Topic-based Community Search over Spatial-Social Networks

Ahmed Al-Baghdadi and Xiang Lian
Department of Computer Science, Kent State University
Kent, OH 44242, USA
{aalbaghd, xlian}@kent.edu

ABSTRACT

Recently, the community search problem has attracted significant attention, due to its wide spectrum of real-world applications such as event organization, friend recommendation, advertisement in e-commence, and so on. Given a query vertex, the community search problem finds dense subgraph that contains the query vertex. In social networks, users have multiple check-in locations, influence score, and profile information (keywords). Most previous studies that solve the CS problem over social networks usually neglect such information in a community. In this paper, we propose a novel problem, named *community search over spatial-social networks* (TCS-SSN), which retrieves community with high social influence, small traveling time, and covering certain keywords. In order to tackle the TCS-SSN problem over the spatial-social networks, we design effective pruning techniques to reduce the problem search space. We also propose an effective indexing mechanism, namely *social-spatial* index, to facilitate the community query, and develop an efficient query answering algorithm via index traversal. We verify the efficiency and effectiveness of our pruning techniques, indexing mechanism, and query processing algorithm through extensive experiments on real-world and synthetic data sets under various parameter settings.

1. INTRODUCTION

With the increasing popularity of *location-based social networks* (e.g., Twitter, Foursquare, and Yelp), the community search problem has drawn much attention [5, 58, 25, 56] due to its wide usage in many real applications such as event organization, friend recommendation, advertisement in e-commence, and so on. In order to enable accurate community retrieval, we need to consider not only social relationships among users on social networks, but also their spatial closeness on spatial road networks. Therefore, it is rather important and useful to effectively and efficiently conduct the community search over a so-called *spatial-social network*, which is essentially a social-network graph integrated with spatial
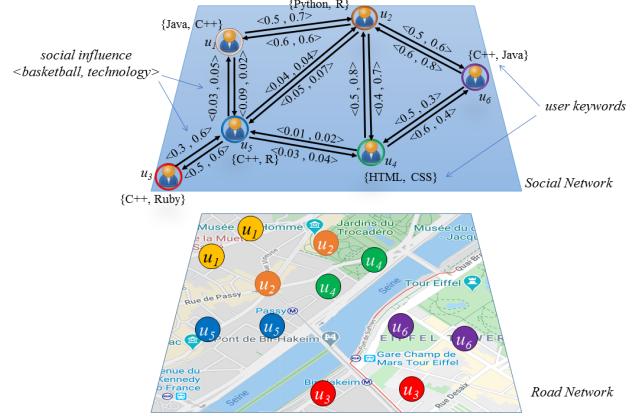
Figure 1: An Example of Spatial-Social Networks.

road networks, where social-network users are mapped to their check-in locations on road networks.

In reality, social-network users are very sensitive to post/ propagate messages with different topics [15]. Therefore, with different topics such as movie, food, sports, or skills, we may obtain different communities, which are of particular interests to different domain users (e.g., social scientists, sales managers, headhunting companies, etc.). In this paper, we will formalize and tackle a novel problem, namely *topic-based community search over spatial-social networks* (TCS-SSN), which retrieves topic-aware communities, containing a query social-network user, with high social influences, social connectivity, and spatial/social closeness.

Below, we provide a motivation example of finding a group (community) of spatially/socially close people with certain skills (keywords) from spatial-social networks to perform a task together.

EXAMPLE 1. **(Building a Project Team)** *Figure 1 illustrates an example of a spatial-social network, G_{rs}, which combines social networks G_s with road networks G_r. In social networks G_s, users, $u_1 \sim u_6$, are vertices, and directed edges (e.g., e_{u_1,u_2}) represent friend relationships between any two users. Each user (e.g., u_1) is associated with a set of keywords that represent one's skills (e.g., programming skills). Furthermore, each directed edge e_{u_i,u_j} has a weight vector with respect to two topics, (basketball, technology), each weight representing the social influence of user u_i towards user u_j with respect to a topic. For example, for the* technology *topic, the social influence of user u_2 on user u_4 (i.e., edge e_{u_2,u_4}) is given by 0.7, whereas the influence of user u_4 on user u_2 (i.e., edge e_{u_4,u_2}) is 0.8. That is, influence probabilities on the two directed edges between users u_2 and u_4 may be asymmetric.*

Moreover, in road networks G_r, vertices are intersection points and edges indicate road segments containing connecting those in-

tersection points. Each social-network user from social network G_s has multiple check-in locations on the spatial network G_r.

In order to accomplish a programming project related to basket-ball websites, a project manager u_2 may want to find a voluntary (non-profit) team of developers who have the programming skills such as $\{Python, HTML, C++\}$ (i.e., topics), are socially and spatially close to each other, and highly influence each other on basketball topics. In this case, the manager can issue a TCS-SSN query over spatial-social networks G_{rs} and call for a community of developers who can meet and complete the project task.

In Figure 1, although user u_6 has the query keyword $C++$ and high influence score with u_2, he/she resides in a place far away from u_2. Thus, u_6 will not be considered. Similarly, user u_5 will not be considered, since its influence score based on `basketball` *and* `technology` *topic is very low. Therefore, in this running example, a community $\{u_2, u_1, u_4\}$ will be returned as potential team members.* ∎

As described in the example above, it is important that relationships among team members (programmers) to be high, so this will encourage them to join and better communicate with their friends. Also, we would like people with certain skills (topics) such as programming skills or front-end and back-end skills. Furthermore, we want programmers to reside within a certain road-network distance such that team members do not have to drive too long to meet.

Prior works on the community search usually consider the community semantics either by spatial distances only [16] and/or structural cohesiveness [25]. They did not consider topic-aware social influences. Moreover, some works [32] considered communities based on topics of interests (e.g., attributes), however, topic-based social influences among users are ignored.

In contrast, in this paper, our TCS-SSN problem will consider the community semantics by taking into account degree of interests (sharing similar topics of interest) among users, degree of interactions (interacting with each other frequently), degrees of mutual influences (users who influence each other), structural cohesiveness (forming a strongly connected component on the social network), and spatial cohesiveness (living in places nearby on road networks).

It is rather challenging to efficiently and effectively tackle the TCS-SSN problem, due to the large scale of spatial-social networks and complexities of retrieving communities under various constraints. Therefore, in this paper, we will propose effective pruning mechanisms that can safely filter out false alarms of candidate users and reduce the search space of the TCS-SSN problem. Moreover, we will design cost-model-based indexing techniques to enable our proposed pruning methods, and propose an efficient algorithm for TCS-SSN query answering via the index traversal.

Specifically, we make the following contributions in this paper.

1. We define the problem of the *topic-based community search over spatial-social networks* (TCS-SSN) in Section 2.

2. We propose effective pruning strategies to reduce the search space of the TCS-SSN problem in Section 3.

3. We design effective, cost-model-based indexing mechanisms to facilitate the TCS-SSN query processing in Section 4.

4. We propose an efficient query procedure to tackle the TCS-SSN problem in Section 5.

5. We demonstrate through extensive experiments the efficiency and effectiveness of our TCS-SSN query processing approach over real/synthetic data sets in Section 6.

Section 7 reviews previous works on query processing in social and/or road networks. Finally, Section 8 concludes this paper.

Table 1: Frequently used symbols and their descriptions.

Symbol	Description
G_r, G_s, and G_{rs}	a spatial network, a social network, and a spatial-social network, respectively
S	a set (community) of social-network users
u, v, u_j, or v_j	a social-network user
$u.key$	a vector of possible keywords associated with user u_j
$u.L$	a set of check-in locations, $u.loc_i$, by user u
$e_{u,v}$	a directed edge from user u to user v
$e_{u,v}.\mathcal{T}$	a vector of topics of interests associated with edge $e_{u,v}$
$sup(e)$	the support of an edge e
$tp_{u,v}^j$	a weight probability on the topic j from user u to v
$infScore(.,.)$	an influence score function
$avg\_dist_{RN}(u,v)$	the average road-network distance between users u and v
$dist_{SN}(u,v)$	No. of hops between users u and v on social networks
$\mathcal{P}_{RN}$	a set of l road-network pivots, $rpiv_i$
$\mathcal{P}_{SN}$	a set of h social-network pivots, $spiv_i$
$\mathcal{P}_{index}$	a set of ι index pivots

2. PROBLEM DEFINITION

In this section, we provide formal definitions and data models for social networks, spatial networks, and their combination, spatial-social networks, and then define our novel query of **t**opic-based **c**ommunity **s**earch over **s**patial-**s**ocial **n**etworks (TCS-SSN).

2.1 Social Networks

We formally define the data model for social networks, as well as structural cohesiveness and social influence in social networks.

DEFINITION 1. (**Social Networks, G_s**) *A social network, G_s, is a triple $(V(G_s), E(G_s), \phi(G_s))$, where $V(G_s)$ is a set of M users $u_1, u_2, \ldots,$ and u_M, $E(G_s)$ is a set of edges $e_{u,v}$ (friendship between two users u and v), and $\phi(G_s)$ is a mapping function: $V(G_s) \times V(G_s) \to E(G_s)$.*

In Definition 1, each user, u_j (for $1 \leq j \leq M$), in the social network G_s is associated with a vector of possible keywords (or skills) $u_j.key = (key_1^j, key_2^j, \ldots, key_{|key|}^j)$.

Each edge (friendship), $e_{u,v}$ ($\in E(G_s)$), in social networks is associated with a vector of topics of interest $e_{u,v}.\mathcal{T} = (tp_{u,v}^1, tp_{u,v}^2, \ldots, tp_{u,v}^{|\mathcal{T}|})$, where $tp_{u,v}^j$ is the influence probability (weight) of interested topic j and $|\mathcal{T}|$ is the size of the topic set $e_{u,v}.\mathcal{T}$.

Modeling Structural Cohesiveness: Previous works usually defined the community as a subgraph in social networks G_s with high structural cohesiveness. In this paper, to capture structural cohesiveness in G_s, we consider the connected (k, d)-truss [32].

Specifically, we first define a *triangle* in G_s, which is a cycle of length 3 denoted as $\triangle_{u_a u_b u_c}$, for user vertices $u_a, u_b, u_c \in V(G_s)$; the support, $sup(e)$, of an edge $e \in E(G_s)$ is given by the number of triangles containing e in G_s [52]. Then, the connected (k, d)-truss is defined as follows.

DEFINITION 2. (**Connected (k,d)-Truss [32]**): *Given a graph G_s, and an integer k, a connected subgraph $S \in G_s$ is called a (k,d)-truss if two conditions hold: (1) $\forall e \in E(S), sup(e) \geq (k - 2)$, and (2) $\forall u, v \in V(S), dist_{SN}(u, v) < d$, where $sup(e)$ is the number of triangles containing e and $dist_{SN}(u, v)$ is the shortest path distance (the minimum number of hops) between users u and v on social networks.*

Modeling Social Influences: Now, we discuss the data model for social influences in social networks. Each edge $e_{u,v}$ is associated with a vector, $\mathcal{T} = (tp_{u,v}^1, tp_{u,v}^2, \ldots, tp_{u,v}^{|\mathcal{T}|})$, of influence probabilities on different topics. Further, we denote the path $path_{u,v}$ as a path on the social network connecting two users u and v such that $path_{u,v} = (u = a_1 \to a_2 \to \cdots \to a_{|path_{u,v}|} = v)$. It is worth noting that, Barbieri et al. [6] extended the classic IC and LT models to be topic-aware and introduced a novel topic-aware influence-driven propagation model that is more accurate in

describing real-world cascades than standard propagation models. In fact, users have different interests and items have different characteristics, thus, we follow the text-based topic discovery algorithm [6, 15] to extract user's interest topics and their distribution on each edge. Specifically, for each edge $e_{u,v}$, we obtain an influence score vector, for example, (basketball:0.1, technology:0.8), indicating that the influence probabilities of user v influenced by user u on topics, basketball and technology, are 0.1 and 0.8, respectively.

Below, we define the influence score function.

DEFINITION 3. (*Influence Score Function [15]*). *Given a social-network graph G_s, a topic vector $\mathcal{T}$, and two social-network users $u, v \in V(G_s)$, we define the influence score from u to v as follows:*

$$infScore(u, v|\mathcal{T}) = \max_{\forall path_{u,v} \in G_s} \{infScore(path_{u,v}|\mathcal{T})\}. \quad (1)$$

For two vertices $u, v \in V(G_s)$ such that $path_{u,v} = (u = a_1 \to a_2 \to \cdots \to a_{|path_{u,v}|} = v)$, and a topic vector $\mathcal{T}$, we define the influence score on $path_{u,v}$ as follows:

$$infScore(path_{u,v}|\mathcal{T}) = \prod_{i=1}^{|path_{u,v}|-1} f(a_i, a_{i+1}|\mathcal{T}), \quad (2)$$

where $f(a_i, a_{i+1}|\mathcal{T})$ is the influence score from a_i to a_{i+1} of the two adjacent vertices a_i and a_{i+1} based on the query topic vector $\mathcal{T}$. We compute the influence score between any two adjacent vertices a_i and a_{i+1} as follows:

$$f(a_i, a_{i+1}|\mathcal{T}) = \sum_{j=1}^{|\mathcal{T}|} tp_{u,v}^j \cdot \mathcal{T}^j, \quad (3)$$

where $tp_{u,v}^j$ is the weight probability on topic j, $\mathcal{T}^j$ is the j-th query topic, and $|\mathcal{T}|$ is the length of the topic set $\mathcal{T}$.

In Definition 3, we define the influence score between any two users in the social network G_s. Given a topic vector $\mathcal{T}$ and a subset $S \subseteq G_s$, we define the influence score between subgraph S and a user v as follows:

$$infScore(S, v|\mathcal{T}) = \min_{\forall u \in S} \{infScore(u, v|\mathcal{T})\}, \quad (4)$$

where $infScore(u, v|\mathcal{T})$ is defined in Eq. (2).

Note that, the pairwise influence (or mutual influence) in our TCS-SSN problem indicates the influence of one user on another user in the community. In particular, the pairwise influence is not symmetric, in other words, for two users u and v, the influence of u on v can be different from that of v on u. Thus, in our TCS-SSN community definition, we require both influences, from u to v and from v to u, be greater than the threshold θ (i.e., mutual influences between u and v are high), which ensures high connectivity or interaction among users in the community. Other metrics such as pairwise keyword similarity [7] (e.g., Jaccard similarity) are usually symmetric (providing a single similarity measure between two users), which cannot capture mutual interaction or influences. Most importantly, users u and v may have common keywords/topics, however, it is possible that they may not have high influences to each other in reality.

2.2 Spatial Road Networks

Next, we give the formal definition of spatial road networks.

DEFINITION 4. (**Spatial Road Networks, G_r**) *A spatial road network, G_r, is represented by a triple $(V(G_r), E(G_r), \phi(G_r))$, where $V(G_r)$ is a set of N vertices $w_1, w_2, \ldots,$ and w_N, $E(G_r)$ is a set of edges $e_{j,k}$ (i.e., roads between vertices w_j and w_k), and $\phi(G_r)$ is a mapping function: $V(G_r) \times V(G_r) \to E(G_r)$.*

In Definition 4, road network G_r is modeled by a graph, with edges as roads and vertices as intersection points of roads.

2.3 Spatial-Social Networks

In this subsection, we define spatial-social networks, as well as the spatial cohesiveness over spatial-social networks.

DEFINITION 5. (**Spatial-Social Networks, G_{rs}**) *A spatial-social network, G_{rs}, is given by a combination of spatial road networks G_r and social networks G_s, where users u_j on social networks G_s are located on some edges of spatial road networks G_r.*

From Definition 1, each social-network user, $u_j \in G_s$ (for $1 \leq j \leq M$), is associated with a 2D location on the spatial network $u_j.L$, where $u_j.L = \{u_{(loc_1, time_1)}^j, \ldots, (u_{(loc_{|u.L|}, time_{|u.L|})}^j)\}$, where $u_{loc_i}^j$ has its spatial coordinates (x_i^j, y_i^j) along $x-$ and $y-$ axes, respectively on G_r at timestamp $u_{time_i}^j$.

Modeling Spatial Cohesiveness: Next, we discuss modeling of spatial cohesiveness over spatial-social networks. In real-world social networks, users change their locations frequently due to mobility. As a result, users' spatially close communities change frequently as well [25]. Social-network users' check-in information can be recorded with the help of GPS and WiFi technologies. To measure the spatial cohesiveness, we define an average spatial distance function, $avg\_dist_{RN}(.)$. The average spatial distance function utilizes social-network users' locations $u.loc$ on the spatial network to measure the spatial cohesiveness.

DEFINITION 6. (***The Average Spatial Distance Function***). *Since each social-network user $u \in V(G_s)$ has multiple locations on spatial networks G_r, $u.loc$, at different timestamp $time$, we define the shortest path distance between any two users $u, v \in V(G_s)$ on the spatial networks as follows:*

$$avg\_dist_{RN}(u, v) \quad (5)$$
$$= \frac{\sum_{\forall u.loc_i} \sum_{\forall v.loc_j} dist_{RN}(u.loc_i, v.loc_j)}{|u.L| \cdot |v.L|},$$

where $|u.L|$ is the number of check-ins by user u, and $dist_{RN}(., .)$ is the shortest path distance between two road-network locations.

2.4 Topic-based Community Search over Spatial- Social Network (TCS-SSN)

In this subsection, we first propose a novel spatial-social structure, ss-truss, and then formally define our TCS-SSN problem.

Spatial-Social Structure, ss-truss. In this work, we consider both spatial and social networks to produce compact communities with respect to spatial cohesiveness, social influence, structural cohesiveness, and user keywords. We propose a novel spatial-social (k, d, σ, θ)-truss, or ss-truss.

DEFINITION 7. (**Spatial-Social (k, d, σ, θ)-Truss**, *ss-truss*). *Given a spatial-social network G_{rs}, a query topic set $\mathcal{T}_q$, integers k and d, a spatial distance threshold σ, and an influence score threshold θ, we define the spatial-social (k, d, σ, θ)-truss, or ss-truss, as a set, S, of users from the social network G_s such that:*

- *S is a (k, d)-truss (as given in Definition 2);*

- *the average spatial distance between any two users u and v in S is less than σ, $u, v \in S | avg\_dist_{RN}(u, v) < \sigma$, and;*

- *the influence score $infScore(.|.)$, $\{\forall u, v \in S : \exists path_{u,v} \in S, infScore(path_{u,v}|\mathcal{T}_q) \geq \theta$.*

Note that, the ss-truss satisfies the nested property that: if $k' \leq k, d' \geq d, \sigma' \leq \sigma$, and $\theta' \leq \theta$ hold, then we have: (k, d, σ, θ)-truss is a subgraph of some $(k', d', \sigma', \theta')$-truss.

Next, we define our novel query topic-based community search over spatial-social networks.

DEFINITION 8. *(**Topic-based Community Search Spatial- Social Community, TCS-SSN**). Given a spatial-social network G_{rs}, a query user q, a keyword set query K_q, and a topic query set $\mathcal{T}_q$, the topic-based community search over spatial-social networks (TCS-SSN) retrieves a maximal set, S, of social-network users such that:*

- $q \in S$;
- S is a (k, d, σ, θ)-truss, and;
- $\forall u \in S, u.key \cap K_q \neq \emptyset$.

Discussions on the Parameter Settings: Note that, parameter θ ($\in [0, 1]$) is an influence score threshold that specifies the minimum score that any two users influence each other based on certain topics in the user group S. Larger θ will lead to a user group S with higher social influence.

The topic query set, $\mathcal{T}_q$, contains a set of topics specified by the user. The influence score between any two users in the user group S is measured based on topics in $\mathcal{T}_q$. The larger the topic set query $\mathcal{T}_q$, the higher the influence score among users in the resulting community S.

The parameter σ controls the maximum (average) road-network distance between any two users in the user group S, that is, any two users in S should have road-network distance less than or equal to σ. The larger the value of σ, the farther the driving distance between any two users in the community community S.

The parameter d limits the maximum number of hops between any two users in the user group S on social networks. The larger the value of d, the larger the diameter (or size) of the community S.

The integer k controls the structural cohesiveness of the community (subgraph) S in social networks. That is, k is used in (k, d)-truss to return a community S with each connection (edge) (u, v) endorsed by $(k - 2)$ common neighbors of u and v. The larger the value of k, the higher the social cohesiveness of the resulting community S.

The keyword query set K_q, is a user-specified parameter, which contains the keywords or skills a user u must have in order to be included in the community. In real applications (e.g., Example 1) each user in the resulting community S must have at least one keyword in K_q.

To assist the query user with setting the TCS-SSN parameters, we provide the guidance or possible fillings of parameters θ, $\mathcal{T}_q$, σ, d, and k, such that the TCS-SSN query returns a non-empty answer set. Specifically, for the influence threshold θ, we can assist the query user by providing a distribution of influence scores for pairwise users, or suggesting the average (or x-quantile) influence score of those user groups selected in the query log. To suggest the topic query set $\mathcal{T}_q$, we can give the user a list of topics from the data set, and the user can choose one or multiple query topics of one's interest. Furthermore, to decide the road-network distance threshold σ, we can also show the query user a distribution of the average road-network distance between any neighbor users (or close friends) on social networks. In addition, we suggest the setting of value k, by providing a distribution of supports, $sup(e)$, on edges e (between pairwise users) of social networks, and let the user tune the social-network distance threshold d, based on the potential size of the resulting subgraph (community). Finally, we

assist the query user setting the keyword query set K_q by providing a list of frequent keywords appearing in profiles of users surrounding the query issuer q.

Challenges: The straightforward approach to tackle the TCS-SSN problem is to enumerate all possible social-network users, check query predicates on spatial-social networks (as given in Definition 8), and return TCS-SSN query answers. However, this method incurs high time complexity, since the number of possible users in a community is rather large. Although some of users with unwanted keywords can be directly discarded, still there will be a very large group of users satisfying the query keyword set. Thus, in the worst case, there is an exponential number of possible combination of users groups. For each user group, spatial-cohesiveness, structural-cohesiveness, and influence score have to be measured to obtain final TCS-SSN answers, which is not efficient. Applying such measures to many group of users may not be even feasible with nowadays social networks containing millions of nodes and edges.

Therefore, in this work, we will design effective pruning strategies to reduce the search space of the TCS-SSN problem. Then, we will devise indexing mechanisms and develop efficient TCS-SSN query answering algorithms by traversing the index.

3. PRUNING METHODS

In this section, we introduce effective pruning techniques that utilize the topic-based community search properties to reduce the search space and facilitate the online community search query processing.

3.1 Spatial Distance-Based Pruning

For any ss-truss community S, the average spatial distance between any pair of users is less than σ (as given in Definition 7). Based on that, for any two social-network user, if the average spatial distance between their check-in locations in the spatial network is greater than σ, then they cannot be in the same community. We propose our spatial distance-based pruning that prunes false alarms w.r.t. σ threshold in the ss-truss.

Intuitively, if the average spatial distance between a vertex v and a candidate vertex u is greater than σ, it means that user v resides in a place far from v. By the lemma, v can be discarded. However, the computation of the average spatial distance is costly. Next, we present our method of computing the average spatial distance between social-network users.

Computing the Average Spatial Distance: For two social-network vertices u and v, the average spatial road distance is computed by applying Eq. (5). Since each social-network user may have multiple check-in locations on the spatial network, Eq.(5) enumerates all possible shortest path combinations between the check-in locations of the two users.

From Figure 1, assume that we would like to compute the average spatial shortest path distance between u_6 and u_4. Since each user has 2 check-in locations, 4 shortest path distance computations on road networks are required. Clearly, Eq. (5) cannot be applied to large graphs due to its high time complexity. Thus, we will develop a pruning method to reduce the computation cost and tolerate real-world large graphs.

To reduce computational costs, we avoid the computation of the exact average spatial distance between two users by estimating the upper bound of the average spatial distance between them.

LEMMA 1. *For any user u in the ss-truss community S, and a user v to be in S, if the upper bound $ub\_avg\_dist_{RN}(u, v)$ of the average shortest path distance is greater than the spatial distance threshold, $ub\_avg\_dist_{RN}(u, v) > \sigma$, then user v cannot be in S and can be safely pruned.*

We will utilize the triangle inequality [3] to estimate the upper bound $ub\_avg\_dist_{RN}(.,.)$ of the average spatial distance between any two vertices. We rely on the spatial distance offline pre-computation of road network pivots $\mathcal{P}_R$ to estimate the upper bound of the average spatial distance between any two social-network users. We offline pre-compute the shortest path distance from each user's check-in locations $u.loc_i(1 \leq i \leq |u.L|)$ to all pivot locations $\mathcal{P}_{RN} = \{rpiv_1, \ldots, rpiv_l\}$.

By the triangle inequality, we have: $dist_{RN}(u.loc_i, v.loc_j) \leq dist_{RN}(u.loc_i, piv_k) + dist_{RN}(piv_k, v.loc_j)$, where $dist_{RN}(u.loc_i, piv_k)$ (or $dist_{RN}(piv_k, v.loc_j)$) is the shortest path distance on the road network between the i-th location of user u (or the j-th location of user v) and the k-th pivot, $1 \leq i \leq |u.L|$, $1 \leq j \leq |v.L|$, and $1 \leq k \leq l$. Then, at the query time, we utilize this triangle inequality property to estimate the average spatial distance upper bound, $avg\_dist_{RN}(u, v)$, of any two social-network users u and v in Eq. (5).

$$
\begin{aligned}
&avg\_dist_{RN}(u,v) \leq \\
&\min_{k=1}^{l} \left\{ \frac{\sum_{i=1}^{|u.L|} \sum_{j=1}^{|v.L|} (dist_{RN}(u.loc_i, rpiv_k) + dist_{RN}(rpiv_k, v.loc_j))}{|u.L| \cdot |v.L|} \right\} \quad (6)\\
&= \min_{k=1}^{l} \left\{ |u.L| \cdot \sum_{i=1}^{|u.L|} dist_{RN}(u.loc_i, rpiv_k) \right. \\
&\qquad\qquad\qquad\qquad \left. +|v.L| \cdot \sum_{j=1}^{|v.L|} dist_{RN}(rpiv_k, v.loc_j) \right\} \\
&= ub\_avg\_dist_{RN}(u,v).
\end{aligned}
$$

where $dist_{RN}(\cdot, \cdot)$ is the shortest path distance on the road network, l is the number of road-network pivots $\mathcal{P}_{RN}$, and $|u.L|$ (or $|v.L|$) is the number of check-in locations by user u (or v).

3.2 Influence Score Pruning

In Definition 7, for a set S of social-network users to be an ss-truss, the influence score between any pair of users should be greater than a certain threshold θ. This ss-truss property ensures that the resulting communities have high influence scores, that is, users in communities highly influence each other. In the sequel, we propose a pruning method that utilizes this property to reduce the search space by filtering out users with low influence score.

For a user v to be in a spatial-social community (ss-truss) S, based on influence score, the influence score between v and each vertex in S has to be greater than or equal to θ.

We propose an effective influence score pruning with respect to influence score upper bounds below.

LEMMA 2. (***Influence Score Pruning***). *Given a social network G_s, a spatial-social community (ss-truss) S, a topic query $\mathcal{T}_q$, and a candidate vertex v to be in S, the vertex v can be safely pruned if there exists a vertex $u \in S$ such that $ub\_infScore(u, v|\mathcal{T}_q) < \theta$.*

For each user u in the social network G_s, we utilize the influence score upper bounds to efficiently prune false alarms. Next, we describe our method of computing a tight upper bound of the influence score between any two vertices.

The Computation of the Influence Score Upper Bound: We denote the in-degree of a vertex u as $u.deg_{in}$, where $u.deg_{in}$ is a set of users $v \in V(G_s)$ such that $e_{v,u} \in E(G_s)$, and $u.deg_{out}$ is the out-degree as a set of users $v \in V(G_s)$ such that $e_{u,v} \in E(G_s)$. We denote $ub\_inf^{in}(u)$ and $ub\_inf^{out}(u)$ as the upper bound of in/out-influence of the vertex u. We compute the $ub\_inf^{in}(.)$ and

$ub\_inf^{out}(.)$ as follows:

$$ub\_inf^{in}(u|\mathcal{T}) \tag{7}$$
$$= \forall_{v \in u.deg_{in}} \forall_{t \in \mathcal{T}} \{\max\{tp_{v,u}^t\}, \ldots, \max\{tp_{v,u}^{|\mathcal{T}|}\}\}$$

$$ub\_inf^{out}(u|\mathcal{T}) \tag{8}$$
$$= \forall_{v \in u.deg_{out}} \forall_{t \in \mathcal{T}} \{\max\{tp_{u,v}^t\}, \ldots, \max\{tp_{u,v}^{|\mathcal{T}|}\}\}$$

For any two nonadjacent vertices $u, v \in V(G_s)(i.e., e_{u,v} \notin E(G_s))$, we estimate the influence score upper bound from u to v:

$$ub\_infScore(u, v|\mathcal{T}) = ub\_inf^{out}(u|\mathcal{T}) \cdot ub\_inf^{in}(v|\mathcal{T}). \tag{9}$$

Estimating the upper bound of the influence score is very critical for the influence score pruning to perform well. In Eq. (9), we utilize one hop friends to estimate the upper bound of the influence score. This method has proven to be effective and we will show in the experimental evaluation, Section 6.

3.3 Structural Cohesiveness Pruning

In fact, the ss-truss communities have high structural cohesiveness. From Definition 2, the support of an edge in ss-truss community S has to be greater than or equal $k - 2$, $sup(e) \geq k - 2$. We refer $\Phi(u), u \in V(G_s)$, as the maximum support of an edge induced by u, mathematically,

$$\Phi(u) = \max_{\forall v \in u.deg} \begin{cases} sup(u,v) & if(v \in u.deg_{out}); \\ sup(v,u) & if(v \in u.deg_{in}), \end{cases} \tag{10}$$

where $u.deg = \{u.deg_{in} \cup u.deg_{out}\}$.

LEMMA 3. (***Structural Cohesiveness Pruning***). *Given a social network G_s, a spatial-social community (ss-truss) S, and a candidate vertex v to be in S, vertex v can be directly pruned, if $\Phi(v) < k - 2$.*

Computing the edge support is a key issue to apply Lemma 3. In this regard, we rely on Wang et al. [52] to compute the maximum edge support for all edges in the graph, $sup(e), \forall e \in E(G_s)$, in $O(E(G_s)^{1.5})$.

3.4 Social Distance-Based Pruning

For a spatial-social community $S \in G_s$, Definition 7 ensures that for any two vertices $u, v \in S$, the shortest path distance connecting u and v over the social network G_s must be less than d, $dist_{SN}(u, v) < d, \forall u, v \in S$. In the social distance-based pruning, we filter out vertices with distances greater than d from the candidate set S.

LEMMA 4. (***Social Distance-Based Pruning***). *Given a social network G_s, a spatial-social community (ss-truss) S, and a candidate vertex v to be in S, the vertex v can be directly filtered out if $ub\_dist_{SN}(S, v) \geq d$, where $ub\_dist_{SN}(S, v) = \max_{\forall u \in S}\{ub\_dist_{SN}(u, v)\}$.*

The Computation of the Social Distance Upper Bound: For a two social-network users u and v, the social network distance is the minimum number of hops connecting u and v. The upper bound of the social-network distance between u and v can be computed by utilizing triangle inequality. We offline pre-compute the social-network distance from user to all social-network pivots $\mathcal{P}_{SN} = \{spiv_1, \ldots, spiv_h\}$. At query time, use triangle inequality

to estimate the social-network distance between any two social-network users u and v as follows:

$$dist_{SN}(u,v) \leq \min_{k=1}^{h}\{dist_{SN}(u,spiv_k) + dist_{SN}(v,spiv_k)\}$$
$$= ub\_dist_{SN}(u,v), \quad (11)$$

where $dist_{SN}(u,spiv_k)$ is the shortest path social-network distance between user u and the k-th social-network pivot, $(1 \leq k \leq h)$, and h is the number of the social-network pivots $\mathcal{P}_{SN}$.

3.5 Keyword-based Pruning

For a user v to join the candidate ss-truss community S, the user keyword set $v.key$ has to cover at least one keyword in the keyword query set K_q. If the candidate vertex $v.key$ shares no keyword with the query set K_q, then v can be discarded.

LEMMA 5. (*Keyword-based Pruning*). *Given a social-network graph G_s, a spatial-social network set S, a keyword query set K_q, and a user v to be in S, user v can be safely pruned, if $v.key \cap K_q = \emptyset$.*

4. INDEXING MECHANISM
4.1 Social-Spatial Index, $\mathcal{I}$, Structure

We build our social-spatial index $\mathcal{I}$ over social-network vertices $V(G_s)$. Specifically, we utilize information from both spatial and social networks to partition the social network vertices into subgraphs. The subgraphs can be treated as leaf nodes of the index $\mathcal{I}$. Then, connected subgraphs in leaf nodes are recursively grouped into non-leaf nodes, until a final root is obtained.

Leaf Nodes. Each leaf node in the social-spatial index $\mathcal{I}$ contains social-network users u. Each user u in leaf nodes is associated with a vector of the user's 2D check-in locations $u.L$, a set of keywords $u.key$, a vector of the maximum out-influence topics $u.inf^{out}$, a vector of the maximum in-influenced topics $u.inf^{in}$, and the minimum value of edge support associated with the user u, $\Phi(u)$. To save the space cost, we hash each keyword $k \in u.key$ into a position in a bit vector $u.V_{key}$.

Furthermore, we choose h social-network pivots $\mathcal{P}_{SN} = \{spiv_1, spiv_2,\ldots,$ and $spiv_h\}$ in G_s. Similarly, we choose l road-network pivots $\mathcal{P}_{RN} = \{rpiv_1, rpiv_2,\ldots,$ and $rpiv_l\}$ in G_r. Each social-network user u in leaf nodes maintains its social-network distance to the social-network pivots, that is, $dist_{SN}(u,spiv_i)(1 \leq i \leq h)$. The case of road-network pivots $avg\_dist_{RN}(u,rpiv_j)(1 \leq j \leq l)$ is similar. A cost model is proposed in Section 4.5 of our technical report [2] to guide how to choose good social-network or road-network pivots.

Non-Leaf Nodes. Each entry e of non-leaf nodes in index $\mathcal{I}$ is a *minimum bounding rectangle* (MBR) for all subgraphs under $e_{\mathcal{I}}$. In addition, e is associated with a keyword super-set $e.key$ $(= \bigcup_{\forall u \in e} u.key)$ and $lb\_\Phi(e)$ $(= \min_{\forall u \in e} \Phi(u))$. We maintain a bit vector $e.V_{key}$ for entry e which is a bit-OR of bit vectors $u.V_{key}$ for all $u \in e$. In addition, we store a lower bound of edge support that is associated with each user under the node e as follows:

$$lb\_\Phi(e) = \min_{\forall u \in e} \Phi(u). \quad (12)$$

Finally, we store an upper bound of in-influence and out-influence scores, that is,

$$ub\_inf^{out}(e|\mathcal{T}) \quad (13)$$
$$= \forall_{u \in e} \forall v \notin e, v \in u.deg_{out} \forall_{t \in \mathcal{T}}\{\max\{tp_{u,v}^t\},\ldots,\max\{tp_{u,v}^{|\mathcal{T}|}\}\},$$

$$ub\_inf^{in}(e|\mathcal{T}) \quad (14)$$
$$= \forall_{u \in e} \forall v \notin e, v \in u.deg_{in} \forall_{t \in \mathcal{T}}\{\max\{tp_{v,u}^t\},\ldots,\max\{tp_{v,u}^{|\mathcal{T}|}\}\}.$$

We also store upper/lower bounds of actual road-network shortest path distance from each user's check-in locations to all road-network pivots $\mathcal{P}_{RN}$, and to social-network distances (the number of hops) to the social-network pivots $\mathcal{P}_{SN}$, that is,

$$lb\_dist_{RN}(e,rpiv_k) = \min_{\forall u \in e}\{avg\_dist_{RN}(u,rpiv_k)\}, \quad (15)$$

$$ub\_dist_{RN}(e_{\mathcal{I}},rpiv_k) = \max_{\forall u \in e}\{avg\_dist_{RN}(u,rpiv_k)\}, \quad (16)$$

$$lb\_dist_{SN}(e,spiv_k) = \min_{\forall u \in e}\{dist_{SN}(u,spiv_k)\}, \quad (17)$$

$$ub\_dist_{SN}(e,spiv_k) = \max_{\forall u \in e}\{dist_{SN}(u,spiv_k)\}. \quad (18)$$

4.2 Index-Level Pruning

In this subsection, we discuss the pruning on the social-spatial index $\mathcal{I}$ which can be used for filtering out (a group of) false alarms on the level of index nodes.

Spatial Distance-based Pruning for Index Nodes: We utilize the road-network distance for ruling out index node $e_{\mathcal{I}_i}$ where users reside far away from locations of users in the candidate set S. Specifically, we have the following lemma.

LEMMA 6. (*Spatial Distance-based Pruning for Index Nodes*). *Given a spatial-social community S of users from social network G_s, and a node e_i from the social-spatial index $\mathcal{I}$. Node $e_i \in \mathcal{I}$ can be safely pruned, if $lb\_dist_{RN}(S,e_i) > \sigma$ holds, where $lb\_dist_{RN}(S,e_i)$ is the lower bound of the average road distance between users in the community S and the index node e_i.*

Discussion on Obtaining Lower Bounds of $dist_{RN}(S,e)$: Next, we discuss how to derive the lower bound, $lb\_dist_{RN}(S,e)$, of the average road network distance which is used in Lemma 6.

$$lb\_dist_{RN}(S,e) \quad (19)$$
$$= \max_{k=1}^{l} \begin{cases} |dist_{RN}(u_q,rpiv_k) - lb\_dist_{RN}(e,rpiv_k)|, \\ \quad if\, dist_{RN}(u_q,rpiv_k) < lb\_dist_{RN}(e,rpiv_k); \\ |dist_{RN}(u_q,rpiv_k) - ub\_dist_{RN}(e,rpiv_k)|, \\ \quad if\, dist_{RN}(u_q,rpiv_k) > ub\_dist_{RN}(e,rpiv_k); \\ 0, \quad otherwise, \end{cases}$$

where u_q is the query vertex assigned at query time, and $lb\_dist_{RN}(e,rpiv_k)$ and $ub\_dist_{RN}(e,rpiv_k)$ are given in Eqs. (15) and (16), resp.

Influence Score Pruning for Index Nodes: The TCS-SSN query aims to produce communities where users highly influence each other. For an ss-truss community S and an index node $e \in \mathcal{I}$, node e can be entirely pruned, if the influence score between the community S and e is less than threshold θ.

LEMMA 7. (**Influence Score Pruning for Index Nodes**). *Given a spatial-social community S and an index node $e \in \mathcal{I}$, e can be safely pruned, if $lb\_infScore(S,e|\mathcal{T}) < \theta$ or $lb\_infScore(e,S|\mathcal{T}) < \theta$.*

The Computatio of the Influence Score Lower Bound $lb\_infScore(S,e|\mathcal{T})$ on the Index $\mathcal{I}$: We define the lower bound of the influence score between a spatial-social community S and an index node $e \in \mathcal{I}$, with respect to the query vertex $u_q \in S$ as follows.

$$lb\_infScore(S,e|\mathcal{T}_q) = ub\_inf^{out}(u_q|\mathcal{T}) \cdot ub\_inf^{in}(e|\mathcal{T}), \quad (20)$$

$$lb\_infScore(e,S|\mathcal{T}_q) = ub\_inf^{out}(e|\mathcal{T}) \cdot ub\_inf^{in}(u_q|\mathcal{T}), \quad (21)$$

where $ub\_inf^{in}(u_q|\mathcal{T})$ and $ub\_inf^{Out}(u_q|\mathcal{T})$ are given resp. in Eqs. (8) and (9), and $ub\_inf^{in}(e|\mathcal{T})$ and $ub\_inf^{out}(e|\mathcal{T})$ are given in Eqs. (14) and (13), resp.

Social Distance-based Pruning for Index Nodes: An index node e of index $\mathcal{I}$ can be filtered out by applying the social distance-based pruning, if the number of hops between the candidate community S and users in e is greater than a threshold d.

LEMMA 8. *(**Social Distance-based Pruning for Index Nodes**). Given a community S of candidate users from social network G_s, and a node e from index $\mathcal{I}$, a node $e \in \mathcal{I}$ can be safely pruned, if $lb\_dist_{SN}(S, e) > d$ holds, where $lb\_dist_{SN}(S, e)$ is the lower bound of the number of hops between users in S and index node e.*

<u>*Discussion on Obtaining Lower Bounds of $dist_{SN}(S, e)$*</u>: Next, we discuss how to derive lower bound to derive the lower bound, $lb\_dist_{SN}(S, e)$, of the social-network distance (i.e., No. of hops) between the ss-truss S and index node e.

To estimate the lower bound $lb\_dist_{SN}(S, e)$ of the social-network distance, we utilize social-network pivots as follows:

$$lb\_dist_{SN}(S, e) \qquad (22)$$
$$= \max_{k=1}^{h} \begin{cases} |dist_{SN}(u_q, spiv_k) - lb\_dist_{SN}(e, spiv_k)|, \\ \quad if\ dist_{SN}(u_q, spiv_k) < lb\_dist_{SN}(e, spiv_k); \\ |dist_{SN}(u_q, spiv_k) - ub\_dist_{SN}(e, spiv_k)|, \\ \quad if\ dist_{SN}(u_q, spiv_k) > ub\_dist_{SN}(e, spiv_k); \\ 0, \quad otherwise, \end{cases}$$

where u_q is the query vertex assigned at query time, and $lb\_dist_{SN}(e, spiv_k)$ and $ub\_dist_{SN}(e, spiv_k)$ are offline pre-computed in Eqs. (17) and (18), respectively.

Structural Cohesiveness Pruning for Index Nodes. Similar to the structural cohesiveness pruning discussed in Section 3.3, if the lower bound of edge support associated with users under node $e \in \mathcal{I}$ is less than threshold k, then there is no edge under e satisfying structural cohesiveness, and the node e can be directly pruned.

LEMMA 9. *(**Structural Cohesiveness Pruning for Index Nodes**) Given a social-spatial index node $e \in \mathcal{I}$, if $lb\_\Phi(e) < k$, then the node e can be safely filtered out.*

In Lemma 9, $lb\_\Phi(e)$ is the lower bound of edge support of the index node e, defined in Section 4.1. Intuitively, if all the edges associated with vertices under the node e has a maximum support value that is less than k, then all vertices (users) under e cannot be in the query result. The lower bound of edge support of the node e is computed in Eq. (12).

Keyword-based Pruning for Index Nodes: Definition 8, ensures that each user in the returned community contains at least one keyword query that appears in the query set K_q. Therefore, for an index node $e \in \mathcal{I}$ can be safely pruned, if all users under e share no keywords with the keyword query set K_q.

LEMMA 10. *(**Keyword-based Pruning for Index Nodes**) Given an index node $e \in \mathcal{I}$ and a set K_q of query keywords, node e can be safely ruled out, if $e.V_{key} \cap K_q = \emptyset$.*

For an index node e, if it holds that $e.V_{key} \cap K_q = \emptyset$, then node e does not contain any keywords in K_q, and thus e can be pruned.

4.3 The Construction of a Social-Spatial Index

Algorithms 1 and 2 will be running simultaneously to generate the social-spatial index $\mathcal{I}$. The general idea of building the social-spatial index is to; First, find a number ι of index pivots (social network users); Second, partition the social network users (vertices) around those pivots.

We first start by describing Algorithm 2, where the input is a social network G_s, a spatial network G_r, and a set $\mathcal{P}_{index}$ of ι pivots (social-network vertices). The goal is to generate ι subgraphs around $\mathcal{P}_{index}$.

For each social-network vertex v, we compute the quality with each social-network pivot $piv_i \in \mathcal{P}_{index}$ (lines 1-4). The quality

---

Algorithm 1: Index Pivot Selection

Input: a road network G_r, a social network G_s, and the number ι of pivots
Output: the set, $\mathcal{P}_{index}$, of pivots

1   $global\_cost = -\infty, \mathcal{P} = \emptyset$;
2   **for** $a = 1$ *to* $global\_iter$ **do**
3   randomly select ι initial pivots and form a pivot set $\mathcal{S}_p$
4   generate subgraphs based on pivots $\mathcal{G}$ = Gen_Subgraphs$(G_r, G_s, \mathcal{S}_p)$
5   set $local\_cost = Cost\_\mathcal{P}_{index}(\mathcal{G})$
6   **for** $b = 1$ *to* $swap\_iter$ **do**
7   select a random pivot $piv \in \mathcal{S}_p$
8   randomly choose a non-pivot new_piv
9       $\mathcal{S}'_p = \mathcal{S}_p - \{piv\} + \{new\_piv\}$
10      $\mathcal{G}'$ = Gen_Subgraphs$(G_r, G_s, \mathcal{S}'_p)$
11   evaluate the new cost $Cost\_\mathcal{P}_{index}^{new}(\mathcal{G}')$ w.r.t. $\mathcal{S}'_p$
12   **if** *the new cost $Cost\_\mathcal{P}_{index}^{new}(\mathcal{G}')$ is better than $local\_cost$* **then**
13        $local\_cost = Cost\_\mathcal{P}_{index}^{new}(\mathcal{G}')$
14        $\mathcal{S}_p = \mathcal{S}'_p$
15   **if** $local\_cost$ *is better than* $global\_cost$ **then**
16       $\mathcal{P}_{index} = \mathcal{S}_p$
17       $global\_cost = local\_cost$
18 **return** $\mathcal{P}_{index}$

---

function $quality(v, piv_i)$ computes the number of hops and road-network distance between v and piv_i (line 4). Then, assign the vertex v to the pivot where the quality is the best (lines 5-8). Finally, the set of partitions in returned (line 9).

Algorithm 1, illustrates the details of the pivot selection. At the beginning, two parameters $global_{cost}$ and $\mathcal{P}$ will be set to store the globally optimal cost value and the corresponding pivot set, resp. (line 1). We randomly select a pivot set $\mathcal{S}_p$ from social-network users (vertices) (line 3). Next, we partition the social network around $\mathcal{S}_p$ by Algorithm 2 and partitions $\mathcal{G}$ (line 4). Then, we evaluate the cost function $Cost\_\mathcal{P}_{index}(\mathcal{G})$ of the resulting partitions (please refer to the cost model for selecting good index pivots in Section 4.5.3 of our technical report [2]). After that, each time we swap a $piv \in \mathcal{S}_p$ with a non-pivot new_piv, which results in a new pivot set $\mathcal{S}'_p$ (lines 7-9), and generate new graph partitions $\mathcal{G}'$ by Algorithm 2 and evaluate it (lines 10-11). If the new cost is better than the best-so-far cost $local\_cost$, then we can accept the new pivot set with its cost (lines 12-14). We repeat the process of swapping a pivot with a non-pivot for $swap\_iter$ times (line 6). To avoid the local optimal solution, we consider selecting different initial pivot sets for $globa\_liter$ times (lines 2-3), and record the globally optimal pivot set and its cost (lines 15-17). Finally, we return the best pivot set $\mathcal{P}_{index}$.

Finally, we pass the optimal pivot set $\mathcal{P}_{index}$ to Algorithm 2 to generate subgraphs, which are treated as leafs of the social-spatial index. Then, the connected subgraphs in leaf nodes are recursively grouped into non-leaf nodes, until a final root is obtained.

Remarks. We also propose evaluation measures of grouping social-network users, based on their spatial closeness, structural cohesiveness, and social influence, and provide cost models for selecting good road-network pivots $\mathcal{P}_{RN}$, social-network pivots $\mathcal{P}_{SN}$, and index pivots $\mathcal{P}_{index}$. We omit them here due to space limitations. For interested readers, please refer to Sections 4.4 and 4.5 in our technical report [2].

5. TCS-SSN COMMUNITY SEARCH

Algorithm 3 illustrates the pseudo code of TCS-SSN answering, which process TCS-SSN queries over the spatial-social network G_{rs} via the social-spatial index $\mathcal{I}$. Specifically, we traverse index $\mathcal{I}$, and apply index level pruning over the index node and objects level pruning over the social network object, and refine a candidate set to return the actual TCS-SSN query answer.

Algorithm 2: Gen_Subgraphs

Input: a spatial network G_r, a social network G_s, and a set $\mathcal{P}_{index} = piv_1, \ldots, piv_h$ of pivots
Output: a set $\mathcal{G} = g_1, \ldots, g_\iota$ of subgraphs

1 **for** $v \in V(G_s)$ **do**
2     $best\_quality = \infty$
3   **for** $i = 1$ *to* ι **do**
4         $quality(v, piv_i) =$
        $\dfrac{avg\_dist_{RN}(v, piv_i)}{max\_avg\_dist_{RN}} + \dfrac{dist_{SN}(v, piv_i)}{max\_Dist_{SN}}$
5   **if** $quality(v, piv_i) < best\_quality$ **then**
6             $j = i$
7             $best\_quality = quality(v, piv_i)$
8   assign v to g_j
9 **return** $\mathcal{G}$

Pre-Processing. Initially, we set S_{cand} to an empty set, initialize an empty minimum heap $\mathcal{H}$, and add the root, $root(\mathcal{I})$, of index $\mathcal{I}$ to $\mathcal{H}$ (lines 1-3).

Index Traversal. In Algorithm 3, after we insert the heap entry $(root(\mathcal{I}), 0)$ into the heap $\mathcal{H}$, we traverse the social-spatial index $\mathcal{I}$ from root to leaf nodes (lines 4-15). In particular, we will use heap $\mathcal{H}$ to enable the tree traversal. Each time we pop out an entry (e_i, key) with the minimum key from heap $\mathcal{H}$, where e_i is an index node $e_i \in \mathcal{I}$, and key is a lower bound of road-network distance, $key = lb\_dist_{RN}(e, e_i)$. If key is greater than spatial distance threshold σ, all entries in $\mathcal{H}$ must have their lower bounds of maximum road-network distances greater than threshold σ. Then, we can safely prune all entries in the heap and terminate the loop.

When entry e_i is a leaf node, we consider each object (social-network user) $u \in e_i$, and apply object-level pruning *spatial distance-based pruning, influence score pruning, structural cohesiveness pruning, social distance-based pruning,* and *keyword-based pruning* to reduce the search space (line 9). If a user u cannot be pruned, we will add it to the candidate set $\mathcal{S}_{cand}$ (line 10).

When entry e_i is a non-leaf node, for each child $e_x \in e_i$, we will apply index-level pruning (e.g., spatial distance-based pruning influence score pruning, social distance-based pruning, structural cohesiveness pruning, and keyword-based pruning for index nodes) (line 14). If a node e_x cannot be pruned in line 14, then we insert heap entry $(e_i, lb\_dist_{RN}(q, e_x))$ into heap $\mathcal{H}$ for further investigation (line 15).

Refinement. After the index traversal, we refine the candidate set $\mathcal{S}_{cand}$ to obtain/return actual TCS-SSN answers $\mathcal{S}$ (line 17).

Complexity Analysis. Next, we discuss the time complexity of our TCS-SSN query answering algorithm in Algorithm 3. The time cost of Algorithm 3 processing consists of two portions: index traversal (lines 4-15) and refinement (lines 16-18).

Let $PP^{(j)}$ be the pruning power on the j-th level of index $\mathcal{I}$, where $1 \le j \le height(\mathcal{I})$. Denote f as the average fanout of non-leaf nodes in the social-spatial index $\mathcal{I}$. Then, the filtering cost of lines 4-15 is given by $O\left(\sum_{j=1}^{height(I)} f^j \cdot (1 - PP^{(j-1)})\right)$, where $PP^{(0)} = 0$.

Moreover, let S_{cand} be a subgraph containing users left after applying our pruning methods. The main refinement cost in lines 16-18 is on the graph traversal and constraint checking (e.g., average spatial distance, social distance, and social influence). In particular, the average spatial distance on road networks can be computed by running the Dijkstra algorithm starting from every vertex in S_{cand}, which takes $O(|V_{RN}(S_{cand})| \cdot (|E_{RN}(S_{cand})| \cdot log(|V_{RN}(S_{cand})|)))$ cost; the social distance computation takes $O(|V_{SN}(S_{cand})| \cdot |E_{SN}(S_{cand})|)$ by BFS traversal from each user in S_{cand}; the k-truss computation takes $O(p \cdot |E_{SN}(S_{cand})|)$, where $p < min(d_{max}, \sqrt{|E_{SN}(S_{cand})|})$ [32];

Algorithm 3: TCS-SSN Query Answering Algorithm

Input: a spatial-social network G_{rs}, social-spatial index $\mathcal{I}$, a query issuer q, a topic query set $\mathcal{T}_q$, a keyword query set K_q, a truss value k, social distance threshold d, spatial distance threshold σ, influence threshold θ
Output: a community S, satisfying TCS-SSN query predicates in Definition 8

1 set $S\_cand = \emptyset$
2 initialize a min-heap $\mathcal{H}$ accepting entries in the form (e, key)
3 insert entry $(root(\mathcal{I}), 0)$ into heap $\mathcal{H}$
4 **while** $\mathcal{H}$ *is not empty* **do**
5   (e_i, key) = de-heap $\mathcal{H}$
6   **if** $key > \sigma$, **then** terminate the loop;
7   **if** e_i *is a leaf node* **then**
8   **for** *each user* $u \in e_i$ **do**
9   **if** u *cannot be pruned by Lemma 1, 2, 3, 4, or 5 w.r.t.* q **then**
10   add u to S_{cand}
11   **else**
        // e_i is a non-leaf node
12   obtain the entry $e_q \in \mathcal{I}$ that contains q
13   **for** *each entry* $e_x \in e_i$ **do**
14   **if** e_x *cannot be pruned by Lemma 6, 7, 8, 9, or 10 w.r.t.* e_q **then**
15   insert $(e_x, lb\_dist_{RN}(q, e_x))$ into the heap $\mathcal{H}$

16 **while** *no users or edges are pruned* **do**
17   start BFS search from q and apply social-network distance pruning and influence score pruning on social network vertices
18   apply truss decomposition directly on remaining edges to prune edges with truss value less than or equal to $k - 2$
19 **return** S

Table 2: Statistics of real data sets $Gow\&Cali$ and $Twi\&SF$.

social network	$|V(G_s)|$	$|E(G_s)|$	road network	$|V(G_r)|$	$|E(G_r)|$
Gowalla (Gow)	196K	1.9M	California ($Cali$)	21K	44K
Twitter (Twi)	349K	2.1M	San Francisco(SF)	175K	446K

Table 3: Experimental settings.

Parameter	Values
the size of keyword set query K_q	2, 3, **5**, 8, 10
the size of topic set query $\mathcal{T}_q$	1, **2**, 3
the spatial distance threshold σ	0.5, 1, **2**, 3, 5
the influence score threshold θ	0.1, 0.3, **0.5**, 0.7, 0.9
the number of triangles k	2, 3, **5**, 7, 10
the social distance threshold d	1, 2, **3**, 5, 10
the number of vertices in road network G_r and social network G_s	10K, 20K, **30K**, 40K, 50K, 100K , 200K

the mutual influence score computation takes $O(|V_{SN}(S_{cand})| \cdot |E_{SN}(S_{cand})|)$ by BFS traversal from each user in S_{cand}. Thus, the overall time complexity of the refinement is given by $O(|V_{RN}(S_{cand})| \cdot (|E_{RN}(S_{cand})| \cdot log(|V_{RN}(S_{cand})|)) + p \cdot |E_{SN}(S_{cand})| + 2 \cdot (|V_{SN}(S_{cand})| \cdot |E_{SN}(S_{cand})|))$.

Discussions on Handling Multiple Query Users. The TCS-SSN problem considers the standalone community search issued by one query user q. In the case where multiple users issue the TCS-SSN queries at the same time, we perform batch processing of multiple TCS-SSN queries, by traversing the social-spatial index only once (applying our pruning methods) and retrieving candidate users for each query user. In particular, an index node can be safely pruned, if for each query there exists at least one pruning rule that can prune this node. After the index traversal, we refine the resulting candidate users for each query and return the TCS-SSN answer sets to query issuers.

6. EXPERIMENTAL EVALUATION
6.1 Experimental Settings

We test the performance of our TCS-SSN query processing approach (i.e., Algorithm 3) on both real and synthetic data sets.

(a) CPU time   (b) I/O cost
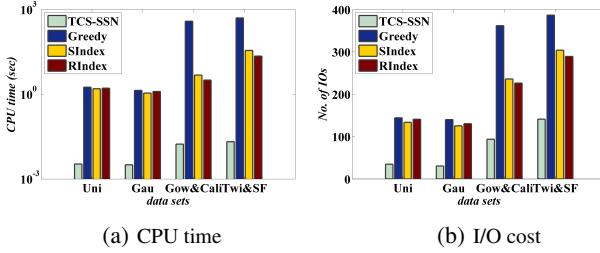
Figure 2: The TCS-SSN performance vs. real/synthetic data sets.
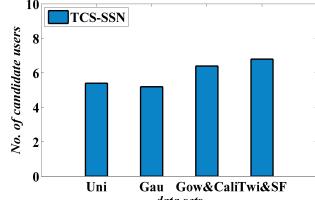
Figure 3: The number of the remaining candidate users after the pruning vs. real/synthetic data sets.

(a) index construction time   (b) index space cost

Figure 4: The index construction time and space cost vs. data sets.

(a) CPU time   (b) I/O cost

Figure 5: The performance vs. road-network distance threshold σ.

Real Data Sets. We evaluate the performance of our proposed TCS-SSN algorithm (as given in Algorithm 3) with two real data sets, denoted as $Gow\&Cali$ and $Twi\&SF$, for spatial-social networks. The first data set, $Gow\&Cali$, is a spatial-social network, which combines Gowalla social network [39] with California road networks [38]. The second spatial-social network $Twi\&SF$ integrates the Twitter [39] with San Francisco road networks [38]. Table 2 depicts statistics of spatial/social networks.

Each user u in social networks (i.e., Gowalla or Twitter) is associated with multiple check-in locations (i.e., places visited by the user u). The user u also has a keyword vector $u.key$, which contains keywords collected from one's social-media profile. The directed edge $e_{u,v}$ between users u and v has a weight that reflects the influence of user u on user v based on a certain topic. We map each user u from social networks (Gowalla or Twitter) to 2D locations on road networks (i.e., California or San Francisco, resp.).

Synthetic Data Sets. We also generate two synthetic spatial-social data sets as follows. Specifically, for the spatial network G_r, we first produce random vertices in the 2D data space following either Uniform or Gaussian distribution. Then, we randomly connect vertices nearby through edges, such that all vertices are reachable in one single connected graph and the average degree of vertices is within $[3, 4]$. This way, we can obtain two types of graphs with Uniform and Gaussian distributions of vertices.

To generate a social network G_s, we randomly connect each user u with other users, such that the degrees of users follow Uniform or Gaussian distribution within a range $[1, 10]$. Each user u has a set, $u.key$, of interested keywords, where keywords are represented by integers within $[1, 10]$ following Uniform or Gaussian distribution. Furthermore, each social-network edge $e_{u,v}$ is associated with a set of topics (we consider 3 topics by default), and each topic has a probability (within $[0, 1]$ following Uniform or Gaussian distribution) that user u can influence user v, similar to [15].

Finally, we combine social network G_s with road network G_r, by randomly mapping social-network users to 2D spatial locations on road networks, and obtain a spatial-social network G_{rs}. With Uniform or Gaussian distributions during the data generation above, we can obtain two types of synthetic spatial-social networks G_{rs}, denoted as Uni and Gau, respectively.

Measures. To evaluate the performance of our TCS-SSN approach, we report the CPU time and the I/O cost. The CPU time is the time cost of retrieving TCS-SSN answer candidates by traversing the index (as illustrated in Algorithm 3), whereas the I/O cost is the
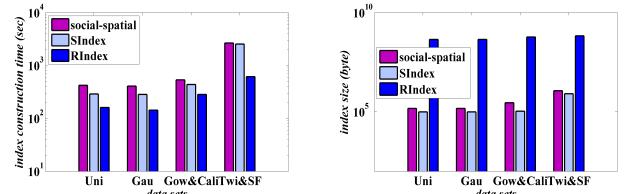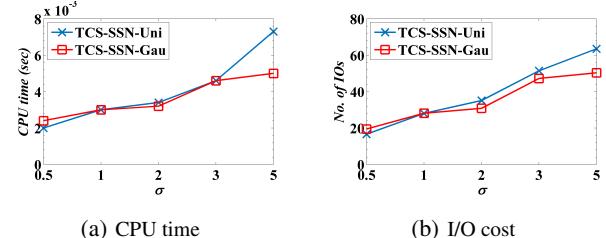
number of page accesses during TCS-SSN query answering.

Competitors: To the best of our knowledge, prior works did not study the problem of community search (CS) over spatial-social networks by considering (k, d)-truss communities with user-specified topic keywords, high influences among users, and small road-network distances among users. Thus, we develop three baseline algorithms, $Greedy$, $SIndex$, and $RIndex$.

$Greedy$ first runs the BFS algorithm to retrieve all users with social distance less than d from the query vertex q in social networks. Meanwhile, it prunes those users without any query keywords in K_q. Then, it runs another BFS algorithm over road networks to filter out all users with average spatial distance to q greater than σ. After that, we iteratively apply the pruning on social-network edges for k-truss and under other constraints (e.g., influence score, social distance, and spatial distance), and refine/return the resulting connected subgraph.

The $SIndex$ baseline offline constructs a tree index over social-network users and their corresponding social information (e.g., truss values and social-distance information via pivots). In particular, it first partitions users on social networks into subgraphs, which can be treated as leaf nodes, and then recursively groups connected subgraphs in leaf nodes into non-leaf nodes until a final root is obtained. For online TCS-SSN query, $SIndex$ traverses this social-network index by applying the pruning w.r.t. the social-network distance d and the truss value k, and refine the resulting subgraphs, similar to the refinement step in Algorithm 3.

$RIndex$ offline constructs an R^*-tree over users' spatial and textual information on road networks. Specifically, we first divide social-network users into partitions based on (1) spatial closeness and (2) keyword information. Then, we treat each partition as a leaf node of the R^*-tree, whose spatial locations are enclosed by a minimum bounding rectangles (MBRs). This way, we can build an R^*-tree with aggregated keyword information in non-leaf nodes. $RIndex$ traverses the R^*-tree and applies pruning based on spatial distance (via pivots) and textual keywords. Finally, the retrieved users (with spatial closeness and keywords) will be refined, as mentioned in the refinement step of Algorithm 3.

Experimental Setup: Table 3 depicts parameter settings in our experiments, where bold numbers are default parameter values. In each set of experiments, we vary one parameter while setting other parameters to default values. We ran our experiments on a PC with Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz (4 CPUs), 2.8GHz and 32 GB memory. All algorithms were implemented by C++.
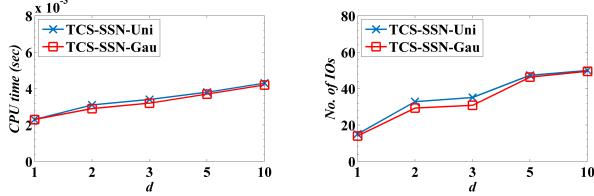
(a) CPU time
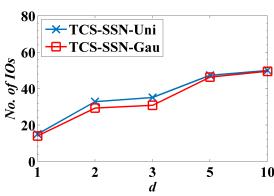
(b) I/O cost

Figure 6: The performance vs. social-network distance threshold d.

(a) CPU time

(b) I/O cost

Figure 8: The performance vs. influence score threshold θ.
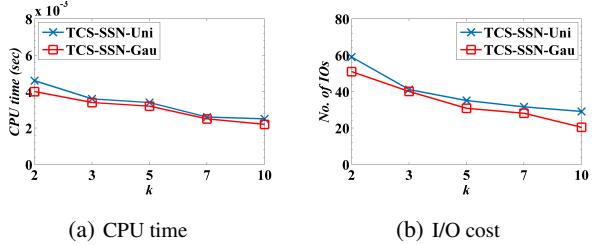
(a) CPU time

(b) I/O cost

Figure 7: The performance vs. the triangle number threshold k.

(a) CPU time

(b) I/O cost

Figure 9: The performance vs. the size, $|K_q|$, of the keyword query set.

6.2 TCS-SSN Performance Evaluation

The TSC-SSN Performance vs. Real/Synthetic Data Sets.
Figure 2 compares the performance of our TCS-SSN query processing algorithm with three baseline algorithms $Greedy$, $SIndex$, and $RIndex$ over synthetic and real data sets, Uni, Gau, $Gow\&Cali$, and $Twi\&SF$, in terms of the CPU time and I/O cost, where we set all the parameters to their default values in Table 3. From the experimental results, we can see that our TCS-SSN approach outperforms baselines $Greedy$, $SIndex$, and $RIndex$. This is because TCS-SSN applies effective pruning methods with the help of the social-spatial index. In particular, for all the real/synthetic data, the CPU time of our proposed TCS-SSN algorithm is $0.0035 \sim 0.028\ sec$, and the number of I/Os is around $35 \sim 162$, which are much smaller than any of the three baseline algorithms $Greedy$, $SIndex$, and $RIndex$. Therefore, this confirms the effectiveness of our proposed pruning strategies and the efficiency of our TCS-SSN query answering algorithm on both real and synthetic data.

Figure 3 evaluates the number of the remaining candidate users after the index traversal (applying the pruning methods) over synthetic/real data, where all the parameters are set to their default values. From the figure, we can see that the number of candidate users varies from 5 to 8. This indicates that we can efficiently refine candidate communities with a small number of users.

In Figure 4, we evaluate the index construction time and space cost of our proposed social-spatial index and the two index-based baselines $SIndex$ and $RIndex$ over Uni, Gau, $Gow\&Cali$, and $Twi\&SF$ data sets. Figure 4(a) demonstrates the index construction time for our proposed social-spatial index and the baselines $SIndex$ and $RIndex$. For $Twi\&SF$ data set (with over than $2.1M$ edges), the index construction time of our social-spatial index takes around 45 minutes. The majority of this time cost goes to the computation of the maximum edge support for all edges in the graph, $sup(e)$, which takes $O(E(G_s)^{1.5})$ by applying Wang et al. [52]. Note that, the social-spatial index (as well as $SIndex$ and $RIndex$ indexes) is offline constructed only once. Furthermore, Figure 4(b) shows the index space cost of our proposed social-spatial index and the two baselines $SIndex$ and $RIndex$. From the experimental results, our social-spatial index is much more space efficient than $RIndex$ that uses R^*-tree, and is comparable to $SIndex$.

To show the robustness of our TCS-SSN approach, in subsequent experiments, we will vary different parameters (e.g., σ, d, k, θ, and
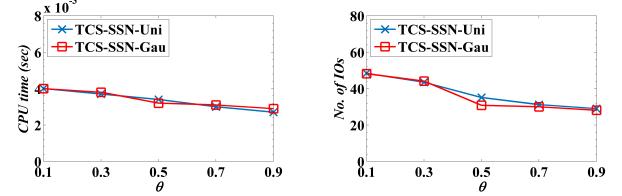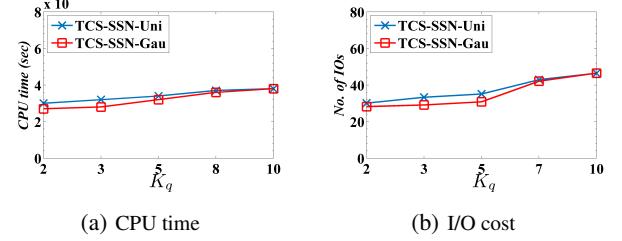
so on, as depicted in Table 3) on synthetic data sets, Uni and Gau.
Effect of the Road-Network Distance Threshold σ. Figure 5 shows the TCS-SSN performance, by varying the road-network distance threshold σ from 0.5 mile to 5 miles, where other parameters are by default. When σ increases, more social-network users will be considered, and thus the CPU time and I/O cost will increase. Nevertheless, for different σ values, both CPU time and I/O cost remain low ($0.002 \sim 0.0075\ sec$ and $17 \sim 64$ I/Os, resp.).
Effect of the Social-Network Distance Threshold d. Figure 6 varies the social-distance threshold d (i.e., the threshold for the number of hops) from 1 to 10, and reports the CPU time and I/O cost of our TCS-SSN approach over Uni and Gau data sets, where other parameters are set to their default values. With the increase of the social-distance threshold d, more candidate communities (with more social-network users) will be retrieved for evaluation. Therefore, the CPU time and I/O cost become higher for larger threshold d. Nonetheless, for different d values, the CPU time remains small (i.e., around $0.0023 \sim 0.005\ sec$), and the I/O cost is low (with $16 \sim 59$ page accesses).
Effect of the Triangle Number Threshold k. Figure 7 examines the TCS-SSN performance with different thresholds k for the number of triangles, in terms of the CPU time and I/O cost, where $k = 2, 3, 5, 7$, and 10, and other parameters are set to their default values. In figures, when k becomes large, many users with low degrees (i.e., $< k$) will be safely pruned, and thus the CPU time and I/O cost are expected to reduce substantially (as confirmed by figures). Nevertheless, the CPU time and the I/O cost remain low (i.e., about $0.002 \sim 0.0065\ sec$ and $20 \sim 61$ I/Os, respectively), which indicates the efficiency of our proposed TCS-SSN approach for different k values.
Effect of the Influence Score Threshold θ. Figure 8 illustrates the CPU time and the I/O cost of our TCS-SSN approach by varying the interest score threshold θ from 0.1 to 0.9, where all other parameter values are set by default. From the experimental results, we can see that both CPU time and I/O cost smoothly decrease with large θ values. This is because larger θ can filter out more edges with low influence scores, which leads to less user candidates for the filtering and refinement. Nonetheless, the time and I/O cost of our TCS-SSN approach remain low (i.e., $0.0025 \sim 0.0041\ sec$ for the CPU time and $25 \sim 50$ page accesses).
Effect of the Size, $|K_q|$, of the Keyword Query Set. Figure 9 demonstrates the performance of our TCS-SSN approach with different numbers of query keywords in K_q, where $|K_q| =$

(a) CPU time

(b) I/O cost
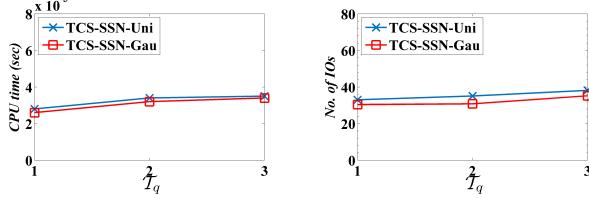
Figure 10: The performance vs. the size, $|\mathcal{T}_q|$, of the topic query set.


(a) CPU time

(b) I/O cost
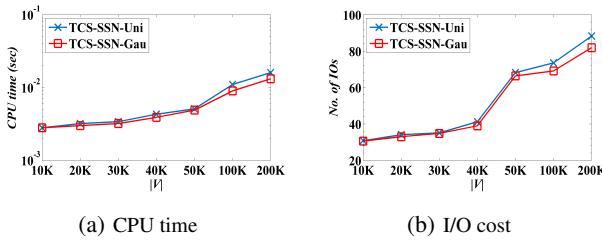
Figure 11: The performance vs. the number, $|V(G_r)|$ (or $|V(G_s)|$), of vertices in spatial or social networks.

$2, 3, 5, 8$, and 10, and default values are used for other parameters. Intuitively, when $|K_q|$ becomes larger (i.e., more query keywords), we need to consider more potential users, which incurs higher CPU time and I/O cost. Despite that, the CPU time and I/O cost of our TCS-SSN approach remain low (i.e., $0.0025 \sim 0.004\ sec$ for the CPU time and $25 \sim 49$ page accesses).

Effect of the Size, $|\mathcal{T}_q|$, of the Topic Query Set. Figure 10 illustrates the performance of our TCS-SSN approach by varying the number of query topics (in $\mathcal{T}_q$) on edges, where $|\mathcal{T}_q| = 1, 2$, and 3, and other parameters are set to their default values. The experimental results show that the TCS-SSN performance is not very sensitive to $|\mathcal{T}_q|$. The CPU time remains low (i.e., $0.003 \sim 0.0035\ sec$) and the I/O cost is around $30 \sim 38$, which indicate the efficiency of our TCS-SSN approach with different $|\mathcal{T}_q|$ values.

Effect of the Number, $|V(G_r)|$ (or $V(G_s)$), of Vertices in Road (Social) Networks. Figure 11 shows the scalability of our TCS-SSN approach with different sizes of spatial/road networks, $|V(G_r)|$ (or $|V(G_s)|$), of spatial/road networks (denoted as $|V|$), where $|V|$ varies from $10K$ to $200K$, and other parameters are set to their default values. In figures, when the number of road-network (or social-network) vertices increases, both CPU time and I/O cost smoothly increase. Nevertheless, the CPU time and I/O costs of our TCS-SSN approach remain low (i.e., $0.0028 \sim 0.017\ sec$ for the time cost and $30 \sim 89$ page accesses, respectively), which confirms the scalability of our TCS-SSN approach against large network sizes.

6.3 A Case Study

Finally, we conduct a case study of our TCS-SSN problem on real-world spatial-social networks, $Twi\&SF$ (i.e., Twitter [39] with San Francisco road networks [38]). As illustrated in Figure 12(a), each social-network user is associated with keywords (i.e., Twitter hashtags) such as $K1 \sim K10$ from user accounts, and has checkin locations on road networks, where the user IDs of vertices and descriptions of keywords are depicted in Figure 12(b).

Assume that we have a TCS-SSN query over $Twi\&SF$, where $ID2$ is a query vertex, truss value $k = 5$, social-network distance threshold $d = 3$, spatial-distance threshold $\sigma = 2\ miles$, influence score threshold $\theta = 0.5$, query topic set $\mathcal{T}_q = (\texttt{sport}, \texttt{health})$, and query keyword set $K_q = \{vegan, vegetarian, eatHealthy, workout, nutritions\}$. Figure 12(a) shows the resulting

a TCS-SSN community of users

vertex ID	user ID
D1	321757
D2	322443
D3	103066
D4	130448
D5	166140
D6	206327
D7	136250
D8	383003

vertex IDs and their associated user IDs

keyword	description
K1	#vegan
K2	#cars
K3	#nytimes
K4	#eatHealthy
K5	#workout
K6	#news
K7	#nutritions
K8	#vegetarian
K9	#apple
K10	#lakers

user keywords and their descriptions

SF road network with tagged users' locations

vertex ID	ID1	ID2	ID3	ID4	ID5	ID6	ID7	ID8
ID1		0.788	0.63	1.55	0.61	0.84	0.63	0.62
ID2	1.15		0.56	0.65	0.88	0.78	0.83	0.57
ID3	0.75	0.76		0.87	0.60	0.89	0.73	0.86
ID4	0.88	0.56	0.66		0.75	0.62	0.56	0.78
ID5	0.57	0.53	0.70	0.99		0.77	0.64	0.97
ID6	0.93	0.75	0.66	0.80	0.63		0.64	0.68
ID7	0.84	0.85	0.72	0.66	0.56	0.94		0.71
ID8	0.65	0.62	0.82	0.57	0.77	0.51	1.02	

*social influence values between users based on topics* `sport` *and* `health`

(a)

(b)

Figure 12: A TCS-SSN case study on spatial-social networks $Twi\&SF$.

TCS-SSN community, which contains a subgraph of 8 users, $ID1 \sim ID8$. Each user in this community is associated with at least one query keyword in K_q, and they show strong structural connectivity (satisfying the $(5, 3)$-truss constraints) and spatial closeness on road networks ($\leq 2\ miles$). Moreover, Figure 12(b) depicts an influence matrix (w.r.t. $\mathcal{T}_q$) for pairwise users in this community, each element of which is above influence threshold θ (i.e., 0.5). This confirms their high influences to each other within our retrieved TCS-SSN community.

7. RELATED WORK

Community Detection. There are many important research problems on graph data management [36, 50] in real applications. The community detection problem aims to discover all communities in a large-scale graph such as social networks or bibliographic networks. Some prior works [27, 47] retrieved communities in large graphs, by considering link information only. More recent work was carried by [54, 45, 46, 60], that devoted for attribute graphs, by using clustering techniques. In [60], for example, the links and keyword vertices are considered to compute the pairwise vertex similarity in order to cluster the large graph. Zang et al. [17] proposed a framework that applies a game-theoretic approach to identify dense communities in large-scale complex networks. Recently, other works carried by [28, 22, 29, 16] focused on detecting communities in spatially constrained graphs, whose vertices are associated with spatial coordinates.

A geo-community is defined as a community, in which vertices are densely connected and loosely connected with other vertices. Techniques such as *average linkage measure* [29] and *modularity maximization* [22, 16] are applied to find geo-communities. However, Lancichinetti et al. [37] argued that modularity-based methods often fail to resolve small-size communities.

Community Search. Community search problem (CS) aims to obtain communities in an "online" manner, based on a query request. Several existing works [51, 20, 19, 42, 33] have proposed efficient algorithms to obtain a community starting from and including a query vertex q. In [51, 20], the *minimum degree* is used to measure the structure cohesiveness of community. Sozio et al. [51] proposed the first algorithm $Global$ to obtain k-core community containing a query vertex q. Cui et al. [20] used local expansion techniques to boost the query performance. Furthermore, Li et al. [41] proposed the most influential community search over large social networks to disclose the $\mathcal{C}^{kr}$ community with the highest outer influences, where the $\mathcal{C}^{kr}$ community contains at least k nodes, and any two nodes in $\mathcal{C}^{kr}$ can be reached at most r hops. Bi et al. [8] proposed an optimal approach to retrieve top-k influential communities (subgraphs),

such that each subgraph g is a maximal connected subgraph with minimum degree of at least γ, and has the highest influence value. Akbas et al. [1] introduced a truss-based indexing approach, where they can in optimal time detect the k-truss communities in large network graphs. Fang et al. [26] studied the community search problem over large heterogeneous information networks, that is, given a query vertex q, find a community from a heterogeneous network containing q, such that all the vertices are with the same type of q and have close relationships, where the relationship between two vertices of the same type is modeled by a $meta-path$, and the cohesiveness of the community is measured by the classic minimum degree metric with the $meta-path$. Note that, the aforementioned previous works [41, 8, 1, 26] did not consider spatial cohesiveness, topic-related social influences, nor keywords, which different from our proposed TCS-SSN problem.

Some other works [24, 42] used minimum degree metric to search communities for attribute graphs. Other well-known structure cohesiveness k-clique [19], k-truss [33] have also been considered for online community search. However, these works are designed for non-spatial graphs. Huang et al. [32] proposed (k, d)-truss for geo-spatial networks, but they did not consider user topic keywords, social influence, neither road-network distance.

Geo-Social Networks. Query processing on location-based social networks has become increasingly important in many real applications. Yang et al. [55] studied the problem of *socio-spatial group query* (SSGQ), which retrieves a group of connected users (friends) with the smallest summed distance to a given query point q. Li et al. [43] proposed another query type that retrieves a group of k users who are interested in some given query keywords and are spatially close to each other. Yuan et al. [57] studied the kNN query which obtains k POIs that are not only closest to query point q, but also recommended by one's friends on social networks under the IC model. Fang et al. [23] introduced the spatial-aware community (or SAC), which retrieves a subgraph (containing a given query vertex q) from geo-social networks that has high structural cohesiveness and spatial cohesiveness. Al-Baghdadi et al. [4] proposed a *group planning query over spatial-social networks* (GP-SSN) that retrieves a set S of users from social networks and a set R of potential POIs from road networks to be visited by the users in S. Chen et al. [14] discussed the co-located community search, that is a subgraph satisfying connectivity, structural cohesiveness, and spatial cohesiveness. Chen et al. [14] considered communities that match query predicates and have the maximum cardinality globally, whereas Fang et al. [23] focused on finding a locally optimal community containing a query vertex. Previous works on geo-social community search neglected the social influence among users and road-network distance. In our proposed TCS-SSN problem, we introduce a new and different definition of communities that not only are spatially and socially close, but also have high social influence and small driving distance among community members.

Keyword Search and Spatial Keyword Queries. The keyword search problem has been extensively studied in both domains of relational databases and graphs. Given a set of query keywords, the keyword search in relational databases [12, 35, 31] usually finds a minimal connected tuple tree that contains all the query keywords. In graph databases [40, 48], the keyword search problem retrieves a subgraph containing the given query keywords. Furthermore, in spatial databases containing both spatial and textual information, another interesting problem is the spatial keyword query, which returns relevant POIs that both satisfy the spatial query predicates and match the given query keywords. Coa et al. [9] categorized the spatial keyword queries based on their ways of specifying spatial and textual predicates, including Boolean Range Queries [30], Boolean kNN Queries [11, 21], and Top-k kNN Queries [10, 18]. Recently, Zhang et al. [59] proposed the keyword-centric community search (KCCS) over an attributed graph, which finds a community (subgraph) with the degree of each node at least k, and the distance between nodes and all the query keywords being minimized. However, Zhang et al. [59] did not consider the spatial cohesiveness neither the social influence. Moreover, Islam et al. [34] proposed the keyword-aware influential community query (KICQ) over an attributed graph, which returns r most influential communities in the attributed graph, such that the returned community has high influence (containing highly influential members) based on certain keywords. An application of KICQ is to find the most influential community of users who are working in "ML" or "DB". Different from KICQ, our proposed TCS-SSN finds a community that not only has a high social cohesiveness and covers certain keywords, but also has a high spatial cohesiveness. Furthermore, our TCS-SSN problem returns the community with high influence score among community members (with respect to specific topics), rather than members with high influences to others in KICQ Chen et al. [13] introduced a parameter-free contextual community model for attributed community search. Given an attributed graph and a set of query keywords describing the desired matching community context, their proposed query returns a community that has both structure and attribute cohesiveness w.r.t. the provided query context. In contrast, different from [13], our TCS-SSN problem returns the community over spatial-social networks (instead of social networks only), which has high social cohesiveness, spatial cohesiveness, social influence, and covers a set of keywords (rather than measuring the context closeness of the community with the query context). Thus, with different data models (relational or graph data) and query types, we cannot directly borrow previous techniques for (spatial) keyword search or community search on attributed graphs to solve our TCS-SSN problem.

To our best knowledge, the TCS-SSN problem has not been studied by prior works on spatial-social networks, which considers (k, d)-truss communities with user-specified topic keywords, high influences among users, and small road-network distances among users. Due to different data models and query types, previous techniques on location-based social networks cannot be directly used for tackling our TCS-SSN problem.

8. CONCLUSIONS

In this paper, we formalize and tackle an important problem, *topic-based community search over spatial-social networks* (TCS-SSN), which retrieves communities of users (including a given query user) that are spatially and socially close to each other. To efficiently tackle this problem, we design effective pruning methods to reduce the search space, propose a novel social-spatial index over spatial-social networks, and develop efficient algorithms to process TCS-SSN queries. Through extensive experiments, we evaluate the efficiency and effectiveness of our proposed TCS-SSN processing approaches over both real and synthetic data. As our future works, we will explore the community search over privacy-preserved [49] or inconsistent spatial-social networks [53, 44].

Acknowledgment

9. REFERENCES

[1] E. Akbas and P. Zhao. Truss-based community search: a truss-equivalence based indexing approach. *PVLDB*, 10(11):1298–1309, 2017.

[2] A. Al-Baghdadi and X. Lian. Topic-based community search over spatial-social networks (technical report). *https://arxiv.org/abs/2007.03014*, 2020.

[3] A. Al-Baghdadi, X. Lian, and E. Cheng. Efficient path routing over road networks in the presence of ad-hoc obstacles. *Information Systems*, 88:101453, 2020.

[4] A. Al-Baghdadi, G. Sharma, and X. Lian. Efficient processing of group planning queries over spatial-social networks. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–1, 2020.

[5] N. Armenatzoglou, S. Papadopoulos, and D. Papadias. A general framework for geo-social query processing. *PVLDB*, 6(10):913–924, 2013.

[6] N. Barbieri, F. Bonchi, and G. Manco. Topic-aware social influence propagation models. *Knowledge and information systems*, 37(3):555–584, 2013.

[7] P. Bhattacharyya, A. Garg, and S. F. Wu. Analysis of user keyword similarity in online social networks. *Social network analysis and mining*, 1(3):143–158, 2011.

[8] F. Bi, L. Chang, X. Lin, and W. Zhang. An optimal and progressive approach to online search of top-k influential communities. *PVLDB*, 11(9):1056–1068, 2018.

[9] X. Cao, L. Chen, G. Cong, C. S. Jensen, Q. Qu, A. Skovsgaard, D. Wu, and M. L. Yiu. Spatial keyword querying. In *International Conference on Conceptual Modeling*, pages 16–29. Springer, 2012.

[10] X. Cao, G. Cong, and C. S. Jensen. Retrieving top-k prestige-based relevant spatial web objects. *PVLDB*, 3(1-2):373–384, 2010.

[11] A. Cary, O. Wolfson, and N. Rishe. Efficient and scalable method for processing top-k spatial boolean queries. In *International Conference on Scientific and Statistical Database Management*, pages 87–95. Springer, 2010.

[12] S. Chaudhuri, S. Agrawal, and G. Das. System for keyword based searching over relational databases, Oct. 5 2004. US Patent 6,801,904.

[13] L. Chen, C. Liu, K. Liao, J. Li, and R. Zhou. Contextual community search over large social networks. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 88–99. IEEE, 2019.

[14] L. Chen, C. Liu, R. Zhou, J. Li, X. Yang, and B. Wang. Maximum co-located community search in large scale social networks. *PVLDB*, 11(10):1233–1246, 2018.

[15] S. Chen, J. Fan, G. Li, J. Feng, K.-l. Tan, and J. Tang. Online topic-aware influence maximization. *PVLDB*, 8(6):666–677, 2015.

[16] Y. Chen, J. Xu, and M. Xu. Finding community structure in spatially constrained complex networks. *International Journal of Geographical Information Science*, 29(6):889–911, 2015.

[17] P. Chopade and J. Zhan. A framework for community detection in large networks using game-theoretic modeling. *IEEE Transactions on Big Data*, 3(3):276–288, 2016.

[18] 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.

[19] W. Cui, Y. Xiao, H. Wang, Y. Lu, and W. Wang. Online search of overlapping communities. In *Proceedings of the 2013 ACM SIGMOD international conference on Management of data*, pages 277–288, 2013.

[20] W. Cui, Y. Xiao, H. Wang, and W. Wang. Local search of communities in large graphs. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 991–1002. ACM, 2014.

[21] I. De Felipe, V. Hristidis, and N. Rishe. Keyword search on spatial databases. In *2008 IEEE 24th International Conference on Data Engineering*, pages 656–665. IEEE, 2008.

[22] P. Expert, T. S. Evans, V. D. Blondel, and R. Lambiotte. Uncovering space-independent communities in spatial networks. *Proceedings of the National Academy of Sciences*, 108(19):7663–7668, 2011.

[23] Y. Fang, R. Cheng, X. Li, S. Luo, and J. Hu. Effective community search over large spatial graphs. *PVLDB*, 10(6):709–720, 2017.

[24] Y. Fang, R. Cheng, S. Luo, and J. Hu. Effective community search for large attributed graphs. *PVLDB*, 9(12):1233–1244, 2016.

[25] Y. Fang, Z. Wang, R. Cheng, X. Li, S. Luo, J. Hu, and X. Chen. On spatial-aware community search. *IEEE Transactions on Knowledge and Data Engineering*, 31(4):783–798, 2018.

[26] Y. Fang, Y. Yang, W. Zhang, X. Lin, and X. Cao. Effective and efficient community search over large heterogeneous information networks. *PVLDB*, 13(6):854–867, 2020.

[27] S. Fortunato. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

[28] M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[29] D. Guo. Regionalization with dynamically constrained agglomerative clustering and partitioning (redcap). *International Journal of Geographical Information Science*, 22(7):801–823, 2008.

[30] R. Hariharan, B. Hore, C. Li, and S. Mehrotra. Processing spatial-keyword (sk) queries in geographic information retrieval (gir) systems. In *19th International Conference on Scientific and Statistical Database Management (SSDBM 2007)*, pages 16–16. IEEE, 2007.

[31] V. Hristidis, Y. Papakonstantinou, and L. Gravano. Efficient ir-style keyword search over relational databases. In *PVLDB*, pages 850–861. Elsevier, 2003.

[32] X. Huang and L. V. Lakshmanan. Attribute-driven community search. *PVLDB*, 10(9):949–960, 2017.

[33] X. Huang, L. V. Lakshmanan, J. X. Yu, and H. Cheng. Approximate closest community search in networks. *arXiv preprint arXiv:1505.05956*, 2015.

[34] M. Islam, M. E. Ali, Y.-B. Kang, T. Sellis, F. M. Choudhury, et al. Keyword aware influential community search in large attributed graphs. *arXiv preprint arXiv:1912.02114*, 2019.

[35] V. Kacholia, S. Pandit, S. Chakrabarti, S. Sudarshan, R. Desai, and H. Karambelkar. Bidirectional expansion for keyword search on graph databases. In *Proceedings of the 31st international conference on Very large data bases*, pages 505–516, 2005.

[36] D. Kempe, J. M. Kleinberg, and É. Tardos. Maximizing the spread of influence through a social network. In *KDD*, 2003.

[37] A. Lancichinetti and S. Fortunato. Limits of modularity maximization in community detection. *Physical review E*,

84(6):066122, 2011.

[38] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On trip planning queries in spatial databases. In *International symposium on spatial and temporal databases*, pages 273–290. Springer, 2005.

[39] G. Li, S. Chen, J. Feng, K.-l. Tan, and W.-s. Li. Efficient location-aware influence maximization. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 87–98, 2014.

[40] G. Li, B. C. Ooi, J. Feng, J. Wang, and L. Zhou. Ease: an effective 3-in-1 keyword search method for unstructured, semi-structured and structured data. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 903–914, 2008.

[41] J. Li, X. Wang, K. Deng, X. Yang, T. Sellis, and J. X. Yu. Most influential community search over large social networks. In *2017 IEEE 33rd International Conference on Data Engineering (ICDE)*, pages 871–882. IEEE, 2017.

[42] R.-H. Li, L. Qin, J. X. Yu, and R. Mao. Influential community search in large networks. *PVLDB*, 8(5):509–520, 2015.

[43] Y. Li, D. Wu, J. Xu, B. Choi, and W. Su. Spatial-aware interest group queries in location-based social networks. In *Proc. of the ACM International Conference on Information and Knowledge Management*, 2012.

[44] X. Lian, L. Chen, and S. Song. Consistent query answers in inconsistent probabilistic databases. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 303–314, 2010.

[45] Y. Liu, A. Niculescu-Mizil, and W. Gryc. Topic-link lda: joint models of topic and author community. In *proceedings of the 26th annual international conference on machine learning*, pages 665–672, 2009.

[46] R. M. Nallapati, A. Ahmed, E. P. Xing, and W. W. Cohen. Joint latent topic models for text and citations. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 542–550, 2008.

[47] M. E. Newman and M. Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[48] L. Qin, J. X. Yu, L. Chang, and Y. Tao. Querying communities in relational databases. In *2009 IEEE 25th International Conference on Data Engineering*, pages

724–735. IEEE, 2009.

[49] S. Song and L. Chen. Differential dependencies: Reasoning and discovery. *ACM Transactions on Database Systems (TODS)*, 36(3):1–41, 2011.

[50] S. Song, B. Liu, H. Cheng, J. X. Yu, and L. Chen. Graph repairing under neighborhood constraints. *The VLDB Journal*, 26(5):611–635, 2017.

[51] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 939–948. ACM, 2010.

[52] J. Wang and J. Cheng. Truss decomposition in massive networks. *PVLDB*, 5(9):812–823, 2012.

[53] Y. Wang, S. Song, L. Chen, J. X. Yu, and H. Cheng. Discovering conditional matching rules. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 11(4):1–38, 2017.

[54] Z. Xu, Y. Ke, Y. Wang, H. Cheng, and J. Cheng. A model-based approach to attributed graph clustering. In *Proceedings of the 2012 ACM SIGMOD international conference on management of data*, pages 505–516, 2012.

[55] D.-N. Yang, C.-Y. Shen, W.-C. Lee, and M.-S. Chen. On socio-spatial group query for location-based social networks. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 949–957, 2012.

[56] Y. Yuan, X. Lian, L. Chen, Y. Sun, and G. Wang. Rsknn: knn search on road networks by incorporating social influence. *IEEE Transactions on Knowledge and Data Engineering*, 28(6):1575–1588, 2016.

[57] Y. Yuan, X. Lian, L. Chen, Y. Sun, and G. Wang. Rs*k*nn: *k*nn search on road networks by incorporating social influence. *IEEE Trans. Knowl. Data Eng.*, 28(6), 2016.

[58] W. Zhang, J. Wang, and W. Feng. Combining latent factor model with location features for event-based group recommendation. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 910–918, 2013.

[59] Z. Zhang, X. Huang, J. Xu, B. Choi, and Z. Shang. Keyword-centric community search. In *2019 IEEE 35th International Conference on Data Engineering (ICDE)*, pages 422–433. IEEE, 2019.

[60] Y. Zhou, H. Cheng, and J. X. Yu. Graph clustering based on structural/attribute similarities. *PVLDB*, 2(1):718–729, 2009.