Density-based Community Detection in Geo-Social Networks

Kai Yao, Dimitris Papadias
kyaoab@ust.hk,dimitris@ust.hk
Dept. of Computer Science and Engineering
The Hong Kong University of Science and Technology
Clearwater Bay, Kowloon
Hong Kong

Spiridon Bakiras
sbakiras@hbku.edu.qa
Division of Information and Computing Technology
College of Science and Engineering
Hamad Bin Khalifa University
Doha, Qatar

ABSTRACT

We propose a density-based model to detect communities of users in geo-social networks that are both socially and spatially cohesive. After formally defining the model and the geo-social distance measure it relies on, we present an algorithm that correctly identifies the underlying communities. We assess the effectiveness of our method using novel quantitative measures on the quality of the discovered communities. We also perform a visual evaluation of the discovered communities, using both real and synthetic datasets. Our results show that the proposed model produces geo-social communities with strong social and spatial cohesiveness, which can not be captured by existing graph or spatial clustering methods.

CCS CONCEPTS

• **Information systems** → **Location based services**; • **Human-centered computing** → **Social networks**; *Social network analysis.*

KEYWORDS

geo-social networks, community detection, graph clustering

ACM Reference Format:
Kai Yao, Dimitris Papadias and Spiridon Bakiras. 2019. Density-based Community Detection in Geo-Social Networks. In *Proceedings of SSTD '19: International Symposium on Spatial and Temporal Databases (SSTD '19).* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Community detection in networks aims at identifying groups of nodes which are inter-connected with edges of high density, compared to connections among nodes in different groups [8], [15]. Assume for instance the social graph of Figure 1. Two natural communities are $C_1 = \{v_1, v_3, v_4, v_5\}$ and $C_2 = \{v_7, v_8, v_9, v_{10}\}$. Recently, community detection has received considerable attention in social networks for advertising and marketing purposes. The dense connections among users in the same community could magnify "word of mouth" effects and facilitate the spread of promotions, news, etc.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
SSTD '19, August 19–21, 2019, Vienna, Austria
© 2019 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Most of the existing work focuses exclusively on the social connections. However, the emergence of geo-social networks (GeoSNs) motivates the integration of location information in community detection. In this context, a community contains a group of users that are tightly connected socially and are situated in the same geographic area. Continuing the example of Figure 1, v_5 is far from the other users in C_1. Accordingly, an alternative community $C_1 = \{v_1, v_3, v_4\}$ may be more suitable for location-based services. Specifically, Density-based Geo-Community Detection (DGCD) aims at identifying groups of users that have high social and spatial density. DGCD has several interesting applications.

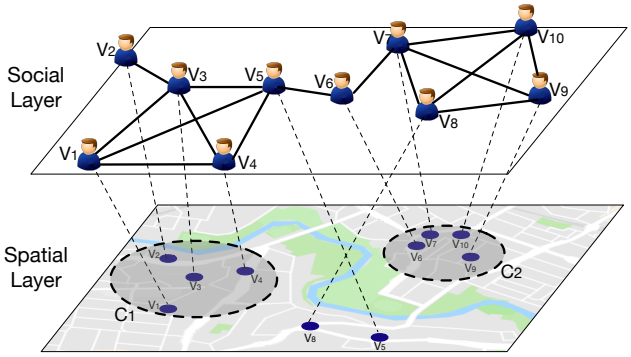


Figure 1: Example of geo-social network

- **Event recommendation.** Online platforms, such as Meetup[1], Eventbrite[2] and Plancast[3], allow social network users to meet physically for various purposes (e.g., events, activities). DGCD could detect potential groups of users that are spatially and socially close, and then recommend events in their vicinity. Intuitively, people who have relatively tight social relations are more likely to participate in a nearby event as a group.
- **Spatial outsourcing.** Given a set of spatial tasks, one needs to distribute them to a set of workers. In order to minimize the coordination cost, the workers should be well-connected according to the underlying social network. Furthermore, all workers should be located near the spatial task. Intuitively, each task could be assigned to the closest community of users that contains enough members to complete the task.

[1] https://www.meetup.com/
[2] https://www.eventbrite.hk/
[3] http://www.plancast.co.uk/

- **Geo-social data analysis.** With the popularity of geo-social networks, such as Facebook and Twitter, it is important to classify users based on their social connections, tags, geographic locations and time stamps. DGCD can be employed to provide concrete geo-social context, e.g., by detecting socially dense communities in a geographic area. Another common data analysis task is to study features about geographic regions. As discussed in [5], these features are often related to the people located there and their interactions.

Previous work on DGCD includes the work by Chen et al. [3], which proposes a geo-distance-based method to discover communities that are densely connected and spatially clustered. However, as shown in Section 5, the communities detected by their approach usually span large geographic areas. In a different study, Fang et al. [7] perform *community search* on geo-social networks. Given a geo-social graph G and a vertex $q \in G$, their goal is to find a subgraph of G, called a spatial-aware community (SAC). However, the spatial-aware community must contain the pre-defined query point, which inevitably limits the exploration of geo-social networks.

In this paper, we address the limitations of previous work, and introduce a novel DGCD model that considers the users' social connections and Euclidean distance. Our model detects all communities in a geo-social network that are socially and spatially cohesive, and outperforms existing approaches in terms of solution quality. Our contributions are summarized as follows:

- We formulate the problem of density-based geo-community detection in geo-social networks.
- We define an effective geo-social distance measure between users in geo-social networks.
- We design an efficient algorithm for DGCD, and analyze its computational complexity.
- We demonstrate the effectiveness of DGCD by case studies and quantitative evaluation.

The remainder of the paper is organized as follows. Section 2 provides a comprehensive survey of existing work that is relevant to our work. Section 3 formally defines the Density-based Geo-Community Detection problem, and Section 4 presents the corresponding DGCD algorithm. Section 5 describes our experimental results, and Section 6 concludes our work.

2 RELATED WORK

Graph Clustering. Community detection in networks and, more generally, graph clustering, is a widely studied problem. Numerous approaches [8, 15, 16, 20, 24, 25] employ *global* or *local* clustering methods, generating groups that are either *flat* or *hierarchical*. The underlying algorithmic solutions are very diverse, including minimum cuts, maximum flows, spectral methods, Markov chains, and random walks. In this work, we focus on global clustering that outputs flat groups of users, such that the density of edges among users in the same group is much higher than the density of edges connecting users in different groups.

Most relevant to our work is the SCAN algorithm [24], whose premise is that densely connected adjacent nodes should be placed in the same cluster. Specifically, SCAN uses the neighborhoods of the vertices as clustering criteria, and defines the structural similarity $\sigma(u, v)$ between vertices u and v as the number of common neighbors normalized by the geometric mean of their degrees. Given parameters ϵ and μ, vertices u and v are structurally similar to each other, if $\sigma(u, v) \geq \epsilon$. Furthermore, u is considered a core vertex if it has at least μ neighbors that are structurally similar to it. Clusters form around core vertices by merging all vertices that are structurally similar to any core vertex in the cluster. SCAN++[19] and pSCAN[2] are variants that increase the efficiency of the basic SCAN algorithm. The main idea behind SCAN++ is that a vertex and its two-hop-away vertices have very similar neighborhoods, due to the high clustering coefficients observed in real-world graphs. Therefore, SCAN++ avoids the computation of the structural similarity between vertices that are shared between the neighbors of a vertex and its two-hop-away vertices. On the other hand, to reduce the number of similarity computations, pSCAN maintains upper and lower bounds for the number neighbors that are similar to each vertex. As such, pSCAN is significantly faster than previous methods. Nevertheless, none of the above algorithms take into account the spatial features of the vertices.

Density-based Spatial Clustering. There is also considerable work on density-based clustering of objects in space [1, 6, 10, 12]. DBSCAN [6] discovers clusters of arbitrary shapes and sizes, where objects in dense regions are grouped into clusters and objects in sparse regions are labeled as outliers. The DBSCAN model finds the spatial ϵ-neighborhood of each point p in the dataset, which is a circular region centered at p with radius ϵ. If the ϵ-neighborhood of p contains at least $MinPts$ points, p is considered a core point. Dense ϵ-neighborhoods are merged into one cluster, if they share at least one core point. This cluster is then expanded, until it retrieves all points that are density-reachable from p. Points that are not reachable by any other point in a cluster are marked as outliers.

OPTICS [1] is an extension of DBSCAN, which generates an augmented ordering of the dataset that captures its density-based clustering structure at different granularities. Furthermore, Jahirabadkar and Kulkarni [11] propose an adaptive algorithm that automatically determines the parameter ϵ of DBSCAN. GDBSCAN [17] is a generalization of DBSCAN that clusters point objects as well as spatially extended objects according to both spatial and non-spatial attributes. A-DBSCAN [14] is an anytime density-based clustering algorithm which is applicable to various data types, such as trajectory and medical data. Finally, DENCLUE [10] clusters based on analytical models of the overall point density, which is computed as the sum of influence functions of the data points.

Other Related Work. Chen et al. [3] study the community detection problem in geo-social networks, and propose a geo-distance-based method for detecting communities in spatially constrained networks. Their goal is to identify communities that are both highly topologically connected and spatially clustered. Gennip et al. [21] use spectral clustering to identify clusters in the graph, corresponding to communities in Hollenbeck. Fang et al. [7] perform community search on geo-social networks. Given a spatial graph G and a vertex $q \in G$, their method returns a subgraph of G, called a spatial-aware community (SAC). Nevertheless, the spatial-aware community must contain the pre-defined query point, which limits the exploration of geo-social networks, and could potentially miss some interesting communities hidden in the GeoSN. Recently, Shi et al. [18, 22] study the problem of Density-based Clustering Places

in Geo-Social Networks (DCPGS) that detects geo-social clusters in GeoSNs. DCPGS extends DBSCAN by replacing the Euclidean distance threshold ϵ with a measure that considers both the spatial and the social distances between places. Different from DCPGS, our model detects clusters of users instead of places.

3 MODEL AND DEFINITIONS

In this section, we propose the density-based geo-community detection model and formally define the DGCD problem.

3.1 DGCD model

A geo-social network graph $G(V, E)$ is an undirected graph with vertex set V and edge set E, where vertices represent users and edges denote their relationships. (Table 1 summarizes the notations used in this paper.) Furthermore, each vertex $v \in V$ has a spatial location $(v.x, v.y)$ in the two-dimensional space.

Table 1: Basic notations

Notation	Definition
$G(V, E)$	a graph with vertex set V and edge set E
$N(v)$	the neighbor set of vertex v in G
$N_s(v)$	the social neighborhood of vertex v in G
$N_{gs}(v)$	the geo-social neighborhood of vertex v in G
$\sigma(u, v)$	the geo-social similarity of vertex u and v
$N_\epsilon(v)$	the ϵ-geo-social neighborhood of v

Our Density-based Geo-Community Detection (DGCD) model is motivated by the notions of *density* [6] and *structural graph clustering* [23]. The DBSCAN model [6] finds the spatial ϵ-neighborhood of each point p in the dataset, which is a circular region centered at p with radius ϵ. If the ϵ-neighborhood of p is dense, i.e., it contains at least $MinPts$ points, p is labeled a *core* point. Dense ϵ-neighborhoods are merged into one community, if they contain the cores points of each other. Structural graph clustering [23] uses the neighborhoods of the vertices as clustering criteria, and defines the *structural similarity* $\sigma(u, v)$ between vertices u and v as the number of common neighbors normalized by the geometric mean of their degrees. Given parameters ϵ and μ, vertices u and v are structurally similar to each other if $\sigma(u, v) \geq \epsilon$. In addition, if u has at least μ structurally similar neighbors, it is marked as a core vertex. Then, clusters grow from core vertices by adding all vertices that are structurally similar to any core vertex in the cluster. Finally, a vertex that does not belong to any cluster is considered a *hub*, if its neighbors belong to two or more clusters, or an *outlier*, otherwise.

A straightforward approach to detect geo-communities, is to come up with a similarity metric that combines the Euclidean distance and social relationship between two users. Let $d_E(u, v)$ be the Euclidean distance and $d_S(u, v)$ be the social distance between users u and v. Then, their geo-social distance is defined as $d_{gs}(u, v) = \omega \cdot d_E(u, v) + (1 - \omega) \cdot d_S(u, v)$, i.e., the weighted average of the two metrics. If we replace the Euclidean distance in the DBSCAN algorithm with $d_{gs}(u, v)$, the geo-social ϵ-neighborhood $N_\epsilon(u)$ of user u includes all users v such that $d_{gs}(u, v) \leq \epsilon$. If the geo-social ϵ-neighborhood of u contains at least $MinPts$ users, then u is regarded as a core user; in this case, u and all other users in its geo-social ϵ-neighborhood, form a community $C(u)$. Furthermore, if another core user v belongs to community $C(u)$, then the communities defined by u and v are merged. After identifying all core users and merging the corresponding communities, the algorithm ends up with a set of communities and a set of outliers (i.e., users that do not belong to the geo-social ϵ-neighborhood of any core user). Nevertheless, one drawback of this approach is that the parameter ω is hard to determine. In addition, it has a high computational cost, because it must compute the Euclidean distance and social relationship strength between all pairs of users. To address these shortcomings, we propose the DGCD model that is based on *structural graph clustering* [23]. DGCD is the product of the following observations.

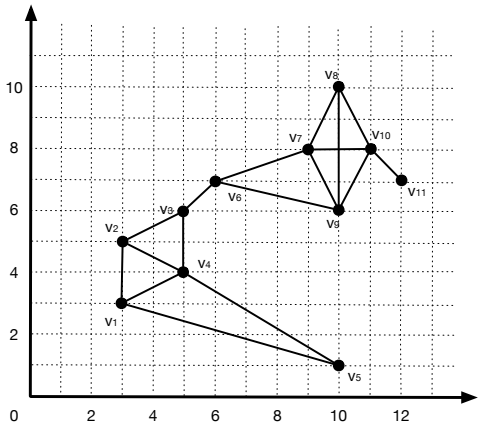


Figure 2: Example of geo-social network

Observation 1. *Given a geo-social network, user pairs with high social strength are more likely to be spatially close to each other.*

Observation 2. *Inside a community, two members normally have many common friends, i.e., common neighbors. Therefore, the neighborhood around two friends is an important factor, which can reflect the stability and cohesiveness of their friendship.*

Observation 3. *A community normally consists of several high-profile users and marginal users. High-profile users play an important role in connecting members to form this community, and they are core users. Marginal users typically have less friends, and they are involved in some communities due to their high-profile friends.*

For each user u in the GeoSN, DGCD finds the geo-social ϵ-neighborhood $N_\epsilon(u)$ of u, which includes all neighbors that are (i) spatially close to u, i.e., their Euclidean distance is less than γ, and (ii) socially close to u, i.e., their structural similarity is larger than ϵ. Note that Constraint (i) is based on Observation 1, while Constraint (ii) is based on Observation 2. In the following, for ease of presentation, we use the terms vertex and user interchangeably.

Definition 3.1. (Social Neighborhood). The social neighborhood of a user u, denoted by $N_s(u)$, is defined as the closed neighborhood [9] of u; that is $N_s(u) = \{v \in V | (u, v) \in E\} \cup \{u\}$.

In this work, we focus on the social (i.e., closed) neighborhood of a vertex; note that the open neighborhood [9] of u is $N(u) = \{v \in$

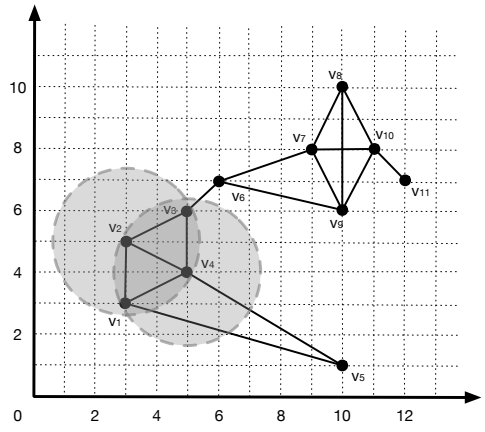


Figure 3: Example of geo-social neighborhood

$V|(u,v) \in E\}\backslash\{u\}$. The degree of u, denoted by $d(u)$, is the cardinality of $N_s(u)$ (i.e., $d(u) = |N_s(u)|$). For example, the social neighborhood of vertex v_4 in Figure 2 is $N_s(v_4) = \{v_1, v_2, v_3, v_4, v_5\}$, its degree is $d(v_4) = |N_s(v_4)| = 5$, and its open neighborhood is $N(v_4) = \{v_1, v_2, v_3, v_5\}$.

Definition 3.2. (Geo-Social Neighborhood). The geo-social neighborhood of a vertex u, denoted by $N_{gs}(u)$, is defined as the set of neighbors of u that are within radius γ from u, including user u; that is, $N_{gs}(u) = \{v \in V|(u,v) \in E \wedge d_E(u,v) \leq \gamma\} \cup \{u\}$, where $d_E(u,v)$ is the Euclidean distance between u and v.

Intuitively, u's geo-social neighborhood includes his nearby friends. In practice, γ is an input parameter and can be varied accordingly. As an example, in Figure 3, suppose $\gamma = \sqrt{5}$. Then, the geo-social neighborhood of v_4 is $\{v_1, v_2, v_3, v_4\}$. Note that v_5 is not in $N_{gs}(v_4)$, since their distance is larger than γ.

Definition 3.3. (Geo-Social Similarity). The geo-social similarity between two users u and v, denoted by $\sigma(u,v)$, is defined as the number of common vertices in $N_{gs}(u)$ and $N_{gs}(v)$, normalized by the geometric mean of their cardinalities; that is, $\sigma(u,v) = \frac{|N_{gs}(u) \cap N_{gs}(v)|}{\sqrt{|N_{gs}(u)| \cdot |N_{gs}(v)|}}$.

Intuitively, the geo-social similarity between two vertices increases with the number of common vertices in their geo-social neighborhoods. Note that, the geo-social similarity value ranges between 0 and 1; that is, $0 \leq \sigma(u,v) \leq 1, \forall u, v \in V$. As shown in Figure 3, when $\gamma = \sqrt{5}$, $N_{gs}(v_2) = \{v_1, v_2, v_3, v_4\}$ and $N_{gs}(v_4) = \{v_1, v_2, v_3, v_4\}$. Thus, $\sigma(v_2, v_4) = \frac{|\{v_1, v_2, v_3, v_4\}|}{\sqrt{4 \cdot 4}} = 1$.

Definition 3.4. (ϵ-Neighborhood). The ϵ-neighborhood of a user u, $N_\epsilon(u)$, includes the set of users in the geo-social neighborhood of u that have geo-social similarity larger than or equal to ϵ: $N_\epsilon(u) = \{v \in N_{gs}(u) | \sigma(u,v) \geq \epsilon\}$.

Definition 3.5. (Core User). A user $u \in V$, is called a core user w.r.t. γ and ϵ, if his ϵ-neighborhood contains at least μ users.

In Figure 3, assume $\gamma = \sqrt{5}$, $\epsilon = 0.7$ and $\mu = 4$. Recall that $\sigma(v_2, v_4) = 1$ and , thus, v_2 is in the ϵ-neighborhood of v_4. Similarly, we can compute $\sigma(v_1, v_4) = 0.86$ and $\sigma(v_3, v_4) = 0.7$. It is easy to see that the ϵ-neighborhood of v_4 is $\{v_1, v_2, v_3, v_4\}$. Therefore, v_4 is a core user, since $|N_\epsilon(v_4)| = 4$. Note that v_5 is not in the ϵ-neighborhood of v_4, because they are far away from each other.

Definition 3.6. (Direct Geo-Social Reachability). A user $v \in V$ is directly geo-socially reachable from a user $u \in V$ if (i) v belongs to the ϵ-neighborhood of u and (ii) u is a core user.

Direct geo-social reachability is symmetric for pairs of core users, but it is asymmetric if either user is not core.

Definition 3.7. (Geo-Social Reachability). A user $u \in V$ is geo-socially reachable from a user $v \in V$ if there is a chain of users, $V' = \{v_1, \ldots, v_n\} \subseteq V$, where $v_1 = v$ and $v_n = u$, s.t. v_i is directly geo-socially reachable from $v_{i-1}, \forall v_i \in V'$.

Geo-social reachability is transitive but asymmetric. It is only symmetric for a pair of core users.

Definition 3.8. (Geo-Social Connectivity). A user $v \in V$ is geo-socially connected to a user $u \in V$, if there is a user $w \in V$, s.t. both v and u are geo-socially reachable from w.

Definition 3.9. (Geo-Socially Connected Cluster). A non-empty subset $C_i \subseteq V, i \in \mathbb{N}$ is a geo-socially connected cluster w.r.t. γ, ε and μ, if all users in C_i are geo-socially connected and C_i is maximal w.r.t. the geo-social reachability.

Definition 3.10. (Geo-Social Clustering). A clustering $\mathbb{C}$, of a geo-social network G, w.r.t. γ, ϵ and μ, includes every geo-socially connected cluster in G, w.r.t. γ, ϵ and μ.

In Figure 3, assume $\gamma = \sqrt{5}$, $\epsilon = 0.7$ and $\mu = 4$. As shown previously, the ϵ-neighborhood of v_4 is $\{v_1, v_2, v_3, v_4\}$ and v_4 is a core user. Thus, v_1, v_2 and v_3 are direct geo-socially reachable from v_4, and v_1 is geo-socially reachable from v_2, since v_1 can reach v_2 via v_4. According to Definition 3.10, the set $\{v_1, v_2, v_3, v_4\}$ forms a new community.

Definition 3.11. (Borderline-User). A user $v \in V$ is a borderline user, if (i) he belongs to at least one cluster and (ii) he is not a core user of any cluster.

Note that a borderline user can belong to more than one clusters at the same time. In fact, if a user appears in more than one clusters, he is regarded as a borderline user.

Definition 3.12. (Outlier). A user $v \in V$ is an outlier, if v does not belong to any cluster.

Given the above definitions, any geo-social graph can yield geo-socially connected clusters by executing the steps below. The correctness of this algorithm is implied by the following two lemmas.

(1) Choose a random core user v.
(2) Find all the geo-socially reachable users starting from v and create a cluster.

Lemma 3.13. *If v is a core user, then the set of users that are geo-socially reachable from v constitute a geo-socially connected cluster $C \subseteq V$.*

Proof. First, we prove that C is a non-empty cluster: v, as a core user, is geo-socially reachable from himself; therefore, he belongs

to cluster C. Next, we show that C is maximal. Let us consider two users, $u \in C$ and $w \in V$, where w is geo-socially reachable from u. User u belongs to C, so he is geo-socially reachable from the core user v. Since geo-social reachability is transitive, w is geo-socially reachable from the core user v, which implies that w is also part of cluster C. Finally, for the connectivity, it holds that, for each pair of users $u \in C$ and $w \in C$, both u and w are geo-socially reachable from the core v, which in turn suggests that u and w are geo-socially connected via v. □

Lemma 3.14. *If $C \subseteq V$ is a geo-socially connected cluster w.r.t. γ, ϵ and μ, and $v \in C$ is a core user, then the users that are geo-socially reachable from v constitute C.*

Proof. Let C' be the set of users that are geo-socially reachable from v. We need to show that $C' = C$. First, it holds that $C' \subseteq C$, since $v \in C'$. Consider a user $u \in C$. Given that C is a geo-socially connected cluster w.r.t. γ, ϵ and μ, there exists a core user $w \in C$, such that (i) w is geo-socially reachable from v and (ii) u is geo-socially reachable from w. This implies that v is geo-socially reachable from w, because geo-social reachability is symmetric for core users. Consequently, u is also geo-socially reachable from v, due to the transitive property of geo-social reachability. As a result, $u \in C'$ and, therefore, $C' = C$. □

4 THE DGCD ALGORITHM

In this section, we introduce our DGCD algorithm that correctly identifies all communities in a geo-social network. We also analyze its computational complexity.

4.1 Algorithm description

The DGCD algorithm is a direct extension of SCAN [23]. First, for each user v in the GeoSN, it computes v's geo-social neighborhood, $N_{gs}(v)$, and checks the geo-social similarity with all its geo-social neighbors. Then, it computes the ϵ-neighborhood of v, $N_\epsilon(v)$, and if $|N_\epsilon(v)| \geq \mu$, v is labeled as a core user and a new community is initialized; otherwise, DGCD moves to the next *unprocessed* user, until all users are processed. For the sake of efficiency, the social network is stored in a hash table. Specifically, each pair of friends in the social network is an entry in the hash table, such that relationship queries incur constant cost.

Algorithm 1 shows the pseudo code of DGCD. Initially, all users in the GeoSN are marked as unprocessed (line 1). Queue Q is used to store potential candidates for the current community, while the identity *cid* of the current community is initialized to 1 (line 2). Then, for each *unprocessed* user v_i, function ϵ-$NEIGHBOR(G, v_i, \gamma, \epsilon)$ is invoked to obtain its ϵ-neighborhood, $N_\epsilon(v_i)$. If $N_\epsilon(v_i)$ contains at least μ users, v_i is labeled as core. Next, all users in $N_\epsilon(v_i)$ are assigned the same community id as v_i (lines 7-8), while all unprocessed users v_j in $N_\epsilon(v_i)$ are inserted into Q for later processing (lines 9-10). Lines 11-19 expand the current community as much as possible, by checking the unprocessed users in Q. When Q becomes empty, the current community can not be expanded any further, so the algorithm increments *cid* and continues its search for the next community (line 20). Note that, Algorithm 1 only outputs the set of communities in G. Nevertheless, it can be easily extended to detect borderline (hub) users and outliers, by traversing the entire graph G in $O(|V| + |E|)$ time.

Algorithm 1 : DGCD (G, γ, ϵ, μ)

```
1:  label all v ∈ V as unprocessed
2:  Q = ∅, H = ∅, cid = 1
3:  for all unprocessed users v_i do
4:      N_ε(v_i) = ε-NEIGHBOR(G, v_i, γ, ε, H)
5:      if |N_ε(v_i)| ≥ μ then
6:          assign cid to v_i
7:          for all user v_j ∈ N_ε(v_i) do
8:              assign cid to v_j
9:              if v_j is unprocessed then
10:                 Q.insert(v_j)
11:     while Q ≠ ∅ do
12:         v_k = Q.pop()
13:         if v_k is unprocessed then
14:             N_ε(v_k) = ε-NEIGHBOR(G, v_k, γ, ε, H)
15:             if |N_ε(v_k)| ≥ μ then
16:                 for all user v_m ∈ N_ε(v_k) do
17:                     assign cid to v_m
18:                     if v_m is unprocessed then
19:                         Q.insert(v_m)
20:     cid++
```

The pseudo code of function ϵ-$NEIGHBOR(G, v_i, \gamma, \epsilon)$ is listed in Algorithm 2. S is an initially empty set that is used to store the ϵ-neighborhood of user v_i. First, the algorithm computes the geo-social neighborhood of v_i, $N_{gs}(v_i)$, which consists of all v_i's neighbors within distance γ. Then, for each user v_j in $N_{gs}(v_i)$, the algorithm computes its geo-social neighborhood $N_{gs}(v_j)$. From the two neighborhood sets, the algorithm evaluates their geo-social similarity, according to Definition 3.3. Finally, if their similarity is larger than ϵ, v_j is inserted into the result set S.

Algorithm 2 : ϵ-$NEIGHBOR(G, v_i, \gamma, \epsilon, H)$

```
1:  S = ∅
2:  compute N_gs(v_i)
3:  for all user v_j in N_gs(v_i) do
4:      if H.exist(v_i, v_j) then
5:          if H[(v_i, v_j)] is true then
6:              S.insert(v_j)
7:      else
8:          compute N_gs(v_j)
9:          compute σ(v_i, v_j) = |N_gs(v_i) ∩ N_gs(v_j)| / sqrt(|N_gs(v_i)| · |N_gs(v_j)|)
10:         if σ(v_i, v_j) ≥ ε then
11:             S.insert(v_j)
12:             H[(v_i, v_j)] = true
13: return S
```

Recall that, geo-social similarity is symmetric. Hence, for a given user v_i, if user v_j is (not) in $N_\epsilon(v_i)$, then v_i is also (not) in $N_\epsilon(v_j)$, and vice versa. Therefore, we use a hash table H to store a boolean value indicating whether a pair of users (v_i, v_j) belong in each other's ϵ-neighborhood (once their geo-social similarity has been

Table 2: Algorithm 1 trace

processing	unprocessed	Q	clusters
v_1	$\{v_2, v_3, \ldots, v_{11}\}$	$\{v_2, v_4\}$	$C_1 = \{v_1, v_2, v_4\}$
v_2	$\{v_3, v_4, \ldots, v_{11}\}$	$\{v_3, v_4\}$	$C_1 = \{v_1, v_2, v_3, v_4\}$
v_3	$\{v_4, v_5, \ldots, v_{11}\}$	$\{v_4, v_6\}$	$C_1 = \{v_1, v_2, v_3, v_4, v_6\}$
v_4	$\{v_5, v_6, \ldots, v_{11}\}$	$\{v_6\}$	$C_1 = \{v_1, v_2, v_3, v_4, v_6\}$
v_5	$\{v_6, v_7, \ldots, v_{11}\}$	$\{v_6\}$	$C_1 = \{v_1, v_2, v_3, v_4, v_6\}$
v_6	$\{v_7, v_8, \ldots, v_{11}\}$	$\emptyset$	$C_1 = \{v_1, v_2, v_3, v_4, v_6\}$
v_7	$\{v_8, v_9, \ldots, v_{11}\}$	$\{v_8, v_9, v_{10}\}$	$C_1 = \{v_1, v_2, v_3, v_4, v_6\}$ $C_2 = \{v_7, v_8, v_9, v_{10}\}$
v_8	$\{v_9, v_{10}, v_{11}\}$	$\{v_9, v_{10}\}$	$C_1 = \{v_1, v_2, v_3, v_4, v_6\}$ $C_2 = \{v_7, v_8, v_9, v_{10}\}$
v_9	$\{v_{10}, v_{11}\}$	$\{v_{10}\}$	$C_1 = \{v_1, v_2, v_3, v_4, v_6\}$ $C_2 = \{v_7, v_8, v_9, v_{10}\}$
v_{10}	$\{v_{11}\}$	$\emptyset$	$C_1 = \{v_1, v_2, v_3, v_4, v_6\}$ $C_2 = \{v_7, v_8, v_9, v_{10}\}$
v_{11}	$\emptyset$	$\emptyset$	$C_1 = \{v_1, v_2, v_3, v_4, v_6\}$ $C_2 = \{v_7, v_8, v_9, v_{10}\}$

computed). In this way, we avoid computing the similarity between the same pair of users twice.

To illustrate the operation of Algorithm 1, consider the example of Figure 4, where $\gamma = \sqrt{5}$, $\epsilon = 0.7$ and $\mu = 3$. The labels on the edges represent the geo-social similarities of the corresponding vertices. Initially, all users are labeled as unprocessed, and assume that the algorithm processes users in the order $v_1, v_2, \ldots, v_{11}$. The first step is to compute the geo-social neighborhood of v_1 as $N_{gs}(v_1) = \{v_1, v_2, v_4\}$. Note that $N_{gs}(v_1)$ does not include v_5, because its Euclidean distance to v_1 is larger than $\sqrt{5}$. Then, since v_2 and v_4 are geo-social similar to v_1, we obtain the ϵ-neighborhood of v_1 as $N_\epsilon(v_1) = \{v_1, v_2, v_4\}$. The cardinality of $N_\epsilon(v_1)$ is 3, so v_1 is regarded as a core user and a new community C_1 is generated. Next, we insert v_2 and v_4 into Q for further processing and, suppose, v_2 is popped from the queue first. We compute $N_\epsilon(v_2) = \{v_1, v_2, v_3, v_4\}$, which makes v_2 a core user, thus assigning v_3 into the current community C_1, i.e., $C_1 = \{v_1, v_2, v_3, v_4\}$. User v_3 is also inserted into Q, whose current members are $\{v_4, v_3\}$. We continue expanding community C_1 in a similar manner until Q becomes empty, which indicates that C_1 is maximal.

The algorithm then increments *cid* and initiates a new search for the next community, by checking the remaining unprocessed users. Specifically, the algorithm will first process user v_7 and compute $N_\epsilon(v_7) = \{v_7, v_8, v_9, v_{10}\}$. The cardinality of $N_\epsilon(v_7)$ is 4, so v_7 is a core user that triggers the formation of a new community C_2. After C_2 is expanded, the algorithm terminates and outputs two communities: $C_1 = \{v_1, v_2, v_3, v_4, v_6\}$ and $C_2 = \{v_7, v_8, v_9, v_{10}\}$. User v_5 is an outlier because he is far away from all his social friends, while v_{11} is an outlier because his geo-social similarity with v_{10} is only 0.63. Table 2 shows a detailed trace of the algorithm, highlighting the contents of the various data structures.

4.2 Algorithm analysis

Given a geo-social network $G = (V, E)$, the time complexity of the DGCD algorithm is $O(\alpha(G) \cdot |E|)$, where $\alpha(G)$ is the arboricity of G. For a graph G, its arboricity equals the minimum number of edge-disjoint forests needed to cover all edges of G, and has an upper bound of $\alpha(G) \leq \sqrt{|E|}$ [4]. Indeed, the basic operation of Algorithm 1 involves the computation of $N_\epsilon(v_i)$ for each $v_i \in V$.

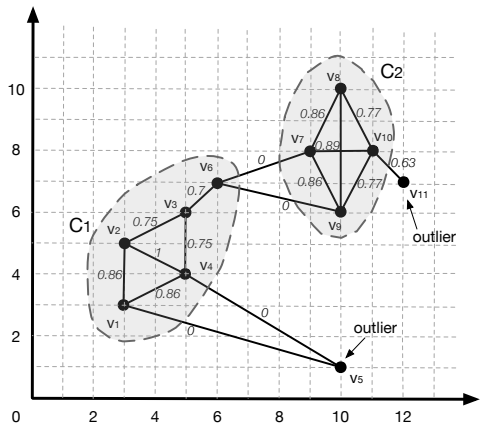


Figure 4: Detected communities

Furthermore, as evident in Algorithm 2, the dominant cost in computing $N_\epsilon(v_i)$ is the evaluation of the geo-social similarity $\sigma(v_i, v_j)$ and, in particular, the set intersection $N_{gs}(v_i) \cap N_{gs}(v_j)$ between the geo-social neighborhoods of v_i and v_j. In the worst case, the overall cost of Algorithm 1 is equal to enumerating all triangles in G. The current best time complexity of triangle enumeration is $O(\alpha(G) \cdot |E|)$ [4], thus bounding the time complexity of our algorithm to $O(|E|^{1.5})$.

5 EXPERIMENTAL EVALUATION

In this section, we experimentally evaluate the performance of our DGCD algorithm. First, we visualize and compare the geo-social communities identified by DGCD and other competitors, using real and synthetic datasets. Then, we introduce several metrics for measuring the social and spatial cohesiveness between users in the discovered geo-social communities, and use them to assess the quality of the solutions produced by the different approaches.

5.1 Setup

We conducted extensive empirical studies to evaluate the effectiveness of our algorithm for density-based geo-community detection. Specifically, we evaluated DGCD against the following methods:

- **DBSCAN:** This is the state-of-the-art algorithm for clustering spatial data [6]. DBSCAN is a density-based clustering algorithm and can discover clusters of arbitrary shapes.
- **SCAN:** This is a structural clustering algorithm for networks [23]. The goal of SCAN is to find clusters, hubs, and outliers in large networks.
- **CNGM:** This approach is based on the fast modularity maximization (CNGM) algorithm [3]. CNGM can identify communities in geo-social networks that are both highly topologically connected and spatially clustered.

We utilized two real datasets in our experiments, namely Dallas and Gowalla[4]. Each vertex in the dataset represents a unique user, and each link represents a social relationship (friendship) between two users. Gowalla is a location-based social networking website, where users share their locations via check-ins. The social network is undirected and was collected using their public API. It consists of 12,748 nodes and 96,838 edges. We collected a total of 6,442,890 check-ins from these users over the period of Feb. 2009 to Oct. 2010, but we only used the first check-in as the user's geographic location. The Dallas dataset is a partial geo-social network in the region (32.63,-97.02) to (32.90,-96.64), extracted from the Gowalla dataset. There are 818 users and 920 relationships, and the degree ranges from 1 to 97.

We also performed experiments with synthetic datasets. We are not aware of any existing spatial graph data generators, so we created our synthetic data as follows. First, we used GTGraph[5], a well-known graph generator, to generate a non-spatial graph (using the default parameter values of GTGraph). The degrees of the vertices follow a power-law distribution, which is often exhibited in social networks. To generate the location of each graph vertex, we first select a random vertex v and assign it to a random position in the $[0, 3000] \times [0, 3000]$ space. Then, we place v's neighbors at random positions, whose distances follow a normal distribution with mean 300 and standard deviation 600. We repeat this step for other vertices, starting from v's neighbors, until every vertex in the graph is associated with a location. We created two spatial graphs of different sizes, namely Syn1 and Syn2. The statistics of all our datasets are summarized in Table 3, where $\hat{d}$ represents the average degree. All algorithms were implemented in Java and the experiments were performed on a 2.2 GHz Intel Core i7 machine with 16GBytes memory, running macOS.

Table 3: Datasets used in our experiments

Type	Name	Vertices	Edges	$\hat{d}$
Real	Dallas	818	920	2.25
	Gowalla	12,748	96,838	15.19
Synthetic	Syn1	200	2,639	26.39
	Syn2	2,000,000	23,398,532	23.39

5.2 Visualization-based analysis

We first visualize and compare the communities discovered by our DGCD algorithm and the three competitors (SCAN, DBSCAN and CNGM) on the Dallas dataset. The visualization of the Dallas geo-social network is shown in Figure 5. Note that Dallas corresponds to a small portion of the Gowalla dataset, in order to facilitate the visualization of the detected communities by the different approaches. In this region, the maximum distance between two users is 11,000 meters. For the first experiment, we set the distance threshold to $\gamma = 1000$, the geo-social similarity threshold to $\epsilon = 0.5$ and the ϵ-neighborhood cardinality threshold to $\mu = 3$.

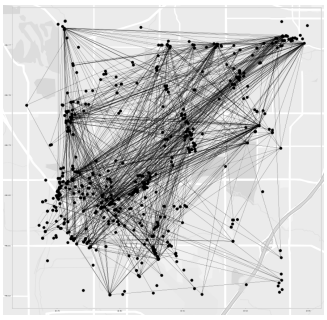


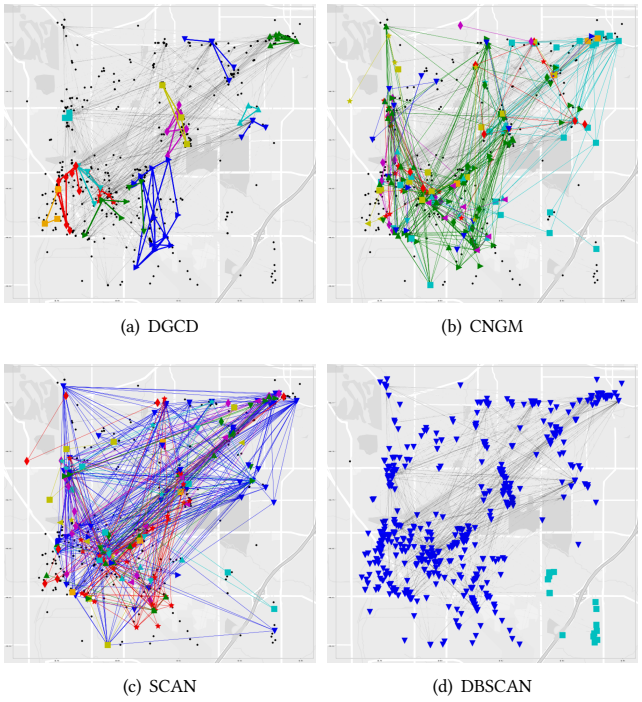
Figure 5: The geo-social network in Dallas

Figure 6 illustrates the communities discovered by DGCD (our model), CNGM (which clusters users according to maximum modularity), DBSCAN (which disregards the social network) and SCAN (which disregards the spatial information). As shown in Figure 6(a), DGCD detected 12 geo-communities, and all of them are both spatially and socially close. These communities are consistent with our density-based model and provide meaningful context to emerging geo-social applications. Consider, for example, the top right corner community, marked with a blue color. It consists of only four users, where each user has strong social connections with the others, and they are all located in close proximity. Identifying such groups is beneficial to targeted advertisements, e.g., a "20% discount on a table for four" offer running at a nearby restaurant. Note that DGCD discovers very few communities compared to the dataset size. This is because the geo-social network of Dallas is sparse and many users turn out to be outliers.

Figure 6(b) shows the clustering output of CNGM. This algorithm begins by labeling every user as a community and then merges them in a bottom-up manner. When a pair of communities is merged, it results in an increase in modularity ΔQ. At every step, the algorithm merges the pair of communities that generates the maximum ΔQ. When the maximum ΔQ becomes negative, the process is stopped and the community structure of the network is revealed. However, as Figure 6(b) shows, this algorithm produces communities whose members are far away from each other, because the terminating condition is hard to decide. Another drawback of CNGM is the expensive computational cost, since, in each iteration, it computes ΔQ for all pairs of users.

The detected communities of SCAN, depicted in Figure 6(c), are spatially sparse. As mentioned previously, this is because the SCAN framework does not take into account the users' geographic locations. Instead, users with more social connections are more likely to be clustered into the same community, even if they are far away from each other. Finally, Figure 6(d) shows the clustering

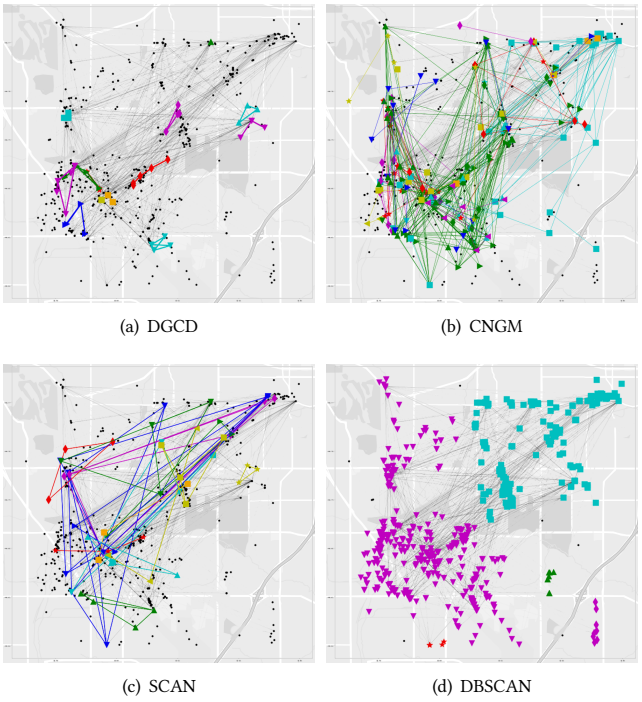
[4] http://snap.stanford.edu/data/index.html
[5] http://www.cse.psu.edu/madduri/software/GTgraph/



(a) DGCD (b) CNGM

(c) SCAN (d) DBSCAN

Figure 6: Visualization on Dallas: $\gamma = 1000, \epsilon = 0.5, \mu = 3$



(a) DGCD (b) CNGM

(c) SCAN (d) DBSCAN

Figure 7: Visualization on Dallas: $\gamma = 700, \epsilon = 0.6, \mu = 3$

output of DBSCAN, which is also not interesting. In particular, DBSCAN produces a very large cluster that dominates all others. This type of clustering is meaningless in geo-social applications, as it lacks social context.

For the next experiment, we strengthen the spatial and social constraints (we set $\gamma = 700$ and $\epsilon = 0.6$) of the community search, which leads to some more interesting observations. As shown in Figure 7(a), the communities detected by the DGCD framework become more compact. That is, for each community, its members are spatially closer to each other and their social relationships are stronger. This has some very practical applications in real life. Consider the context of spatial crowdsourcing, where there are several tasks waiting to be assigned. The DGCD model can be employed to identify uniquely qualified groups willing to undertake these tasks. Intuitively, group members should be spatially close in order to reduce the traveling cost, but at the same time, group members should be familiar with other, which could potentially increase their working efficiency.

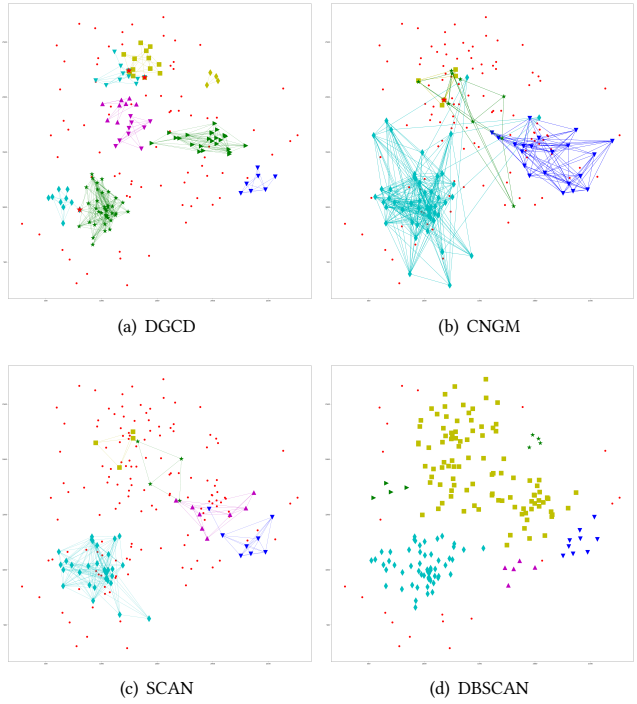
The CNGM algorithm is not sensitive to the γ, ϵ and μ parameters, because its objective function is to maximize modularity. Therefore, the output of CNGM remains unchanged. SCAN detects fewer communities than before, as shown in Figure 7(c), since we tightened the social constraint of the query. As for DBSCAN, Figure 7(d) illustrates that the large dominating community has split into two smaller ones, because of the tighter spatial constraint that we imposed. As a result, DBSCAN outputs more communities than before but, unfortunately, they are still not interesting in the context of geo-social networks.

Note that, the Dallas dataset is very sparse in terms of social connections, having an average degree of 2.25. As such, in the next set of experiments we utilize our synthetic dataset (Syn1) that is considerably more active in terms of social relationships (average degree of 26.39). Figure 8 plots the discovered communities for the four algorithms, using parameters $\gamma = 200$, $\epsilon = 0.7$ and $\mu = 4$. DGCD outputs 9 clusters that are both spatially close and socially cohesive. CNGM's clusters, on the other hand, are spatially sparse, as in the case of the Dallas dataset. This is because the CNGM algorithm tries to assign all users to a community.

Similarly, as shown in Figure 8(c), SCAN's results are spatially sparse. However, they are significantly improved compared to the Dallas dataset, due to the stringent social similarity constraint ($\epsilon = 0.7$). Finally, DBSCAN exhibits very similar behavior as in the Dallas dataset, i.e., it outputs a couple of large clusters that dominate all others, but do not provide any valuable insight to the end-user. Clearly, being a purely spatial-based clustering algorithm, DBSCAN can not identify interesting communities in geo-social networks.

In our final visualization experiment, we weaken the social constraint to $\epsilon = 0.6$, while keeping the other two parameters constant ($\gamma = 200$ and $\mu = 4$). Figure 9 illustrates the discovered communities for the four approaches. DGCD outperforms again all the other algorithms, and identifies 7 communities that are spatially close and socially cohesive. On the other hand, CNGM and DBSCAN are not affected, because they are insensitive to the social parameter ϵ.

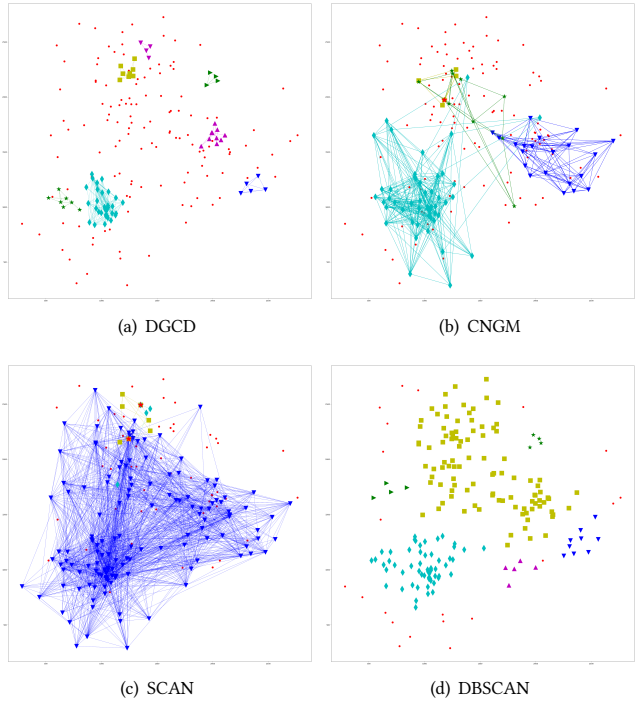
Figure 9(c) shows the most interesting result in this experiment, as the slight weakening of the social constraint in the SCAN algorithm produces a very large community (marked in blue color) that



(a) DGCD (b) CNGM

(c) SCAN (d) DBSCAN

Figure 8: Visualization on Syn1: $\gamma = 200, \epsilon = 0.7, \mu = 4$



(a) DGCD (b) CNGM

(c) SCAN (d) DBSCAN

Figure 9: Visualization on Syn1: $\gamma = 200, \epsilon = 0.6, \mu = 4$

dominates all others. This tells us that SCAN is extremely sensitive to the social constraint ϵ. Furthermore, since SCAN disregards spatial information, an improperly set social constraint can easily produce meaningless results in the context of geo-social networks.

5.3 Social and spatial quality evaluation

In this section, we introduce several metrics for assessing the social and spatial cohesiveness of the discovered geo-social communities. Based on these metrics, we assess the quality of DGCD and its competitors on the two larger datasets, namely Gowalla and Syn2. We also test the quality of the solutions for our DGCD model, using different social similarity metrics to replace $\sigma(u, v)$. In particular, we consider the following two metrics for measuring the similarity of $u, v \in G$, based on their geo-social neighborhoods $N_{gs}(u)$ and $N_{gs}(v)$, respectively:

- **Jaccard similarity:** $J(u, v) = \frac{|N_{gs}(u) \cap N_{gs}(v)|}{|N_{gs}(u) \cup N_{gs}(v)|}$
- **Dice similarity:** $D(u, v) = \frac{2|N_{gs}(u) \cap N_{gs}(v)|}{|N_{gs}(u) + N_{gs}(v)|}$

To quantify the social cohesiveness of a community, we adopt two of the eight network community multi-criterion scores listed in [13]: *internal density* and *conductance*. The other six criteria produced similar results, so we omit them in our study. Given a geo-social network, let C be the set of detected communities. For a given cluster $c_i \in C$, its internal density $f_d(c_i)$ is defined as

$$f_d(c_i) = 1 - \frac{m_{c_i}}{\frac{|c_i|(|c_i|-1)}{2}}$$

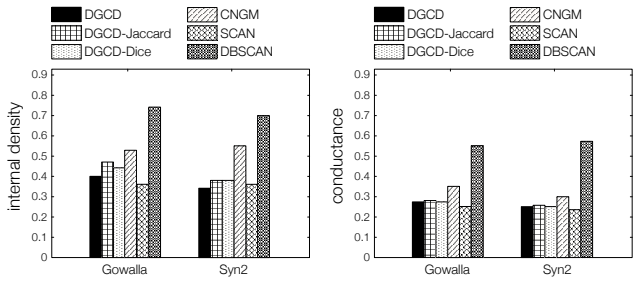
where m_{c_i} is the number of edges in c_i, i.e., $m_{c_i} = |\{(u, v)|u \in c_i, v \in c_i\}|$. On the other hand, its conductance $f_c(c_i)$ is the fraction of edges from c_i's vertices that point outside c_i, that is,

$$f_c(c_i) = \frac{o_{c_i}}{2m_{c_i} + o_{c_i}}$$

where $o_{c_i} = |\{(u, v)|u \in c_i, v \notin c_i\}|$. For both metrics, a lower score indicates better social quality.

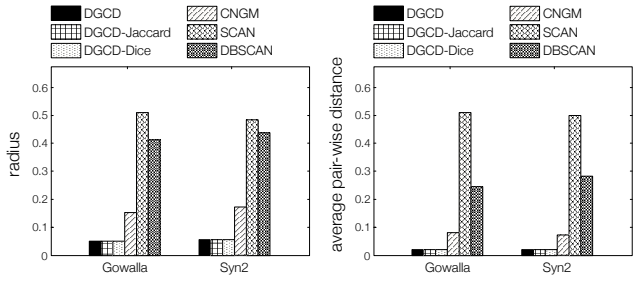
Figure 10 illustrates the community scores for the different approaches. They are computed by averaging the individual scores (either density or conductance) for each discovered community c_i. The query parameters were set as $\gamma = 1000$, $\epsilon = 0.5$ and $\mu = 3$ for the Gowalla dataset, and $\gamma = 200$, $\epsilon = 0.7$ and $\mu = 4$ for the Syn2 dataset. In terms of conductance, SCAN is marginally better than DGCD for both datasets. On the other hand, the internal density of DGCD is slightly better than SCAN for the Syn2 dataset. SCAN's lower community scores are expected, since it clusters users based solely on their social relationships. Nevertheless, DGCD performs almost as well, thus generating communities with high social cohesiveness. Furthermore, DGCD outperforms CNGM under all settings. Naturally, all approaches are superior to DBSCAN, because DBSCAN does not utilize the social graph in its clustering function. Finally, it is worth pointing out that the Jaccard and Dice metrics produced worse results than our default social similarity metric.

To measure the spatial cohesiveness of the detected communities, we introduce two additional metrics: (i) *radius*, which is the average radius of all communities in the final result, and (ii) *aveDist*, which represents the average pair-wise distance among all users inside a community. Figure 11 presents the (normalized) results for the two



(a) Internal density (b) Conductance

Figure 10: Community score evaluation



(a) Radius (b) Average pair-wise distance

Figure 11: Spatial cohesiveness

metrics. DGCD clearly outperforms the other clustering methods by a wide margin. Furthermore, the Jaccard and Dice variants of DGCD produce almost identical results, since they use the same radius constraint γ. DBSCAN is superior to SCAN, but it still results in high values for the radius and aveDist metrics. This is due to the large communities it produces, as evident in our visualization experiments. SCAN has the worst spatial cohesiveness in its communities, because it is based purely on social relationships. CNGM is the closest competitor to DGCD, but its spatial cohesiveness scores are at least 3 times worse than DGCD for both datasets.

6 CONCLUSIONS

In this paper, we studied for the first time the problem of Density-based Geo-Community Detection (DGCD) in geo-social networks. Our model extends the density-based clustering paradigm to consider both the spatial and social relationships between users. We defined a new measure for the geo-social distance between a pair of users that captures well their social strength and also their spatial proximity. Then, we introduced the DGCD algorithm that correctly identifies all communities in a GeoSN, according to a set of spatial and social constraints. We performed an extensive experimental evaluation to assess the effectiveness of the DGCD model, using both real and synthetic datasets. We showed that the DGCD model produces high-quality communities that can not be captured by existing graph or spatial clustering methods. In our future work, we plan to extend the DGCD model to temporal networks.

ACKNOWLEDGEMENTS

This work was supported by grant 16231216 from Hong Kong RGC.

REFERENCES

[1] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. 1999. OPTICS: ordering points to identify the clustering structure. 28, 2 (1999), 49–60.
[2] Lijun Chang, Wei Li, Lu Qin, Wenjie Zhang, and Shiyu Yang. 2017. pSCAN: Fast and Exact Structural Graph Clustering. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 29, 2 (2017), 387–401.
[3] Yu Chen, Jun Xu, and Minzheng Xu. 2015. Finding community structure in spatially constrained complex networks. *International Journal of Geographical Information Science* 29, 6 (2015), 889–911.
[4] Norishige Chiba and Takao Nishizeki. 1985. Arboricity and subgraph listing algorithms. *SIAM J. Comput.* 14, 1 (1985), 210–223.
[5] Yohan Chon, Nicholas D Lane, Fan Li, Hojung Cha, and Feng Zhao. 2012. Automatically characterizing places with opportunistic crowdsensing using smartphones. In *Proc. ACM Conference on Ubiquitous Computing*. 481–490.
[6] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 226–231.
[7] Yixiang Fang, Reynold Cheng, Xiaodong Li, Siqiang Luo, and Jiafeng Hu. 2017. Effective community search over large spatial graphs. *Proc. of the VLDB Endowment* 10, 6 (2017), 709–720.
[8] Santo Fortunato. 2010. Community detection in graphs. *Physics Reports* 486, 3-5 (2010), 75–174.
[9] Alan Gibbons. 1985. *Algorithmic graph theory.* Cambridge University Press.
[10] Alexander Hinneburg, Daniel A Keim, et al. 1998. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, Vol. 98. 58–65.
[11] Sunita Jahirabadkar and Parag Kulkarni. 2014. Algorithm to determine ε-distance parameter in density based clustering. *Expert Systems with Applications* 41, 6 (2014), 2939–2946.
[12] Mehjabin Khatoon and W Aisha Banu. 2019. An efficient method to detect communities in social networks using DBSCAN algorithm. *Social Network Analysis and Mining* 9, 1 (2019), 9.
[13] Jure Leskovec, Kevin J Lang, and Michael Mahoney. 2010. Empirical comparison of algorithms for network community detection. In *Proc. International Conference on World Wide Web (WWW)*. 631–640.
[14] Son T Mai, Xiao He, Jing Feng, Claudia Plant, and Christian Böhm. 2015. Anytime density-based clustering of complex data. *Knowledge and Information Systems* 45, 2 (2015), 319–355.
[15] Mark EJ Newman and Michelle Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69, 2 (2004), 026113.
[16] Andrew Y Ng, Michael I Jordan, and Yair Weiss. 2002. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*. 849–856.
[17] Jörg Sander, Martin Ester, Hans-Peter Kriegel, and Xiaowei Xu. 1998. Density-based clustering in spatial databases: The algorithm DBSCAN and its applications. *Data Mining and Knowledge Discovery* 2, 2 (1998), 169–194.
[18] Jieming Shi, Nikos Mamoulis, Dingming Wu, and David W Cheung. 2014. Density-based place clustering in geo-social networks. In *Proc. ACM International Conference on Management of Data (SIGMOD)*. 99–110.
[19] Hiroaki Shiokawa, Yasuhiro Fujiwara, and Makoto Onizuka. 2015. SCAN++: efficient algorithm for finding clusters, hubs and outliers on large-scale graphs. *Proc. of the VLDB Endowment* 8, 11 (2015), 1178–1189.
[20] Chayant Tantipathananandh, Tanya Berger-Wolf, and David Kempe. 2007. A framework for community identification in dynamic social networks. In *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 717–726.
[21] Yves van Gennip, Blake Hunter, Raymond Ahn, Peter Elliott, Kyle Luh, Megan Halvorson, Shannon Reid, Matthew Valasik, James Wo, George E Tita, et al. 2013. Community detection using spectral clustering on sparse geosocial data. *SIAM J. Appl. Math.* 73, 1 (2013), 67–83.
[22] Dingming Wu, Jieming Shi, and Nikos Mamoulis. 2018. Density-Based Place Clustering Using Geo-Social Network Data. *IEEE Transactions on Knowledge and Data Engineering (TKDE)* 30, 5 (2018), 838–851.
[23] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas AJ Schweiger. 2007. Scan: a structural clustering algorithm for networks. In *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 824–833.
[24] Xiaowei Xu, Nurcan Yuruk, Zhidan Feng, and Thomas A. J. Schweiger. 2007. SCAN: A Structural Clustering Algorithm for Networks. In *Proc. ACM International Conference on Knowledge Discovery and Data Mining (KDD)*. 824–833.
[25] Man Lung Yiu and Nikos Mamoulis. 2004. Clustering objects on a spatial network. In *Proc. ACM International Conference on Management of Data (SIGMOD)*. 443–454.