN of a query point, and let $dst = \sum^c_{i=1} dist(q_i, o_i)$. If $dst > dst_j$ and $\forall o \notin S \cap S'$, it is verifiable that object o is not among the CGNNs and can be discarded from S.

Proof: For each i such that $o' \in E_i$, because $o \notin S \cap S'$ and $o \notin E'_i$, we have $dist(q_i, o) \geq dist(q_i, o')$

Algorithm 4: Verification(H, S, CO, R)

Input: H: min-heap; S: candidate set; CO: common objects; R: objects of network distances
Output: CGNNs

1 **while** $|S| > j$ **do**
2 $e \leftarrow \text{Min}(H)$; $q_{cur} \coloneqq e.q$; get q_{cur}'s next NN $o' \in R_{cur}$;
3 $dst = \Sigma_{i=1 \wedge o' \in E_i}^{c} dist(o', q_i) + \Sigma_{i=1 \wedge o' \notin E_i}^{c} dist(o_{q_i}^1, q_i)$;
4 **if** $dst > dst_j$ **then** $S \coloneqq S \cap (\cup_{i=1}^{c} E_i')$; $\{o'\} \rightarrow E_{cur}$;
5 **else**
6 $\{o'\} \rightarrow E_{cur}$;
7 **if** $o' \in \cap_{i=1}^{c} E_i$ **then**
8 Update S and CO; $\langle o_j, dst_j \rangle \leftarrow \text{Sort}(CO)$;
9 **if** $\exists t \in UP$ **and** $t.dist > dst_j$ **then**
10 $E_i' \leftarrow t.E_i$; $S \coloneqq S \cap (\cup_{i=1}^{c} E_i')$; $UP \setminus \{t\}$;
11 **else** $UP.\text{push}(\langle o', dst, E_1, E_2, \ldots, E_c \rangle)$;
12 $H.\text{push}(\langle q_{cur}, dist(o', q_{cur}) \rangle)$;
13 **return** CGNNs $\coloneqq S$;

by the definition of E_i'. Similarly, for each i such that $o' \notin E_i$, $dist(q_i, o) \geq dist(q_i, o_{q_i}^1)$. Since $dst = \sum_{i=1 \wedge o' \in E_i}^{c} dist(q_i, o') + \sum_{i=1 \wedge o' \notin E_i}^{c} dist(q_i, o_{q_i}^1)$, we have $\sum_{i=1}^{c} dist(q_i, o) \geq dst$, i.e., $dist_{sum}(o, Q) \geq dst$. Thus, $dist_{sum}(o, Q) > dst_j$ because $dst > dst_j$. Hence, object o is not among the CGNNs and can be discarded from S. □

If $dst \leq dst_j$, we need only to place q_i's next NN o' into its expanded set E_i and then determine whether o' is included in the expansions of all query points. If so, we replace $\langle o_j, dst_j \rangle$ with $\langle o', dst \rangle$ in the set of common objects, i.e., CO, where we update dst_j with the newly elected o_j after sorting; otherwise, we save an entry consisting of o', the corresponding dst, and every current E_i in the unpruned set UP. Note that this action enables us to continue verifying the qualified o remaining in S with both the expanding and backtracking strategies based on up-to-date dst_j and not-yet-validated entries in UP. We proceed in this way until there are only j objects in S, which are the top-j CGNNs.

In Algorithm 4, Line 2 first extracts the head of min-heap H and obtains its next NN in every round. Then, we apply our verification strategies to speed up the CGNN query processing. The value of dst is the lower bound on the total network distance from o' to all query points (Line 3). Line 4 verifies the objects that cannot be CGNNs by taking the intersection of S with the union of the E_i'. If $dst \leq dst_j$ and o' is a common object, then Line 8 updates the current set CO and dst_j. Line 12 updates the weight of q_{cur} in H. In Lines 9-10, we use the backtracking verification method to accelerate the convergence of the qualified objects retained from S.

Example 6: Continuing *Example* 5, we obtain q_2's next NN o_8 from R_2, and $dst = dist(q_1, o_8) + dist(q_2, o_8) + dist(q_3, o_6) = 11.3$. Since $dst < dst_2$, we insert o_8 into E_2 (i.e., $E_2 = \{o_5, o_4, o_8\}$), push $\langle o_8, 11.3 \rangle$ with each current E_i into UP, and update $H = \{\langle q_1, 4.5 \rangle, \langle q_3, 5.5 \rangle, \langle q_2, 4.8 \rangle\}$. Next, we obtain q_1's next NN o_1 from R_1 and $dst = dist(q_1, o_1) + dist(q_2, o_5) + dist(q_3, o_6) = 11$. Now, we have $E_1 = \{o_5, o_6, o_8, o_4, o_1\}$, push $\langle o_1, 11, E_1, E_2, E_3 \rangle$ into UP, and update $H = \{\langle q_2, 4.8 \rangle, \langle q_3, 5.5 \rangle, \langle q_1, 6 \rangle\}$. Then, we obtain q_2's next NN o_6 and $dst = dist(q_1, o_6) + dist(q_2, o_6) + dist(q_3, o_6) = 10.5$. Because $dst < dst_2$ and $o_6 \in \cap_{i=1}^{3} E_i$, we discard o_4 from S (i.e., $S = \{o_5, o_6, o_8\}$) and update $CO = \{\langle o_5, 7.5 \rangle, \langle o_6, 10.5 \rangle\}$; we now have $dst_2 = 10.5$. Since there is an entry in UP with $dst > dst_2$, e.g., $\langle o_8, 11.3, E_1, E_2, E_3 \rangle$, we can obtain E_1', E_2' and E_3' in accordance with the corresponding object and E_i of that entry: $E_1' = \{o_5, o_6\}$, $E_2' = \{o_5, o_4\}$, and $E_3' = \emptyset$ such that $S' = \{o_4, o_5, o_6\}$ and $S \cap S' = \{o_5, o_6\}$. Thus, the verified objects o_5 and o_6 are the top-2 CGNNs. □

Correctness and complexity. By Theorem 2, the set A_q of top-j CGNN objects surely belongs to S. Thus, only objects that cannot be in A_q are pruned by Theorem 3, and we will obtain the accurate top-j objects in response to a CGNN query because the entire verification process of Algorithms 3 and 4 corresponds to the expansion of the set R_i of objects within ϵ for each query point, requiring at most $|R|$ operations.

IV. Optimization Methods

In this section, we present four optimization methods to improve the CGNN query processing performance. Section IV-A presents an efficient algorithm for reducing the search space before finding the most cohesive k-core. Section IV-B develops a road network index, based on which we can progressively find the closest object candidates from each query point. Section IV-C proposes a novel technique for rapidly obtaining the candidate set. Section IV-D introduces a round-robin technique for increasing the overall CGNN query efficiency.

A. Social-distance-based Pruning (SD)

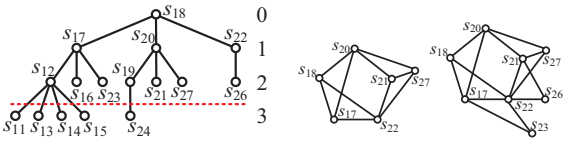
The pruning effect with k-core decomposition [7] alone is not significant. If we can identify vertices that must not appear in the query results, it may greatly reduce the search space and computational complexity. We assume that the coreness of the most cohesive k-core must be no less than 2; otherwise, retrieving only $G_s^1(u_q, c)$ becomes less meaningful. From this perspective, we regard the shortest social distance as the minimum number of edges between two vertices.

Theorem 4: Given a social network G_s, $v \in V_s$ must not be contained in $G_s^{k_{max}}(u_q, c)$ if the shortest social distance from v to u_q is no less than $c-1$.

Proof: Suppose that $v \in V_s$ is a vertex with a shortest social distance to u_q of no less than $c-1$ and is contained in $G_s^{k_{max}}(u_q, c)$. Then, there are at least $c-1$ edges between v and u_q in $G_s^{k_{max}}(u_q, c)$; i.e., the path from u_q to v passes through at least $c-2$ vertices. This implies that the coreness of $G_s^{k_{max}}(u_q, c)$ is 1, contradicting our assumption. □

Based on Theorem 4, a social distance pruning strategy is proposed to refine the cardinality of the potential candidate set by eliminating more distant vertices before finding the most cohesive k-core. Given a query user u_q and a constant c, we construct a breadth-first search (BFS) tree rooted at u_q; then, vertices at a tree height of at least $c-1$ can be pruned directly (the tree height is 0 at the root). When this pruning strategy is used to delete vertices, the degree of the remaining vertices in the underlying social network changes, and more vertices can then be pruned through k-core decomposition [7].

Example 7: Suppose that we wish to find $G_s^{k_{max}}(s_{18}, 4)$, a BFS tree rooted at s_{18}, as shown in Fig. 6(a). Vertices with a tree height of no less than 3, i.e., s_{11}, s_{13}, s_{14}, s_{15}, and s_{24}, can be pruned by Theorem 4. Furthermore, s_{12} can be pruned via k-core decomposition since its degree falls from 5



(a) BFS tree rooted at s_{18}. (b) G_s^3 and G_s^2.

Fig. 6. Social-distance-based pruning.

(in Fig. 2) to 1. Finally, G_s^3 and G_s^2, both containing s_{18}, are obtained as shown in Fig. 6(b). □

B. Road-network-based Indexing

To perform the range search described in Section III-C more efficiently, we adopt a road network index I_{RN}, based on which we can progressively find the closest objects within the network distance threshold ϵ from each query point.

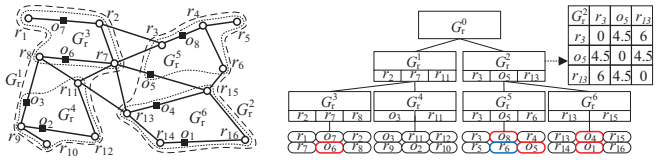
We first construct a new road network $G'_r = (V'_r, E'_r)$ from G_r as follows. Let $V'_r = V_r \cup O$. We add all edges in E_r that do not contain an object in O to E'_r with the same weight. The edges in E_r that contain one or more objects, are subdivided into multiple sub-edges depending on the number of objects; the weights of the resulting sub-edges are proportional.

Index structure. We adopt a state-of-the-art hierarchical tree structure [5] to index the road network G'_r. Initially, let G'_r be the root and partition it into f equal-sized subgraphs as children. Next, we recursively partition the children and repeat this step until each leaf-node's subgraph has no more than τ vertices. During partitioning, for each subgraph, we will add its borders into the corresponding node.

I_{RN} is a balanced search tree with the following properties. (1) Each node of I_{RN} represents a subgraph, and the root corresponds to G'_r. The subgraph represented by a parent node is a supergraph of those represented by its child nodes. All subgraphs at the same level of I_{RN} compose a partition of G'_r; i.e., any two such subgraphs are disjoint, and their union is G'_r. (2) Each nonleaf node has $f(\geq 2)$ children. (3) Each leaf node contains at most $\tau(\geq 1)$ vertices. All leaf nodes appear at the same level. (4) Each node has a border set (the set of vertices at the boundary of a partition) and a distance matrix. In the distance matrix, for a nonleaf node, the columns/rows are all borders of its children, and the value of each entry is the network distance between two borders; for a leaf node, the rows are all borders and the columns are all vertices for this node, and the value of each entry is the network distance between a border and a vertex. (5) The occurrence list (i.e., $\mathcal{L}(n)$) for a leaf node is the list of objects in this node; for a nonleaf node, it is the list of its children that contain objects.

Example 8: Fig. 7(b) shows the tree I_{RN} for a road network G_r. Each nonleaf node corresponds to a subgraph of G_r; e.g., G_r^2 corresponds to the right subgraph in Fig. 7(a). The borders of each node are shown in the rectangular boxes under that node. For instance, G_r^2 has 3 borders $\{r_3, o_5, r_{13}\}$, and its distance matrix is listed beside it. The vertices of each leaf node are shown in rounded rectangles; e.g., G_r^6 has 2 borders $\{r_{13}, r_{15}\}$ and 6 vertices $\{r_{13}, o_4, r_{15}, r_{14}, o_1, r_{16}\}$. □

While traversing I_{RN}, we use a priority queue PQ_i to maintain the nearest objects to each $q_i \in Q$. First, we locate the leaf node of q_i and objects O^* with leaf(n) and construct the occurrence list in a bottom-up manner. Initially, we calculate the network distance from every object $o \in \mathcal{L}(\text{leaf}(q_i))$ to



(a) Graph partition. (b) Index tree I_{RN}.

Fig. 7. Road network index.

q_i, and we place $\langle o, dist(q_i, o)\rangle$ in PQ_i. Next, we iteratively dequeue the first element $\langle e, dis\rangle$ of PQ_i and address it separately according to whether e is an object or a node. This process continues until an element with a dis greater than ϵ is dequeued from PQ_i, indicating that all objects within the network distance threshold ϵ in R_i have been retrieved. The same condition applies for a directed road network.

Example 9: Continuing *Example* 4, the range search process based on I_{RN} is as follows. First, we construct the occurrence list $\mathcal{L}$ based on $\{o_1, o_4, o_5, o_6, o_8\}$, access $G_r^5 = \text{leaf}(r_6)$ as the current node pointer T_n, and determine the minimum network distance T_{min} from q_1 to each border within T_n; here, $T_{min} = 0$ since r_6 is a border. Next, we push $dist(r_6, o_5) = 3.5$ and $dist(r_6, o_8) = 3.8$ into PQ_1 as $o_5, o_8 \in \mathcal{L}(G_r^5)$. Now, because $3.5 > T_{min}$, we update T_n to G_r^2 and $T_{min} = dist(r_6, G_r^2) = 3.5$ and push $dist(r_6, G_r^6) = 1$ into PQ_1 as $G_r^6 \in \mathcal{L}(T_n)$. Then, we access G_r^6, and push $dist(r_6, o_4) = 5.2$ and $dist(r_6, o_1) = 7.5$ into PQ_1 as $o_4, o_1 \in \mathcal{L}(G_r^6)$. We now insert $\langle o_5, 3.5\rangle$ into R_1 because $3.5 \leq T_{min}$. Because $dist(r_6, o_8) > T_{min}$, T_n is updated to G_r^0, $T_{min} = \infty$, and $dist(r_6, G_r^1) = 5$ is pushed into PQ_1 as $G_r^1 \in \mathcal{L}(T_n)$. Finally, we insert $\langle o_8, 3.8\rangle$ into R_1 and terminate the process because $dist(r_6, G_r^1) > \epsilon$. □

C. Rapid Acquisition of the Candidate Set

As described in Section III-D1, a strategy of progressive outward expansion from each query point is employed to obtain the candidate set. However, according to Theorem 2, we actually need only the first j common objects in the intersection of all expansion lists E_i. Accordingly, we propose an optimization process to rapidly obtain the candidate set.

Theorem 5: For all common objects O' in $\cap_{i=1}^{c} R_i$, if we can determine the j-th smallest maximum network distance (i.e., $dist_{max}(o_{jsm}, Q)$) from an object $o_{jsm} \in O'$ to each $q_i \in Q$, then we can construct E_i^* consisting of the objects in R_i with network distances no greater than $dist_{max}(o_{jsm}, Q)$, and the candidate set S^* is the union of all these E_i^*.

Proof: Here, we need only to prove that the object o_j in Theorem 2 is identical to o_{jsm} and that $S \subseteq S^*$. Suppose that o' is a common object in the expansions of all $q_i \in Q$ that is different from o_j. If o' is added after o_j, we can easily deduce that $dist_{max}(o', Q) > dist_{max}(o_j, Q)$ from the properties of the min-heap H in Algorithm 3. The same argument applies when o' is added before o_j since o_j is the j-th common object added to the expansions of all q_i, or equivalently, $dist_{max}(o_j, Q)$ is the j-th smallest maximum network distance among all $dist_{max}(o', Q)$. Because all objects within a distance ϵ of q_i are contained in R_i, i.e., $o' \in O'$, the j-th smallest maximum network distance among all $dist_{max}(o, Q), \forall o \in O'$, is equal to $dist_{max}(o_j, Q)$, i.e., $dist_{max}(o_{jsm}, Q) = dist_{max}(o_j, Q)$.

o_j may have been identified before all entries have been expanded to a distance no greater than $dist_{max}(o_j, Q)$ in H;

in other words, we may find o_j with high probability when an entry is expanded to a distance of $dist_{max}(o_j, Q)$ in H for the first time. However, E_i^* consists of the objects in R_i with network distances no greater than $dist_{max}(o_{jsm}, Q)$, which means that $E_i \subseteq E_i^*$. Thus, we can conclude that $S \subseteq S^*$ without any effect on the top-j CGNNs. □

Example 10: In Example 5, the second common object o_4 is identified when $\langle q_1, 4.5\rangle$ is expanded. While $dist_{max}(o_4, Q) = 5.5$, by Theorem 5, $E_2^* = \{o_5, o_4, o_8\}$ because $dist(q_2, o_8) = 4.8 < 5.5$, with E_1^* and E_3^* unchanged from E_1 and E_3. Thus, S^* is identical to S since o_8 is already contained in S. □

D. Round-robin Optimization

As illustrated in Section III-B, the CE algorithm places every possible expansion into a queue, and both the *Lc* and *Li* strategies may be repeatedly executed until the top-j objects are found for the current user group in a CGNN query. If any selected user can be identified as unlikely to be included in the most cohesive k-core, the search space can be reduced in both the social and spatial domains, and the overall query performance can be improved. In this section, we first deduce an intrinsic distance restriction between query points and then apply this restriction to the maximal $f(\cdot)$ value in the heuristics before a tie in social distance can arise. The procedure is repeated in a round-robin fashion; in other words, the intrinsic distance restriction is re-applied with respect to each selected vertex until the valid most cohesive k-core is obtained. On the one hand, this allows us to filter the invitees to reduce the diversity of the generated candidates, eliminating the need to find and verify objects for $G_s^{k_{max}}(u_q, c)$ that are qualified in terms of social connections but not locations. On the other hand, the intrinsic distance restriction consolidates the attendees such that the objects of interest will be relatively close; thus, the acquisition of the candidate set S is sped up.

Theorem 6: The network distance between the locations of each pair of users from $G_s^{k_{max}}(u_q, c)$ in road network G_r must be less than or equal to 2ϵ.

Proof: For any object $o \in A_q$ among the top-j objects, $dist_{max}(o, Q) \leq \epsilon$. Consider two users u_m and u_n in $G_s^{k_{max}}(u_q, c)$; the network distance between their locations, $L_s(u_m)$ and $L_s(u_n)$, in G_r is greater than 2ϵ. Then, $dist(o, L_s(u_m)) + dist(o, L_s(u_n)) \geq dist(L_s(u_m), L_s(u_n))$ due to triangular inequality, that holds iff o is on the least costly path from $L_s(u_m)$ to $L_s(u_n)$. Hence, $dist_{max}(o, Q)$ must be greater than ϵ because $dist(o, L_s(u_m)) + dist(o, L_s(u_n)) > 2\epsilon$, which contradicts our precondition. □

Example 11: Continuing *Example* 3, if $\epsilon = 4$, we can calculate $dist(r_3, r_9) = 9 > 2\epsilon$ from I_{RN} because s_{21} and s_{26} are at r_3 and r_9, respectively. By Theorem 6, H_2 in Fig. 4 can be discarded via *Lc* or *Li* instead of during verification. Thus, only $G_s(H_1)$ is a valid solution for the most cohesive k-core. □

V. Experimental Evaluation

This section evaluates the effectiveness and efficiency of our algorithms through comprehensive experiments.

[4]http://snap.stanford.edu/data/index.html
[5]https://www.cs.utah.edu/ lifeifei/SpatialDataset.htm
[6]http://www.dis.uniroma1.it/challenge9/index.shtml

TABLE II
Statistics of the Datasets

Dataset	Vertices	Edges	dg_{avg}	dg_{max}	h
California	21,048	21,693	1.03	8	-
San Francisco	174,956	223,001	2.55	8	-
Florida	1,070,376	1,356,399	2.53	12	-
Western USA	6,262,104	15,248,146	2.43	14	-
Facebook	4,039	88,234	43.69	1,045	8
Brightkite	58,228	214,078	7.35	1,134	17
Gowalla	196,591	950,327	9.67	14,730	15
Orkut	3,072,441	117,185,083	76.28	33,313	7
Twitter	17,069,982	476,553,560	55.84	109,214	8

A. Experimental Setting

Datasets. Five real-life social networks[4], Facebook (FB), Brightkite (BR), Gowalla (GO), Orkut (OR) and Twitter (TW), and four real road networks[5,6], California (CA), San Francisco (SF), Florida (FL) and Western USA (WU), are investigated in our experiments. CA and SF contain detailed street networks, whereas FL and WU consist only of highways and main roads. The statistics of these datasets are presented in Table II, where h denotes the average longest social distance.

We map each user v_s to the location v_r in the road network that matches the scale of his/her social network as follows: We first project the spatial locations into the range $[0, 1]$ in each dimension, and we generate $L_s(v_s)$ randomly (or by drawing from recent check-ins). If v_r has the smallest Euclidean distance to $L_s(v_s)$ in the projection space, we assume that v_r is the current location of v_s.

Algorithms. To our knowledge, no previous works have investigated the CGNN problem on general road-social networks. In this paper, we implement and evaluate a baseline algorithm, two advanced algorithms and the round-robin-based algorithm as described in Table III. Since the CE algorithm introduced in Section III-B is relatively slow even on a small road-social network, as illustrated by Exp-1 and Exp-2, we employ the optimal heuristic selection strategy *Li* as the baseline algorithm in this empirical study on the problem of finding the most cohesive k-core and verifying the top-j objects. In Table III, we also define the abbreviation for each technique used in the considered algorithms.

Parameters. We conducted experiments in different settings by varying six parameters, including the number of attendees c, the number of qualified objects j, the network distance threshold ϵ, the degree distribution of u_q, the difference $k_{ini} - k_{max}$ ($k_{ini} = cn(C_{max}(u_q, c))$), and the object ratio. The default values of c and j were 4 and 3, respectively, and the average degree of u_q was 11-20; in CA, SF, FL and WU, the respective default values of ϵ were 1, 4, 40 and 150 km, and the default object ratios were 0.1, 0.1, 0.01 and 0.01.

All programs were implemented in standard C++ and compiled with G++ in Linux. All experiments were performed on an Ubuntu Linux System with an Intel Xeon E7-4820 2 GHz CPU and 1 TB of memory.

B. Performance Evaluation

We investigate the efficiency of all algorithms listed in Table III, and then compare each under different settings. The results shown for each experiment are the average of 50 independent tests.

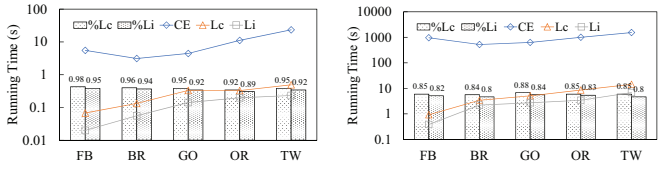
Exp-1: We evaluated the exact algorithm and the two heuristics for finding the most cohesive k-core described in

TABLE III
SUMMARY OF ALGORITHMS

Technique	Description
SD	social-distance-based pruning (Theorem 4)
ϵNN	range search for nearest neighbors within a threshold ϵ by the road network index (Section IV-B)
RC	rapid acquisition of the candidate set (Theorem 5)
IR	intrinsic distance restriction between users in Q (Theorem 6)

Algorithm	Description
Baseline	baseline method; consists of SD, the *Li* selection strategy (Theorem 1 and Equation 3), the range filter (Algorithm 2), and verification techniques (Theorems 2 and 3)
Adv1	Adv1=Baseline+RC; the acquisition of the candidate set S during the verification process (Theorem 2) is replaced with the rapid technique (Theorem 5)
Adv2	Adv2=Baseline+ϵNN+RC; the objects within the threshold distance ϵ are computed with the ϵNN strategy instead of the range filter (Algorithm 2) in addition to the modification adopted in Adv1
RR	round-robin-based method; incorporates IR (Theorem 6) into the *Li* selection strategy (Algorithm 1) for alternative validation in addition to the modifications in Adv2

Section III-B, namely, CE, *Lc* and *Li*, on the five social networks for $c = 4$ and $c = 32$. The accuracy (*%Lc* or *%Li*) is 1 if the entire user group obtained by *Lc*/*Li* is included in the CE results; otherwise, it is proportional to the number of users included. Fig. 8 indicates that CE is three to four orders of magnitude slower than the other two methods, especially when c is large, because CE enters the next iteration if it fails in the current round, while *Lc*/*Li* ends once the size expands to c. Although FB is small, CE takes more time on it than on BR or GO because SD depends on the social distance and c. As shown in Table II, $h=8$ in FB, which is sufficient to support good SD performance only for $c \leq 9$. The average accuracies of *Lc* and *Li* are essentially stable at approximately 0.9, but *Li* is nearly an order of magnitude faster. Thus, *Li* is selected for the Baseline algorithm for CGNN queries.



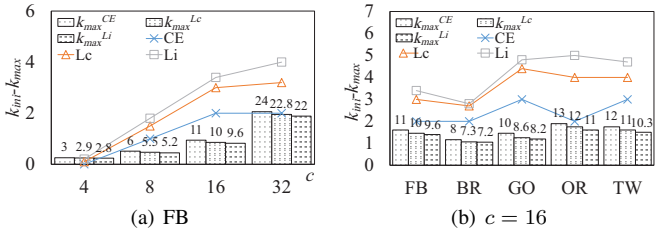
(a) $c = 4$ (b) $c = 32$

Fig. 8. Query performance and accuracy between CE and heuristics.

Exp-2: We examined k_{ini}–k_{max}, representing the difference between $cn(C_{max}(u_q, c))$ and the coreness of the attendees, with respect to c and the various social networks. As shown in Fig. 10(a), all curves increase with increasing c on FB; even the bars (e.g., k_{max}^{CE}, denoting the coreness k_{max} retrieved by CE) increase because with more attendees, guaranteeing their social connections is a greater challenge. Fig. 10(b) shows that the curves of *Lc* and *Li* are very close for the various social networks with $c=16$. This experiment again demonstrates the effectiveness of *Lc*/*Li* for CGNN queries.

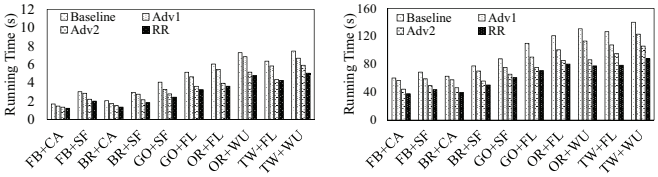
Exp-3: We explored the CGNN query performance of four algorithms, namely, Baseline, Adv1, Adv2 and RR, on real-world social-road networks for $c=4$ and $c=64$. Fig. 11 shows that the algorithms can satisfy user requirements for different numbers of attendees and that the query performance is rather stable as the scale of the road-social network varies.

Exp-4: We examined the query processing time with respect



(a) FB (b) $c = 16$

Fig. 10. $k_{ini} - k_{max}$ with respect to c between CE and heuristics.

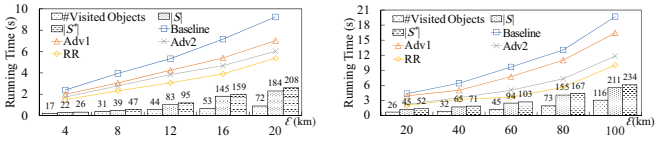


(a) $c = 4$ (b) $c = 64$

Fig. 11. Scalability.

to the number of attendees on scale-commensurate road-social networks: FB+CA, FB+SF, BR+CA, BR+SF, GO+SF, GO+FL, OR+FL, OR+WU, TW+FL and TW+WU. Fig. 9 shows the results; all curves increase as c increases. As long as *Lc* is fast enough, optimizations of the road network dominate the CGNN query efficiency. The more invitees there are, the more objects need to be validated. Among the algorithms, Adv2 and RR are superior to the other two; in particular, RR is twice as fast as Baseline because ϵNN uses the distance matrix of I_{RN} and IR prunes unpromising invitees, eliminating the need for multiple rounds of unnecessary object lookups.

Exp-5: We evaluated the efficiency and efficacy of the 4 algorithms for varying ϵ with the other parameters set to the default. In Fig. 12, all curves (running times) are rising, and the bars (numbers of candidates and visited objects) also rise with increasing ϵ. Although the number of candidate objects $|S^*|$ obtained by RC in Adv1, Adv2 and RR is slightly larger than that ($|S|$) in Baseline, expansion of the next NN is avoided. In Adv2 and RR, ϵNN significantly accelerates the acquisition of objects within ϵ, but the effect of IR is less prominent for larger ϵ. Moreover, the number of visited objects is always smaller than $|S|$ or $|S^*|$ due to our verification technique, which discards most candidate objects.



(a) GO+SF (b) OR+FL

Fig. 12. Query performance with respect to ϵ.

Exp-6: We examined the effect of j, as shown in Fig. 13, with the other parameters set to the default. Because $c = 4$, Adv1 is approximately equivalent to Baseline, and the same is true for Adv2 and RR. Here, the query performance is dominated by the road network techniques because finding the top-j objects requires only a few seconds due to the small values of c and ϵ; in particular, RR is twice as fast as Baseline. In addition, the number of candidate objects ($|S|$ in Baseline and $|S^*|$ in the others) increases with increasing j, but the number of visited objects remains almost unchanged.

Exp-7: We evaluated k_{max} with respect to the degree

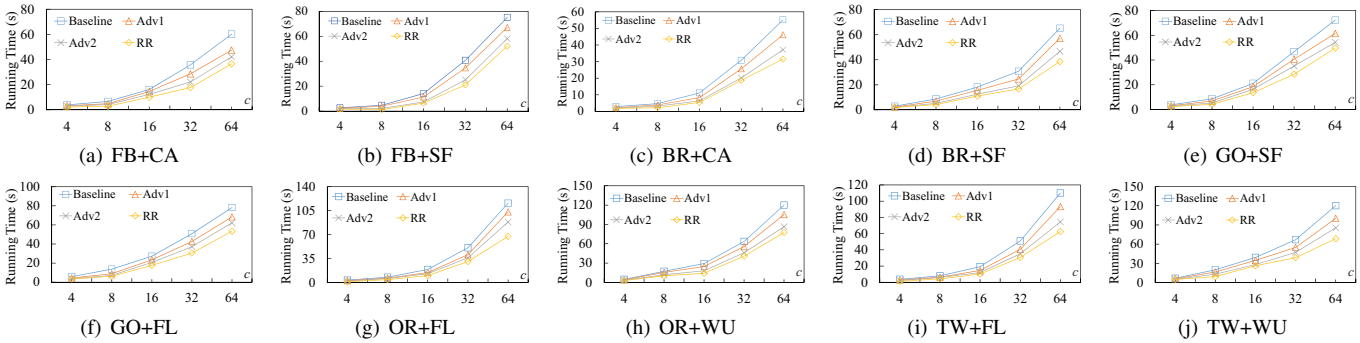


Fig. 9. Query processing time with respect to c.

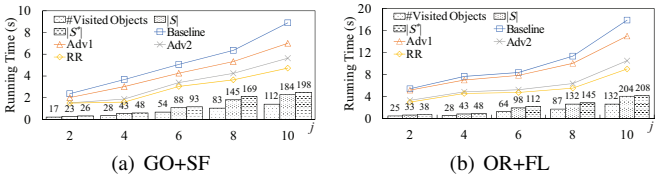


Fig. 13. Query performance with respect to j.

distribution of u_q, representing the variation of c, in the five social networks, as shown in Fig. 14. k_{max} increases when the degree distribution of u_q spans a larger interval, which corresponds to CE/*Lc*/*Li* finding the maximum coreness in $C_{max}(u_q, c)$. For FB, OR and TW, k_{max} is always close to the upper bound of the interval, followed by GO. From the average degrees in Table II, we conclude that FB, OR and TW are of relatively high density, affecting the performance of SD.

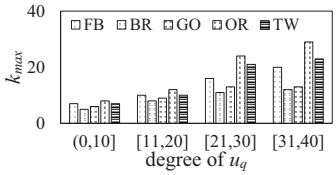


Fig. 14. Variation in k_{max} with the degree distribution of u_q.

Exp-8: We examined the k_{max} of the most cohesive k-core and the total travel cost $dist_{sum}$ of the attendees for the top-1 object with respect to c. In Fig. 15, k_{max} and $dist_{sum}$ increase with increasing c, but the average travel cost per attendee ($dist_{sum}/c$) is essentially steady because ϵ controls the network distance from each attendee to the optimal assembly point.

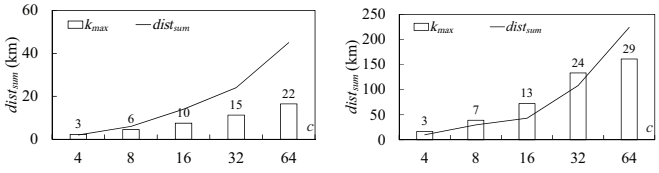


Fig. 15. $dist_{sum}$ and k_{max} with respect to c.

Exp-9: We evaluated the effect of the ratio of the number of objects to the number of edges, E_r, as shown in Fig. 16. This ratio varies from 0.001 to 1 in SF and from 0.0001 to 1 in FL. We then selected the objects with the keyword "restaurant" within each default ϵ. The run times of all algorithms decreased because with fewer objects, the objects are more sparsely and uniformly distributed and the average cost tends to be higher, as observed from $|S|$ or $|S^*|$ and the number of visited objects.

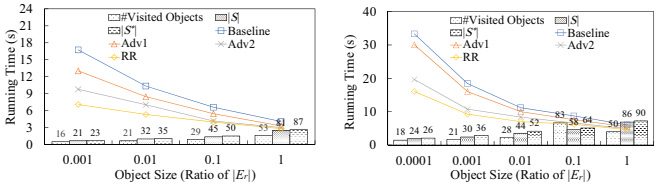


Fig. 16. Query performance with respect to the object ratio.

VI. Related Work

Geo-social query processing. [11] formulates a framework for geo-social query processing that builds queries based on atomic operations, by which some complex queries (e.g., nearest friends and range friends) can be answered. Recently, Li *et al.* [12] have studied spatial-aware interest group queries in location-based social networks and presented efficient processing algorithms. A geo-social k-cover group query [13] retrieves a minimum user group in which each user is socially related to at least k others and their associated regions (e.g., familiar/service regions) jointly cover all the given query points. [14] studies the kNN search on road networks incorporating social influence. Given a location q and a constant k, a socio-spatial group query [15] returns an approximate group of users such that each has no social relationship with at most k others in the group on average and their total distance to q is minimized. This approach focuses only on Euclidean spatial distances, and the assembly point is designated in advance; if queries with the same parameters are sent by diverse users, the results are identical.

k-core. k-core computation, first introduced by Seidman [4], is a fundamental graph problem with a wide spectrum of applications, such as network analysis [16], graph clustering [17], and network visualization [18]. A linear-time in-memory algorithm for computing core numbers of all vertices in a graph is presented in [6]. I/O-efficient algorithms for core number computation on graphs that cannot fit in the main memory of a machine are proposed in [19], [20]. Whether k-core decomposition of large networks can be computed using a consumer-grade PC is explored in [7]. Local computation and estimation of core numbers are studied in [21], [22]. Algorithms for core number maintenance on dynamic graphs are proposed in [23]. However, the most cohesive k-core model is better suited to personalized user group queries (i.e., the attendees invited by different query users are varied), and concurrently has more desirable properties, including society, cohesiveness, connectivity and maximization.

Nearest neighbor search. As one of the most important

queries among NN search variants, a group nearest neighbor (GNN) query [1] retrieves the point(s) in a given set of points P with the smallest sum of distances to all points in another given set of points Q. An aggregate nearest neighbor (ANN) query [2], which returns the point(s) in P that minimizes an aggregate function with respect to Q, was subsequently proposed as an extension of GNN query, which is equivalent to ANN query with an aggregate function of *sum* exclusively. As the methods of [1], [2] are applicable only in Euclidean space, [24] investigated the solution of ANN queries in road networks. However, this approach is not applicable in the case of edge weights that are disproportionate to the corresponding physical lengths, and the query performance is inefficient when the scale of Q is large. Moreover, [25] addresses the problem of ANN query monitoring for moving objects in Euclidean space. [26] discusses ANN queries with moving query points. [27] explores ANN queries for query points with location privacy concerns. [28], [29] addresses ANN queries on uncertain databases and graphs. In these works, neither the social connectivity among the spatial query points in Q nor textual descriptions of the spatial objects in P are considered.

Our work, being more flexible and scalable, is totally different from that described above: (1) the assembly points are analyzed dynamically with regard to the optimal attendees, (2) the unique social topology can be adequately considered due to the limited number of attendees, (3) the core number k is self-optimized to obtain the highest familiarity, and (4) distance restrictions based on a combination of *sum* and *max* functions are simultaneously applied to road networks. In real life, CGNN queries are often the most natural way to express the requests of an activity initiator or mobile user who wishes to organize an offline activity. To our knowledge, this paper proposes the first practical algorithm for solving the CGNN problem on general road-social networks.

VII. Conclusions

In this paper, we define a pragmatic query type, namely, CGNN queries, to identify suitable spatial-textual objects as assembly points for a group of optimal attendees over road-social networks. To our knowledge, no previous solution has addressed such type of scenario in automated offline activity planning services based on the social and geographically spatial relationship of activity attendees. We show that the problem is nontrivial and devise an efficient framework including effective filtering and verification techniques to reduce the processing time. Moreover, it is shown that with optimization, the query performance is improved and less time is required to find the optimal solution in both social and spatial domains. Experimental results for real-world road-social networks significantly demonstrate that our approaches are highly scalable and robust in both efficiency and efficacy. A possible direction for future work is to integrate other user attributes, e.g., user preferences, to filter the activity attendees. Potential future research also includes joint social and road processing on networks stored in a distributed manner.

Acknowledgment

Ye Yuan is supported by the National Key R&D Program of China (NO. 2016YFC1401900), the NSFC (Grant No. 61572119 and 61622202) and the Fundamental Research Funds for the Central Universities (Grant No. N150402005). Guoren Wang is supported by the NSFC (Grant No. U1401256, 61732003 and 61729201). Lei Chen is supported by the NSFC (Grant No. 61732003). Xiang Lian is supported by the NSF OAC No. 1739491, and Lian Start Up No. 220981, Kent State University.

References

[1] D. Papadias, Q. Shen, Y. Tao, and K. Mouratidis, "Group nearest neighbor queries," in *ICDE*, 2004, pp. 301–312.
[2] D. Papadias, Y. Tao, K. Mouratidis, and C. K. Hui, "Aggregate nearest neighbor queries in spatial databases," *TODS*, vol. 30, no. 2, pp. 529–576, 2005.
[3] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao, "Query processing in spatial network databases," in *VLDB*, 2003, pp. 802–813.
[4] S. B. Seidman, "Network structure and minimum degree," *Social networks*, vol. 5, no. 3, pp. 269–287, 1983.
[5] R. Zhong, G. Li, K.-L. Tan, L. Zhou, and Z. Gong, "G-tree: An efficient and scalable index for spatial search on road networks," *TKDE*, vol. 27, no. 8, pp. 2175–2189, 2015.
[6] V. Batagelj and M. Zaversnik, "An o(m) algorithm for cores decomposition of networks," *CoRR, cs.DS/0310049*, 2003.
[7] W. Khaouid, M. Barsky, V. Srinivasan, and A. Thomo, "K-core decomposition of large networks on a single pc," *PVLDB*, vol. 9, no. 1, pp. 13–23, 2015.
[8] S. Luo, Y. Luo, S. Zhou, G. Cong, J. Guan, and Z. Yong, "Distributed spatial keyword querying on road networks." in *EDBT*, 2014, pp. 235–246.
[9] M. Kolahdouzan and C. Shahabi, "Voronoi-based k nearest neighbor search for spatial network databases," in *VLDB*, 2004, pp. 840–851.
[10] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.
[11] N. Armenatzoglou, S. Papadopoulos, and D. Papadias, "A general framework for geo-social query processing," *PVLDB*, vol. 6, no. 10, pp. 913–924, 2013.
[12] Y. Li, D. Wu, J. Xu, B. Choi, and W. Su, "Spatial-aware interest group queries in location-based social networks," *DKE*, vol. 92, pp. 20–38, 2014.
[13] Y. Li, R. Chen, J. Xu, Q. Huang, H. Hu, and B. Choi, "Geo-social k-cover group queries for collaborative spatial computing," *TKDE*, vol. 27, no. 10, pp. 2729–2742, 2015.
[14] Y. Yuan, X. Lian, L. Chen, Y. Sun, and G. Wang, "Rsknn: knn search on road networks by incorporating social influence," *TKDE*, vol. 28, no. 6, pp. 1575–1588, 2016.
[15] D.-N. Yang, C.-Y. Shen, W.-C. Lee, and M.-S. Chen, "On socio-spatial group query for location-based social networks," in *SIGKDD*, 2012, pp. 949–957.
[16] J. I. Alvarez-Hamelin, L. Dall'Asta, A. Barrat, and A. Vespignani, "Large scale networks fingerprinting and visualization using the k-core decomposition," in *NIPS*, 2006, pp. 41–50.
[17] C. Giatsidis, F. D. Malliaros, D. M. Thilikos, and M. Vazirgiannis, "Corecluster: A degeneracy based graph clustering framework." in *AAAI*, vol. 14, 2014, pp. 44–50.
[18] F. Zhao and A. K. Tung, "Large scale cohesive subgraphs discovery for social network visual analysis," *PVLDB*, vol. 6, no. 2, pp. 85–96, 2012.
[19] J. Cheng, Y. Ke, S. Chu, and M. T. Özsu, "Efficient core decomposition in massive networks," in *ICDE*, 2011, pp. 51–62.
[20] D. Wen, L. Qin, Y. Zhang, X. Lin, and J. X. Yu, "I/o efficient core graph decomposition at web scale," in *ICDE*, 2016, pp. 133–144.
[21] W. Cui, Y. Xiao, H. Wang, and W. Wang, "Local search of communities in large graphs," in *SIGMOD*, 2014, pp. 991–1002.
[22] M. P. OBrien and B. D. Sullivan, "Locally estimating core numbers," in *ICDM*, 2014, pp. 460–469.
[23] Y. Zhang, J. X. Yu, Y. Zhang, and L. Qin, "A fast order-based approach for core maintenance," in *ICDE*, 2017, pp. 337–348.
[24] M. L. Yiu, N. Mamoulis, and D. Papadias, "Aggregate nearest neighbor queries in road networks," *TKDE*, vol. 17, no. 6, pp. 820–833, 2005.
[25] H. G. Elmongui, M. F. Mokbel, and W. G. Aref, "Continuous aggregate nearest neighbor queries," *GeoInformatica*, vol. 17, no. 1, pp. 63–95, 2013.
[26] J. Li, J. R. Thomsen, M. L. Yiu, and N. Mamoulis, "Efficient notification of meeting points for moving groups via independent safe regions," *TKDE*, vol. 27, no. 7, pp. 1767–1781, 2015.
[27] T. Hashem, L. Kulik, and R. Zhang, "Privacy preserving group nearest neighbor queries," in *EDBT*, 2010, pp. 489–500.
[28] X. Lian and L. Chen, "Probabilistic group nearest neighbor queries in uncertain databases," *TKDE*, vol. 20, no. 6, pp. 809–824, 2008.
[29] Z. Liu, C. Wang, and J. Wang, "Aggregate nearest neighbor queries in uncertain graphs," *WWW*, vol. 17, no. 1, pp. 161–188, 2014.