This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering

Efficient Processing of Group Planning Queries Over Spatial-Social Networks

Ahmed Al-Baghdadi, Gokarna Sharma, and Xiang Lian
Department of Computer Science, Kent State University
Kent, OH 44242, USA
{aalbaghd, gsharma2, xlian}@kent.edu

Abstract—Recently, location-based social networks, that involve both social and spatial information, have received much attention in many real-world applications such as location-based services (LBS), map utilities, business planning, and so on. In this paper, we seamlessly integrate both social networks and spatial road networks, resulting in a so-called *spatial-social network*, and study an important and novel query type, named *group planning query over spatial-social networks* (GP-SSN), which is very useful for applications such as trip recommendations. In particular, a GP-SSN query retrieves a group of friends with common interests on social networks and a number of spatially close *points of interest* (POIs) on spatial road networks that best match group's preferences and have the smallest traveling distances to the group. In order to tackle the GP-SSN problem, we design effective pruning methods, matching score pruning, user pruning, and distance pruning, to rule out false alarms of GP-SSN query answers and reduce the problem search space. We also propose effective indexing mechanisms to facilitate the GP-SSN query processing, and develop efficient GP-SSN query answering algorithms via index traversals. Extensive experiments have been conducted to evaluate the efficiency and effectiveness of our proposed GP-SSN query processing approaches.

Index Terms—Spatial-social network, group planning query over spatial-social networks, GP-SSN

1 Introduction

NOWADAYS, social networks (e.g., Twitter, Facebook, etc.) have become increasingly popular and important in many people's daily lives. With the proliferation of advanced technologies such as GPS-equipped smart devices and high-speed wireless networks (e.g., WiFi, Bluetooth, or mobile data networks), social-network users can now easily share their spatial locations via mobile devices, and request for *location-based social networking services* (e.g., Yelp [50], Foursquare [19], etc.), such as finding restaurants recommended by friends, and/or spatial locations closest to one's current working place. Therefore, location-based social networks (a.k.a. geo-social networks [3], [18], [52]), that involve both spatial and social information, have recently drawn much attention from the database community, which are useful in numerous real-world applications such as *location-based services* (LBS), map utilities, mobile recommendation systems, and so on.

With the popularity of mobile-based devices, many mobile Apps provide comprehensive location-based functions such as the trip planning, as well as social communications with ones friends on social networks. While many existing works [38], [31], [9] on the trip planning usually recommend *points of interest* (POIs), such as shops or restaurants, for one single user, in practice, some user may prefer to travel together with a group of friends who share common or similar interests (e.g., apparel, food, or places of interest). Previous works on group trip planning [21], [5], [45], [22] always assumed that the user group is known or given at the query time. In reality, however, a user may need to search for such a group of users who are friends of each other with common/similar interests. Inspired by this, in this paper, we will formulate and tackle a *group planning query over spatial-social networks* (GP-SSN), which retrieves a group of friends who share common interests with a query user on social networks, and a number of (spatially close) POIs that best match the group's preferences and have the smallest traveling distances to the group on spatial road networks.

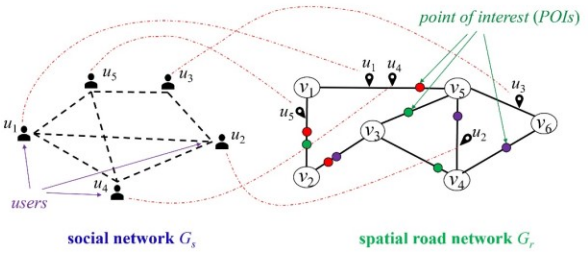


Fig. 1: An illustration of a spatial-social network G_{rs}.

Below, we provide a motivation example of the GP-SSN query to facilitate the plan of visiting POIs for a group of friends.

Example 1. (Destination Planning for a Group of Friends). *Figure 1 illustrates an example of a so-called spatial-social network, denoted as G_{rs}, which combines social networks G_s with spatial road networks G_r. In social networks G_s, users, $u_1 \sim u_5$, are vertices, and edges (e.g., u_1u_2) represent friend relationships between users. In spatial road networks G_r, edges indicate road segments containing POIs such as restaurants or hotels, and vertices (e.g., $v_1 \sim v_6$) are intersection points of roads. In addition, each user u_j ($1 \leq j \leq 5$) on social networks G_r is associated with a link (edge) pointing to some spatial location (e.g., home address) on road networks G_r. This way, spatial-social network G_{rs} is an integrated graph from social and spatial road networks (i.e., G_s and G_r, respectively).*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering

In this spatial-social network G_{rs}, a user, say u_1, may want to utilize social/spatial information available in the spatial-social network, and obtain suggestions/recommendations about forming a group of friends (from social networks) and having a tour of several POIs of their interests that are not far away from their homes (on road networks). ■

The GP-SSN query is also useful in real applications such as online advertising/marketing by recommending discount coupons from several participating merchants to a group of customers.

Example 2. (Online Advertising and Marketing). *Consider an e-commerce company like Groupon [47], [48], which targets at customer groups and offers substantial discounts, coupons, and cashback on group purchases (or called group buying) to a group of people (e.g., with at least 5 people) if they buy products or services together. Assume that a sale manager in Groupon would like to send Groupon deals (e.g., discount coupons for group buying) to a customer, in order to promote products/services of participating merchants. In this case, one can issue a GP-SSN query over users in social networks and merchant location information on road networks, and recommend this customer not only with a group of merchants (i.e., POIs), but also with a group of friends on social networks who share similar interests and match with the merchants' products/services. In such a scenario, the GP-SSN query is very useful and suggestive for online advertising and marketing, such that customers can benefit from discounts and merchants can establish good purchasing habits of customer groups.* ■

There are many other applications for our GP-SSN problem such as recommendation systems (e.g., POI/user recommendations on Yelp), location-based services, and map services.

Our Contributions. In order to efficiently process GP-SSN queries, in this paper, we will propose effective pruning mechanisms, including *matching score pruning*, *user pruning*, and *road-network distance pruning*, that can safely filter out GP-SSN false alarms and reduce the problem search space. Moreover, we will design indexing structures to enable our proposed pruning methods, and propose an efficient algorithm for GP-SSN query answering via the index traversal.

Specifically, we make the following contributions in this paper.

1) We formalize the problem of the group planning query over spatial-social networks (GP-SSN) in Section 2.
2) We propose effective pruning strategies to reduce the search space of the GP-SSN problem in Section 3.
3) We design effective indexing mechanisms to facilitate the GP-SSN query processing in Section 4.
4) We propose an efficient query procedure to tackle the GP-SSN problem in Section 5.
5) We demonstrate through extensive experiments the performance of our GP-SSN approach in Section 6.

Section 6.3 reviews previous works on social networks and/or road networks. Finally, Section 7 concludes this paper.

2 Problem Definition

2.1 Data Model for Spatial-Social Networks

Spatial Road Networks. We first give the definition of spatial road networks (and POIs on them as well).

Definition 1. (Spatial Road Networks, G_r) *A spatial road network, G_r, is a triple $(V(G_r), E(G_r), \phi(G_r))$, where $V(G_r)$ is a set of N vertices $v_1, v_2, \ldots$, and v_N, $E(G_r)$ is a set of edges $e_{j,k}$ (roads between vertices v_j and v_k), and $\phi(G_r)$ is a mapping function: $V(G_r) \times V(G_r) \rightarrow E(G_r)$.*

In Definition 1, the spatial road network G_r can be modeled by a graph, where edges are roads and vertices correspond to intersection points of roads.

Definition 2. (Points of Interest, or POIs) *Given a spatial road network G_r, there are a set, O, of n points of interest (POIs) on G_r, where each POI object $o_i \in O$ is a facility on an edge $e_{j,k} \in E(G_r)$ with the ID $o_i.id$, 2D location $o_i.Loc = (o_i.x, o_i.y)$, and a set, $o_i.K$, of keywords.*

In practice, each POI object o_i (for $1 \leq i \leq n$), as given in Definition 2, may be restaurant, cinema, shop, museum, and so on. The set, $o_i.K$, of keywords describes the POI, including its facility type (e.g., restaurant) and attributes (e.g., French food).

Social Networks. Next, we give the definition of social networks.

Definition 3. (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 $f_{j,k}$ (friendship between users u_j and u_k), and $\phi(G_s)$ is a mapping function: $V(G_s) \times V(G_s) \rightarrow E(G_s)$.*

In Definition 3, each user, u_j (for $1 \leq j \leq m$), in the social network G_s is associated with a vector of d possible interested keywords (or topics) $u_j.w = (w_1^{(j)}.p, w_2^{(j)}.p, ..., w_d^{(j)}.p)$, where the user u_j is interested in the f-th topic $w_f^{(j)}$ with probability $w_f^{(j)}.p \in [0, 1]$. Here, the probability $w_f^{(j)}.p$ can be inferred from user's behavior (e.g., posted/forwarded messages in history). We follow prior works that assume the availability of users' interested topics, for example, [10], where the weighted vector is computed by applying text-based topic discovery algorithms such as [4], [42]. This way, we can obtain a (normalized) weighted vector (distribution) for each user's interests. For other metrics such as Jaccard similarity or Hamming distance, we need to design specific techniques (e.g., pruning with lower/upper bounds of these metrics). We would like to leave the interesting topic of using other metrics such as Jaccard similarity or Hamming distance as our future work. Moreover, in reality, the home of each user resides at a 2D location, $u_j.Loc = (u_j.x, u_j.y)$, on road network G_r.

TABLE 1: Illustration of interest keyword vectors $u_j.w$ for social-network users u_j.

user ID	restaurant	shopping mall	cafe
u_1	0.7	0.3	0.7
u_2	0.2	0.9	0.3
u_3	0.4	0.8	0.8
u_4	0.9	0.7	0.7
u_5	0.1	0.8	0.5

Table 1 depicts an example of the interest keyword (topic) vector (i.e., the row in the table), $u_j.w$ ($1 \leq j \leq 5$), for each social-network user u_j in Figure 1. As an example, u_3 is interested in the *restaurant* topic with probability 0.4, visiting *shopping malls* with probability 0.8, and going to *cafe* with probability 0.8.

Spatial-Social Networks. By integrating both social and road networks, we formally define the spatial-social networks below.

Definition 4. (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, denoted as $G_{rs} = G_r \bigcup G_s$, where users u_j on social networks G_s are located on edges of spatial road networks G_r.*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering

Figure 12 shows an example of spatial-social networks G_{rs} (as given by Definition 4), which integrate both road and social networks in Figure 1. With enriched information from spatial-social networks that bridges the gap between physical and virtual worlds (i.e., road and social networks, resp.), in this paper, we will tackle a novel group trip planning problem (discussed in the next subsection), which cannot be defined on a single type of networks alone (i.e., either road or social networks).

2.2 Group Planning on Spatial-Social Networks

In this subsection, we formalize the problem of the *group planning query over spatial-social networks* (GP-SSN).

Definition 5. (Group Planning Query over Spatial-Social Networks, GP-SSN) *Given a spatial-social network G_{rs}, a query issuer u_q, and a group size τ, a group planning query over spatial-social networks (GP-SSN) is to retrieve a set, S, of τ users from social networks G_s, and a set, R, of POIs from spatial road network G_r, such that:*

- *$u_q \in S$;*
- *all users in S are connected in G_s;*
- *for any two users u_j and u_k in S, the common interest score holds that: $Interest\_Score(u_j, u_k) \geq \gamma$;*
- *POI objects $o_i \in R$ satisfy the condition that the road-network distance of any two POI objects is less than $2r$;*
- *for any user $u_j \in S$, the matching score, $Match\_Score(u_j, R) \geq \theta$; and*
- *the maximum distance between S and R, $maxdist_{RN}(S, R) = \max_{\forall u_j \in S} \max_{\forall o_i \in R} dist_{RN}(u_j, o_i)$, is minimized.*

where γ is an interest score threshold between any two users, θ is a matching threshold between user and POIs, r is the threshold on the traveling distance between POIs, and $dist_{RN}(u_j, o_i)$ is the shortest path distance on road networks G_r between user $u_j \in S$ and POI object $o_i \in R$.

In Definition 5, a GP-SSN query retrieves a group, S, of τ users who are friends and share common interests on social networks, as well as a group, R, of spatially close POI objects that match with these users' interests with the smallest (maximum) road-network distance, $maxdist_{RN}(S, R)$, between S and R. Intuitively, GP-SSN provides the user group S with a POI set R, and users have the flexibility to freely select their preferred POIs in R to visit.

Discussions on the GP-SSN Query Predicates. In our GP-SSN problem definition (Definition 5), the 6 conditions are natural query predicates that are needed in real applications such as group planning or social marketing. For example, the first 3 conditions require the returned user group S must contain the query issuer u_q, include users that are friends of each other (i.e., at least connected and know each other), and have users with common interests (so that they can group buy the same products). The fourth and fifth conditions require the returned group, R, of POI objects be spatially close and match with users' interests, respectively. Finally, the sixth condition aims to find a POI group closest to the user group (i.e., with small traveling time or low fuel consumption). Therefore, these 6 conditions are quite necessary to define our GP-SSN problem

Discussions on the Parameter Tuning. Note that, parameter γ ($\in [0, 1]$) is an interest score threshold that specifies the minimum score that any two users have common/similar interests in the user group S. Larger γ will lead to a user group S with higher social connections.

The matching score threshold θ provides a constraint on the matching score, $Match\_Score(u_j, R)$, between a user u_j and a set, R, of POIs. Intuitively, if a user u_j has interest keywords/topics matching with POIs in R, then this user tends to have a higher matching score with R. Thus, larger interest score threshold θ will result in better matching pairs of user-POI groups.

The parameter r controls the maximum road-network distance between any two POIs in the set R, that is, any two POIs in R have road-network distance less than or equal to $2r$. The larger the value of r, the farther user would have to drive.

It is worth mentioning that, γ, θ, and r are system parameters, which can be tuned from historical query logs or data distributions of users/POIs. In particular, interest score threshold γ can be tuned by the average interest score of pairwise users for those user groups selected in the query log, or the x-th percentile over the distribution of common interest scores for pairwise users in social networks. Similarly, the matching score threshold θ can be specified by taking the average (or x-percentile) of the matching scores between users and POI groups from query logs or spatial-social networks. Further, we can set $2r$ to the maximum road-network distance that a user (or user group) may travel between any two POIs, based on the query history of their trip planning.

In contrast, τ is a user-specified parameter, which indicates the number of users (i.e., user group size) the query user u_q would like to invite for group visiting to POIs. In real applications such as online advertising and marketing (Example 2), this parameter τ can be set to the required number of customers on the coupon (e.g., by Groupon) in order to receive group buying deals.

Semantics of the Interest Score and Matching Score. In Definition 5, we define the score, $Interest\_Score(u_j, u_k)$, of common interests for users as follows:

$$Interest\_Score(u_j, u_k) = \sum_{l=1}^{d} (w_l^{(j)}.p \cdot w_l^{(k)}.p). \quad (1)$$

Furthermore, the matching score, $Match\_Score(u_j, R)$, between user u_j and a set, R, of POIs is given by:

$$Match\_Score(u_j, R) = \sum_{l=1}^{d} \left(w_l^{(j)}.p \cdot \chi \left(w_l^{(j)} \in \bigcup_{\forall o_i \in R} o_i.K \right) \right), \quad (2)$$

where $\chi(z) = 1$, if z is true; $\chi(z) = 0$, if z is false.

Challenges. One straightforward method to tackle the GP-SSN problem is to enumerate all possible groups of users and POI objects, check the query predicates in Definition 5, and return GP-SSN results. However, this method incurs high time complexity, since the number of all possible user/POI groups is rather large.

In particular, there are three major challenges to solve the GP-SSN problem. First, there are an exponential number of possible combinations with τ users (sharing common interests) in spatial-social networks G_{rs}. Second, there are also an arbitrary number of possible circular spatial regions (i.e., circles centered at any locations in the 2D space), and in turn sets, R, of POI objects, which is not efficient to enumerate for answering the GP-SSN query. Third, it is not efficient either to examine all combinations of user and POI groups, S and R, respectively, and obtain the best combination satisfying the conditions given in Definition 5.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering

To retrieve GP-SSN query answers, in this paper, we will design effective pruning strategies (in Section 3) to reduce the search space of the GP-SSN problem. Then, we will devise indexing mechanisms (in Section 4) and develop efficient GP-SSN query answering algorithms (in Section 5) by traversing the index. Please refer to the commonly-used symbols and their descriptions in Table 4 of the supplemental materials.

3 Pruning Strategies

3.1 Matching Score Pruning

We first present the *matching score pruning* method, which utilizes an upper bound, $UB\_Match\_Score(u_j, R)$, of the matching score, $Match\_Score(u_j, R)$, to filter out the false alarms. Specifically, given an upper bound, $UB\_Match\_Score(u_j, R)$, of the matching score between a user u_j and a POI set R, if it holds that $UB\_Match\_Score(u_j, R)$ is smaller than the matching threshold θ, then the POI set R can be safely pruned (since the POI set R does not match with user u_j's interests).

Lemma 1. *(Matching Score Pruning) Given an upper bound, $UB\_Match\_Score(u_j, R)$, of the matching score, $Match\_Score\ (u_j,\ R)$, between a user u_j and a POI set R, if $UB\_Match\_Score\ (u_j,\ R) < \theta$ holds, then the POI set R can be safely pruned.*

Proof. Please refer to Appendix B in the supplemental materials for the detailed proof. □

Derivation of Matching Score Upper Bound. One important issue to enable the matching score pruning is to derive an upper bound of the matching score. In the sequel, we will use the monotonic property of the matching score $Match\_Score(u_j, R)$ to obtain its upper bound.

Lemma 2. *Let R' be a superset of the POI set R, that is, $R \subseteq R'$. Then, we have:*

$$Match\_Score(u_j, R) \leq Match\_Score(u_j, R'). \quad (3)$$

Proof. Please refer to Appendix C in the supplemental materials for the detailed proof. □

From Lemma 2, $Match\_Score(u_j, R')$ is an upper bound of the matching score $Match\_Score(u_j, R)$ (as given in Eq. (3)), which can be used to replace $UB\_Match\_Score(u_j,\ R)$ in Lemma 1.

Discussions on Obtaining a Superset R' of POI Objects in R. To compute the upper bound $Match\_Score(u_j, R')$, one remaining issue is how to obtain the superset, R', of the POI set R (within circles with radii r of road network distance). However, in reality, there are an infinite number of circular regions with radius r in the data space. Thus, it is not efficient, or feasible, to enumerate all of them (with any possible circle centers).

In order to find a superset, R', of a POI object set R (within a circular region with radius r of road network distance), we will consider a larger circular region, $\bigodot(o_i, 2r)$, centered at each POI object o_i with radius $2r$, and obtain all objects falling into region $\bigodot(o_i, 2r)$ as a candidate superset R'. Intuitively, a POI object set R (containing o_i) in any circle with radius r must fall in the region $\bigodot(o_i, 2r)$.

Figure 2 illustrates an example of the (outer) circular region, $\bigodot(o_i, 2r)$, which contains 5 POI objects (forming a superset R'). The inner circle with radius r containing o_i captures a subset, $R \subseteq R'$, with 4 POIs.

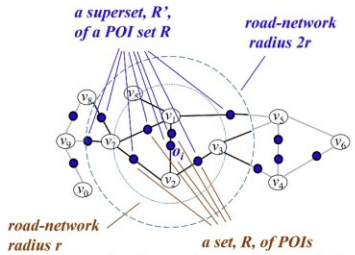


Fig. 2: Illustration of the circular region, $\bigodot(o_i, 2r)$, for the candidate superset R'.

3.2 User Pruning

In this subsection, we design the *user pruning* method, which rules out those users with low interest scores between two users or far away from the query user u_q in spatial-social networks G_{rs}.

Interest Score Pruning. Due to the large scale of graph G_{rs}, it is not space-efficient to offline pre-compute and store all the interest scores between pairwise users. Thus, our basic idea is to derive a pruning region in the data space of the interest score, which rules out pairs of users u_j and u_k with interest scores below threshold γ.

We have the following lemma about the pruning method with respect to interest scores.

Lemma 3. *(Interest Score Pruning) Given a user u_j in the set S and a candidate user u_k, based on Definition 5, if $Interest\_Score\ (u_j,\ u_k) < \gamma$ holds, then u_k can be safely pruned from the set S.*

Proof. Please refer to Appendix D for the detailed proof. □

Note that, according to Definition 5, the query user u_q must be in set S (i.e., $u_j \equiv u_q$ in Lemma 3). Thus, we can apply Lemma 3 to prune other users u_k with u_q.

Next, we aim to derive a pruning region for filtering out false alarms of users u_k. Specifically, we first provide an equivalent form of the interest score $Interest\_Score(u_j, u_k)$, and then use it to derive the pruning condition as given by Lemma 3.

An Equivalent Form of the Interest Score. The interest score function $Interest\_Score(u_j, u_k)$ between two users u_j and u_k (given in Eq. (1)) is equivalent to the *cosine similarity* [13] between two vectors $u_j.w$ and $u_k.w$. That is, we have:

$$\begin{aligned} Interest\_Score(u_j, u_k) &= u_j.w \cdot u_k.w \\ &= ||u_j.w|| \cdot ||u_k.w|| \cdot cos\theta, \end{aligned} \quad (4)$$

where $||V||$ is the length of vector V in the d-dimensional space, and θ is the angle between two vectors $u_j.w$ and $u_k.w$.

Figure 3 shows an example of the cosine similarity between two vectors $u_j.w$ and $u_k.w$ in the d-dimensional space. We consider a plane that contains the origin O and two d-dimensional points $u_j.w$ and $u_k.w$, which are plotted in the figure. Figure 3 depicts the lengths, $||u_j.w||$ (i.e., the distance from origin O to point $u_j.w$) and $||u_k.w|| \cdot cos\theta$ (i.e., the projected line length of vector $u_k.w$ on vector $u_j.w$). Based on Eq. (4), the interest score $Interest\_Score(u_j, u_k)$ can be computed by the multiplication of these two lengths, that is, $||u_j.w||$ times $(||u_k.w|| \cdot cos\theta)$.

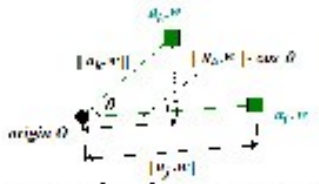


Fig. 3: The cosine similarity between vectors $u_j.w$ and $u_k.w$.

Derivation of a User Pruning Region. By substituting Eq. (4) into the interest score pruning method in Lemma 3, we can obtain the following lemma about the user pruning region.

Definition 6. *(User Pruning Region, $PR(u_i)$) Given a user $u_i \in S$, we define a user pruning region, $PR(u_i)$, w.r.t. u_i as a halfplane that satisfies the following conditions: (1) origin $O \in PR(u_i)$; (2) $PR(u_i)$ is perpendicular to vector $u_i.w$, intersecting with line from O to $u_i.w$ at point A; and (3) the Euclidean distance, $dist(O, A)$, between O and A is given by $\frac{\gamma}{\|u_i.w\|}$.*

Figure 4 shows an example of the user pruning region, $PR(u_i)$, with respect to a user u_i in a 2D space. Point A is on the line segment from origin O to $u_i.w$ such that $dist(O, A) = \frac{\gamma}{\|u_i.w\|}$ holds. The line perpendicular to the line segment $\overline{OA}$ divides the 2D space into two halfplanes, and the pruning region $PR(u_i)$ is given by the halfplane containing the origin O (i.e., the region with sloped lines).

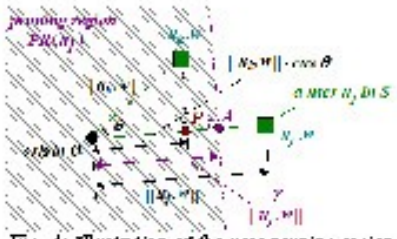


Fig. 4: Illustration of the user pruning region.

With the user pruning region, we have the following pruning corollary.

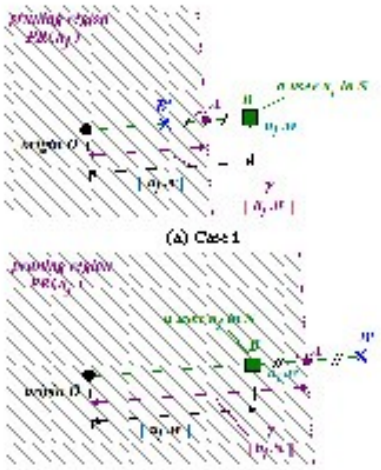
Corollary 1. *(User Pruning Condition 1) Given a user u_i in the set S and an interest score threshold γ, any candidate user u_k whose interest vector $u_k.w$ falls in the user pruning region $PR(u_i)$ can be safely pruned.*

Proof. Please refer to Appendix E in the supplemental materials for the detailed proof. □

Note that, according to Definition 5, the query user u_i must be in the user set S of our GP-SSN answers. Thus, we can apply Corollary 1 by replacing u_i with u_i.

Discussions on How to Obtain the Pruning Region. Figure 5 illustrates two cases of computing the halfplane (that defines the pruning region) in multidimensional data space. Specifically, Case 1 corresponds to the case that $dist(O, A) \leq dist(O, B)$ (i.e., $\|u_i.w\|^2 \geq \gamma$) holds, whereas Case 2 that $dist(O, A) > dist(O, B)$ (i.e., $\|u_i.w\|^2 < \gamma$) holds, where point B corresponds to vector $u_i.w$.

In order to obtain the pruning region $PR(u_i)$ defined by the halfplane (especially in d-dimensional space), we use point B and its mapping point B' in two cases to define the plane (or perpendicular bisector between points B and B'), where



(a) Case 1

(b) Case 2

Fig. 5: The computation of the halfplane w.r.t. a user u_i.

$dist(A, B) = dist(A, B')$. In particular, in both Cases 1 and 2, on each dimension z, we have:

$$B'[z] = w_i^{(j)}.p \cdot \frac{dist(O, B')}{dist(O, B)} = w_i^{(j)}.p \cdot \frac{2 \cdot \gamma - \|u_i.w\|^2}{\|u_i.w\|^2}.$$

Therefore, with respect to B (= $u_i.w$) and B', for any vector $u_k.w$, if it holds that $dist(u_k.w, B') < dist(u_k.w, B)$ in Case 1, we can safely prune user u_k; similarly, if it holds that $dist(u_k.w, B') > dist(u_k.w, B)$ in Case 2, then we can also safely prune user u_k.

Corollary 2. *(User Pruning Condition 2) Let S' be a superset of the user set S of the GP-SSN answers (i.e., $S' \supseteq S$), where $|S'|$ is the size of the set S'. Then, if a user $u_k \in S'$ is in the pruning regions, $PR(u_i)$, of at least $(|S'| - \tau + 1)$ candidates u_i in S', then user u_k can be safely pruned.*

Proof. Please refer to Appendix F in the supplemental materials for the detailed proof. □

Social-Network Distance Pruning. According to Definition 5, our GP-SSN problem retrieves τ connected users in S from social network G_s. Thus, given a query user u_i, the maximum number of possible hops from any user u_j to u_i cannot exceed $(\tau - 1)$ (i.e., $< \tau$). Therefore, we can safely prune those users that are more than $(\tau - 1)$ hops away from u_i.

Lemma 4. *(Social-Network Distance Pruning) For any user $u_k \in G_s$, if it holds that $lb\_dist_{SN}(u_k, u_i) \geq \tau$, then user u_k can be safely pruned, where $lb\_dist_{SN}(u_k, u_i)$ is a lower bound of the number of hops between users u_k and u_i.*

Proof. Please refer to Appendix G in the supplemental materials for the detailed proof. □

Derivation of Distance Lower Bound on Social Networks. We now illustrate how to obtain a distance lower bound between users u_j and u_k. Our basic idea is to select h pivots (users), piv_1, piv_2, .., and piv_h, from the social networks and pre-compute the network distances from each user to these h pivots.

Then, by applying the triangle inequality, we can obtain a lower bound of network distance $dist_{SN}(u_k, u_j)$ in G_s as follows.

$$lb\_dist_{SN}(u_k, u_j) = \max_{z=1}^{h} |dist_{SN}(u_k, piv_z) - dist_{SN}(piv_z, u_j)|.$$

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering

The selection of pivots can be guided by our proposed cost model that gradually finds better pivots (with tighter distance lower bound $lb\_dist_{SN}(u_k, u_q)$ above).

3.3 Road-Network Distance Pruning

In this subsection, we present the road-network distance pruning method, which filters out those false alarms of set pairs, (S'', R''), with large maximum distances $maxdist_{RN}(S'', R'')$. Specifically, we utilize the bounds of the maximum distance to enable the pruning below.

Lemma 5. *(Road-Network Distance Pruning) Denote lower and upper bounds of $maxdist_{RN}(S, R)$ as $lb\_maxdist_{RN}(S, R)$ and $ub\_maxdist_{RN}(S, R)$, respectively. Given two pairs of candidate sets, (S', R') and (S'', R''), if it holds that $ub\_maxdist_{RN}(S', R') \leq lb\_maxdist_{RN}(S'', R'')$, then the pair (S'', R'') can be safely pruned.*

Proof. Please refer to Appendix H in the supplemental materials for the detailed proof. □

Derivation of Distance Lower/Upper Bounds. To enable the distance pruning method in Lemma 5, the remaining issue is how to obtain the upper/lower distance bounds, $ub\_maxdist_{RN}(S', R')$ and $lb\_maxdist_{RN}(S'', R'')$.

In particular, assume that R' is a set of POI objects in a circle centered at o_i with radius $2r$ of road network distance. We compute the upper bound of the maximum distance $ub\_maxdist_{RN}(S', R')$ as follows.

$$\begin{aligned} & ub\_maxdist_{RN}(S', R') \\ = & \max_{\forall u_j \in S'} \{dist_{RN}(u_j, o_i)\} + \max_{\forall o_j \in R'} \{dist_{RN}(o_i, o_j)\} \end{aligned} \quad (5)$$

Similarly, since user u_q belongs to set S'', we can calculate the lower bound of the maximum distance $lb\_maxdist_{RN}(S'', R'')$ below:

$$lb\_maxdist_{RN}(S'', R'') = \max_{\forall o \in R''} dist_{RN}(u_q, o). \quad (6)$$

4 Indexing Mechanism

4.1 Index Structures

To support the GP-SSN query processing and simultaneously prune false alarms of POIs and users in road networks and social networks, respectively, we will build two indexes, $\mathcal{I}_\mathcal{R}$ and $\mathcal{I}_\mathcal{S}$, over POI objects on road networks G_r and users on social networks G_s, respectively. Specifically, we traverse both indexes $\mathcal{I}_\mathcal{R}$ and $\mathcal{I}_\mathcal{S}$ at the same time, apply our proposed pruning methods on both indexes, retrieve candidate pairs, and refine candidate pairs to return actual GP-SSN query answers.

Index $\mathcal{I}_\mathcal{R}$ Over POI Objects on Road Networks G_r. For road networks G_r, we use the R*-tree [6], denoted as $\mathcal{I}_\mathcal{R}$, to index POI objects o_i on road networks. In particular, we insert the 2D locations $o_i.Loc = (o_i.x, o_i.y)$ of POI objects o_i into the R*-tree by using a normal "insert" operator.

Leaf Nodes. Each leaf node in index $\mathcal{I}_\mathcal{R}$ contains POI objects o_i. Each object o_i is associated with 2 pre-computed keyword sets, $o_i.sup\_K$ $(= \bigcup_{\forall o_j \in R'} o_j.K)$ and $o_i.sub\_K$ $(= \bigcup_{\forall o_j \in R''} o_j.K)$. Here R' and R'' are sets of POIs within circular regions of road distance $\odot(o_i, 2r_{max})$ and $\odot(o_i, r_{min})$, respectively, where r_{max} and r_{min} are maximum and minimum possible values of the user-specified radius r, respectively. The two keyword sets will be used for overestimating / underestimating the matching scores. To save the space cost, we hash each keyword $w \in o_i.sup\_K$ (or $w \in o_i.sub\_K$) into a position in a bit vector $o_i.V_{sup}$ (or $o_i.V_{sub}$).

Further, we choose h road-network vertices as pivots $rp_1, rp_2, \ldots,$ and rp_h in G_r, and each POI object o_i in leaf nodes also maintains its road-network distances to these pivots, that is, $dist_{RN}(o_i, rp_k)$ $(1 \leq k \leq h)$. A cost model is proposed in Appendix M of the supplemental materials to guide how to choose good pivots.

Non-Leaf Nodes. Each entry e_R of non-leaf nodes in index $\mathcal{I}_\mathcal{R}$ stores a *minimum bounding rectangle* (MBR) for all POIs under e_R. In addition, e_R is also associated with a keyword superset $e_R.sup\_K$ $(= \bigcup_{\forall o_i \in e_R} o_i.sup\_K)$, a keyword subset $e_R.sub\_K$ $(= o_i.sub\_K$, for some object $o_i \in e_R)$, and lower/upper bounds of road-network distances from POIs under e_R to each pivot, rp_k, as follows:

$$lb\_dist_{RN}(e_R, rp_k) = \min_{\forall o_i \in e_R} dist_{RN}(o_i, rp_k), \quad (7)$$

$$ub\_dist_{RN}(e_R, rp_k) = \max_{\forall o_i \in e_R} dist_{RN}(o_i, rp_k). \quad (8)$$

Furthermore, we maintain a bit vector $e_R.V_{sup}$ for entry e_R, which is a bit-OR of bit vectors $o_i.V_{sup}$ for all $o_i \in e_R$. In addition, we also keep a bit vector $e_R.V_{sub}$, hashed by keywords in set $e_R.sub\_K$.

Index $\mathcal{I}_\mathcal{S}$ for Users on Social Networks G_s. We also build a tree index, $\mathcal{I}_\mathcal{S}$, over users from social networks G_s, which can help access users efficiently for GP-SSN query processing. Specifically, we partition the graph structure of social networks G_s into subgraphs (via standard graph partitioning methods such as [28]), which can be treated as leaf nodes of index $\mathcal{I}_\mathcal{S}$. Then, connected subgraphs in leaf nodes are recursively grouped into non-leaf nodes, until a final root is obtained.

Leaf Nodes. Each user u_j in leaf nodes of index $\mathcal{I}_\mathcal{S}$ is associated with one's interest vector $u_j.w$, and road-network distances, $dist_{RN}(u_j, rp_k)$, from user's home location to road-network pivots $rp_1 \sim rp_h$.

Moreover, we choose l social-network users as pivots sp_1, sp_2, ..., and sp_l in G_s, according to our proposed cost model (see Appendix L in supplemental materials). Then, for each user u_j, we also store social-network distances (i.e., No. of hops), $dist_{SN}(u_j, sp_k)$ (for $1 \leq k \leq l$), to l social-network pivots $sp_1 \sim sp_l$, which can be used for social-network distance pruning.

Non-Leaf Nodes. In each entry e_S of non-leaf nodes, we keep lower/upper bounds of user interest probabilities $w_l^{(j)}.p$ (w.r.t. topics $w_l^{(j)}$) for all users u_j under e_S, given as follows.

$$e_S.lb\_w = (w_1^{(e_S)}.lb\_p, w_2^{(e_S)}.lb\_p, \ldots, w_d^{(e_S)}.lb\_p), \quad (9)$$

$$e_S.ub\_w = (w_1^{(e_S)}.ub\_p, w_2^{(e_S)}.ub\_p, \ldots, w_d^{(e_S)}.ub\_p), \quad (10)$$

where for $1 \leq f \leq d$ we have $w_f^{(e_S)}.lb\_p = \min_{\forall u_j \in e_S}(w_f^{(j)}.p)$, and $w_f^{(e_S)}.ub\_p = \max_{\forall u_j \in e_S}(w_f^{(j)}.p)$.

Furthermore, we also store lower/upper bounds of distances

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering

from (all POIs in) e_S to social-network and road-network pivots:

$$lb\_dist_{SN}(e_S, sp_k) = \min_{\forall u_j \in e_S} dist_{SN}(u_j, sp_k), \quad (11)$$

$$ub\_dist_{SN}(e_S, sp_k) = \max_{\forall u_j \in e_S} dist_{SN}(u_j, sp_k); \quad (12)$$

$$lb\_dist_{RN}(e_S, rp_k) = \min_{\forall u_j \in e_S} dist_{RN}(u_j, rp_k), \quad (13)$$

$$ub\_dist_{RN}(e_S, rp_k) = \max_{\forall u_j \in e_S} dist_{RN}(u_j, rp_k). \quad (14)$$

Discussion on Indexing Mechanisms Over Spatial-Social Networks. Please refer to Appendix O in supplemental materials.

4.2 Index-Level Pruning

4.2.1 Road-Network Index Pruning

We first propose the pruning with the road-network index $\mathcal{I}_\mathcal{R}$.

Matching Score Pruning for Road-Network Index Nodes. Given a user $u_j \in S$ and a node e_R from road-network index $\mathcal{I}_\mathcal{R}$, the basic idea of our index-level matching score pruning is to derive an upper bound, $ub\_Match\_Score(u_j, e_R)$, of matching scores, $Match\_Score(u_j, o_i)$, for all objects $o_i \in e_R$. If this upper bound is below the matching threshold θ (i.e., $ub\_Match\_Score(u_j, e_R) < \theta$), then we can safely discard index node e_R.

Lemma 6. (Matching Score Pruning for Road-Network Index Nodes) *Given a user $u_j \in S$ and a node $e_R \in \mathcal{I}_R$, if $ub\_Match\_Score(u_j, e_R) < \theta$ holds, node e_R can be pruned.*

Proof. Please refer to Appendix I in the supplemental materials for the detailed proof. □

Derivation of the Matching Score Upper Bound. To derive an upper bound of the matching score $ub\_Match\_Score(u_j, e_R)$, we consider the keyword superset, $e_R.sup\_K$, stored in node e_R. Then, we can obtain the upper bound of the matching score below.

$$ub\_Match\_Score(u_j, e_R) = \sum_{f=1}^{d} \left(w_f^{(j)}.p \cdot \chi\left(w_f^{(j)} \in e_R.sup\_K \right) \right), \quad (15)$$

where $\chi(z) = 1$, if z is true; $\chi(z) = 0$, if z is false.

Road-Network Distance Pruning for Index Nodes. Next, we consider the road-network distance pruning, which rules out road-network nodes e_{Ri} that are far away from locations of users in set S. Specifically, we have the following lemma.

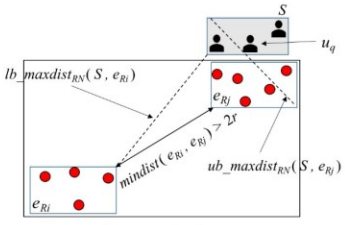


Fig. 6: Illustration of road-network distance pruning with respect to the user set S.

Lemma 7. (Road-Network Distance Pruning for Index Nodes) *Given a set, S, of users from social networks G_s, a node e_{Rj} from index $\mathcal{I}_\mathcal{R}$ (containing at least one candidate set R satisfying the matching score condition with users in S), and a radius r, as illustrated in Figure 6, any node $e_{Ri} \in \mathcal{I}_R$ can be safely pruned, if $lb\_maxdist_{RN}(S, e_{Ri}) > ub\_maxdist_{RN}(S, e_{Rj})$ and $mindist(e_{Ri}, e_{Rj}) > 2r$ hold, where $mindist(e_{Ri}, e_{Rj})$ is the minimum Euclidean distance between nodes e_{Ri} and e_{Rj}.*

Proof. Please refer to Appendix J in the supplemental materials for the detailed proof. □

Discussion on Obtaining Upper/Lower Bounds of $maxdist_{RN}(S, e_R)$. Next, we discuss how to derive upper and lower bounds, $ub\_maxdist_{RN}(S, e_R)$ and $lb\_maxdist_{RN}(S, e_R)$, resp., of maximum road-network distance $maxdist_{RN}(S, e_R)$, which are used in Lemma 7.

For the upper bound, we overestimate the maximum possible distance between S and e_R, via pivots rp_k ($1 \le k \le h$; by the triangle inequality) as follows:

$$ub\_maxdist(S, e_R) = \min_{k=1}^{h} \left\{ \max_{\forall u_j \in S} \{dist_{RN}(u_j, rp_k)\} + ub\_dist_{RN}(e_R, rp_k) + 2r \right\}. \quad (16)$$

Moreover, for the lower bound of the maximum distance, we utilize the query user $u_q \in S$ as follows:

$$lb\_maxdist(S, e_R) = \max_{k=1}^{h} \begin{cases} |dist_{RN}(u_q, rp_k) - lb\_dist_{RN}(e_R, rp_k)|, \\ \quad if\ dist_{RN}(u_q, rp_k) < lb\_dist_{RN}(e_R, rp_k); \\ |dist_{RN}(u_q, rp_k) - ub\_dist_{RN}(e_R, rp_k)|, \\ \quad if\ dist_{RN}(u_q, rp_k) > ub\_dist_{RN}(e_R, rp_k); \\ 0, \quad otherwise. \end{cases} \quad (17)$$

Discussions on How to Guarantee a Candidate Set R in Node e_{Rj}. In Lemma 7, we need to guarantee that there exists at least one candidate set R in index node e_{Rj} that satisfies the matching score condition w.r.t. S. In order to check this condition, we will derive a lower bound, $lb\_Match\_Score(S, e_R)$, of the matching score. If this lower bound is greater than threshold θ for all $u_j \in S$, then there exists at least one candidate set R in node e_{Rj}.

To derive the lower bound $lb\_Match\_Score(S, e_R)$, we maintain a number of samples o_i in node e_R, as well as their keyword subsets $o_i.sub\_K = \bigcup_{\forall o_w \in \bigcirc(o_i, r_{min})} o_w.K$, where r_{min} is the minimum possible radius value that can be specified by the query issuer. Specifically, with samples $o_i \in e_R$, we obtain the lower bound as follows:

$$lb\_Match\_Score(S, e_R) = \max_{\exists \text{ sample } o_i \in e_R} \left\{ \min_{\forall u_j \in S} \sum_{l=1}^{d} \left(w_l^{(j)}.p \cdot \chi\left(w_l^{(j)} \in o_i.sub\_K \right) \right) \right\}, \quad (18)$$

where $\chi(z) = 1$, if z is true; $\chi(z) = 0$, if z is false.

4.2.2 Social-Network Index Pruning

Next, we discuss pruning techniques that filter out nodes e_S on the social-network index $\mathcal{I}_\mathcal{S}$.

Interest Score Pruning for Social-Network Index Nodes. We next discuss our interest score pruning on social-network index $\mathcal{I}_\mathcal{S}$. Similar to the interest score pruning discussed in Section 3.2, we only need to check if an index node e_S is completely in the pruning region.

Lemma 8. (Interest Score Pruning for Social-Network Index Nodes) *Given a query user u_q and an index node $e_S \in \mathcal{I}_S$, node e_S can be safely pruned, if it holds that $e_S \in PR(u_q)$, where $PR(u_q)$ is the user pruning region w.r.t. $u_q.w$ and γ.*

Proof. Please refer to Appendix K in the supplemental materials for the detailed proof. □

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering

To check if all interest score vectors under e_S fall into the pruning region $PR(u_q)$, we combine lower/upper bound of interest vectors, $e_S.lb\_w$ and $e_S.ub\_w$, into an MBR, denoted as $e_S.w$, in the data space of interests. As illustrated in Figure 5, if it holds that $maxdist\ (e_S.w, B') < mindist(e_S.w, B)$ for Case 1 (or $maxdist(e_S.w,\ B) < mindist(e_S.w, B')$ for Case 2), then node e_S can be safely pruned.

Distance Pruning for Social-Network Index Nodes. Given a query user u_q, an index node e_S from social network G_s, and a group size τ, our index-level distance pruning method rules out node e_S, if the distance lower bound $lb\_dist_{SN}(u_q, e_S)$ between u_q and any user under entry e_S on social network G_s is greater than or equal to τ (i.e., $lb\_dist_{SN}(u_q, e_S) \geq \tau$).

Lemma 9. (Distance Pruning for Social-Network Index Nodes) *Given a query user u_q, an index node $e_S \in \mathcal{I}_S$, and a group size τ, node e_S can be safely pruned, if $lb\_dist_{SN}(u_q, e_S) \geq \tau$ holds.*

Proof. Please refer to Appendix L in the supplemental materials for the detailed proof. □

Discussion on Obtaining the Lower Bound of the Social-Network Distance. The only remaining issue is how to obtain the lower bound, $lb\_dist_{SN}(u_q, e_S)$, of the social-network distance. Specifically, each index node $e_S \in \mathcal{I}_S$ stores lower and upper bounds of distances $dist_{SN}(u_j, sp_k)$ (for $1 \leq k \leq l$) from all users $u_j \in e_S$ to l pivots sp_k (as mentioned in Section 4.1), denoted as $lb\_dist_{SN}(e_S, sp_k)$ and $ub\_dist_{SN}(e_S, sp_k)$, respectively.

Then, we can apply the triangle inequality (via pivots) to obtain $lb\_dist_{SN}(u_q, e_S)$ as follows.

$$lb\_dist_{SN}(u_q, e_S) \tag{19}$$
$$= \max_{k=1}^{l} \begin{cases} |dist_{SN}(u_q, sp_k) - lb\_dist_{SN}(e_S, sp_k)|, \\ \quad if\ dist_{SN}(u_q, sp_k) < lb\_dist_{SN}(e_S, sp_k); \\ |dist_{SN}(u_q, sp_k) - ub\_dist_{SN}(e_S, sp_k)|, \\ \quad if\ dist_{SN}(u_q, sp_k) > ub\_dist_{SN}(e_S, sp_k); \\ 0, \quad otherwise. \end{cases}$$

4.2.3 Pivot Selection Algorithm

Algorithm 1 illustrates the details of pivot selection, in light of our proposed cost model, on road or social networks. Specifically, our algorithm first initializes two parameters, $global\_cost$ and $\mathcal{S}_p$, which store the globally optimal cost value and the corresponding pivot set, respectively (line 1). Then, we first randomly select a pivot set $\mathcal{P}$ from road (social) networks, and evaluate the cost function $Cost_{RN}$ in Eq. (20) (or $Cost_{SN}$ in Eq. (21)), recorded by variable $local\_cost$ (lines 3-5). Next, each time we swap a pivot $piv \in \mathcal{P}$ with a non-pivot $npiv$, which results in a new pivot set $\mathcal{P}^{new}$ with new cost $Cost_{RN}^{new}$ (or $Cost_{SN}^{new}$) (lines 6-10). If the new cost is better than the best-so-far cost $local\_cost$, then we can accept the new pivot set and its cost (lines 11-13). 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 $global\_iter$ times (lines 2-3), and record the globally optimal pivot set and its cost (lines 14-16). Finally, we return the best pivot set $\mathcal{S}_p$.

Algorithm 1: Pivot Selection for Road Networks or Social Networks

Input: a road (social) network G_r (or G_s) and the number, h (or l), of pivots
Output: the set, $\mathcal{S}_p$, of pivots
1 $global\_cost = -\infty, \mathcal{S}_p = \emptyset$;
2 **for** $a = 1$ *to* $global\_iter$ **do**
3 randomly select h (or l) initial pivots and form a pivot set $\mathcal{P}$
4 evaluate the cost function $Cost_{RN}$ (or $Cost_{SN}$) of $\mathcal{P}$
5 set $local\_cost = Cost_{RN}$ (or $Cost_{SN}$)
6 **for** $b = 1$ *to* $swap\_iter$ **do**
7 select a random pivot $piv \in \mathcal{P}$
8 randomly choose a non-pivot $npiv$
9 $\mathcal{P}^{new} = \mathcal{P} - \{piv\} + \{npiv\}$
10 evaluate the new cost $Cost_{RN}^{new}$ (or $Cost_{SN}^{new}$) w.r.t. $\mathcal{P}^{new}$
11 **if** *the new cost* $Cost_{RN}^{new}$ *(or* $Cost_{SN}^{new}$*) is better than* $local\_cost$ **then**
12 $local\_cost = Cost_{RN}^{new}$ (or $Cost_{SN}^{new}$)
13 $\mathcal{P} = \mathcal{P}^{new}$
14 **if** $local\_cost$ *is better than* $global\_cost$ **then**
15 $\mathcal{S}_p = \mathcal{P}$
16 $global\_cost = local\_cost$
17 **return** $\mathcal{S}_p$

5 GP-SSN Query Answering

Algorithm 2 illustrates the pseudo code of GP-SSN answering, which processes GP-SSN queries over spatial-social networks G_{rs} via indexes $\mathcal{I}_\mathcal{R}$ and $\mathcal{I}_\mathcal{S}$. Specifically, we traverse both indexes $\mathcal{I}_\mathcal{R}$ and $\mathcal{I}_\mathcal{S}$ at the same time, retrieve candidate POI/user pairs (R, S), and refine candidate pairs to return actual GP-SSN query answers.

In particular, we maintain two candidate sets, R_{cand} and S_{cand}, which store POI object sets (i.e., $R'(o_i)$) in road networks and users in social networks, respectively. Moreover, parameter δ keeps the smallest upper bound of the maximum distance between $R'(o_i)$ and S_{cand}, which can be used for the distance pruning.

Pre-Processing. Initially, we set R_{cand} to an empty set, add the root, $root(\mathcal{I}_\mathcal{S})$, of index $\mathcal{I}_\mathcal{S}$ to set S_{cand}, and let parameter δ be $+\infty$ (line 1). Then, we create an empty minimum heap $\mathcal{H}_\mathcal{R}$ for index traversal, which contains entries (e_R, key), where e_R is the node of index $\mathcal{I}_\mathcal{S}$ (for POI objects), and key is the lower bound of the maximum distance between sets $R'(o_i)$ of POI objects o_i in e_R and users in S_{cand} (line 2).

Traversal of Index $\mathcal{I}_\mathcal{S}$. In Algorithm 2, we first insert a heap entry $(root(\mathcal{I}_\mathcal{R}),\ 0)$ into heap $\mathcal{H}_\mathcal{R}$ (line 3), and then parallel traverse both indexes $\mathcal{I}_\mathcal{R}$ and $\mathcal{I}_\mathcal{S}$ from roots to leaf nodes (lines 4-28). We start from root, $root(\mathcal{I}_\mathcal{S})$, of index $\mathcal{I}_\mathcal{S}$ on level $height(\mathcal{I}_\mathcal{S})$, and visit nodes level by level to leaf nodes on level 0 (line 4). On each level k, for each node e_S in candidate set S_{cand}, we consider its child nodes $e_x \in e_S$, and apply the user pruning method (discussed in Section 3.2). If e_x cannot be pruned, then we add e_x to a candidate set S'_{cand} (lines 5-9). After traversing all nodes $e_S \in S_{cand}$, we obtain a new candidate set S'_{cand} containing their children on a lower level, and update S_{cand} with S'_{cand} (line 10).

Traversal of Index $\mathcal{I}_\mathcal{R}$. At the same time, we will search for candidate POI objects o_i (and objects in their surrounding regions $R'(o_i)$) in index $\mathcal{I}_\mathcal{R}$ (lines 11-26). In particular, we will use heap $\mathcal{H}_\mathcal{R}$ to enable the tree traversal. Each time we pop out an entry (e_R, key) with the minimum key in heap $\mathcal{H}_\mathcal{R}$ (lines 12-13). If this key key is greater than parameter δ, then all entries in heap $\mathcal{H}_\mathcal{R}$ must have their lower bounds of maximum distances greater than the upper bound distances of candidates pairs from R_{cand} and S_{cand}, which indicates that all entries can be safely pruned and the loop can be terminated (line 14).

When e_R is a leaf node, we consider each POI object $o_i \in e_R$, and apply matching score and distance pruning methods to reduce the search space (lines 15-18). If POI objects o_i cannot be pruned, then we will add o_i to candidate set R_{cand}, and meanwhile update the value of parameter δ w.r.t. S_{cand} and $R'(o_i)$ (lines 19-20).

When e_R is a non-leaf node, for each child $e_y \in e_R$, we will

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering

Algorithm 2: GP-SSN Query Answering Algorithm

Input: a spatial-social network G_{rs}, indexes $\mathcal{I}_{\mathcal{R}}$ and $\mathcal{I}_{\mathcal{S}}$, a query issuer u_q, a group size τ, an interest score threshold γ, a matching threshold θ, and a spatial radius threshold r
Output: a pair of sets, (R, S), satisfying GP-SSN query predicates in Def. 5

1 $R_{cand} = \emptyset$; $S_{cand} = root(\mathcal{I}_{\mathcal{S}})$; $\delta = +\infty$;
2 initialize a min-heap $\mathcal{H}_{\mathcal{R}}$ accepting entries in the form (e_R, key)
3 insert entry $(root(\mathcal{I}_{\mathcal{R}}), 0)$ into heap $\mathcal{H}_{\mathcal{R}}$
4 **for** $k = height(\mathcal{I}_{\mathcal{S}})$ *to 0* **do**
5 $\quad S'_{cand} = \emptyset$
6 $\quad$ **for** *each node* $e_S \in S_{cand}$ **do**
7 $\quad\quad$ **for** *each entry* $e_x \in e_S$ **do**
8 $\quad\quad\quad$ **if** e_x *cannot be pruned by the user pruning w.r.t.* u_q **then**
9 $\quad\quad\quad\quad$ add e_x to S'_{cand} // Section 3.2
10 $\quad S_{cand} = S'_{cand}$
11 $\quad$ initialize an empty min-heap $\mathcal{H}_{\mathcal{R}}{}'$
12 $\quad$ **while** $\mathcal{H}_{\mathcal{R}}$ *is not empty* **do**
13 $\quad\quad$ (e_R, key) = de-heap $\mathcal{H}_{\mathcal{R}}$
14 $\quad\quad$ **if** $key > \delta$, **then** terminate the loop;
15 $\quad\quad$ **if** e_R *is a leaf node* **then**
16 $\quad\quad\quad$ **for** *each POI object* $o_i \in e_R$ **do**
17 $\quad\quad\quad\quad$ **if** o_i *cannot be pruned by matching score pruning w.r.t.* u_q *(Section 3.1)* **then**
18 $\quad\quad\quad\quad\quad$ **if** o_i *cannot be pruned by road-network distance pruning w.r.t.* S_{cand} *(Section 3.3)* **then**
19 $\quad\quad\quad\quad\quad\quad$ add o_i to R_{cand}
20 $\quad\quad\quad\quad\quad\quad$ update δ with $\min_{\forall o_i \in R_{cand}} ub\_maxdist(S_{cand}, R'(o_i))$
21 $\quad\quad$ **else**
$\quad\quad\quad$ // e_R is a non-leaf node
22 $\quad\quad\quad$ **for** *each entry* $e_y \in e_R$ **do**
23 $\quad\quad\quad\quad$ **if** e_y *cannot be pruned by matching score pruning w.r.t.* u_q *(Section 3.1)* **then**
24 $\quad\quad\quad\quad\quad$ **if** e_y *cannot be pruned by road-network distance pruning w.r.t.* S_{cand} *(Section 3.3)* **then**
25 $\quad\quad\quad\quad\quad\quad$ insert $(e_y, lb\_maxdist(S_{cand}, e_y))$ into the heap $\mathcal{H}_{\mathcal{R}}{}'$
26 $\quad \mathcal{H}_{\mathcal{R}} = \mathcal{H}_{\mathcal{R}}{}'$
27 **while** $\mathcal{H}_{\mathcal{R}}$ *is not empty* **do**
28 $\quad$ execute lines 13-25 (except that in line 25 the heap entry is inserted into heap $\mathcal{H}_{\mathcal{R}}$)
29 apply the user pruning (Corollary 2) to prune users in S_{cand}
30 apply matching score and distance pruning to prune POIs in R_{cand} (w.r.t. S_{cand})
31 refine $(S_{cand}, R'(o_i))$ (for $o_i \in R_{cand}$) to obtain actual GP-SSN query answers

also use matching score and distance pruning methods to filter out false alarms (lines 21-24). If a node e_y cannot be pruned, then we will insert heap entry $(e_y, lb\_maxdist(S_{cand}, e_y))$ into an initially empty heap $\mathcal{H}_{\mathcal{R}}{}'$ for further checking (line 25).

After the loop (lines 12-25), we keep all candidate children of nodes in heap $\mathcal{H}_{\mathcal{R}}$, and obtain a new candidate set $\mathcal{H}_{\mathcal{R}}{}'$ at a lower level. Then, we will update $\mathcal{H}_{\mathcal{R}}$ with $\mathcal{H}_{\mathcal{R}}{}'$ (line 26).

If the height of index $\mathcal{I}_{\mathcal{S}}$ is smaller than that of index $\mathcal{I}_{\mathcal{R}}$, then we will continue to process the remaining levels in index $\mathcal{I}_{\mathcal{R}}$ by executing lines 13-25 with a slight modification in line 25 (i.e., inserting entry into heap $\mathcal{H}_{\mathcal{R}}$ instead of $\mathcal{H}_{\mathcal{R}}{}'$; lines 27-28).

Refinement. After the index traversals, we will check candidates in S_{cand} and R_{cand} by filtering out false alarms via user, matching score, and distance pruning methods (lines 29-30). Finally, we will refine candidate pairs $(S_{cand}, R'(o_i))$ (for $o_i \in R_{cand}$), and compute/return actual GP-SSN query answers (line 31).

It is worth mentioning that after the index traversal (with user pruning in social networks), we can obtain a candidate set, S_{cand}, of users. We will enumerate subsets (connected subgraphs), S, of S_{cand} with size τ (and a POI set R as well), and refine the POI-user group pairs (R, S). To enhance the efficiency of the enumeration, we can apply subset sampling by randomly expanding the subgraph starting from the query vertex u_q, which we will leave as our future work

TABLE 2: Statistics of real data sets Bri+Cal and Gow+Col.

social network	$\|V(G_s)\|$	$deg(G_s)$	road network	$\|V(G_r)\|$	$deg(G_r)$
Brightkite (Bri)	40K	10.3	California (Cal)	21K	2.1
Gowalla (Gow)	40K	32.1	Colorado (Col)	30K	2.4

6 Experimental Evaluation

6.1 Experimental Settings

We evaluate the performance of our GP-SSN query answering algorithm on both real and synthetic data sets.

Real Data Sets: Specifically, we tested 2 real data sets of spatial-social networks, namely Bri+Cal and Gow+Col. The first real data set, Bri+Cal, is a spatial-social network, which combines social network, Brightkite [7], with California road network [8]; the second spatial-social network, Gow+Col, integrates social network, Gowalla [20], and Colorado road network [12]. Statistics of these social/road networks are depicted in Table 2. Each user u_j in Brightkite or Gowalla social networks has a number of check-in locations (i.e., POIs). Intuitively, if a user often visits some POIs, s/he is more interested in topics/keywords of those POIs. Thus, we build a vector, $u_j.w$, of interested keywords for user u_j, where each element $w_f^{(j)}.p$ corresponds to the percentage of times user u_j visit locations with keyword $w_f^{(j)}$.

We map each user u_j in Brightkite (or Gowalla) social networks to a 2D spatial location (e.g., home address) on California (or Colorado) road network, which is set to the centroid of POIs that s/he checked in.

Synthetic Data Sets: We also generate 2 synthetic spatial-social networks (i.e., graphs) as follows. For spatial road network G_r, we first obtain random intersection points (vertices) in a 2D data space. Then, we produce road segments (edges) by randomly connecting vertices that are spatially close to each other (without introducing new intersection points, since the road network is a plannar graph). Furthermore, we generate n POI objects, by first selecting random edges on road network G_r and then generating w POIs on each edge, where $w \in [0, 5]$ follows the Uniform or Zipf distribution. Each POI o_i is associated with a keyword set $o_i.K$, where each keyword has the value domain $[0, 4]$ with the Uniform/Zipf distribution.

To generate a social network G_s with m users, we randomly connect each user u_j with $deg(G_s)$ users via edges, where degree $deg(G_s)$ follows the Uniform or Zipf distribution within the range $[1, 10]$. Each user u_j is associated with an interest keyword vector $u_j.w$ of size d, containing interest probabilities $w_f^{(j)}.p$ of keywords $w_f^{(j)}$. The interest probability $w_f^{(j)}.p$ has the Uniform/Zipf distribution within $[0, 1]$. Finally, we combine social network G_s with road network G_r, by randomly mapping social-network users to a 2D spatial location on the road network, and obtain a spatial-social network G_{rs}.

With the Uniform or Zipf distribution during the data generation above, we can obtain two types of synthetic spatial-social networks G_{rs}, denoted as UNI and $ZIPF$.

Measures: In order to evaluate the performance of our GP-SSN algorithm, we report the CPU time and the I/O cost. In particular, the CPU time measures the time cost of retrieving the GP-SSN answer candidates by traversing the index (as illustrated in

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering

TABLE 3: Experimental settings.

Parameter	Values
the interest score threshold γ	0.2, **0.3**, 0.5, 0.7, 0.9
the user group size τ	2, 3, **5**, 7, 10
the number, n, of POI objects	3K, 5K, **10K**, 15K, 30K
the number, $\|V(G_r)\|$, of vertices in G_r	10K, 20K, **30K**, 40K, 50K
the number, $\|V(G_s)\|$, of vertices in G_s	10K, 20K, **30K**, 40K, 50K
the matching score threshold θ	0.2, 0.3, **0.5**, 0.7, 0.9
the spatial radius r	0.5, 1, **2**, 3, 4
the number of pivots l or h	2, 3, **5**, 7, 10

Algorithm 2), whereas the I/O cost is the number of page accesses during the GP-SSN query answering.

Competitor: To the best of our knowledge, prior works did not study the group planning problem over spatial-social networks. A straightforward baseline method, $Baseline$, is as follows. We first find all user sets S of size τ (containing query user u_q) from social networks G_s that satisfy the constraint of the interest score threshold γ. Then, we obtain all sets R of POIs in a circular region with radius r, which θ-match with user sets S. Finally, we return a pair, (S, R), with the smallest maximum distance.

Parameter Settings: Table 3 depicts the parameter settings in our experiments, where bold numbers are default parameter values. In each set of our subsequent experiments, we will vary one parameter while setting other parameters to their default values. We ran our experiments on a machine with Intel(R) Core(TM) i7-6700 CPU 3.40 GHz (8 CPUs) and 64 GB memory. All algorithms were implemented by C++.

6.2 Effectiveness Evaluation of the GP-SSN Approach

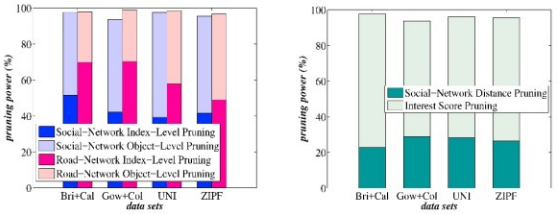
In this subsection, we measure the effectiveness of our proposed GP-SSN pruning strategies over real/synthetics data sets, in terms of the *pruning power*.

The Pruning Power of Index-Level and Object-Level Pruning: Figure 7(a) examines the pruning power of index-level pruning on social-network index, $\mathcal{I}_\mathcal{S}$, and road-network index, $\mathcal{I}_\mathcal{R}$, as well as that of the object-level pruning. In particular, the pruning power of the index-level pruning is given by the percentage of users/POIs that can be ruled out, when we traverse index nodes and apply our pruning methods. Moreover, the pruning power of object-level pruning is defined as the percentage of users/POIs (in leaf nodes) that can be pruned after the index-level pruning.

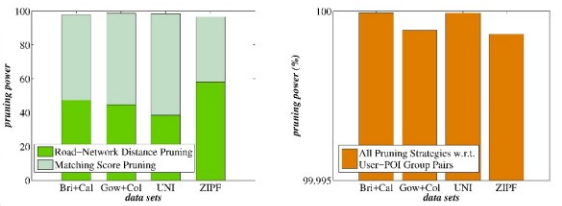
For real/synthetic data sets $Bri + Cal$, $Gow + Col$, UNI, and $ZIPF$, all parameters are set to their default values. From the figure, social-network index-level pruning has 40%∼50% pruning power, and social-network object-level pruning can achieve 50%∼58% pruning power (with an overall pruning power 94%∼97%). Furthermore, the pruning power of road-network index-level pruning is 48%∼70%, and that of road-network object-level pruning is 30%∼42% (i.e., the total pruning power is around 96%∼98%). Therefore, our experimental results confirm the effectiveness of our indexing mechanism and pruning methods.

The Pruning Power of User Pruning on Social Networks: Figure 7(b) illustrates the percentages of users on social networks that can be pruned by social-network distance pruning and interest score pruning over real/synthetic data sets, where parameters are set to their default values. In this figure, we can see that social-network distance pruning can achieve 24%∼30% pruning power, whereas the pruning power of the interest score pruning varies between 65%∼75%. We can clearly see the effectiveness of our pruning methods on the social-network index, $\mathcal{I}_\mathcal{S}$.

The Pruning Power of POI Pruning on Road Networks: Figure 7(c) shows the pruning powers of road-network distance pruning and matching score pruning on real/synthetic data sets $Bri +$

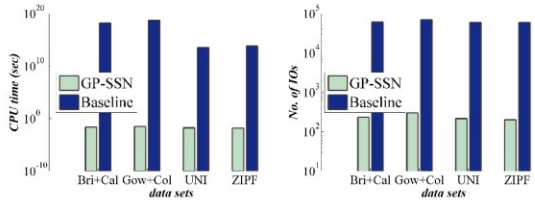


(a) index- and object-level pruning (b) user pruning on social networks



(c) POI pruning on road networks (d) pruning with user-POI group pairs

Fig. 7: The pruning powers of the GP-SSN pruning strategies vs. real/synthetic data sets.



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

Fig. 8: Performance comparisons of GP-SSN with $Baseline$ over real/synthetic data sets.

Cal, $Gow + Col$, UNI, and $ZIPF$, where all parameters are set to their default values. Here, the pruning power is given by the percentage of the remaining POI objects after applying the pruning methods. From the figure, the pruning power of the road-network distance pruning is around 38%∼58%, and that of the matching score pruning is about 55%∼68% (after applying road-network distance pruning), which indicates the effectiveness of our proposed road-network distance and matching score pruning methods.

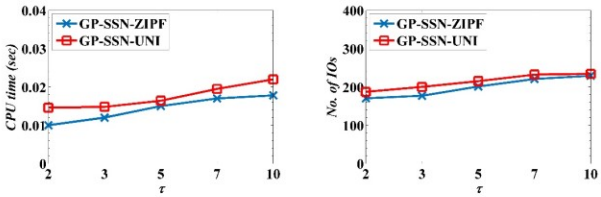
The Pruning Power of User-POI Group Pairs: Figure 7(d) shows the overall pruning power of the user-POI group pairs on real/synthetic spatial-social networks, which is defined as the number of user-POI group pairs that can be pruned divided by the total number of user-POI group pairs, where default parameter values are used. Specifically, from the figure, we can see that the overall pruning power of the user-POI group pairs is high (i.e., 99.9993%∼99.9999%), which confirms the effectiveness of our proposed pruning strategies.

6.3 Efficiency Evaluation of the GP-SSN Approach

In this subsection, we evaluate the GP-SSN performance by testing different real/synthetics data sets.

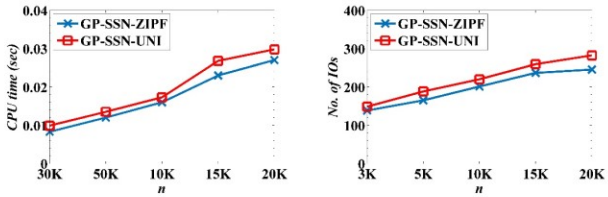
The GP-SSN and $Baseline$ Performances vs. Real/Synthetic Data Sets: Figure 8 compares the performance of our GP-SSN query processing approach with that of the $Basline$ algorithm

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TKDE.2020.3004153, IEEE Transactions on Knowledge and Data Engineering



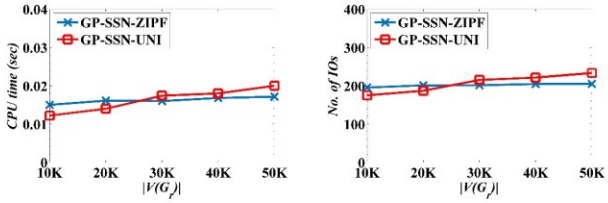
(a) CPU time (b) I/O cost

Fig. 9: The GP-SSN performance vs. the user group size τ.



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

Fig. 10: The GP-SSN performance vs. the number, n, of POI objects.



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

Fig. 11: The GP-SSN performance vs. the number, $|V(G_s)|$, of vertices in road network.

over 4 real/synthetic data sets, Bri+Cal, Gow+Col, UNI, and $ZIPF$, in terms of the CPU time and I/O cost, where all parameters are set to their default values. Since it is not feasible to obtain the exact time cost for $Baseline$ (e.g., almost 1.9×10^{13} days as shown in Figure 8), we will alternatively use a sampling method to estimate the time cost. In particular, we take 100 sample sets S, obtain the average CPU time and I/O cost for retrieving GP-SSN answers w.r.t. one pair (S, R), and estimate the overall $Baseline$ cost by multiplying the average time (or I/O cost) with the total number, $\binom{L}{\tau}$, of candidate pairs.

From the experimental results, for all the real/synthetic data, the CPU time of our proposed GP-SSN algorithm is 0.017∼0.035 sec, and the number of I/Os is around 201∼303. This confirms the efficiency of our GP-SSN query answering algorithm on real/synthetic data. In contrast, the $Baseline$ algorithm takes years (with high CPU time and I/O costs) to obtain the results. Therefore, our GP-SSN approach can outperform the $Baseline$ algorithm by orders of magnitude.

In order to show the performance trend of our GP-SSN approach for different parameters, in subsequent experiments, we will not report the similar results of the $Baseline$ algorithm. Below, we will vary different parameters (e.g., $\gamma, \theta, r, n, |V(G_r)|$, etc., as depicted in Table 3) on synthetic data sets UNI and $ZIPF$ to evaluate the robustness of our GP-SSN approach.

Effect of the User Group Size τ: Figure 9 examines the GP-SSN performance with different sizes, τ, of the user group, in terms of the CPU time and I/O cost, where $\tau = 2, 3, 5, 7$, and 10, and other parameters are set to their default values. From figures, when the user group size τ becomes larger, both CPU time and I/O cost smoothly increase. This is because larger user group may lead to higher computation cost to retrieve more user/POI candidates. Nevertheless, the CPU time and the I/O cost remain low (i.e., 0.01 $\sim$ 0.022 sec and around 170 $\sim$ 235 page accesses, respectively), which indicates the efficiency of our proposed GP-SSN approach for different user group sizes τ.

Effect of the Number, n, of POIs: Figure 10 demonstrates the scalability of our GP-SSN query processing algorithm with different numbers of POIs, n, where $n = 3K, 5K, 10K, 15K$, and $20K$, and default values are used for other parameters. Intuitively, when n becomes larger (i.e., with more POIs), we need more effort to retrieve POI candidates, which thus leads to higher CPU time and I/O cost. From the experimental results, we can see that both CPU time and I/O cost smoothly increase with larger n, which confirms the scalability of our GP-SSN approach w.r.t. n. Nonetheless, the time and I/O costs of our GP-SSN approach remain low (i.e., 0.009∼0.03 sec and 138∼285 I/Os, resp.).

Effect of the Number, $|V(G_r)|$, of Vertices in Road Network: Figure 11 shows the performance of our GP-SSN approach with different sizes of the spatial road network, where the number, $|V(G_r)|$, of vertices in road network varies from $10K$ to $50K$, and we set other parameters to their default values. The experimental results show that the GP-SSN performance is not very sensitive to the size of the road network (i.e., $|V(G_r)|$). This is because we utilize offline pre-computed road-network pivots to enable fast online pruning and GP-SSN query processing. With different $|V(G_r)|$, the CPU time is around 0.014∼0.02 sec, whereas the I/O cost is about 200∼270 I/Os, which indicates the efficiency of our GP-SSN algorithm with large road networks.

Please refer to more experimental results (w.r.t., θ, r, No. of pivots, and γ) in Appendix P, and comprehensive related works in Appendix S of supplemental materials.

7 Conclusions

In this paper, we formulate and tackle the GP-SSN problem on spatial-social networks. To efficiently and effectively tackle the GP-SSN problem, we design effective pruning and indexing mechanisms, and propose efficient GP-SSN query processing algorithms. Through extensive experiments, we confirm the efficiency and effectiveness of our proposed GP-SSN approaches over real/synthetic data sets under various parameter settings. In future, we will consider group planning over privacy-preserved [40] or inconsistent spatial-social networks [46], [33].

Acknowledgments

Xiang Lian was supported by NSF OAC (No. 1739491) and Lian Startup (No. 220981) from Kent State University.

References

[1] R. Agrawal, S. Dar, and H. V. Jagadish. Direct transitive closure algorithms: design and performance evaluation. *TODS*, 15, 1990.
[2] 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.

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

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

[5] S. Barua, R. Jahan, and T. Ahmed. Weighted optimal sequenced group trip planning queries. In *MDM*, pages 222–227. IEEE, 2017.

[6] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The R*-tree: an efficient and robust access method for points and rectangles. In *SIGMOD*, 1990.

[7] Brightkite. *https://en.wikipedia.org/wiki/Brightkite*. 2018.

[8] California Road Network. *http://www.cs.utah.edu/ lifeifei/SpatialDataset.htm*. 2018.

[9] H. Chen, W.-S. Ku, M.-T. Sun, and R. Zimmermann. The multi-rule partial sequenced route query. In *SIGSPATIAL*, pages 1–10, 2008.

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

[11] W. Chen, Y. Yuan, and L. Zhang. Scalable influence maximization in social networks under the linear threshold model. In *ICDM*, 2010.

[12] Colorado Road Networks. *http://www.dis.uniroma1.it/challenge9/index.shtml*. 2018.

[13] Cosine Similarity. *http://en.wikipedia.org/wiki/Cosine_similarity*. 2018.

[14] W. Cui, Y. Xiao, H. Wang, Y. Lu, and W. Wang. Online search of overlapping communities. In *SIGMOD*, pages 277–288, 2013.

[15] K. Deng, X. Zhou, and H. T. Shen. Multi-source skyline query processing in road networks. In *ICDE*, 2007.

[16] E. W. Dijkstra. A note on two problems in connexion with graphs. In *Numerische Mathematik*, 1959.

[17] P. Domingos and M. Richardson. Mining the network value of customers. In *KDD*, 2001.

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

[19] Foursquare. *https://foursquare.com/*. 2018.

[20] Gowalla. *http://snap.stanford.edu/data/loc-gowalla.html*. 2018.

[21] T. Hashem, S. Barua, M. E. Ali, L. Kulik, and E. Tanin. Efficient computation of trips with friends and families. In *CIKM*, pages 931–940, 2015.

[22] T. Hashem, T. Hashem, M. E. Ali, and L. Kulik. Group trip planning queries in spatial databases. In *SSTD*, pages 259–276. Springer, 2013.

[23] X. Huang and L. V. Lakshmanan. Attribute-driven community search. *Proceedings of the VLDB Endowment*, 10(9):949–960, 2017.

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

[25] Y.-W. Huang, N. Jing, and E. A. Rundensteiner. Integrated query processing strategies for spatial path queries. In *ICDE*, 1997.

[26] Y. Ioannidis, R. Ramakrishnan, and L. Winger. Transitive closure algorithms based on graph traversal. *TODS*, 18, 1993.

[27] H. Jeung, M. L. Yiu, X. Zhou, and C. S. Jensen. Path prediction and predictive range querying in road network databases. *The VLDB Journal*, 19, 2010.

[28] G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392, 1998.

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

[30] R.-M. Kung, E. Hanson, Y. Ioannidis, T. Sellis, L. Shapiro, and M. Stonebraker. Heuristic search in data base systems. In *Expert Database Systems*, 1986.

[31] 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.

[32] Y. Li, D. Wu, J. Xu, B. Choi, and W. Su. Spatial-aware interest group queries in location-based social networks. In *CIKM*, 2012.

[33] 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.

[34] D. Papadias, J. Zhang, N. Mamoulis, and Y. Tao. Query processing in spatial network databases. In *VLDB*, 2003.

[35] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. In *KDD*, pages 228–238, 2005.

[36] M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, 2002.

[37] C. Shahabi, M. R. Kolahdouzan, and M. Sharifzadeh. A road network embedding technique for k-nearest neighbor search in moving object databases. In *GIS*, 2002.

[38] M. Sharifzadeh, M. Kolahdouzan, and C. Shahabi. The optimal sequenced route query. *The VLDB journal*, 17(4):765–787, 2008.

[39] C.-Y. Shen, D.-N. Yang, L.-H. Huang, W.-C. Lee, and M.-S. Chen. Socio-spatial group queries for impromptu activity planning. *IEEE Transactions on Knowledge and Data Engineering*, 28(1):196–210, 2015.

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

[41] 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.

[42] S. Song, H. Zhu, and L. Chen. Probabilistic correlation-based similarity measure on text records. *Information Sciences*, 289:8–24, 2014.

[43] M. Sozio and A. Gionis. The community-search problem and how to plan a successful cocktail party. In *KDD*, pages 939–948. ACM, 2010.

[44] Y. Sun, J. Qi, Y. Zheng, and R. Zhang. K-nearest neighbor temporal aggregate queries. In *EDBT*, 2015.

[45] A. Tabassum, S. Barua, T. Hashem, and T. Chowdhury. Dynamic group trip planning queries in spatial databases. In *SSDBM*, pages 1–6, 2017.

[46] 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.

[47] Wikipedia contributors. Groupon — Wikipedia, the free encyclopedia, 2020. [Online; accessed 25-April-2020].

[48] Wikipedia contributors. Investopedia — Wikipedia, the free encyclopedia, 2020. [Online; accessed 25-April-2020].

[49] D. Yang, C. Shen, W. Lee, and M. Chen. On socio-spatial group query for location-based social networks. In *KDD*, 2012.

[50] Yelp. *https://www.yelp.com/*. 2018.

[51] M. L. Yiu, D. Papadias, N. Mamoulis, and Y. Tao. Reverse nearest neighbors in large graphs. In *ICDE*, 2005.

[52] 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.

[53] Q. Zhu, H. Hu, C. Xu, J. Xu, and W.-C. Lee. Geo-social group queries with minimum acquaintance constraints. *The VLDB Journal*, 26(5):709–727, 2017.

Ahmed Al-Baghdadi received his BS degree from the Dept. of Computer Science, Uni. of Al-Qadissyah, Iraq, and the MSc degree in computer science from the Dept. of Computer Science, Kent State University, USA. He is now a PhD candidate at the Dept. of Computer Science, KSU, USA. His research interests include query processing on uncertain/certain graph databases, and social and road networks.

Gokarna Sharma is an Assistant Professor in the Department of Computer Science at Kent State University, Ohio, USA. He research interests are in distributed systems and algorithms, robotics, and emerging technologies such as blockchain. At Kent State, he directs Scalable Computer Architecture & Emerging Technologies (SCALE) lab that focuses on research in these areas. Additionally, he teaches fundamental courses such as multi-core computing, internet of things, and blockchain.

Xiang Lian received the BS degree from the Department of Computer Science and Technology, Nanjing University, China, and the PhD degree in computer science from the Hong Kong University of Science and Technology, Hong Kong. He is now an assistant professor in the Computer Science Department at Kent State University, USA. His research interests include uncertain/certain graph databases, spatio-temporal databases, probabilistic databases, and so on.