Spatial-Aware Local Community Detection Guided by Dominance Relation

Li Ni*, Hefei Xu, Yiwen Zhang*, and Wenjian Luo*, Senior Member, IEEE

Abstract—The problem of finding the spatial-aware community for a given node has been defined and investigated in geosocial networks. However, existing studies suffer from two limitations: 1) the criteria of defining communities are determined by parameters, which are difficult to set, and 2) algorithms may require global information and are not suitable for situations where the network is incomplete. Therefore, we propose spatial-aware local community detection (SLCD), which finds the spatial-aware local community with only local information and defines the community based on the difference in terms of the sparseness of edges inside and outside the community. Specifically, to address the SLCD problem, we design a novel spatial aware local community detection algorithm based on dominance relation, but this algorithm incurs high cost. To further improve the efficiency, we propose a greedy algorithm. Experimental results demonstrate that the proposed greedy algorithm outperforms the comparison algorithms.

Index Terms—Dominance relation, geosocial networks, local community detection, spatial-aware local community detection.

NOMENCLATURE

G Geosocial network.
C Local community C.
Ni Set of neighbor nodes of node i.
NC Set of neighbor nodes community C.
MC M of community C.
SC S of community C.
ND Set of nondominated communities.
NDE Set of nondominated communities that have not been expanded.
HND Set of nondominated communities that have been expanded.
D Set of derived communities that are expanded from nondominated communities.

I. INTRODUCTION

WITH the increasing popularity of location-based services, geosocial networks have emerged [1]. Geosocial networks contain users’ social relations and geographic location information. In geosocial networks, one of the most critical tasks is detecting spatial-aware communities (SACs) [2], which has broad application prospects in many location-based social services, such as event recommendation, social marketing, and geosocial data analysis [3].

In this article, we study the problem of spatial-aware local community detection (SLCD) in geosocial networks. Specifically, for a geosocial network and a given node, the objective is to find the spatial-aware local community to which the given node belongs. The following two properties hold: 1) only local information is used in the process of detecting the community and 2) the community satisfies both structural and spatial cohesiveness. Structural cohesiveness means that nodes inside the community are relatively tightly connected, and nodes inside and outside the community are relatively sparsely connected, while spatial cohesiveness means that the locations of nodes in the same community are close to each other.

Prior Work: The studies on finding communities contain global community detection [4], [5], local community detection [6], [7], and community search [8], [9]. Global community detection algorithms aim to detect all communities in social networks [4], [10], [11]. Most global community detection studies only use topology information to detect communities [5], [12]. In real life, users’ spatial location can affect social relationships because offline social activities are constrained by geography [13], [14], [15]. As studied in [3] and [16], users often meet physically for different purposes, such as dinner and party; geosocial applications (e.g., Meetup and Meetin) wish to provide location-based events for users. Without considering users’ location, users who are far away (e.g., belonging to another province or country) may be invited to dinner or location-based events, which is inappropriate. In these two scenarios, users who are spatially close are more likely to attend the dinner/location-based events. Therefore, some work has considered the user’s location information [14], [15], [17], [18]. Global community detection methods often require global information of the network, such as the total number of edges [19]. However, because of trade secrets, global information about entire networks may be unavailable.

Manuscript received 9 January 2022; revised 10 August 2022; accepted 15 September 2022. Date of publication 10 October 2022; date of current version 3 April 2023. This work was supported in part by the National Key Research and Development Plan of China under Grant 2019YFB1704101; in part by the National Natural Science Foundation of China under Grant 62106004, Grant U1936220, and Grant 61872002; and in part by the Natural Science Foundation of Anhui Province of China under Grant 2108085QP273.

Li Ni, Hefei Xu, and Yiwen Zhang are with the School of Computer Science and Technology, Anhui University, Hefei, Anhui 230601, China (e-mail: nili@ahu.edu.cn; e20301226@stu.ahu.edu.cn; zhangyiwen@ahu.edu.cn).

Wenjian Luo is with the School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China (e-mail: luowenjian@hit.edu.cn).

Digital Object Identifier 10.1109/TCSS.2022.3209976

2329-924X © 2022 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.
or expensive to obtain. In addition, when users want to know the local community to which the given node belongs, it is not necessary to mine all the communities in the network [19].

To compensate for these shortcomings, local community detection has been investigated, which can quickly detect the community that contains the given node with only local information [7], [20], [21], [22]. Similar to the local community detection problem, community search aims to find a subgraph containing a set of given nodes. Some community search works need global information of social networks [8], and some do not [23], [24]. These above studies consider only the link between nodes [7], [8], [20], [21], [23], ignoring the nodes’ locations, so the detected communities may not be spatially cohesive and may not be suitable for some location-based services [3], [15], [25].

To obtain a community that satisfies both structural and spatial cohesiveness, SAC search has attracted attention [1], [3]. Existing studies adopt different spatial constraints to restrict the geographic location of nodes to ensure spatial cohesiveness and place a minimum degree constraint on nodes to guarantee structural cohesiveness [1], [2], [3], [26]. For example, Fang et al. [27] regarded a $k$-core structure with the minimum coverage circle (MCC) as an SAC, which considers the location information of nodes and could obtain a spatially cohesive community.

However, existing studies need to set parameters such as parameter $k$ in the $k$-core, which is not easy for users to set [28]. If $k$ is set to be large, the $k$-core structure does not exist. On the other hand, if $k$ is set to be small, the $k$-core structure may not be structurally cohesive. Taking Eve in Fig. 1 as an example, when $k \geq 4$, the community that contains Eve does not exist; when $k = 2$ or $1$, the $k$-core structure is not sufficiently cohesive. When $k = 3$, the three-core containing Eve is suitable, but the three-core containing Ann does not exist. In addition, some methods require global information [27] and are not suitable for incomplete networks.

For situations in which global information about the network is unavailable or the user is only interested in the community to which the given node belongs, we propose SLCD, which uses only local information to detect the community. To avoid setting the parameter, we adopt the difference between inside and outside the community to detect communities and propose a parameter-free spatial-aware local community detection algorithm with dominance relation (SLDR). SLDR can obtain the community of spatial cohesiveness and structural cohesiveness. However, obtaining the derived communities by the community derivation process incurs a high cost. To address this bottleneck, we propose a greedy community derivation process. The basic idea is to reduce the number of derived communities.

The contributions of this work are summarized as follows.

1) We propose the SLCD problem and define the dominance relation between communities based on the structural and spatial cohesiveness of the communities.
2) We propose SLDR without parameters, which iteratively performs community derivation and community filtration. We further propose the greedy SLDR (SLDRG) with a process of merging multiple communities.
3) We evaluate the effectiveness of the proposed algorithm on synthetic and real datasets. The experimental results show that the greedy algorithm outperforms other comparison algorithms.

The rest of this article is organized as follows. Section II reviews the related works. Section III presents the preliminaries, including the local modularity and dominance relation. Section IV first introduces the SLD problem and then designs SLDR and its greedy algorithm. Section V conducts experiments on synthetic and real geosocial networks. Section VI concludes our work.

II. RELATED WORK

Studies on finding communities contain community detection [4], [7] and community search [23]. Community detection can be divided into global community detection and local community detection [29].

A. Global Community Detection

Community detection aims to detect all communities in social networks [5], [30], [31]. Most community detection works consider only link information when detecting the community [32], [33]. Newman [4] defined modularity and designed the greedy algorithm based on modularity to partition communities in the network. Pizzuti [34] designed a multiobjective evolutionary algorithm (EA) for community detection. It divides the nodes into different communities by optimizing the tightness of intraconnections inside each community and the sparsity of interconnections between different communities. Li et al. [35] proposed a method based on network representation learning, which combines node embedding and community embedding to detect communities in social networks.

Some works on geosocial networks consider the effect of users’ spatial information, aiming at finding communities that satisfy spatial and structural cohesiveness [15], [36], such as the reports in [17] and [18]. The works in [15] and [16] use the location information of the node to weight the edges to transform the unweighted network into a weighted network and then detect the communities. Zhang et al. [18] adopted $k$-core and the pairwise similarity between users based on attribute values (e.g., users’ geolocations) to guarantee the community detection algorithm with dominance relation (SLDR).
cohesiveness of a community from both structural and vertex attributes. Kim et al. [36] proposed geosocial coclustering problem, which aims to cocluster the users in social networks and users’ locations. Furthermore, they designed an efficient geosocial coclustering framework to address the geosocial coclustering problem.

All of these works detect all communities from a global view rather than a local view, which is different from our work.

B. Local Community Detection and Community Search

The goal of local community detection is to obtain the local community that contains the given node with only local information [20], [21]. Researchers have proposed various approaches for local community detection [19], [37], [38], [39], [40]. Luo et al. [7] proposed a local modularity $LQ$ and designed modularity optimization algorithms based on $LQ$. He et al. [29] developed a community detection method based on the local spectral subspace, which is defined based on the Krylov subspace. Lyu et al. [39] proposed an EA-based method for local community detection with some effective strategies in terms of individual representation, fitness evaluation, local search operation, and so on. Most studies on local community detection do not consider the location information of nodes and are not suitable for detecting communities in geosocial networks.

Another related line of work on finding the community that contains the query node is community search. For a graph and a set of query nodes, the task of community search is finding a connected subgraph based on query parameters [28], [41]. Huang et al. [42] proposed a $k$-truss community model and designed a tree index to detect local communities in the network efficiently. Liu et al. [43] studied the SCkT search problem, which aims to find a triangle-connected $k$-truss containing query nodes with sizes no larger than a given threshold as a community. They investigated expansion and shrinking strategies to detect communities in networks. These studies on community search employ only link information to detect communities without considering the location information of nodes, which is not suitable for detecting communities that satisfy both structural and spatial cohesiveness.

Local community detection is similar to community search; both detect the community that contains the given/query node. The difference between local community detection and community search includes the definition of the community and whether to restrict access to only local networks. The latter means that local community detection only visits local networks around a given node, while community search does not have this limitation.

C. SAC Search

Recently, Fang et al. [44] have written a very informative review on community search, in which the authors summarized some spatially aware community search methods on geosocial networks. Most existing studies employ the $k$-core model [1], [2], [3], [27], [45] to ensure the structural cohesiveness of communities. These studies use different spatial constraints, such as the minimum MCC constraint, range constraint, and $k$-nearest neighbor constraint, to ensure the spatial cohesiveness of communities [2], [27]. For example, Fang et al. [3], [27] proposed the exact solution to detect the $k$-core structure covered by the smallest MCC as a community. Wang et al. [1] ensured that the nodes in the same community are geographically close by a radius-bounded circle. However, the value of $k$ in these methods [1], [27], [45] is hard to set. When $k$ is set larger than the degree of the query node, the community does not exist. Even if $k$ is less than the degree of the query node, the community may not be found because the query node and its neighbors may not satisfy the minimum degree constraint; on the other hand, when $k$ is set to be small, too many nodes satisfy the minimum degree constraint, thus making the detected community possibly not structurally cohesive enough.

To our knowledge, existing studies on finding SACs containing given nodes focus mainly on SAC search. SLCD has not received much attention. The goal of SLCD is to detect the local community that satisfies both structural and spatial cohesiveness only with local information. The differences between SAC and SLCD are evident in the following two aspects.

1) For SAC, the criteria of the definition of structural cohesiveness are based on query parameters (e.g., $k$-core [23]). In contrast, the criteria of defining structural cohesiveness for SLCD usually take advantage of the difference in terms of the sparseness of edges inside and outside the community (e.g., local modularity).

2) The SLCD algorithms use only local information when detecting communities, while SAC algorithms have no restriction on whether to use global information, that is, some algorithms require global information [42] and some do not [46].

III. PRELIMINARIES

In this section, we first introduce the local modularity and then introduce some relevant definitions about dominance relation.

A. Local Modularity

Luo et al. [21] proposed the local modularity called $M$ to evaluate the quality of the community. Local modularity $M$ is based on internal and external edges of the community, defined as follows:

$$M = \frac{e_I}{e_O}$$

where $e_I$ is the number of internal edges of the community and $e_O$ is the number of external edges of the community. If the number of internal edges is larger and the number of external edges is smaller, the quality of the community is better.

B. Domination Relation

Given the objective function space $F$ and the solution space $X$, some relevant definitions about dominance relation are given as follows.
Definition 1 (Dominance Relation [47]): Given maximization objective functions: \( f_1(x), f_2(x), f_3(x), \ldots, f_n(x) \in F \), two solutions: \( x_1, x_2 \in X \), \( \forall i \in [1, 2, \ldots, n], \exists j \in [1, 2, \ldots, n] \), if \( f_i(x_1) \leq f_i(x_2) \) and \( f_j(x_1) < f_j(x_2) \), then solution \( x_2 \) dominates solution \( x_1 \) or solution \( x_1 \) is dominated by solution \( x_2 \), denoted as \( x_1 < x_2 \).

Definition 2 (Nondominated Solution and Dominated Solution [47]): Among solution space \( X \), solution \( x \in X \) is a nondominated solution or Pareto-optimal solution if it cannot be dominated by any solution in \( X \); otherwise, \( x \) is a dominated solution.

Dominance relation is often used in multiobjective optimization to find the nondominated solution [48]. To obtain a set of nondominated solutions, Liu et al. [49] proposed a fast bi-objective non-dominated sorting algorithm (BNSA). For the set of multiple solutions \( \{x_1, x_2, x_3, \ldots, x_n\} \) and two maximization objective functions \( f_1 \) and \( f_2 \), the BNSA algorithm [49] first sorts solutions in \( \{x_1, x_2, x_3, \ldots, x_n\} \) in descending order of the \( f_1 \) value. If the two solutions have the same value of \( f_1 \), the solutions are sorted in descending order of \( f_2 \) value. Then, starting from the second solution, each solution \( x_i \) is processed as follows. If it is dominated by the previous solution, \( x_i \) is the dominated solution, which is deleted from sorted solutions; otherwise, \( x_i \) is a nondominated solution, which is retained in sorted solutions. The first solution is a nondominated solution because it has the maximum value of \( f_1 \), which implies that no other solution can dominate it. After all solutions except the first solution are processed, we can obtain nondominated solutions. Supposing that \( n \) is the size of the solution set, the time complexity of the sorting step and comparison step is \( O(n \log n) \) and \( O(n) \), respectively. Therefore, the time complexity of the BNSA algorithm is \( O(n \log n) \) [49].

After executing the BNSA algorithm, a set of nondominated solutions will be obtained. The minimum Manhattan distance (MMD) [50] is often used for selecting one solution from a set of nondominated solutions. For the set of nondominated solutions \( \{x_1, x_2, x_3, \ldots, x_n\} \) and two maximization objective functions \( f_1 \) and \( f_2 \), the MMD of each solution is calculated as follows. First, the values of objective functions \( f_1 \) and \( f_2 \) of each solution are normalized

\[
    f'_i(x) = \frac{f_i(x) - f_i^{\text{max}}}{f_i^{\text{min}} - f_i^{\text{max}}}
\]

where \( f_i^{\text{min}} \) (\( f_i^{\text{max}} \)) is the minimum (maximum) value of all nondominated solutions and \( f_i(x) \) (\( f'_i(x) \)) denotes the objective function value (the normalized objective function value). After values of objective functions \( f_1 \) and \( f_2 \) of each solution are normalized by formula (2), the MMD of each solution \( x_i \) is calculated as

\[
    \text{MMD}_{x_i} = f'_1(x_i) + f'_2(x_i).
\]

In this article, MMD is used for ranking the nondominated solutions in Section IV-C.

IV. PROPOSED ALGORITHM

A. Problem Statement and Community Dominance Relation

We start this section with an introduction to geosocial networks. Then, we introduce the SLCD problem and community dominance relation.

A geosocial network is a graph with node location information. Let \( G(V, E) \) represent a geosocial network, where \( E \) represents the edge set and \( V \) represents the node set. Each node in \( V \) has location information, which is often represented by horizontal and vertical coordinates.

Problem 1 (SLCD Problem): For a geosocial network and a given node, SLCD aims to find the spatial-aware local community, satisfying the following properties.

1) Connectivity: The community containing the given node and nodes in the community is directly or indirectly connected.

2) Structural Cohesiveness: The nodes inside the community are relatively tightly connected to each other, while nodes inside and outside the community are relatively sparsely connected.

3) Spatial Cohesiveness: The locations of nodes in the same community are close to each other.

4) Only Local Information: Only local information is used when detecting the community. For example, only nodes and edges in or near the community are accessed.

The differences between SLCD and SAC problem [3] are mainly in two aspects. One is structural cohesiveness. The criteria of structural cohesiveness of the SLCD problem are based on the difference in the sparseness of edges inside and outside the community. The criteria of structural cohesiveness of the SAC problem are based on query parameters (e.g., k-core). The other is that the SLCD problem uses only local information, while the SAC problem has no restrictions on global information of the network. For better understanding, here, we take the Exact algorithm [3] and the proposed method (Section IV-B) as examples. The first step of the Exact algorithm is to traverse all nodes in the network to extract the k-core subgraph. All nodes in geo-networks are accessed, so the Exact algorithm uses the global information. Our proposed method only accesses the nodes in or near the detected community, that is, only local information is utilized.

Here, we model the SLCD problem with two objectives. The first objective is to maximize the structural cohesiveness of the community. The other is to maximize the spatial cohesiveness of the community. Specifically, we use local community \( M \) [calculated by (1)] to measure structural cohesiveness, while \( S \) is adopted for measuring spatial cohesiveness, formulated as

\[
\begin{align*}
    \max f_1 &= M = \frac{e_L}{e_O} \\
    \max f_2 &= S = -\frac{\sum_{i,j \in C} d(i,j)}{|C| \times (|C| - 1)}
\end{align*}
\]

where \( |C| \) denotes the size of community \( C \) and \( d(i,j) \) denotes the distance between nodes \( i \) and \( j \). \( S \) is a variant of average distance between nodes within the community, which is used to measure the spatial cohesiveness of communities.
in [3] and [27]. Here, we use $S$ as the optimized objective to improve the spatial cohesiveness of communities. Maximizing $M$ and $S$ of the community could make the links within the community dense and the locations of nodes in the same community are close.

When detecting communities in geosocial networks, maximizing the first objective may make the second objective worse and vice versa. Specifically, maximizing the first objective is to add nodes that are more structurally connected to the community into the community. If the node is far away from the community, the second objective will worsen. In summary, the first objective and the second objective are potentially conflicting.

Based on the $M$ value and $S$ value of the community, we define the dominance relation between communities, nondominated community, dominated community, and derived community.

Definition 3 (Community Domination Relation): Given community $C_1$ and community $C_2$, we say that community $C_2$ is dominated by community $C_1$ or $C_1$ dominates $C_2$ (denoted as $C_2 < C_1$) if $M_{C_1} \geq M_{C_2}$ and $S_{C_1} > S_{C_2}$, or $S_{C_1} \geq S_{C_2}$ and $M_{C_1} > M_{C_2}$, where $M_{C_1}$ ($S_{C_1}$) is $M$ ($S$) of the community $C$.

Definition 4 (Nondominated Community and Dominated Community): Among a given set of communities, community $C$ is a nondominated community if it cannot be dominated by any communities; otherwise, $C$ is a dominated community.

Definition 5 (Derived Community): Given a nondominated community $C$ and its neighbor nodes set $N_C$, the community (e.g., $C \cup \{v\}$) expanded by adding one node $v$ in $N_C$ to community $C$ is called the derived community.

B. Basic Algorithm

To address the SLCD problem, we design a novel SLDR. We first introduce SLDR and then provide a detailed description of its two key processes.

Some notations used in this article are listed in the Nomenclature.

1) SLDR: Based on the definitions described in Section IV-A, we propose SLDR for the SLCD problem, as shown in Algorithm 1. The basic idea of this algorithm is to maximize $S$ and $M$ of the community by iteratively performing community derivation (Section IV-B2) and community filtration (Section IV-B3). The community derivation process aims to expand the community by generating communities derived from nondominated communities. The community filtration process aims to obtain the nondominated communities from derived communities.

The general process of SLDR is given as follows. Initially, community $C$ contains $v$ and $NDE$ contains community $C$ (lines 1–6). First, Algorithm 1 expands the communities in $NDE$ to obtain derived communities $D$ by the community derivation process (Section IV-B2). Some nondominated communities in $ND$ may be dominated by communities in $D$ and become dominated communities. Therefore, the algorithm removes the dominated communities from $D \cup ND$ to obtain nondominated communities $ND$ by the community filtration process (Section IV-B3). Then, the algorithm updates $HND$ (line 10), including removing the dominated community in $HND$ (e.g., $ND \cap HND$) and adding the newly expanded nondominated community (e.g., $NDE \cap ND$) to $HND$. Next, the algorithm obtains the nondominated community set $NDE$ by deleting the processed nondominated communities from $ND$ (line 11). If $NDE$ is not empty, the algorithm continues the processes of derivation and filtration. By iteratively performing community derivation and community filtration, the local communities are continuously expanded and optimized; otherwise, the algorithm jumps out of the loop because the communities will not be optimized further at this point. Finally, we select one community with minimum MMD from $ND$ as the final community (line 13), where the MMD [50] (introduced in Section III-B) of community is calculated as

$$\text{MMD} = M' + S'$$

where $M'$ and $S'$ refer to the $M$ and $S$ values of the community after normalizing the $M$ and $S$ of all communities in $ND$ by formula (2), respectively. It may happen that the MMD of two communities is equal. In this case, the community with minimum $\text{dev} = (S/S_{\text{mean}}) - (M/M_{\text{mean}})$ is selected, where $M_{\text{mean}}$ and $S_{\text{mean}}$ denote the average of the $M$ and $S$ of communities in $ND$, respectively.

Example 1: We use node $a$ as the given node and the geosocial network in Fig. 2 to illustrate the process of SLDR. The states of $D$, $ND$, $HND$, and $NDE$ after the first and second loops are shown in Fig. 3. Initially, the community is $\{a\}$, and $NDE$ contains $\{a\}$. Since $NDE$ is not empty, the algorithm enters the first loop. After derived community set $D$ is obtained, the communities in $D \cup HND$ are screened to obtain nondominated communities $ND = \{\{a, b\}, \{a, d\}\}$. 

| Algorithm 1 SLDR | Input: $G, v$ | Output: community that $v$ belongs to |
|------------------|-------------|----------------------------------|
| 1: $C \leftarrow \{v\}$ |  | |
| 2: $N_C \leftarrow N_v$ |  | |
| 3: $M_C \leftarrow 0, S_C \leftarrow -\infty$ |  | |
| 4: $HND \leftarrow \emptyset$ |  | |
| 5: $ND \leftarrow \{C\}$ |  | |
| 6: $NDE \leftarrow \{C\}$ |  | |
| 7: while $NDE \neq \emptyset$ do |  | |
| 8: $D \leftarrow \text{Community Derivation (NDE)}$ (Alg. 2) |  | |
| 9: $ND \leftarrow \text{Community Filtration (D \cup ND)}$ (Alg. 3) |  | |
| 10: $HND \leftarrow (ND \cap HND) \cup (NDE \cap ND)$ |  | |
| 11: $NDE \leftarrow ND - HND$ |  | |
| 12: end while |  | |
| 13: Select one community with minimum MMD from $ND$ |  | |
Then, \( H_{ND} \) is empty, and \( N_{DE} \cap N_{D} = \emptyset \), so \( H_{ND} \) is empty. Since community set \( N_{DE} = N_{D} - H_{ND} = \{a, b\}, \{a, d\} \) is not empty, the algorithm enters the next loop. The second loop performs community derivation and community filtration to obtain \( D \) and \( N_{D} \), respectively. Since \( (N_{DE} \cap H_{ND}) = \emptyset \) and \( N_{DE} \cap N_{D} = \{a, b\}, \{a, d\} \), \( H_{ND} \) is \{a, b\}, \{a, d\}. Communities in \( H_{ND} \) have been processed before and do not be added to the \( N_{DE} \). The community set \( N_{DE} = \{a, b, c\}, \{a, b, d\}, \{a, c, d\} \) is not empty, the algorithm continues community derivation and community filtration processes, omitted to save space.

Although the geosocial network \( G \) is the input of SLDR, SLDR visits only the neighbor nodes of the communities to obtain the derived communities. Therefore, the proposed algorithm only uses the local information instead of the global information of the geosocial network.

2) Community Derivation Process: The community derivation process aims to expand the community by generating derived communities of nondominated communities in \( N_{DE} \). The essential operation of obtaining a derived community is to add one neighbor node to \( N_{C} \) to form a new derived community \( C' \).

Algorithm 2 shows the process of community derivation, which is described as follows. First, the derived community set \( D \) is empty (line 1). Then, for each community \( C \) in \( N_{DE} \) and each node \( u \) in \( N_{C} \), the following steps are performed (lines 4–10): 1) the algorithm obtains the derived community \( C' \) by adding node \( u \) to \( C \) and 2) if \( C' \) is not in \( D \), the algorithm calculates \( M_{C} \) and \( S_{C} \), obtains \( N_{C} \), and adds the derived community \( C' \) to \( D \). Finally, the derived community set \( D \) is obtained.

Example 2: We continue with Example 1. In the first loop, initially, \( N_{DE} = \emptyset \) and \( D = \emptyset \). For node \( b \) in \( N_{[a]} \) where \( N_{[a]} = \{b, c, d, f, g\} \), the following steps are performed: 1) the algorithm obtains a derived community \( C' = \{a, b\} \); 2) since \( \{a, b\} \) is not in \( D \), the algorithm computes \( M_{[a,b]} \) and \( S_{[a,b]} \) to be 0.17 and \(-1\), respectively; and 3) the algorithm obtains \( N_{[a,b]} = \{c, d, f, g\} \) and adds the derived community \( \{a, b\} \) to \( D \). Similarly, \( c, d, f, \) and \( g \) are added to community \( C \). Therefore, \( D = \{a, b\}, \{a, d\}, \{a, c\}, \{a, f\}, \{a, g\} \).

3) Community Filtration Process: In the community filtration process, we apply BNSA [49] (Section III-B) to obtain nondominated communities from derived communities.

Algorithm 3 shows the process of community filtration. For convenience, let \( \text{sortedND} \) represent the list of sorted communities, \( N_{D}' \) represent the nondominated community set, and \( C_{\text{prior}} \) represent the last community that was added to \( N_{D}' \). First, the algorithm sorts the communities in \( D \cup N_{D} \) in descending order of \( M \) and \( S \) values to obtain the sorted...
Algorithm 3 Community Filtration

Input: \( D \cup ND \)
Output: \( ND' \)
1: \( sortedND \leftarrow \) sort communities in \( D \cup ND \)
2: \( C_{prior} \leftarrow \) the first community in \( sortedND \)
3: \( ND' \leftarrow \emptyset \)
4: remove \( C_{prior} \) from \( sortedND \) and add \( C_{prior} \) to \( ND' \)
5: for each community \( C \) in \( sortedND \) do
6: \( \) if \( C \preceq C_{prior} \) is not satisfied then
7: \( \) add \( C \) to \( ND' \)
8: \( C_{prior} \leftarrow C \)
9: end if
10: end for
11: return \( ND' \)

Example 3: Continue with Example 2. In the first loop, sort communities in \( D \cup ND \) to obtain \( sortedND = \{ \{a, b\}, \{a, d\}, \{a, c\}, \{a, f\}, \{a, g\} \} \). The first community \( \{a, b\} \) in \( sortedND \) is a nondominated community and added to \( ND' \). For the second community \( \{a, d\} \), since \( M_{\{a,d\}} \) and \( S_{\{a,d\}} \) are equal to \( M_{\{a,b\}} \) and \( S_{\{a,b\}} \), respectively, \( \{a, d\} \) is not dominated by \( \{a, b\} \). Therefore, \( \{a, d\} \) is a nondominated community and added to \( ND' \). For the third community \( \{a, c\} \), since \( S_{\{a,c\}} \) is less than \( S_{\{a,d\}} \), \( \{a, c\} \) is not a nondominated community. Similarly, \( \{a, f\} \) and \( \{a, g\} \) are dominated communities. Therefore, \( ND' = \{ \{a, b\}, \{a, d\} \} \).

C. Greedy Algorithm

In this section, we first analyze SLDR. Then, we introduce a greedy algorithm, called SLDRG.

In the community derivation process of SLDR, for a nondominated community \( C \), each neighbor node in \( N_C \) is added to community \( C \) to obtain a derived community. The number of derived communities of a community is equal to the number of neighbor nodes of this community. For each derived community, the community derivation process needs to compute \( M \) and \( S \). If there are too many nondominated communities or too many neighbor nodes of a nondominated community, community derivation and community filtration processes have high computational costs.

1) Observation: We start with an important phenomenon. During the execution of SLDR, we observe a phenomenon. The community \( \{a, b, d\} \) can be derived from \( \{a, b\} \) or \( \{a, d\} \). This phenomenon shows that a nondominated community can be derived from multiple communities. This means that the community derivation process spends considerable time generating the same communities. To reduce the computational cost, SLDRG reserves only one nondominated community for the next community derivation and merges nondominated communities based on MMD [50] calculated by (5), which is used to sort nondominated communities.

Algorithm 4 shows the process of SLDRG. To show the difference between SLDR and SLDRG, we focus on the steps of SLDRG that are different from that of SLDR. In the community derivation process, only one nondominated community (i.e., the local community \( C \)) performs community derivation (line 6). After the community filtration process, some nondominated communities in \( ND \) are obtained. If only community \( C \) is in \( ND \), it means that all nondominated communities derived from \( C \) will be dominated by \( C \), that is, \( C \) cannot be further optimized and the algorithm stops (lines 8 and 9); otherwise, the communities in \( ND \) are merged to get a new community \( C \) (Algorithm 5). For convenience, let history record the merged communities returned from Algorithm 5. If the quality of the merged community \( C \) becomes poor (i.e., \( C \) is dominated by a community in history), return the community in history that dominates community \( C \); otherwise, continue with the next loop.

The process of community merging (Algorithm 5) is given as follows. For convenience, let \( sortedND \) represent the list of communities sorted by MMD, \( C_{temp} \) temporarily store the merged community, and \( Clist \) store all merged communities. First, the algorithm calculates the MMD [50] [calculated by (5)] of each community in \( ND \) (line 1). Then, the algorithm sorts communities in \( ND \) by MMD values in ascending order to obtain \( sortedND \) (line 2), and the first community in \( sortedND \) is recorded as \( C_{temp} \) (line 3). Next, the community \( C_{temp} \) is merged with other communities in \( sortedND \) in turn.
Algorithm 5 Community Merging

Input: \( ND \)

Output: \( MergedC \)

1: calculate \( MMD \) for each community in \( ND \)
2: \( sorted\ ND \) ← sort the communities in \( ND \) by \( MMD \)
3: \( C_{temp} \) ← select the first community in \( sorted\ ND \)
4: \( Clist \) ← \( \emptyset \)
5: for each community \( C' \) in \( sorted\ ND \) do
6: \( C_{temp} \leftarrow C' \cup C_{temp} \)
7: compute \( M_{C_{temp}} \) and \( S_{C_{temp}} \)
8: add \( C_{temp} \) to \( Clist \)
9: end for
10: \( ND \) ← Community Filtration \( (\ ND \cup Clist \) (Alg. 3))
11: calculate \( MMD \) for each community in \( ND \)
12: \( C \) ← the community with minimum \( MMD \) in \( ND \)
13: Return \( C \)

D. Complexity Analysis

In this section, we analyze the time complexity and space of the proposed algorithms (i.e., SLDR and SLDRG). Assume that the size of the community is \( c \) and the average degree of nodes in the network is \( d \). SLDR contains two main steps: community derivation and community filtering. The cost of these two steps is analyzed as follows.

1) Line 8: The cost of Algorithm 2 is given as follows. First, there is one node in the community. The sizes of \( NDE \) and \( N_c \) are 1 and \( d \), respectively, so the cost of lines 2–10 is \( d * d \). Then, there are two nodes in the community. The sizes of \( NDE \) and \( N_c \) are 1d and 2d, respectively, so the cost of lines 2–10 is \( 2d^2 * 2d \). Next, there are three nodes in the community. The sizes of \( NDE \) and \( N_c \) are \( 2d^2 \) and \( 3d \), respectively, so the cost of lines 2–10 is \( 3d^3 * 3d \) and so on. Generally, when there are \( c \) nodes in the community, the sizes of \( NDE \) and \( N_c \) are \( (c - 1)d^{c-1} \) and \( cd \), respectively, so the cost of lines 2–10 is \( cc'd^{c+1} \). In summary, the cost is \( O(\sum_{i=1}^{n} i!d^{c+1}) = O(c^2c'd^{c+1}) \).

2) Line 9: The cost of Algorithm 3 is given as follows. The cost of finding nondominated solutions is \( O(n log n) \), where \( n \) is the number of solutions [49]. First, there is one node in the community and the size of \( D \cup ND \) is \( d + 1 \). Next, there are two nodes in the community, and the size of \( D \cup ND \) is \( 1d * 2d + 1d \). Generally, when there are \( c \) nodes in the community, the size of \( D \cup ND \) is \( cd^c + (c - 1)d^{c-1} \). In summary, the cost of line 9 is \( O(\sum_{i=1}^{c} i!d^{c+1}) = O((c^2c'd^{c+1})log(c'd^c)) \).

To sum up, the time complexity of SLDR is \( O(c^2c'd^{c+1} + (c^2c'd^{c+1})log(c'd^c)) \). The maximum space needed for SLDR is an array with the size of \( D \), so the space complexity is \( c'd^c \). The time complexity analysis of SLDRG is given as follows.

1) Line 6: The analysis process of the cost of line 6 is similar to that of line 8 in SLDR. The difference is that only one community in SLDRG is derived at a time. It is easy to deduce that the cost of line 6 is \( \sum_{i=1}^{c} i^2 * d^2 \).

2) Line 7: The analysis process of the cost of line 7 is similar to that of line 9 in SLDR. First, there is one node in the community, and the size of \( D \cup ND \) is \( d + 1 \). Next, there are two nodes in the local community, and the size of \( D \cup ND \) is \( 2d + 1 \). Generally, when
there are \( c \) nodes in the local community, the size of 
\( D \cup ND \) is \( c \times d + 1 \). Therefore, the cost of line 7 is
\( O(\sum_{i=1}^{c}(i(i)d\log(id))) = O((i(i)d\log(id))\).

3) Line 11: The cost of Algorithm 5 is given as follows.
Suppose that the size of the community is \( j \). In line 1,
the time complexity of computing MMD and dev for each
community in \( ND \) is \( O(jd) \). In line 2, the time
complexity of sorting is \( O((jd)\log(jd)) \). Lines 6–8
need to be repeated \( j \times d \) times. The cost of lines 6–8 is
\( O(\sum_{i=1}^{j}(j+i+j) = O(j^2d^2)) \), \( O(jd^2) \), and \( O(jd) \),
respectively. As the size of community grows from 1 to 
\( c \), the cost is \( O(\sum_{i=1}^{c}i^2d^2) \).

To sum up the above analysis, the time complexity of SLDRG
is \( O(\sum_{i=1}^{c}i^2d^2) = O(c^3d^2) \). The maximum space needed for
SLDRG is an array with the size of \( D \), so the space complexity
is \( O(cd) \).

V. EXPERIMENTS

In this section, we conduct comprehensive experiments to
test the proposed algorithms. We first introduce the experi-
mental settings, including datasets, evaluation metrics, and
comparison algorithms. Implementation of this work was
carried out using Centos7 (CPU: Intel Xeon CPU E5-2630
v3 @ 2.40 GHz, memory: 200 GB). The algorithms were
implemented using Python 3.7 programming language, and
the code is available.\(^1\)

A. Experimental Settings

1) Datasets: We tested our algorithm on four real and
two synthetic datasets. The statistics of these datasets are
summarized in Table I. We select 200 nodes evenly from each
dataset and exclude isolated nodes. These 200 given nodes are
selected with a step size of 250, 500, 1000, 10000, 500, and
10000 from the Brightkite, Gowalla, Flickr, Foursquare, Syn1,
and Syn2 datasets, respectively. Each node is selected as the
given node for SLCD, and then, the average values of metrics
are calculated for all selected nodes. The four real datasets are
Brightkite,\(^2\) Gowalla,\(^2\) Flickr,\(^3\) and Foursquare.\(^4\) In the above
four real datasets, each node is a user, and each edge is the
friendship between two users. We implement the processing
of these datasets referring to [27], introduced as follows.

1) Brightkite contains 4491 143 checkin records collected
from 772783 different places from April 2008 to October
2010. The user’s geographic location is the place
that the user checks most frequently.

2) Gowalla contains 6442892 checkin records collected
from 1 280 969 places. Similar to the Brightkite dataset,
the place that the user checks most often is marked as
the user’s geographic location.

3) Flickr contains locations where photographs were taken.
The location where a user took photographs most fre-
quently is marked as the user’s geographic location.

4) Foursquare contains 33 278 683 checkin records,
obtained by crawling the Foursquare website. Each
user in the Foursquare dataset has a location of he/her
hometown position, which is regarded as the user’s
geographic location.

We also conducted experiments on synthetic datasets. Syn-
thetic networks are generated by a graph generator named
GTGraph,\(^5\) following the method in [1] and [27]. We obtained
the synthetic datasets by the following two steps.

1) Generate a social network without node location inform-
ation by the R-MAT graph generator in GTGraph. The
degrees of the nodes in the generated network obey a
power-law distribution, and the default parameter values
of the GTGraph are adopted.

2) Generate location information for all nodes in the social
network. We randomly generate a location coordinate
for each node with a location range of \([0, 1] \times [0, 1]\).
This process is repeated for each node until all nodes
have location coordinates.

Based on these steps, we generated two synthetic datasets:
Syn1 and Syn2.

2) Evaluation Metrics: When measuring the quality of the
community, we take both the structural cohenees and spa-
tial coheseness of the community into account. The metric
\( \text{comm} \)\( \text{mutude} \) [51], [52] is adopted to measure the degree of
structural coheseness of the community; \( d_{\text{avg}} \) and \( d_{\text{lo}} \) are
adopted to measure the degree of spatial coheseness of the
community. These three metrics are introduced as follows.

1) The metric \( \text{comm} \)\( \text{mutude} \) [51], [52] measures the struc-
tural coheseness based on internal and external struc-
tural differences of communities, and it is calculated as
follows:

\[
\text{comm} \text{mutude}(C) = \frac{e[C]}{m} - \left(\frac{D[C]}{2m}\right)^2 \\
\sqrt{\left(\frac{D[C]}{2m}\right)^2 \left(1 - \frac{D[C]}{2m}\right)^2}
\]

where \( m \) is the total number of edges in the network,
\( e[C] \) is the number of internal edges of \( C \), and \( D[C] \) is
the sum of degrees of the nodes in \( C \). A higher value
of \( \text{comm} \)\( \text{mutude} \) indicates better structural coheseness
of the community.

2) Here, \( d_{\text{avg}} \) measures the spatial proximity of nodes in
the community. It is defined as the average distance between
node pairs in \( C \). The smaller the value of \( d_{\text{avg}} \) is, the
better the spatial coheseness of the community.

3) In addition, \( d_{\text{lo}} \) measures spatial coheseness based on
the geographical proximity of internal nodes and
external nodes of the community, which is calculated by
the following equation:

\[
d_{\text{lo}} = \frac{d_{\text{avg}}}{\sum_{i \in C} \sum_{j \in N_C} d(i, j) / (|C| * |N_C|)}
\]

where \( d(i, j) \) is the distance between nodes \( i \) and \( j \).
The smaller the value of \( d_{\text{lo}} \) is, the better the spatial coheseness
of the community.

\(^1\)http://bigdata.ahu.edu.cn/paper
\(^2\)http://snap.stanford.edu/data/index.html
\(^3\)https://www.flickr.com/
\(^4\)https://archive.org/details/201309_foursquare_dataset_umn
\(^5\)http://www.cse.psu.edu/?madduri/software/GTgraph/
We briefly describe these comparison algorithms. Modularity (i.e., AppAcc [27]), and an SAC search method (i.e., Geomod [15]). We briefly describe these comparison algorithms.

1) M Method [21]: This method starts with a given node and then finds a community with the largest local modularity $M$.

2) Geomod [15]: It is a global SAC detection algorithm that detects all communities in geosocial networks. In our experiments, we select the community containing the given node from all detected communities. In addition, parameter $n$ is set to 2.

3) AppAcc [3]: The parameters of the AppAcc algorithm follow the experimental setting in [3]. Specifically, $k$ is set to 4, and $\epsilon_A$ is set to 0.5. Since the exact algorithm (Exact+) in [3] is slow on large datasets, we take the AppAcc algorithm (the most accurate approximation algorithm [3]) as the comparison algorithm.

Since SLDR runs slowly, we use SLDRG instead of SLDR to compare with other algorithms. In addition, if the runtime of SLDRG for a node is longer than 2 h, we terminate it and select one community from the current communities in ND as the final community.

### B. Results

Considering that communities detected by the AppAcc method for some nodes are empty, when comparing M, AppAcc, Geomod, and SLDRG, the average values of metrics are calculated for nodes whose communities detected by AppAcc are not empty. Moreover, the average values of metrics calculated for all selected nodes are also given for M, Geomod, and SLDRG. In addition, Geomod was executed on the Foursquare and Syn2 dataset for more than a week and still did not terminate, so its results are not given.

1) Structural Cohesiveness: Metric communitude (Section V-A2) is adopted to evaluate the structural cohesiveness of the community. Table II shows the average communitude of the communities detected for nodes whose communities detected by AppAcc are not empty. Table III shows the average communitude of the communities detected for all selected nodes.

Table II shows that, in most cases, SLDRG outperforms the AppAcc and Geomod algorithms. The communitude value of SLDRG is more than 1.5 times that of the AppAcc method. SLDRG uses the idea of maximizing both goals of the community, which can climb out of a local optimum to find a community with better structural cohesiveness. The M method performs better than AppAcc and Geomod, as it is designed only for the link-based analysis of nodes and does not consider the spatial cohesiveness of the community, so it focuses more on detecting communities with better structural cohesiveness. From Table II, we see that the values of AppAcc are relatively low. The reason is that the AppAcc algorithm guarantees only the closeness of connections within the community without considering the sparsity of the connections inside and outside the community. The performance of Geomod is worse than that of our algorithm. This is because Geomod uses global modularity (i.e., $Q^{GR}$ [15]) to partition the network into several communities to find the global optimum.

Tables II and III show that SLDRG is competitive with M method on real datasets and better than the Geomod method, which indicates that SLDRG does not lose much structural cohesiveness of the community due to the consideration of spatial cohesiveness. Compared with Table II, we notice some numerical fluctuations in the experimental results on the Brightkite, Gowalla, and Foursquare datasets. It is because many nodes with empty community detected by the AppAcc algorithm are not considered in Table II. For example, for the Brightkite dataset, Table II shows the average communitude of 77 nodes whose communities detected by AppAcc are not empty, and Table III shows the average communitude of all 200 selected nodes. On the Syn1 dataset, the difference between the communitude values in Table II and that in Table III is small.

2) Spatial Cohesiveness: Here, $d_{avg}$ and $d_{IO}$ are adopted to measure the spatial cohesiveness of the community. Table IV shows the average metrics values of the communities detected for nodes whose communities detected by AppAcc are not empty. Table V shows the average metrics values of the communities detected for all selected nodes.

Table IV shows that on the six datasets, SLDRG has smaller $d_{avg}$ and $d_{IO}$ values than other methods, which means that nodes in the community found by SLDRG have closer distances. For $d_{avg}$ and $d_{IO}$, the M method performs worse than the other comparison methods because it does not consider the spatial location information of nodes when detecting community structure. Among the methods that consider the spatial location of nodes, Geomod finds communities with the largest values of $d_{avg}$ and $d_{IO}$ because Geomod detects all communities in the network from the perspective of global

### Table I

| Type  | Datasets | #Vertices | #Edges | Average Degree |
|-------|----------|-----------|--------|----------------|
| Real  | Brightkite | 51,406    | 197,167 | 7.67           |
|       | Gowalla   | 107,092   | 456,830 | 8.53           |
|       | Flicker   | 214,698   | 2,096,306 | 19.5          |
|       | Foursquare| 2,127,093 | 8,460,352 | 8.12          |
| Synthetic | Syn1      | 200,000   | 800,000 | 8              |
|       | Syn2      | 3,000,000 | 12,000,000 | 8             |
TABLE II
COMPARISON OF COMMUNITIES FOR NODES WHOSE COMMUNITIES DETECTED BY APPACC ARE NOT EMPTY. "NUM" MEANS THE NUMBER OF NODES WHOSE COMMUNITIES DETECTED BY APPACC ARE NOT EMPTY.

| Dataset   | Methods | M       | AppAcc  | Geomod  | SLDHG   | Num |
|-----------|---------|---------|---------|---------|---------|-----|
| Brightkite|         | 0.419±0.013 | 0.304±0.016 | 0.449±0.016 | 0.455±0.025 | 77  |
| Gowalla   |         | 0.439±0.021 | 0.294±0.020 | 0.431±0.022 | 0.461±0.029 | 76  |
| Flicker   |         | 0.262±0.006 | 0.125±0.004 | 0.190±0.004 | 0.275±0.012 | 143 |
| Foursquare|         | 0.406±0.019 | 0.256±0.008 | /         | 0.374±0.007 | 53  |
| Syn1      |         | 0.320±0.002 | 0.080±0.000 | 0.153±0.001 | 0.230±0.002 | 181 |
| Syn2      |         | 0.336±0.002 | 0.074±0.000 | /         | 0.249±0.003 | 158 |

* "/" means that the algorithm cannot obtain results because the dataset is too large.

TABLE III
COMPARISON OF COMMUNITIES FOR ALL SELECTED NODES.

| Dataset   | Methods | M       | Geomod  | SLDHG   |
|-----------|---------|---------|---------|---------|
| Brightkite|         | 0.510±0.024 | 0.507±0.030 | 0.512±0.027 |
| Gowalla   |         | 0.529±0.030 | 0.490±0.041 | 0.500±0.030 |
| Flicker   |         | 0.287±0.125 | 0.198±0.007 | 0.289±0.013 |
| Foursquare|         | 0.464±0.031 | /         | 0.396±0.014 |
| Syn1      |         | 0.322±0.002 | 0.154±0.001 | 0.233±0.002 |
| Syn2      |         | 0.345±0.003 | /         | 0.258±0.005 |

* "/" means that the algorithm cannot obtain results because the dataset is too large.

TABLE IV
COMPARISON OF $d_{avg}$ AND $d_{IO}$ FOR NODES WHOSE COMMUNITIES DETECTED BY APPACC ARE NOT EMPTY.

| Methods | Brightkite | Gowalla | Flicker | Foursquare | Syn1 | Syn2 |
|---------|------------|---------|---------|------------|------|------|
|         | $d_{avg}$  | 0.098±0.009 | 0.045±0.004 | 0.172±0.013 | 0.066±0.004 | 0.433±0.005 | 0.426±0.006 |
|         | $d_{IO}$   | 0.796±0.180 | 0.620±0.267 | 0.780±0.110 | 0.836±0.207 | 1.001±0.014 | 0.996±0.019 |
| AppAcc  | $d_{avg}$  | 0.024±0.0029 | 0.017±0.002 | 0.036±0.002 | 0.012±0.001 | 0.240±0.002 | 0.257±0.003 |
|         | $d_{IO}$   | 0.128±0.0500 | 0.108±0.036 | 0.139±0.023 | 0.075±0.0230 | 0.612±0.005 | 0.642±0.008 |
| Geomod  | $d_{avg}$  | 0.061±0.001 | 0.037±0.001 | 0.105±0.001 | / | 0.353±0.003 | / |
|         | $d_{IO}$   | 0.378±0.028 | 0.295±0.027 | 0.415±0.010 | / | 0.861±0.011 | / |
| SLDHG   | $d_{avg}$  | 0.008±0.000 | 0.003±0.000 | 0.024±0.001 | 0.008±0.000 | 0.211±0.002 | 0.220±0.002 |
|         | $d_{IO}$   | 0.064±0.001 | 0.066±0.010 | 0.136±0.011 | 0.054±0.033 | 0.530±0.012 | 0.554±0.013 |

* "/" means that the algorithm cannot obtain results because the dataset is too large.

optimization. Table V shows the average values of metrics calculated for all selected nodes. Although there are some differences between the values in Table V and those in Table IV, similar conclusions are drawn from Table V, i.e., SLDHG performs better than the M and Geomod methods.

As shown in Tables II–V, SLDHG performs competitively on all the datasets. Specifically, SLDHG detected communities on community, $d_{avg}$, and $d_{IO}$ substantially outperformed other SAC detection methods. In addition, SLDHG does not depend on the size of the dataset, and it can detect communities on large datasets. As a comparison, Geomod cannot detect communities on Foursquare and Syn2 datasets because the size of these two datasets is too large.

C. Discussion

1) Effect of k on the AppAcc Algorithm: As mentioned in Section I, the performance of community search algorithms is affected by the parameter $k$. We use the performance of AppAcc on the Brightkite and Gowalla datasets to illustrate the effect of $k$ on the final results. A total of 200 given nodes are selected. Fig. 5 shows the number of communities found by the AppAcc algorithm.

For the Brightkite dataset, when $k = 1, 5,$ and 10, the number of communities is 200, 67, and 24, respectively. Correspondingly, the number of nodes for which no community is found is 0, 133, and 176. Similar results are obtained on
TABLE V
COMPARISON OF $d_{avg}$ AND $d_{IO}$ FOR ALL SELECTED NODES

| Methods | Brightkite | Gowalla | Flickr | Foursquare | Syn1 | Syn2 |
|---------|------------|---------|--------|------------|------|------|
| $d_{avg}$ | 0.083±0.011 | 0.045±0.006 | 0.164±0.015 | 0.076±0.008 | 0.435±0.005 | 0.425±0.008 |
| $d_{IO}$ | 0.799±0.280 | 0.627±0.323 | 0.743±0.136 | 0.845±0.166 | 1.009±0.016 | 0.986±0.028 |
| Geomod | 0.062±0.003 | 0.037±0.002 | 0.103±0.001 | / | 0.382±0.002 | / |
| $d_{avg}$ | 0.386±0.057 | 0.311±0.060 | 0.408±0.013 | / | 0.863±0.011 | / |
| SLDG | 0.13±0.002 | 0.004±0.000 | 0.024±0.001 | 0.016±0.001 | 0.214±0.002 | 0.223±0.003 |
| $d_{avg}$ | 0.141±0.067 | 0.089±0.021 | 0.136±0.011 | 0.168±0.041 | 0.536±0.012 | 0.556±0.015 |

"/" means that the algorithm cannot obtain results because the dataset is too large.

TABLE VI
COMPARISON OF SLDR AND SLDG

| Dataset | Methods | SLDR | SLDG |
|---------|---------|------|------|
| Brightkite | time(s) | communitude | $d_{avg}$ | time(s) | communitude | $d_{avg}$ |
| Gowalla | 3069.36 | 0.569±0.020 | 0.010±0.001 | 41.10 | 0.512±0.027 | 0.013±0.002 |
| Gowalla | 2908.14 | 0.552±0.034 | 0.005±0.000 | 151.84 | 0.500±0.030 | 0.004±0.000 |

Fig. 5. Number of communities detected by AppAcc as $k$ varies.

Fig. 6. Expansion of AppAcc and SLDG.

the Gowalla dataset. From this phenomenon, we can see that $k$ greatly affects the performance of the AppAcc algorithm. Our algorithm has no parameters. For the 200 nodes selected, SLDG could find communities for each given node. From this point of view, compared with AppAcc algorithm, SLDG is more robust.

2) Difference Between AppAcc and SLDG in Structural Cohesiveness: Here, expansion [52] is adopted to measure the sparsity of external edges of individual communities, calculated as follows:

$$\text{expansion} = \frac{|E_{out}|}{|C|}$$

(8)

where $|C|$ is the size of community $C$. The smaller the value of expansion is, the better the structural cohesiveness of the community.

Fig. 6 shows that the expansion of AppAcc is tens times larger than that of SLDG, which means that the community detected by SLDG is more sparsely connected to the external nodes. The reason is that AppAcc considers only the closeness within the community, while SLDG considers the closeness within the community as well as the differences inside and outside the community.

3) Comparison of SLDR and SLDG: We also compare SLDR with SLDG in terms of runtime, communitude, and $d_{avg}$. Due to the slow speed of SLDR, we only compare SLDR and SLDG on two datasets, Brightkite and Gowalla. Table VI shows the results of SLDR and SLDG.

Table VI shows that SLDR is better than SLDG. Although SLDG is slightly worse than SLDR in terms of community quality, the speed of SLDG is much faster than that of SLDR. Specifically, we analyzed the runtime of the algorithms on the 200 nodes. On the Brightkite (Gowalla) dataset, for SLDG, the runtime of 92.5% (92%) nodes is within 10 s. As a comparison, for SLDR, the runtime of only 36% (44%) nodes is within 10 s on the Brightkite (Gowalla) dataset. The reasons for the long runtime of some nodes are given as follows. The degrees of these nodes or their neighbor nodes are greater than several thousands, which causes our algorithms to spend much overhead to generate derived communities.
4) Scalability: To test the scalability of SLDRG, we randomly extract the subgraphs containing 20%, 40%, 60%, 80%, and 100% of the nodes from the dataset and test SLDRG on these subgraphs. Fig. 7(a)–(d) shows the performance of SLDRG in terms of $d_{avg}$, $d_{IO}$, and runtime.

Fig. 7(a) shows that $commumitude$ increases as the percentage of nodes increases. It is because more edges are added to the network and the community structure is more obvious. The nodes in spatial become denser as the percentage increases, which makes the spatial cohesiveness of the communities better. Therefore, $d_{avg}$ and $d_{IO}$ become better as the percentage increases. The running time of SLDRG on Syn1, Syn2, and Brightkite datasets increases as the value of percentage increases. This is because the average degree of nodes in networks increases and the time complexity is proportional to $d^2$, where $d$ means the average degree. For Gowalla, Flickr, and Foursquare datasets, the runtime of SLDRG fluctuates in the runtime of a few nodes is long (more than 1 h) on some subgraphs, while the runtime of these nodes may be short (within a few hundred seconds) on other subgraphs, resulting in the average time to fluctuate. For example, node 610001 in Foursquare takes 3538.19 s on the subgraph with 20% nodes and 968.35 s on the subgraphs with 40% nodes.

VI. CONCLUSION

In this article, we study the SLCD problem, which aims to detect a spatial-aware local community with only local information. To address this problem, we propose SLDR and its efficient greedy algorithm called SLDRG. Extensive experiments on synthetic and real-world datasets demonstrate that SLDRG substantially outperforms other methods in both structural cohesiveness and spatial cohesiveness. In the future, protecting users’ geolocation privacy in the process of detecting community is worthy of studying. Protecting users’ geolocation has received much attention [53]. However, mining communities from geosocial networks under the premise of protecting users’ location privacy has not been well studied yet. This is an important and significant topic in both global and local SAC detection.

REFERENCES

[1] K. Wang, X. Cao, X. Lin, W. Zhang, and L. Qin, “Efficient computing of radius-bounded k-cores,” in Proc. IEEE 34th Int. Conf. Data Eng. (ICDE), Apr. 2018, pp. 233–244.
[2] Q. Zhu, H. Hu, C. Xu, J. Xu, and W.-C. Lee, “Geo-social group queries with minimum acquaintance constraints,” VLDB J., vol. 26, no. 5, pp. 709–727, 2017.
[3] Y. Fang, R. Cheng, X. Li, S. Luo, and J. Hu, “Effective community search over large spatial graphs,” Proc. VLDB Endowment, vol. 10, no. 6, pp. 709–720, Feb. 2017.
[4] M. E. Newman, “Fast algorithm for detecting community structure in networks,” Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 69, no. 6, 2004, Art. no. 066133.
[5] S. Fortunato, “Community detection in graphs,” Phys. Rep., vol. 486, nos. 3–5, pp. 75–174, 2009.
[6] L. Chen, C. Liu, R. Zhou, J. Li, X. Yang, and B. Wang, “Maximum co-located community search in large scale social networks,” Proc. VLDB Endowment, vol. 11, no. 10, pp. 1233–1246, 2018.
[7] W. Luo, D. Zhang, L. Ni, and N. Lu, “Multiscale local community detection in social networks,” IEEE Trans. Knowl. Data Eng., vol. 30, no. 3, pp. 1102–1112, Mar. 2021.
[8] M. Sozio and A. Gionis, “The community-search problem and how to plan a successful cocktail party,” in Proc. 16th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, 2010, pp. 938–948.
[9] K. Yao and L. Chang, “Efficient size-bounded community search over large networks,” Proc. VLDB Endowment, vol. 14, no. 8, pp. 1441–1453, 2021.
[10] P. Li, L. Huang, C. Wang, J. Lai, and D. Huang, “Community detection by motif-aware label propagation,” ACM Trans. Knowl. Discovery From Data, vol. 14, no. 2, pp. 1–29, Jun. 2020.
[11] K. Nath, R. Shamugam, and V. Vijayakumar, “Ma-code: A multi-phase approach on community detection in evolving networks,” Inf. Sci., vol. 569, pp. 326–343, 2021.
[12] R. Guimerà and L. A. N. Amaral, “Functional cartography of complex metabolic networks,” Nature, vol. 433, no. 7028, pp. 895–900, Feb. 2005.
[13] D. Guo, “Regionalization with dynamically constrained agglomerative clustering and partitioning (redcap),” Int. J. Geographical Inf. Sci., vol. 22, no. 7, pp. 801–823, 2008.
[14] P. Expert, T. S. Evans, V. D. Blondel, and R. Lambiotte, “Uncovering space-independent communities in spatial networks,” Proc. Nat. Acad. Sci. USA, vol. 108, no. 19, pp. 7663–7668, 2011.
[15] Y. Chen, J. Xu, and M. Z. Xu, “Finding community structure in spatially constrained complex networks,” Int. J. Geographical Inf. Sci., vol. 29, no. 6, pp. 889–911, Jun. 2015.
[16] W. Zhang, J. Wang, and W. Feng, “Combining latent factor model with location features for event-based group recommendation,” in Proc. 19th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining. Chicago, IL, USA: ACM, 2013, pp. 910–918.
[17] L. Wang, Y. Zeng, Y. Li, Z. Liu, J. Ma, and X. Zhu, “Research on resolution limit of community detection in location-based social networks,” in Proc. Int. Conf. Netw. Netw. Appl. (NaNa), Oct. 2019, pp. 90–95.
[18] F. Zhang, Y. Zhang, L. Qin, W. Zhang, and X. Lin, “When engagement meets similarity: Efficient (k,r)-core computation on social networks,” Proc. VLDB Endowment, vol. 10, no. 10, pp. 998–1009, Jun. 2017.
[19] W. Luo, N. Lu, L. Ni, W. Zhu, and W. Ding, “Local community detection by the nearest nodes with greater centrality,” Inf. Sci., vol. 517, pp. 377–392, May 2020.
[20] A. Clauset, “Finding local community structure in networks,” Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top., vol. 72, no. 2, Aug. 2005, Art. no. 026132.
[21] F. Luo, J. Z. Wang, and E. Proukakis, “Exploring local community structures in large networks,” Web Intell. Agent Syst., Int. J., vol. 6, no. 4, pp. 387–400, 2008.
[22] D. Luo, Y. Bian, Y. Yan, X. Liu, J. Huan, and X. Zhang, “Local community detection in multiple networks,” in Proc. 26th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining, Aug. 2020, pp. 266–274.
[23] W. Cui, Y. Xiao, H. Wang, and W. Wang, “Local search of communities in large graphs,” Proc. ACM SIGMOD Int. Conf. Manage. Data, Jun. 2014, pp. 991–1002.
[24] B. Viswanath, A. Post, K. P. Gummadi, and A. Mislove, “An analysis of social network-based Sybil defenses,” ACM SIGCOMM Comput. Commun. Rev., vol. 40, no. 4, pp. 363–374, 2010.
[25] M. Barthelemy, “Spatial networks,” Phys. Rep., vol. 499, no. 1, pp. 1–101, 2011.
[26] J. Luo, X. Cao, X. Xie, Q. Qu, Z. Xu, and C. S. Jensen, “Efficient attribute-constrained co-located community search,” in Proc. IEEE 36th Int. Conf. Data Eng. (ICDE), Apr. 2020, pp. 1201–1212.

[27] M. Liu, W. Zeng, and J. Zhao, “A fast bi-objective non-dominated sorting algorithm,” Pattern Recognit. Artif. Intell., vol. 24, no. 4, pp. 538–547, 2011.

[28] Y. Fang et al., “On spatial-aware community search,” IEEE Trans. Knowl. Data Eng., vol. 31, no. 4, pp. 783–798, Apr. 2019.

[29] Y. Fang, “An evolutionary multiobjective approach for community discovery in dynamic networks,” IEEE Trans. Cybern., vol. 51, no. 6, pp. 277–288, Jun. 2021.

[30] Q. Liu, Y. Zhu, M. Zhao, X. Huang, J. Xu, and Y. Gao, “VAC: Vertex-centric attributed community search,” in Proc. IEEE 36th Int. Conf. Data Eng. (ICDE), Apr. 2021, pp. 957–948.

[31] C. Pizzuti, “Evolutionary computation for community detection in complex networks: A review,” IEEE Trans. Evol. Comput., vol. 22, no. 3, pp. 464–483, Jun. 2018.

[32] M. Okada, S. Sato, Y. Sato, and Y. Kidawara, “Community detection using restrained random-walk similarity,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 43, no. 1, pp. 89–103, Jan. 2021.

[33] X. Luo, Z. Liu, M. Shang, J. Lou, and M. Zhou, “Highly-accurate community detection via pointwise mutual information-incorporated symmetric non-negative matrix factorization,” IEEE Trans. Netw. Sci. Eng., vol. 8, no. 1, pp. 463–476, Jan. 2021.

[34] J. Luo, X. Cao, X. Xie, Q. Qu, Z. Xu, and C. S. Jensen, “Efficient attribute-constrained co-located community search,” in Proc. IEEE 36th Int. Conf. Data Eng. (ICDE), Apr. 2020, pp. 1201–1212.

[35] F. Folino and C. Pizzuti, “An evolutionary multiobjective approach for community discovery in dynamic networks,” IEEE Trans. Knowl. Data Eng., vol. 26, no. 8, pp. 1838–1852, Aug. 2014.

[36] M. Liu, W. Zeng, and J. Zhao, “A fast bi-objective non-dominated sorting algorithm,” Pattern Recognit. Artif. Intell., vol. 24, no. 4, pp. 538–547, 2011.

[37] W. Li, G. Zhang, T. Zhang, and S. Huang, “Knee point-guided multiobjective optimization algorithm for microgrid dynamic energy management,” Complexity, vol. 2020, pp. 1–11, Nov. 2020.

[38] Y. Li, K. He, K. Kloster, D. Bindel, and J. E. Hopcroft, “Local spectral community detection based on modularity and k-plexes,” Inf. Sci., vol. 513, pp. 127–142, Mar. 2020.

[39] C. Pizzuti, “Evolutionary computation for community detection in networks: A review,” IEEE Trans. Evol. Comput., vol. 16, no. 2, pp. 418–430, Jun. 2012.

[40] M. Li, S. Lu, L. Zhang, Y. Zhang, and B. Zhang, “A community detection method for social network based on community embedding,” IEEE Trans. Computat. Social Syst., vol. 8, no. 2, pp. 308–318, Apr. 2021.

[41] J. Kim, J. Lee, B. S. Lee, and J. Liu, “Geo-social co-clustering: A novel framework for geosocial community detection,” ACM Trans. Intell. Syst. Technol., vol. 11, no. 4, pp. 45:1–45:26, 2020.

[42] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, “Querying k-truss community in large and dynamic graphs,” in Proc. IEEE 36th Int. Conf. Data Eng. (ICDE), Apr. 2021, pp. 957–948.

[43] C. Pizzuti, “A multiobjective genetic algorithm to find communities in complex networks,” IEEE Trans. Evol. Comput., vol. 16, no. 2, pp. 418–430, Jun. 2012.

[44] M. Li, S. Lu, L. Zhang, Y. Zhang, and B. Zhang, “A community detection method for social network based on community embedding,” IEEE Trans. Computat. Social Syst., vol. 8, no. 2, pp. 308–318, Apr. 2021.

[45] J. Kim, J. Lee, B. S. Lee, and J. Liu, “Geo-social co-clustering: A novel framework for geo-social community detection,” ACM Trans. Intell. Syst. Technol., vol. 11, no. 4, pp. 45:1–45:26, 2020.

[46] X. Li, K. He, K. Kloster, D. Bindel, and J. E. Hopcroft, “Local spectral clustering for overlapping community detection,” ACM Trans. Knowl. Discovery From Data, vol. 12, no. 2, pp. 17:1–17:27, 2018.

[47] L. Ni, W. Luo, W. Zhu, and B. Hua, “Local overlapping community detection,” ACM Trans. Knowl. Discovery From Data, vol. 14, no. 1, pp. 3:1–3:25, 2020.

[48] C. Lyu, Y. Shi, and L. Sun, “A novel local community detection method using evolutionary computation,” IEEE Trans. Cybern., vol. 51, no. 6, pp. 3348–3360, Jun. 2021.

[49] W. Cheng, C. Wang, X. Zhang, and Y. Yang, “A local-neighborhood information based overlapping community detection algorithm for large-scale complex networks,” IEEE/ACM Trans. Netw., vol. 29, no. 2, pp. 543–556, Apr. 2021.

[50] K. Wang, W. Zhang, X. Lin, Y. Zhang, L. Qin, and Y. Zhang, “Efficient and effective community search on large-scale bipartite graphs,” in Proc. IEEE 37th Int. Conf. Data Eng. (ICDE), Apr. 2021, pp. 85–96.

[51] A. Miyauchi and Y. Kawase, “What is a network community?: A novel quality function and detection algorithms,” in Proc. 24th ACM Int. Conf. Inf. Knowl. Manage., Oct. 2015, pp. 1471–1480.

[52] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly, “Metrics for community analysis: A survey,” ACM Comput. Surv., vol. 50, no. 4, pp. 1–37, 2017.

[53] J. Yang, A. Chakrabarti, and Y. Vorobeychik, “Protecting geolocation privacy of photo collections,” in Proc. 34th AAAI Conf. Artif. Intell. New York, NY, USA: AAAI Press, 2020, pp. 524–531.

[54] Hefei Xu received the B.S. degree from the Anhui University of Finance and Economics, Bengbu, China, in 2020. He is currently pursuing the M.Sc. degree with the School of Computer Science and Technology, Anhui University, Hefei, China. His research interests include machine learning and data mining.

[55] Wenjian Luo (Senior Member, IEEE) received the B.S. and Ph.D. degrees from the Department of Computer Science and Technology, University of Science and Technology of China, Hefei, China, in 1998 and 2003, respectively.

[56] Yiwen Zhang received the Ph.D. degree in management science and engineering from the Hefei University of Technology, Hefei, Anhui, China, in 2013.

[57] Dr. Luo is a Senior Member of Association for Computing Machinery (ACM) and China Computer Federation (CCF). He has been a member of the organizational team of more than ten academic conferences, in various functions, such as the program chair, the symposium chair, and the publicity chair. He serves as the Chair for the IEEE CIS ECTC Task Force on Artificial Immune Systems. He also serves as an Associate Editor or an Editorial Board Member for several journals, including Swarm and Evolutionary Computation Journal, Journal of Information Security and Applications, Applied Soft Computing Journal, and Complex & Intelligent Systems journal.

[58] Li Ni received the B.E. degree from Anhui University, Hefei, China, in 2015, and the Ph.D. degree from the School of Computer Science and Technology of China, Hefei, in 2020.

[59] She is currently a Lecturer with the School of Computer Science and Technology, Anhui University. Her research interests include machine learning and data mining.