Clustering in Geo-Social Networks

Dingming Wu, Nikos Mamoulis, and Jieming Shi
Department of Computer Science, The University of Hong Kong
Pokfulam Road, Hong Kong
{dmwu,nikos,jmshi}@cs.hku.hk

Abstract

The rapid growth of Geo-Social Networks (GeoSNs) provides a new and rich form of data. Users of GeoSNs can capture their geographic locations and share them with other users via an operation named checkin. Thus, GeoSNs can track the connections (and the time of these connections) of geographic data to their users. In addition, the users are organized in a social network, which can be extended to a heterogeneous network if the connections to places via checkins are also considered. The goal of this paper is to analyze the opportunities in clustering this rich form of data. We first present a model for clustering geographic locations, based on GeoSN data. Then, we discuss how this model can be extended to consider temporal information from checkins. Finally, we study how the accuracy of community detection approaches can be improved by taking into account the checkins of users in a GeoSN.

1 Introduction

Clustering is a common task of data mining, which divides a set of objects into groups such that objects in the same group (called a cluster) are similar to each other while objects in different clusters are dissimilar. Clustering finds applications in machine learning, pattern recognition, image analysis, information retrieval, and bioinformatics. Specific applications include grouping homologous sequences into gene families in bioinformatics, partitioning the general population of consumers into groups in market research, recognizing communities within large groups of people in social networks, dividing a digital image into distinct regions for border detection or object recognition. Clustering can be achieved by various algorithms that may differ significantly in how they define clusters. Popular definitions of clusters are groups with small distances among the cluster members, dense areas of the data space, intervals or particular statistical distributions. The distance function, the density threshold or the number of expected clusters to use depend on the data to be clustered and the intended use of the clustering results.

The enormous growth of Geo-Social Networks (GeoSNs) not only brings more interesting data to clustering, but also poses challenges. In GeoSNs, such as Gowalla[1], Foursquare[2], and Facebook Places[3], users are allowed to capture their geographic locations and share them by an operation named *checkin*. A checkin is a triplet

Bulletin of the IEEE Computer Society Technical Committee on Data Engineering

[1] http://gowalla.com
[2] https://foursquare.com
[3] https://www.facebook.com/about/location

$\langle u, p, time \rangle$ modeling the fact that user u visited place with point location $p = \langle x, y \rangle$ at a certain $time$. Thus, on one hand, GeoSNs provide geographic places (e.g. points of interest) an opportunity to be (temporally) connected with social networks. On the other hand, the users of social networks are associated with their checkin point locations. The purpose of this paper is to investigate how clustering can be applied on this rich form of data.

Different from the traditional clustering of geographic locations, where only the spatial dimension is considered, clustering places in a GeoSN involves geo-social or geo-social-temporal dimensions. The geo-social place clusters discovered in a GeoSN find important application in the generalization and characterization of places. For example, discovering regions populated with similar places with respect to the people who live in them or visit them is a common task in geographic data analysis. Taking another example in urban planning, land managers are interested in identifying regions which have consistent demographic statistics, e.g., areas where elderly people prefer to visit, or, in general, people who belong to certain communities and have special transportation or living needs. The place clusters found in GeoSNs may benefit marketing as well. The fact that two (or more) commercial places belong to the same cluster indicates that there is a high likelihood that a user who likes one place would also be interested to visit the other(s). Therefore, campaigns may be initiated to users who visited other places in the same cluster, or a set of places could do collaborative promotion (e.g., a discount for users who visit multiple places in the cluster). By considering also the temporal information in the data (i.e., when did users checkin at the various places), the discovered clusters can be further refined and can become valuable for urban activity analysis, local authorities, service providers, decision makes, etc. For example, a certain set of places (e.g., shopping spots) may be characterized as a cluster for only restricted time periods or intervals (e.g., during Saturday morning hours). In addition, the user-groups that are relevant to a cluster could be relative to certain time periods. For example, shopping places in downtown are visited during the evening by people who have to work and could not shop at daytime, while supermarkets and small shops in the suburbs are usually visited by housewives in the daytime. Such geo-social-temporal clusters can be useful to marketing or advertising companies, which may benefit from understanding the (time sensitive) shopping habits of various social groups.

GeoSN data can also be used for clustering social network users. Different to classic social networks, which do not have checkin information, GeoSNs allow users to be clustered not only based on their social links but also based on their checkin behavior. Using both the social relationships and the checked in places by users can help discovering user clusters (called *local communities*) such that users in the same cluster not only have close social relationships, but also have similar mobility behavior in terms of their checkin places. The discovered local communities may provide useful information to local advertisers and social travel recommendation services such as facebook.com/36hrs.in and gogobot.com.

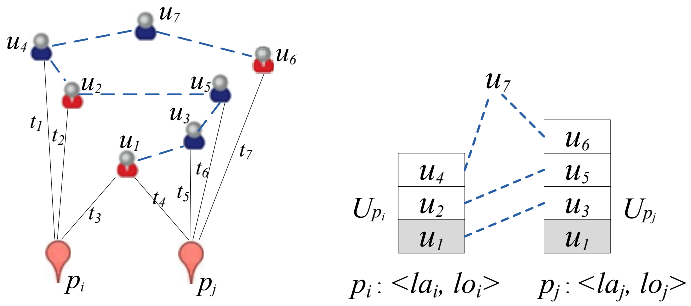
In this paper we investigate the possibilities of clustering geographic locations (i.e., places) and users based on the rich information tracked by GeoSNs. We first present the Density-based Clustering Places in Geo-Social Networks (DCPGS) model in Section 2 that detects geo-social place clusters in GeoSNs, considering both the *spatial* and the *social distances* between places. The DCPGS model (originally, proposed in [12]) extends traditional density-based clustering for spatial locations to consider the social relationships of users who visit them in a GeoSN. Next, we discuss possible definitions and future research directions for the geo-social-temporal place clustering and the local community detection problems in GeoSNs, in Sections 3 and 4, respectively. Finally, Section 5 concludes the paper.

2 Geo-Social Place Clustering

Among various clustering techniques, density-based clustering [4] is an effective approach for spatial data with low dimensionality [13]. It discovers arbitrary shaped clusters and is robust to outliers. The DCPGS model extends the density-based clustering framework by introducing a new distance function that takes both the spatial proximity and the social relationship between places into account. Section 2.1 formulates the DCPGS problem and defines the social distance measure between places that we use. DCPGS algorithms based on R-tree and grid partitioning are proposed in Section 2.2. We report part of our findings in Section 2.3.

2.1 Model and Definitions

The input of the DCPGS model includes a social network G and the set of checkins CK of a set of users U to a set of places P. The social network is an undirected graph $G = (U, E)$, where U is the set of all users and each edge $(u_i, u_j) \in E$ indicates that users $u_i, u_j \in U$ are friends. Each place $p_k \in P$ is identified by a unique GPS coordinate. Set $CK = \{\langle u_i, p_k, t_r\rangle | u_i \in U, p_k \in P\}$ includes all checkins generated by users in U. For a place p_k, its *visiting user set* is defined by $U_{p_k} = \{u_i | \langle u_i, p_k, *\rangle \in CK\}$, where $*$ means any time.



(a) A toy example (b) Abstraction

Figure 1: Example and storage structure of GeoSNs

2.1.1 DCPGS Model

DCPGS extends the model of DBSCAN [4]; for each place $p_i \in P$, DCPGS finds its *geo-social ϵ-neighborhood* $N_\epsilon(p_i)$, which includes all places p_j such that $D_{gs}(p_i, p_j) \leq \epsilon$, $D_S(p_i, p_j) \leq \tau$, and $E(p_i, p_j) \leq maxD$. For two places p_i, p_j, $E(p_i, p_j)$ is the *Euclidean distance*, $D_S(p_i, p_j)$ is the *social distance*, and $D_{gs}(p_i, p_j) = f(D_S(p_i, p_j), E(p_i, p_j))$ is the *geo-social* distance, defined as a function of $E(p_i, p_j)$ and $D_S(p_i, p_j)$. Parameter ϵ is geo-social distance threshold, while τ and $maxD$ are two *sanity* constraints for the social and the spatial distances between places, respectively. If the geo-social ϵ-neighborhood of a place p_i contains at least $MinPts$ places, then p_i is a *core* place; in this case, p_i and all places in its geo-social ϵ-neighborhood should belong to a cluster $r(p_i)$. If another core place p_j belongs to cluster $r(p_i)$, then $r(p_i) = r(p_j)$, i.e., the clusters defined by p_i and p_j are merged. After identifying all core places and merging the corresponding clusters, DCPGS ends up with a set of (disjoint) clusters and a set of outliers (i.e., places that do not belong to the geo-social ϵ-neighborhood of any core place).

Parameters. ϵ and $MinPts$ are the main parameters of DCPGS. $MinPts$ (i.e., the minimum number of places in the neighborhood of a core place) is set as in the original DBSCAN model (see [4]); a typical value is 5. ϵ takes a value between 0 and 1, because, as we explain later on, we define $D_{gs}(p_i, p_j)$ to take values in this range. Since the geo-social distance $D_{gs}(p_i, p_j)$ is a function of a spatial and a social distance, τ and $maxD$ constrain these individual distances to avoid the following two cases that negatively affect the quality of geo-social clusters.

- The geo-social distance between two places p_i and p_j could be less than ϵ if they are extremely close to each other in space, but have no social connection at all. This may lead to putting places close to each other spatially, but having no social relationship, into the same cluster.

- The geo-social distance between two places p_i and p_j could be less than ϵ if they have very small social distance, but they are extremely far from each other spatially. This may lead to putting places with close social distances, but large spatial distances, into the same cluster.

Constraints τ and $maxD$ are defined for quality control and can be set by experts or according to the analyst's experience. We experimentally study how clustering quality is affected by the two constraints and ϵ in Section 2.3.

Distance Functions. The social distance $D_S(p_i, p_j)$ takes in the visiting user sets U_{p_i} and U_{p_j} of places p_i and p_j, respectively, and returns a value between 0 and 1. In Section 2.1.2, we present our definition for $D_S(p_i, p_j)$. Before defining the geo-social distance $D_{gs}(p_i, p_j)$, we *normalize* the Euclidean distance $E(p_i, p_j)$ to a *spatial* distance $D_P(p_i, p_j) = \frac{E(p_i,p_j)}{maxD}$ that takes values between 0 and 1. Finally, $D_{gs}(p_i, p_j)$ is defined as weighted sum of $D_S(p_i, p_j)$ and $D_P(p_i, p_j)$, i.e.,

$$D_{gs}(p_i, p_j) = \omega \cdot D_P(p_i, p_j) + (1 - \omega) \cdot D_S(p_i, p_j), \tag{27}$$

where $\omega \in [0, 1]$.

2.1.2 Social Distance Between Places

The social distance $D_S(p_i, p_j)$ between p_i and p_j naturally depends on the social network relationships between the visiting user sets U_{p_i} and U_{p_j} of places p_i and p_j, respectively. Our definition for $D_S(p_i, p_j)$ is based on the set CU_{ij} of *contributing users* between two places p_i and p_j:

Definition 1: (Contributing Users) Given two places p_i and p_j with visiting user sets U_{p_i} and U_{p_j}, respectively, the set of contributing users CU_{ij} for the place pair (p_i, p_j) is defined as $CU_{ij} = \{u_a \in U_{pi} | u_a \in U_{p_j} \vee \exists u_b \in U_{p_j}, (u_a, u_b) \in E\} \cup \{u_a \in U_{p_j} | u_a \in U_{p_i} \vee \exists u_b \in U_{p_i}, (u_a, u_b) \in E\}$

Specifically, if a user u_a has visited both p_i and p_j, then u_a is a contributing user. Also if u_a has visited place p_i, u_b has visited p_j, and u_a and u_b are friends, both u_a and u_b are contributing users. Users in CU_{ij} contribute positively (negatively) to the social similarity (distance) between p_i and p_j. Formally:

Definition 2: (Social Distance) Given two places p_i and p_j with visiting users U_{p_i} and U_{p_j}, respectively, the *social distance* between p_i and p_j is defined as

$$D_S(p_i, p_j) = 1 - \frac{|CU_{ij}|}{|U_{p_i} \cup U_{p_j}|} \tag{28}$$

The above definition of $D_S(p_i, p_j)$ takes both the set similarity between sets U_{p_i} and U_{p_j} and the social relationships among users in U_{p_i} and U_{p_j} into account. In addition, the distance measure penalizes pairs of places p_i and p_j which are popular (i.e., U_{p_i} and/or U_{p_j} are large) but their set of contributing users is relatively small (see Equation 28). The reason is that such place pairs are not characteristic to their (loose) social connections.

As an example, consider places p_i and p_j of Figure 1. Figure 1(b) shows U_{p_i} and U_{p_j} for the two places p_i and p_j of the toy example in Figure 1(a). The figure also connects the user pairs in the two sets who are linked by friendship edges in the social network. Note that user u_8 does not belong to either U_{p_i} or U_{p_j}, but connects users u_4 and u_7 in the social graph.

To compute $D_S(p_i, p_j)$, we first set $U_{p_i} = \{u_1, u_2, u_4\}$ and $U_{p_j} = \{u_1, u_3, u_5, u_6\}$. All users in U_{p_i} and U_{p_j} are checked one by one to obtain the contributing users between p_i and p_j. We derive $CU_{ij} = \{u_1, u_2, u_3, u_5\}$, since (i) u_1 have visited both p_i and p_j, (ii) user u_2, who visited p_i, has a friend u_5 who visited p_j, (iii) symmetrically, user u_5, who visited p_j, has a friend u_2 who visited p_i, and (iv) u_3 ($\in U_{p_j}$) has a friend u_1 having been to p_i. According to Definition 2, the social distance $D_S(p_i, p_j)$ between p_i and p_j in Figure 1 is $1 - |CU_{ij}|/(|U_{p_i} \cup U_{p_j}|) = 1 - 4/6 \approx 0.3333$.

2.2 Algorithms

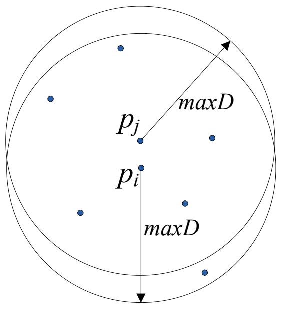
We propose two algorithms for computing geo-social clusters using DCPGS model. Algorithm DCPGS-R (Section 2.2.1) is based on the R-tree index, while algorithm DCPGS-G (Section 2.2.2) uses a grid partitioning.

2.2.1 Algorithm DCPGS-R: R-tree based

Algorithm DCPGS-R is a direct extension of the DBSCAN algorithm; it uses an R-tree to facilitate the search of the geo-social ϵ-neighborhood for a given place. Initially, all places are bulk-loaded into an R-tree. Then, DCPGS-R examines all places and, given a place p_i, it performs a range query centered at p_i with radius $maxD$ to get a set of *candidate places* that may fall in the geo-social ϵ-neighborhood of p_i, i.e., $N_\epsilon(p_i)$. Recall that $maxD$ is the maximum allowed spatial distance between place p_i and places in its geo-social ϵ-neighborhood. Then, DCPGS-R keeps in $N_\epsilon(p_i)$ only the candidates that satisfy the social distance constraint τ and the geo-social distance threshold ϵ. Clusters are identified by merging core places and their geo-social ϵ-neighborhoods.

2.2.2 Algorithm DCPGS-G: Grid based

DCPGS-R conducts a spatial range query for each place to obtain the candidate places for the purpose of discovering geo-social clusters. Even though individual R-tree based range queries are very efficient, discovering geo-social clusters in a GeoSN with millions of places requires millions of such queries (e.g., there are 1,280,969 places in the Gowalla dataset used in our experiments). Given two places p_i and p_j that are spatially close to each other, as Figure 2(a) shows, the results of the two range queries with radius $maxD$ centered at p_i and p_j, respectively, are almost identical. In algorithm DCPGS-R, independently issuing similar range queries on the R-tree searches almost the same space, resulting in redundant traversing paths and computations. To overcome this drawback, we develop a dynamic grid partitioning technique and a new algorithm DCPGS-G.



(a) Nearby spatial range queries (b) Grid partitioning

Figure 2: Nearby spatial range queries and grid partitioning

Grid Partitioning. The area covered by the whole data set is partitioned using a grid of size $maxD/\sqrt{2} \times maxD/\sqrt{2}$. The non-empty grid cells are indexed by a hash table with the grid cell coordinates as search keys.

Neighbor Cells. The neighbor cells of a cell c are the cells that intersect the union of four circles, each centered at a corner of cell c with radius $maxD$. For example, in Figure 2(b), the 20 gray cells (except c) are the neighbor cells of c, denoted as $NC(c)$. We can trivially show that for any place p inside c, the content of p's geo-social ϵ-neighborhood is contained in $NC(c)$ and c itself.

Cluster Discovery. Algorithm DCPGS-G includes three phases. First, it maps all places into grid cells. Second, it computes the geo-social ϵ-neighborhoods of places at the grid cell level. Specifically, for each non-empty and *unprocessed* cell c, its neighbor cells $NC(c)$ are retrieved. This operation filters out the pairs of places (p_i, p_j) with spatial distance greater than $maxD$. A cell is 'unprocessed' if its neighbor cells have not been retrieved before. Then the pairs of places (p_i, p_j) that satisfy the social distance constraint τ and the geo-social distance threshold ϵ are identified and the geo-social ϵ-neighborhoods $N_\epsilon(p_i)$ and $N_\epsilon(p_j)$ are updated. After all cells have been processed, meaning that the geo-social ϵ-neighborhoods of all places in the GeoSN are acquired, the third phase discovers all geo-social clusters following the framework of algorithm DCPGS-R, except that the $N_\epsilon(p_i)$ of each place p_i has already been computed in the second phase.

Complexity. With the help of grid partitioning, the geo-social ϵ-neighborhood of all places in cell c can be obtained by checking all c's neighbor cells; the whole process can be completed within a single pass of the data. Thus, the complexity of DCPGS-G is $O(n)$, as each of its three phases makes one pass over the data. However, algorithm DCPGS-R computes the geo-social ϵ-neighborhoods of each place one by one. Hence its cost is $O(n \log n)$, given that the expected cost of a single range query on the R-tree is $O(\log n)$.

2.3 Evaluation Results

In [12], we evaluated the DCPGS model and algorithms using two data sets from real geo-social networks[4] from two perspectives: effectiveness and efficiency. To assess effectiveness, we conducted a visualization-based analysis and a social quality evaluation in terms of two measures: social entropy and community score. In general, it has been demonstrated that the social relationships between users who visit places have great impact in place clustering and cannot be overlooked. The social distance measure we propose is more effective compared to competitor measures. To evaluate efficiency, we implemented the R-tree based and the grid-based DCPGS algorithms to apply using alternative distance measures and compared their performance under various parameter settings. The results show that the grid-based implementation is more efficient than the R-tree based implementation and our proposed social distance measure between places is more efficient to compute compare to more complex alternatives. The detailed evaluation results can be found in [12]. In this section, we show part of our visualization-based analysis, which compares the clusters found by DCPGS and competitor methods in the area of Manhattan on the Gowalla dataset (Figures 3(a)–3(f)) and also in the area of Chicago, on the Brightkite dataset (Figures 4(a)–4(b)).

Competitor DBSCAN [15] disregards the social network and finds density-based clusters using only the Euclidean distance between places. Competitor PureSocialDistance is an extreme case of DCPGS where ω is set to 0 in Equation 27. Competitor LinkClustering constructs a place network PN where two places p_i and p_j are connected if $E(p_i, p_j) \leq maxD$ and $D_S(p_i, p_j) \leq \tau$. The edge weight is set to $W_{gs}(p_i, p_j) = 1 - D_{gs}(p_i, p_j)$. Then, an offline community detection algorithm [1, 2] is applied on PN to discover place clusters. Competitor Jaccard replaces the social distance in DCPGS with the Jaccard similarity between the visiting user sets of two places. Finally, competitor SimRank applies the Minimax version of SimRank [7] to measure the similarity between the visiting user sets of two places. Compared to these five competitors, our proposed DCPGS finds geo-social clusters with the following features.

Geo-Social Splitting/Merging Criteria. Clusters found by DBSCAN due to their spatial closeness are split by DCPGS because of their weak social relationships, while clusters split by DBSCAN due to relatively low spatial density are merged by DCPGS due to their strong social ties. For example, Figure 3(a) and 3(b) shows that the layouts of the clusters discovered by DCPGS and DBSCAN are totally different. Specifically, comparing region A in Figures 3(a) with the corresponding region A′ in Figure 3(b), DCPGS and DBSCAN detect different cluster structures. The clusters found by DCPGS cannot be discovered by DBSCAN even if the parameters are tuned, since the densities of the small clusters in the half bottom of region A′ are similar and they are close to each other. Hence, DBSCAN will consider the places in the half bottom of region A′ as either a single cluster or several fragmented clusters (Figure 3(b)), under different parameter settings. Sometimes, DCPGS is able to split spatially dense clusters due to some natural barriers, such as rivers, and walls. It is inconvenient for the users to travel from one side of the barrier to the other side, so that the social ties between the places from the two sides

[4] snap.stanford.edu/data/index.html

of the barrier are weak, resulting in a splitting effect. As an example, in Figures 4(a) and 4(b), a cluster (region C) found by DBSCAN is split into two DCPGS clusters (regions C_1 and C_2) by the river. Although it might be possible for DBSCAN to detect the two DCPGS clusters by reducing the value of eps, such parameter settings will make some existing significant clusters disappear, resulting in too many outliers.

Spatially Loose Clusters. Some geo-social clusters found by DCPGS in region B of Figure 3(a) are considered as outliers by DBSCAN, shown as region B′ in Figure 3(b), since the places in region B′ is spatially too sparse to satisfy the density requirement of DBSCAN, and thus most places inside it are considered as outliers. However, these places (in region B of Figure 3(a)) are grouped into clusters by DCPGS due to the reason that the users who checked in those places have strong social relationships. If reducing the density parameters of DBSCAN, such spatially loose clusters can also be discovered. Nevertheless, many other clusters may be merged, making denser clusters indistinguishable.

Fuzzy Boundary Clusters. The boundaries of some DCPGS geo-social clusters are fuzzy, which makes sense in the real world, since groups of socially connected users may spatially overlap. In contrast, the clusters detected by DBSCAN have clearly strict boundaries. For instance, in Figure 3(a), no strict boundary exists between the four clusters enclosed in region A. Competitor PureSocialDistance also produces clusters with fuzzy boundaries (shown in Figure 3(c)). However, these clusters are spatially indistinguishable and of no interest, i.e., for the applications mentioned in the Introduction.

Competitor LinkClustering produces thousands of small clusters with average size around 3, shown in Figure 3(d), which are typically not well-separated spatially. Because of the sparse geo-social network data, the constructed place network consists of a lot of connected components that are disconnected with each other. For example, the place network built given $\tau = 0.7$, $maxD = 100$, and $\omega = 0.5$ contains 34,496 connected components with 4.3 nodes and 8.2 edges on average.

Competitors Jaccard and SimRank replace our D_S definition (Definition 2) by the Jaccard and the SimRank based measures. Figures 3(e) and 3(f) shows their clustering results. Competitor Jaccard produces small clusters and too many outliers, since large distance values are given for most pairs of places p_i and p_j due to the reason that the set of common users for two places in Jaccard (i.e., $U_{p_i} \cap U_{p_j}$) is expected to be small. On the contrary, competitor SimRank produces clusters of slightly larger sizes compared to DCPGS. We observed that the probability distribution of the SimRank-based measure is skewed towards small values, so that a lot of pairs of places are given low bipartite minimax SimRank social distance.

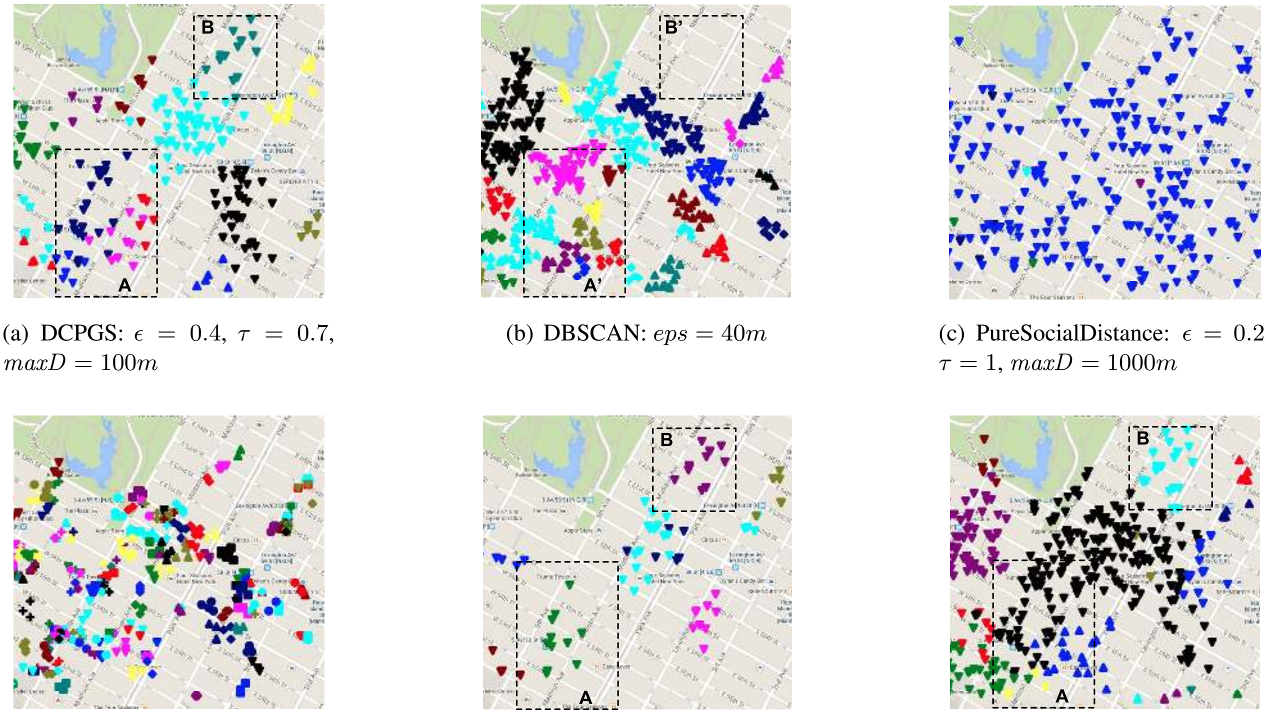
3 Geo-Social-Temporal Place Clustering

A checkin in GeoSNs is a triplet $\langle u, p, time\rangle$ modeling the fact that user u visited place with point location $p = \langle x, y\rangle$ at a certain $time$. The geo-social clusters found by the DCPGS model (presented in the previous section) compute the social distance between places based on the social network relationships between the visiting user sets of the places. However, temporal information is completely disregarded by the DCPGS model. It would be interesting to extend the DCPGS model such that the temporal information is taken into account in clustering and investigate how the temporal information affects the clustering result. In this section, we investigate the discovery of *geo-social-temporal* clusters in GeoSNs, which are spatio-temporal regions visited by groups of socially connected users.

In order to compute such geo-social-temporal clusters, a possible method would be to extend the definition of social distance between places to a socio-temporal distance D_{ST}. Using the social-temporal distance D_{ST}, the DCPGS model can then replace the geo-social distance D_{gs} by a newly defined geo-social-temporal distance as follows:

$$D_{gst}(p_i, p_j) = \omega \cdot D_P(p_i, p_j) + (1 - \omega) \cdot D_{ST}(p_i, p_j). \tag{29}$$

An intuitive definition of the socio-temporal distance D_{ST} would be to consider a pair of places socio-temporally close if they share many visiting users that have checked in the places within a small time period. On the other



(a) DCPGS: $\epsilon = 0.4$, $\tau = 0.7$, $maxD = 100m$

(b) DBSCAN: $eps = 40m$

(c) PureSocialDistance: $\epsilon = 0.2$, $\tau = 1$, $maxD = 1000m$

(d) LinkClustering: $\tau = 0.7$, $maxD = 100m$

(e) Jaccard: $\epsilon = 0.4$, $\tau = 0.7$, $maxD = 100m$

(f) SimRank: $\epsilon = 0.3$, $\tau = 0.7$, $maxD = 100m$

Figure 3: Place clusters of Gowalla found in Manhattan

hand, two places are socio-temporally far from each other if they do not have common visitors within a short time interval. The temporal dimension captures the evolution of place visits, and thus reflects the changes of the social distance between places. Based on the above, we suggest that the following three possible definitions of D_{ST} should be investigated.

Temporal Threshold. The socio-temporal distance extends the social distance (Equation 28) by replacing the contributing users CU_{ij} with the temporally contributing users TCU_{ij}, i.e.,

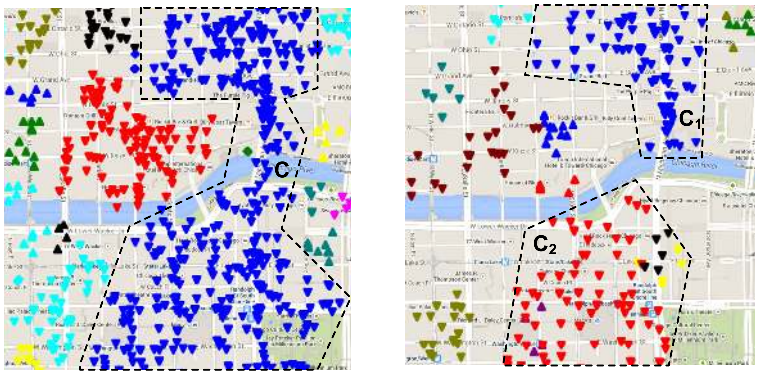
$$D_S(p_i, p_j) = 1 - \frac{|TCU_{ij}|}{|U_{p_i} \cup U_{p_j}|}. \tag{30}$$

The temporally contributing users are socially connected users who checked in p_i and p_j within a time interval θ. Let $T(u_a, p_i)$ be the time when user u_a checked in place p_i; formally:

Definition 3: (Temporally Contributing Users) Given two places p_i and p_j with visiting user sets U_{p_i} and U_{p_j}, respectively, the set TCU_{ij} of temporally contributing users for the place pair (p_i, p_j) is defined as $TCU_{ij} = \{u_a \in U_{p_i} | (u_a \in U_{p_j} \wedge |T(u_a, p_i) - T(u_a, p_j)| \leq \theta) \vee (\exists u_b \in U_{p_j}, (u_a, u_b) \in E \wedge |T(u_a, p_i) - T(u_b, p_j)| \leq \theta\} \cup \{u_a \in U_{p_j} | (u_a \in U_{p_i} \wedge |T(u_a, p_i) - T(u_a, p_j)| \leq \theta) \vee (\exists u_b \in U_{p_i}, (u_a, u_b) \in E \wedge |T(u_b, p_i) - T(u_a, p_j)| \leq \theta)\}$.

This definition of TCU_{ij} favors place pairs to which socially connected users paid visits what were close in time.

Damping Window. This method assigns each contributing user u_a an exponential decay factor $e^{t_c - T(u_a, p_i)}$, where t_c is the current time and $T(u_a, p_i)$ is the time when user u_a checked in place p_i. The contributing users who made checkins recently are weighed high. Instead of counting 1 for each user when computing the social



(a) DBSCAN: $eps = 60m$

(b) DCPGS: $\epsilon = 0.4$ s.t. $\tau = 0.7$, $maxD = 120m$

Figure 4: Clusters of Brightkite found by DBSCAN and DCPGS in Chicago

distance (Equation 28), this method counts the exponential decay factor of each user when computing the social-temporal distance D_{ST}. This definition favors place pairs to which the socially connected users have paid recent visits.

History-frame Clustering. This method performs geo-social clustering for each time period separately. For example, we can generate a different clustering of places for each month, by only using the checkin data recorded in that month. The clustering results would be useful in finding out how the place clusters evolve over time. It is also possible to track which place enters or leaves a cluster at a particular month and which parts of the clusters are time-insensitive.

4 Local Community Detection in GeoSNs

Community detection is an analytics tool for studying the social relationships among users. When detecting communities, there are two possible sources of information one can use: the social network structure and the features and attributes of users. Existing algorithms, however, typically focus on one of these two data modalities: community detection algorithms traditionally consider only on the network structure, while clustering algorithms mostly consider only user (node) attributes. Recently, algorithm CESNA [16] has been proposed to detect overlapping communities in networks with node attributes. CESNA statistically models the interaction between the network structure and the node attributes, which leads to more accurate community detection as well as improved robustness in the presence of noise in the structure. Later, Shakarian et al. [11] used a variant of Newman-Girvan modularity with the Louvain algorithm to address the problem of mining for geographically dispersed communities.

In GeoSNs, it would be interesting to detect local user communities taking both the social network structure and the checkin information into account, so that groups of socially connected users that checkin in the same or geographically close places are discovered. Existing algorithms that use either the network structure or node attributes cannot achieve the goal of local community detection. In addition, although CESNA could be applied by considering the checkin places as attributes of nodes (i.e., users), it may not achieve satisfactory results, because the proximity of places is not taken into account (i.e., only users that check in identical places would be considered as similar). Typically, the probability that two users have a significant overlap in their visiting places is low, therefore it makes sense to consider proximity as a factor of similarity between users in local community detection. Finally, although Shakarian et al. [11] provide a way to leverage spatial information in addition to network connection topology when mining networks for communities, they assume that each node in the social network is associated with only one home location. This approach is not applicable for the case when the users in GeoSNs have multiple check-in locations.

To implement local community detection in GeoSNs, we first need to model the users' mobile behaviors according to their checkin locations. Next, in the social network, we assign weights on the edges based on the similarity between the mobility behaviors of the corresponding users. Then, the resulting edge-weighted graph can be fed to existing community detection algorithms for weighted graphs, such as ISCoDe [6] and the algorithm proposed by Liu et al. [9], to identify the local communities. We suggest the following three ways for modeling the mobility behaviors of users.

Trajectory-based. The checkin locations of each user can be connected according to the time of the checkins to form a trajectory for the user. This trajectory models the mobility behavior of the user. Trajectory similarity can then be used to model the similarity between two users that are connected in a social network. Measures for trajectory similarity include Euclidean distance [10], dynamic time warping [8], edit distance [3], and longest common subsequence [5].

Image-based. The mobile behavior of each user is modeled as a black-and-white image where each pixel corresponds to the coordinates of a checkin location. The light intensity (gray value) of a pixel is determined by the frequency of visits at the corresponding place. The similarity between two images can be computed using the Minkowski metric on their contained pixels or more complicated measures incorporating specific task-dependent features [14].

Frequently Visited Region based. By analyzing the checkin locations of users, we can identify one or multiple frequently visited regions or areas for each of them. The granularity of the frequently visited regions can be determined by superimposing a grid on the map that includes all checkin locations. Then, each user is associated with one or multiple regions (cells) which s/he has frequently visited. For two users, we can use the spatial relationships (e.g., Euclidean distance or overlapping ratio) between their frequently visited regions to determine the similarity between the users.

5 Conclusions

Although geographic data clustering and community detection have been extensively studied for decades and many effective algorithms have been proposed, the rapid growth of the geo-social networks bring to these two problems a new and rich form of data together with new challenges. Clustering places by considering both their spatial proximity and the users who visit them (as well as the ties between these users) results in significantly different clusters compared to just using place locations. The time of the user checkins to places can be used to further refine the clusters. Differences and interesting insights can also be found in the user communities discovered when both the social relationships between users and the proximity between places they check in are considered.

In this paper, we have presented the Density-based Clustering Places in Geo-Social Networks (DCPGS) model [12] that discovers spatially and socially relevant place clusters. Our empirical studies prove the effectiveness of the model. We also discussed how to extend the DCPGS model to consider temporal information in the check-in data. Finally, we introduced the local community detection problem in GeoSNs, where the users forming a cluster are not only socially close but also exhibit similar mobility behavior in terms of their check-in locations.

References

[1] Y.-Y. Ahn, J. P. Bagrow, and S. Lehmann. Link communities reveal multiscale complexity in networks. *Nature*, 466:761–764, 2010.

[2] I. R. Brilhante, M. Berlingerio, R. Trasarti, C. Renso, J. A. F. de Macêdo, and M. A. Casanova. Cometogether: Discovering communities of places in mobility data. In *MDM*, 2012.

[3] L. Chen and R. Ng. On the marriage of lp-norms and edit distance. In *VLDB*, pages 792–803. VLDB Endowment, 2004.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, 1996.

[5] T. Ichiye and M. Karplus. Collective motions in proteins: a covariance analysis of atomic fluctuations in molecular dynamics and normal mode simulations. *Proteins*, 11(3):205–217, 1991.

[6] E. Jaho, M. Karaliopoulos, and I. Stavrakakis. Iscode: A framework for interest similarity-based community detection in social networks. In *Computer Communications Workshops (INFOCOM WKSHPS), IEEE Conference on*, pages 912–917, 2011.

[7] G. Jeh and J. Widom. Simrank: A measure of structural-context similarity. Technical Report 2001-41, Stanford InfoLab, 2001.

[8] J. B. Kruskal. An overview of sequence comparison. In *Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison*, pages 1–44. Addison-Wesley, Reading, MA, 1983.

[9] R. Liu, S. Feng, R. Shi, and W. Guo. Weighted graph clustering for community detection of large social networks. *Procedia Computer Science*, 31(0):85 – 94, 2014.

[10] A. C. Sanderson and A. K. C. Wong. Pattern trajectory analysis of nonstationary multivariate data. *IEEE Transactions on Systems, Man, and Cybernetics*, 10(7):384–392, 1980.

[11] P. Shakarian, P. Roos, D. Callahan, and C. Kirk. Mining for geographically disperse communities in social networks by leveraging distance modularity. In *KDD*, pages 1402–1409, 2013.

[12] J. Shi, N. Mamoulis, D. Wu, and D. W. Cheung. Density-based place clustering in geo-social networks. In *SIGMOD*, pages 99–110, 2014.

[13] P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison-Wesley, 2005.

[14] A. B. Watson, editor. *Digital Images and Human Vision*. MIT Press, Cambridge, MA, USA, 1993.

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

[16] J. Yang, J. J. McAuley, and J. Leskovec. Community detection in networks with node attributes. In *ICDM*, pages 1151–1156, 2013.