Density-based Place Clustering Using Geo-Social Network Data

Dingming Wu, Jieming Shi, and Nikos Mamoulis,

Abstract—Spatial clustering deals with the unsupervised grouping of places into clusters and finds important applications in urban planning and marketing. Current spatial clustering models disregard information about the people and the time who and when are related to the clustered places. In this paper, we show how the density-based clustering paradigm can be extended to apply on places which are visited by users of a geo-social network. Our model considers spatio-temporal information and the social relationships between users who visit the clustered places. After formally defining the model and the distance measure it relies on, we provide alternatives to our model and the distance measure. We evaluate the effectiveness of our model via a case study on real data; in addition, we design two quantitative measures, called social entropy and community score to evaluate the quality of the discovered clusters. The results show that temporal-geo-social clusters have special properties and cannot be found by applying simple spatial clustering approaches and other alternatives.

Index Terms—Clustering, Similarity Measurements, Algorithms.

1 Introduction

Clustering is commonly used as a method for data exploration, characterization, and summarization. Density-based clustering [1], in particular, divides a large collection of points into densely populated regions and it is the most appropriate clustering paradigm for spatial data, which have low dimensionality [2]. Density-based clusters have arbitrary shapes and sizes and exclude objects in areas of low density (i.e., outliers). The DBSCAN model [1] finds the spatial eps-neighborhood of each point p in the dataset, which is a circular region centered at p with radius eps. If the eps-neighborhood of p is dense, meaning that it contains no less than $MinPts$ places, p is called a *core* point. Dense eps-neighborhoods are put into the same cluster if they contain the cores of each other.

In this paper, we investigate the extension of traditional density-based clustering for spatial locations to consider their *relationship to a social network* of people who visit them *and the time* when they were visited. In specific, we consider the places of a *Geo-Social Network* (GeoSN) application, which allows users to capture their geographic locations and share them in the social network, by an operation called *checkin*. Online social networks with this functionality include Gowalla[1], Foursquare[2], and Facebook Places[3]. A checkin is a triplet $\langle uid, pid, time\rangle$ modeling the fact that user uid visited place pid at a certain $time$.

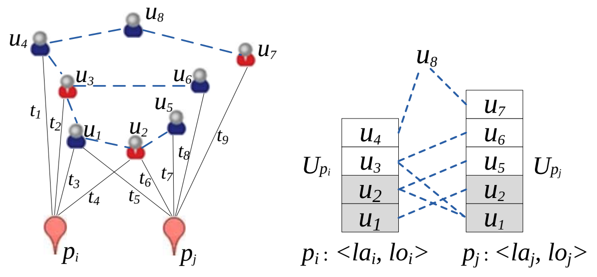
We define the new problem of Density-based Clustering Places in Geo-Social Networks (DCPGS), to detect geo-social clusters in GeoSNs. DCPGS extends DBSCAN by replacing the Euclidean distance threshold eps for the extents of dense regions by a threshold ϵ, which considers both the *spatial* and the *social distances* between places. For two places p_i and p_j, the spatial distance is considered to be the Euclidean distance between p_i and p_j, while the social distance should consider the social relationships between *the two sets of users* U_{p_i} and U_{p_j} which have checked in p_i and p_j, respectively. We define and use such a social distance measure, based on the intuition that two places are socially similar if they share many common users in their checkin records or the users in these records are linked by friendship edges. Figure 1(a) illustrates the data of a GeoSN that includes eight users (u_1–u_8) and two places (p_i and p_j). The dashed lines represent user friendships and the solid lines annotated with timestamps (t_1–t_9) illustrate checkins of users at places. For instance, user u_1 is a friend with u_2 and u_3 and has visited place p_i at time t_3 and p_j at t_5. As Figure 1(b) shows, each place is modeled by its spatial coordinates (e.g., $\langle la_i, lo_i\rangle$ for the latitude and longitude of p_i) and the set of users that have visited it (e.g., U_{p_i} for p_i). For the spatial distance between p_i and p_j, we can use the Euclidean distance between $\langle la_i, lo_i\rangle$ and $\langle la_j, lo_j\rangle$, while for the social distance component we use the set of common users in U_{p_i} and U_{p_j} (e.g., $\{u_1, u_2\}$) and the users in one place's record who are friends with visitors of the other place (e.g., $u_3 \in U_{p_i}$ and $u_6 \in U_{p_j}$ who are friends with each other). The intuition is that users who are friends can influence each other to visit the places included in their checkin history. The details of our distance measure are presented in Section 2.

In our previous work [3], the time of the check-ins made by users is not considered during the clustering process. The social connections established between places may be either out-dated or based on distant checkins in terms of time. For instance, the place clusters discovered according to the checkins one year ago may not interest analysts that value recent relationships between places. In addition, a cluster may have low quality if the check-in times of the different places in it vary significantly. In this paper, we investigate three methods of incorporating the temporal information (i.e., when did users checkin at the various places)

- *D. Wu is with the College of Computer Science & Engineering, Shenzhen University, China. E-mail: dingming@szu.edu.cn*
- *J. Shi (corresponding author) is with the Lenovo Big Data Lab, Hong Kong E-mail: jshi@lenovo.com*
- *N. Mamoulis is with the Department of Computer Science and Engineering, University of Ioannina. E-mail: nikos@cs.uoi.gr*

Manuscript received 2016; revised .

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



(a) A toy example (b) Abstraction

Fig. 1. Example and storage structure of GeoSNs

in the process of geo-social clustering, i.e., (1) history-frame geo-social clustering, (2) damping window, and (3) temporally contributing users, yielding temporal-geo-social clusters. The three methods of incorporating temporal information result in different clusters that satisfy different analysis needs. The history-frame method reveals how clusters evolve over time; its result can be used to analyze relationships between places at different periods of time. The damping window method weighs the recently checkins higher than old ones, so it generates clusters relevant to the current time. The temporally contributing user method only considers the checkins that have been made within a period of time, and ignores distant checkins. Hence, the places in the same cluster are visited by socially related users in the same periods of time, which indicates a stronger relationship between them. We compare the temporal-geo-social clusters and the geo-social clusters based on visualization and social entropy evaluation. The result shows that the temporal information helps improving the quality of clusters.

To the best of our knowledge, there is no previous work on clustering GeoSN places. While there exists a significant body of research on analyzing and querying GeoSN data [4], [5], [6], [7], most of these works are centered around users; i.e., they study user behavior, user link prediction or recommendation, or the evaluation of user queries. Thus, the places and checkins are only regarded as some auxiliary information to facilitate user-centered analysis. On the other hand, GeoSNs provide a new and rich form of geographical data, affiliated with the social network graph, the analysis of which can provide new and interesting insights, compared to raw spatial data. In specific, clustering of places in a GeoSN network finds a number of interesting applications:

Generalization and characterization of places. In geographic data analysis, a common task is to define regions (especially in urban areas), which include similar places with respect to the people who live in them or visit them. For example, in urban planning, land managers are interested in identifying regions which have uniform (i.e., consistent) demographic statistics, e.g., areas where elderly people prefer to visit, or people who belong to certain religious communities and have special transportation or living needs. Our DCPGS framework is especially useful for such spatial generalization and characterization tasks, because it can identify geographic regions where places form dense regions and the people who visit them are also socially connected to each other. 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).

Data cleaning. Intuitively, places that belong in the same geo-social cluster, according to our DCPGS framework, should have similar semantics. Therefore, our clustering results can help toward the cleaning of semantics (e.g., tags), which are given to places being in the same cluster (e.g., inspect tags that are inconsistent with the ones given to the majority of places in the cluster). In addition, as already mentioned, nearby GeoSN locations collected by user checkins could belong to the same physical place (e.g., a large restaurant) and our clusters can help toward identifying such cases and integrating multiple locations to the same physical place (i.e., region), possibly with the joint help of map-matching tools. Besides, taking the temporal dimension into account can also help to provide more accurate cleaning.

Marketing. GeoSN places may be commercial (e.g., restaurants). The fact that two (or more) such places belong to the same geo-social 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, by having knowledge of a place's geo-social cluster, the management of the place may initiate campaigns 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). 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.

Compared to conventional density-based spatial clusters, geo-social clusters detected by DCPGS exhibit larger intra-cluster social strength, as we confirm experimentally in Section 3. DCPGS also has additional advantages. First, DCPGS uses geo-social splitting criteria; for example, DCPGS splits clusters, which are spatially dense, but they are separated by barriers, such as rivers or walls, or visitors' weak social connections. Second, DCPGS finds spatially loose clusters that include sets of places that (pairwise) are not very close to each other (and therefore violate a typically tight spatial density threshold), but have very tight social relationships with each other. Such places satisfy our DCPGS criteria. On the other hand, DBSCAN is less flexible in including them in the same cluster as loosening its spatial distance threshold would result in putting everything in a single huge cluster. Third, DCPGS can discover geo-social clusters with fuzzy spatial boundaries; such clusters cannot be identified by spatial clustering, which defines strict spatial boundaries between clusters. Our evaluation is based on case studies and on the use of quantitative measures that we also propose in this paper. In addition, we demonstrate that the social distance measure we propose and use in DCPGS is more effective compared to alternative measures (based on node-to-node proximity). Overall, the results of our evaluation indicate that the social relationships between users who visit places have great impact in the clustering of places and cannot be overlooked. Furthermore, considering the temporal information in the clustering not only reveals how the clusters evolve with time, but also helps finding groups of places having more coherent social relationships.

Summing up, the contributions of this paper are as follows:

- We propose and formulate the problem of density-based clustering GeoSN places.

- We define a simple but effective social distance measure between places in GeoSNs.
- We demonstrate the effectiveness of DCPGS by case studies and quantitative evaluation through two quality measures that are also devised in this paper.
- We study three ways of incorporating temporal information in DCPGS and evaluate the resulting temporal-geo-social clusters.

The rest of the paper is organized as follows. Section 2 formulates the DCPGS problem, defines the social distance measure between places that we use, and introduces three methods of incorporating temporal information into DCPGS. The effectiveness of our framework are analyzed in Sections 3. Related work is reviewed in Section 4. Finally, Section 5 concludes this paper and discusses future work. The efficient algorithms in our previous work [3] for clustering GeoSN places and the performance evaluations are presented in Appendices B and C.

2 Model and Definitions

Our data input includes three components: a *social network*, a set of *places* and the *checkins* of users to the places. 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. Set P is the set of all places visited by users, in the form of $\langle$*latitude, longitude*$\rangle$ GPS points. Thus, identifiers are assigned to places according to their distinct GPS coordinates. Set $CK = \{\langle u_i, p_k, t_r\rangle | u_i \in U \text{ and } p_k \in P\}$ includes all checkins generated by users in U. A checkin in CK is a triplet $\langle u_i, p_k, t_r\rangle$ modelling the fact that user u_i visited place p_k at a certain t_r. For a place p_k, the set U_{p_k} of *visiting users* of p_k is defined by $U_{p_k} = \{u_i | \langle u_i, p_k, *\rangle \in CK\}$, where $*$ means any time. 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.

2.1 DCPGS Model

Our Density-based Clustering Places in Geo-Social Networks (DCPGS) model extends the model of DBSCAN [1]; for each place p_i in the GeoSN, DCPGS finds the **geo-social ϵ-neighborhood** $N_\epsilon(p_i)$ of p_i, which includes all places p_j such that $D_{gs}(p_i, p_j) \le \epsilon$, $D_S(p_i, p_j) \le \tau$, and $E(p_i, p_j) \le maxD$. 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. We will give detailed definitions for all above distance functions and parameters later on. If the geo-social ϵ-neighborhood of a place p_i contains at least $MinPts$ places, then p_i is a **core** place. In DCPGS, w.r.t. ϵ and $MinPts$; a place p_i is **directly density-reachable** from a place p_j if $p_i \in N_\epsilon(p_j)$ and $N_\epsilon(p_j) \ge MinPts$; a place p_i is **density reachable** from a place p_j if there is a chain of places $p_1, \cdots, p_n, p_1 = p_i, p_n = p_j$ such that p_{k-1} is directly density-reachable from p_k $(1 < k \le n)$; a place p_i is **density connected** to a place p_j if there is a place p_k, such that both p_i and p_j are density-reachable from p_k. A **cluster** C in DCPGS w.r.t. ϵ and $MinPts$ is a non-empty subset of places satisfying the following conditions: 1) $\forall p_i, p_j$: if $p_i \in C$ and p_j is density-reachable from p_i then $p_j \in C$. 2) $\forall p_i, p_j \in C$: p_i is density-connected to p_j. Outliers are the places that do not belong to any cluster.

Parameters. ϵ and $MinPts$ are the main parameters of DCPGS. $MinPts$ (i.e., the minimum number of places in the neighborhood of a core point) is set as in the original DBSCAN model (see [1]); 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 3.

Distance Functions. The social distance $D_S(p_i, p_j)$ takes as inputs the sets of users U_{p_i} and U_{p_j} who have visited p_i and p_j, respectively, and returns a value between 0 and 1. In Section 2.2, we present our definition for $D_S(p_i, p_j)$ and alternative ways to define it based on previous work. Before defining the geo-social distance $D_{gs}(p_i, p_j)$, we convert the Euclidean distance $E(p_i, p_j)$ into a *spatial* distance $D_P(p_i, p_j) = \frac{E(p_i, p_j)}{maxD}$ so that any place p_j in the geo-social ϵ-neighborhood of p_i has spatial distance no larger than 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.,

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

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

2.1.1 Alternatives to DCPGS

Using the proposed geo-social distance, our place clustering model (DCPGS) extends density-based clustering in spatial databases. We next present two alternatives to DCPGS, which can be also used to cluster GeoSN places.

SNN-based Clustering. The SNN clustering algorithm [8] is an improvement of DBSCAN that can find clusters of widely differing shapes, sizes, and densities. It first finds the k nearest neighbors of each data point and then redefines the similarity between pairs of points in terms of how many nearest neighbors the two points share. Using this definition of similarity, the SNN algorithm identifies core points and then builds clusters around the core points as DBSCAN does.

We extend SNN to cluster places in the geo-social network as follows. Firstly, the neighborhood of a place p in SNN is defined as $NN(p) = \{q \in P \wedge q \neq p | E(p, q) \le maxD \text{ and } D_S(p, q) \le \tau\}$. Then, according to SNN, the similarity between two places p_i and p_j is defined as the cardinality of the common places in their neighborhood, i.e., $S_{snn}(p_i,p_j) = |NN(p_i) \cap NN(p_j)|$. Given a user specified similarity parameter eps_{snn}, the SNN geo-social neighborhood of a place p_i is defined as $N_{snn}(p_i) = \{p_j \in P | S_{snn}(p_i,p_j) \geq eps_{snn}\}$. Place p_i is a core place if $|N_{snn}(p_i)| \geq MinPts_{snn}$, where $MinPts_{snn}$ is the user specified density threshold. If two core places are within each other's SNN geo-social neighborhood, then they are put into the same cluster.

Graph-based Clustering. GeoSN places can alternatively be clustered by the use of graph clustering models. The main idea of such a model is to construct a *place network* PN, which connects places according to their social and spatial distances and then apply an off-the-shelf community detection algorithm on PN. Specifically, given two places p_i and p_j, if $E(p_i,p_j) \leq maxD$ and $D_S(p_i,p_j) \leq \tau$, an *undirected* and *weighted* edge with geo-social weight $W_{gs}(p_i,p_j) = 1 - D_{gs}(p_i,p_j)$ is added between p_i and p_j. Community detection algorithms like Link Clustering [9], [10] or Metis [11] can then be applied to derive the clusters. Link Clustering constructs a dendrogram of network communities (that may overlap) in a hierarchical manner. Metis is another multilevel graph partition paradigm that includes three phases: graph coarsening, initial partitioning, and uncoarsening; Metis divides a network into k non-overlapping communities. As we will show in Section 3, these graph clustering methods are inferior to DCPGS.

2.2 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 sets U_{p_i} and U_{p_j} of users who visited p_i and p_j, respectively. $D_S(p_i,p_j)$ 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 users 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*

$$\begin{aligned} CU_{ij} = &\{u_a \in U_{p_i} | u_a \in U_{p_j} \ or \ \exists u_b \in U_{p_j}, (u_a,u_b) \in E\} \\ \cup &\{u_a \in U_{p_j} | u_a \in U_{p_i} \ or \ \exists u_b \in U_{p_i}, (u_a,u_b) \in E\} \end{aligned} \quad (2)$$

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*

$$D_S(p_i,p_j) = 1 - \frac{|CU_{ij}|}{|U_{p_i} \cup U_{p_j}|} \quad (3)$$

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 3). 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. To compute $D_S(p_i,p_j)$, we first set $U_{p_i} = \{u_1,u_2,u_3,u_4\}$ and $U_{p_j} = \{u_1,u_2,u_5,u_6,u_7\}$. 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 $CU_{ij} = \{u_1,u_2,u_3,u_5,u_6\}$, since (i) both u_1 and u_2 have visited p_i and p_j, (ii) user u_3, who visited p_i, has a friend u_6 who visited p_j, (iii) symmetrically, user u_6, who visited p_j, has a friend u_3 who visited p_i, and (iv) u_5 ($\in U_{p_j}$) has a friend u_2 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 $1 - |CU_{ij}|/(|U_{p_i} \cup U_{p_j}|) = 1 - 5/7 \approx 0.2857$.

Observe that only direct friendship edges between users of U_{p_i} and U_{p_j} are considered in our social distance definition. Longer network paths, such as friend-of-friend relationships, are ignored (e.g., the case of users u_4 and u_7 in Figure 1(b) who are connected via user u_8). According to the small world effect [12], a user in a social network can reach a large portion of other users within only few hops. For instance, the 90-percentile effective diameter [13] of Gowalla GeoSN, used in our experiments, is just 5.7.[4] This means within quite a few hops most users can reach a very large percentage of all users. In Gowalla, only 8 users can access more than 1% of all the users in 1 hop, while 40516 users (20.61% of all the users) can access more than 1% of all the users in 2 hops and 141582 users (72.02% of all the users) can reach more than 1% of all the users in 3 hops. The number of users who can reach more than 1% users in 2 or 3 hops increases dramatically compared to the percentage of those visited within 1 hop. Thus, paths longer than 1 hops are too common and cannot be considered as (indirect) user relationships; i.e., their impact is much weaker compared to direct friendship edges. Hence, Definition 2 introduces a simple, but powerful social distance measure. Properties of $D_S(p_i,p_j)$ include *symmetry* (i.e., $D_S(p_i,p_j) = D_S(p_j,p_i)$) and *self-minimality* (i.e., $D_S(p_i,p_j) \in [0,1]$, and $D_S(p_i,p_j) = 0$ for $U_{p_i} = U_{p_j}$). On the other hand, $D_S(p_i,p_j)$ does not obey the triangular inequality, but this does not affect our clustering algorithm.

2.2.1 *Alternatives to D_S*

Our DCPGS model is independent of the social distance definition between places (i.e., D_S). As an alternative to our Definition 2, the following measures can be used. In Section 3, we evaluate the effectiveness of these alternatives.

Jaccard. Based on the Jaccard similarity $J(p_i,p_j) = (|U_{p_i} \cap U_{p_j}|)/(|U_{p_i} \cup U_{p_j}|)$ between the sets of visiting users for p_i and p_j, we define $D_S^{Jac}(p_i,p_j) = 1 - J(p_i,p_j)$, which is not intuitive; it disregards the social network, assuming that two users who are friends do not affect each other in visiting GeoSN places.

SimRank. SimRank is a structural-context model for measuring the similarity between nodes in a graph. The idea is that two nodes are equivalent if they relate to equivalent nodes. We can define a SimRank-based social distance $D_S^{sim}(p_i,p_j)$, using the Minimax version of SimRank [14].[5] This measure compares each of p_i's visiting users $u_r^{p_i}$ with the visiting user $u_s^{p_j}$ of p_j who is the most similar to $u_r^{p_i}$, to compute the similarity between places $s(p_i,p_j)$. The similarity between users $s(u_r^{p_i},u_s^{p_j})$ is computed in an analogous way. Specifically, $D_S^{sim}(p_i,p_j) = 1 - s(p_i,p_j)$, where $s(p_i,p_j) = \min(s_{p_i}(p_i,p_j), s_{p_j}(p_i,p_j))$, $s_{p_i}(p_i,p_j) = \frac{\phi}{|U_{p_i}|} \sum_{u_r \in U_{p_i}} \max_{u_s \in U_{p_j}} s(u_r,u_s)$, where $\phi = 0.8$ is a decay factor [14], $s(u_r,u_s) = \min(s_{u_r}(u_r,u_s), s_{u_s}(u_r,u_s))$,

4. http://snap.stanford.edu/data/index.html

5. The original SimRank measure is only meant for node-to-node similarity; in our case, we need a measure between U_{p_i} and U_{p_j}.

and assuming that P_{u_r} is the set of places visited by u_r, $s_{u_r}(u_r, u_s) = \frac{\phi}{|P_{u_r}|} \sum_{p_i \in P_{u_r}} \max_{p_j \in P_{u_s}} s(p_i, p_j)$.

Katz. The Katz similarity measure [15] sums over all possible paths from user u_r to u_s with exponential damping by length, i.e., $\mathcal{K}(u_r, u_s) = \sum_{l=1}^{\infty} \beta^l |paths^l_{u_r,u_s}|$, where $paths^l_{u_r,u_s}$ is the set of all length-l paths from u_r to u_s, and damping factor β is typically set to 0.05. Due to the poor scalability of this measurement, in practice only paths up to length L are considered [16]; i.e., an approximated Katz score $\mathcal{K}_a(u_r, u_s) = \sum_{l=1}^{L} \beta^l |paths^l_{u_r,u_s}|$ can be used. Accordingly, we can define a Katz-based social distance between places p_i and p_j, by averaging the normalized Katz similarities between all pairs of users from U_{p_i} and U_{p_j}:

$$D_S^{Katz}(p_i, p_j) = 1 - \frac{1}{|U_{p_i}||U_{p_j}|} \sum_{u_r \in U_{p_i}} \sum_{u_s \in U_{p_j}} \mathcal{K}_a(u_r, u_s)$$

CommuteTime. The hitting time $h(u_r, u_s)$ from u_r to u_s is the expected number of steps required for a random walk starting at u_r to reach u_s. The commute time between u_r and u_s is defined by $ct(u_r, u_s) = h(u_r, u_s) + h(u_s, u_r)$. However, the commute time is sensitive to long paths and favors nodes of high degree. Thus, the *truncated commute time* [17], which considers only paths of length no longer than L, can be used to model the social distance between a pair of users. Finally, we can define a commute time based social distance between places p_i and p_j as follows:

$$D_S^{ct}(p_i, p_j) = \frac{1}{|U_{p_i}||U_{p_j}|} \sum_{u_r \in U_{p_i}} \sum_{u_s \in U_{p_j}} ct^L(u_r, u_s)$$

where $ct^L(u_r, u_s)$ is the normalized truncated commute time.

2.3 Incorporating Temporal Information

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, while the temporal information is disregarded. Taking the advantage of the temporal information in the checkin data may help improving the quality of discovered geo-social clusters. In this section, we investigate how the temporal information affects the clustering result. We investigate the discovery of temporal-geo-social clusters in GeoSNs, which are spatio-temporal regions visited by groups of socially connected users. In order to compute such temporal-geo-social clusters, we extend the definition of social distance between places to a temporal-social distance D_{TS}. Using the temporal-social distance D_{TS}, the DCPGS model can then replace the geo-social distance D_{gs} by a newly defined temporal-geo-social distance as follows:

$$D_{tgs} = \omega \cdot D_P(p_i, p_j) + (1 - \omega) \cdot D_{TS}(p_i, p_j) \qquad (4)$$

An intuitive definition of the temporal-social distance D_{TS} would be to consider a pair of places temporal-socially close if they share many socially connected visiting users that have checked in the places within a small time period. On the other hand, two places are temporal-socially far from each other if they do not have socially connected 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_{TS} should be investigated.

2.3.1 History-Frame Geo-Social Clustering

This method performs geo-social clustering for each time period separately. The time period can be either continuous or periodic (i.e., calendric). For example, we can generate a different clustering of places for each month, by only using the checkin data recorded in that month. We can also generate clusterings for working days and weekends. The clustering results in consecutive continuous periods can be used to study the evolution of clusters 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. Similarly, calendric clusters of different time periods can be compared; for instance, some clusters may only appear in holiday periods.

2.3.2 Damping Window

Recall that the social distance (Equation 3) between two places is calculated based on the users who checked in the two places. The social distance between two places is small if their visiting users are socially connected well. In this method, given a place p, each user u who has visited p is assigned an exponential decay factor $u^w(p) = \exp^{-x}$, where $x = (t_c - t(u, p))/T$, t_c is the current time, $t(u, p)$ is the last time when user u checked in place p, and T is the time range of the data (i.e., t_c minus the earliest time of the data), which is used for normalization purpose. The users who made checkins recently are weighed high. This method favors place pairs to which the socially connected users have paid recent visits. The temporal-social distance is defined as

$$\begin{aligned} D_{TS}(p_i, p_j) = 1 - \Big(& \sum_{u \in CU_{ij} \cap U_{p_i} \cap U_{p_j}} \max\{u^w(p_i), u^w(p_j)\} \\ & + \sum_{u \in CU_{ij} \cap (U_{p_i} \setminus U_{p_j})} u^w(p_i) \\ & + \sum_{u \in CU_{ij} \cap (U_{p_j} \setminus U_{p_i})} u^w(p_j) \Big) / |U_{p_i} \cup U_{p_j}| \end{aligned}$$

As an example, consider the social network and places p_i and p_j in Figure 1. Table 1 shows the visiting users' check-in time. Let the current time be 10 and the time range of the data be $T = 10$. Table 2 shows the exponential decay factors of the visiting users. For example, the exponential decay factor of user u_4 for place p_i is computed as $u_4^w(p_i) = \exp^{-(10-3)/10} = 0.5$. According to Definition 1, $CU_{ij} = \{u_1, u_2, u_3, u_5, u_6\}$. Hence, the temporal-social distance between places p_i and p_j is $D_{TS}(p_i, p_j) = 1 - (0.5 + 0.82 + 0.61 + 0.67 + 0.55)/7 = 0.55$.

TABLE 1
Checkin Time

Checkin Time	u_1	u_2	u_3	u_4	u_5	u_6	u_7
p_i	2	8	5	3	-	-	-
p_j	3	7	-	-	6	4	9

TABLE 2
Exponential Decay Factor

$\exp^{-x}$	u_1	u_2	u_3	u_4	u_5	u_6	u_7
p_i	0.45	0.82	0.61	0.50	-	-	-
p_j	0.50	0.74	-	-	0.67	0.55	0.90

2.3.3 Temporally Contributing Users

In this method, we consider temporally contributing users that are socially connected users who checked in places p_i and p_j within a time interval θ.

Definition 3. *(Temporally Contributing Users) Let E be the edge set of the social network. Given two places p_i and p_j with visiting* *user sets U_{p_i} and U_{p_j}, the contributing user set CU_{ij} is defined in Definition 1. The temporally contributing user set is a subset of the contributing user set, i.e., $TCU_{ij} \subseteq CU_{ij}$. Each user in the temporally contributing user set satisfies one of the following conditions $\forall u \in TCU_{ij}$*

1) $|t(u,p_i) - t(u,p_j)| \le \theta$
2) $\exists u' \in TCU_{ij}((u,u') \in E \wedge |t(u,p_i) - t(u',p_j)| \le \theta)$
3) $\exists u' \in TCU_{ij}((u,u') \in E \wedge |t(u',p_i) - t(u,p_j)| \le \theta)$

The temporal-social distance between places p_i and p_j is defined as $D_{TS}(p_i,p_j) = 1 - |TCU_{ij}|/|U_{p_i} \cup U_{p_j}|$.

3 Qualitative Analysis

This section analyzes the quality of the geo-social clusters and the temporal-geo-social clusters discovered by our proposed framework. Firstly, we compare with two extreme versions of DCPGS: PureSocialDistance applies density-based clustering by using the social distance $D_S(p_i,p_j)$ only, while DBSCAN uses only the Euclidean distance $E(p_i,p_j)$. This comparison shows the appropriateness of using both social and spatial distances in clustering. In the implementation of PureSocialDistance, we do not put place pairs with spatial distance more than 1000 meters in the same cluster; otherwise this method becomes too expensive. Secondly, we compare DCPGS with the graph clustering and SNN-based approaches discussed in Section 2.1.1, in order to demonstrate the suitability of the density-based clustering model for this application. Thirdly, we assess the suitability of our social distance measure (Section 2.2) by evaluating versions of DCPGS, which use the alternative social distance definitions discussed in Section 2.2.1. Finally, we analyze the temporal effects on the geo-social clusters found by DCPGS. All tested methods were implemented in C++ and the experiments were performed on a 3.4 GHz quad-core machine running Ubuntu 12.04 with 16 GBytes memory.

Data. We use two publicly available datasets[6] from historical geo-social networks. Gowalla contains a social network with $|U|$ =196,591 users and $|E|$ =950,327 undirected friendship edges. There are $|CK|$ =6,442,892 checkins performed by those users on $|P|$ =1,280,969 places over a period from Feb. 2009 to Oct. 2010. Brightkite includes a social network of $|U|$ =58,228 users and $|E|$ =214,078 undirected friendship edges. It contains $|CK|$ =4,491,143 checkins on $|P|$ =772,783 distinct places collected over the period from Apr. 2008 to Oct. 2010.

Default Parameter Settings. The density requirement of the clustering is determined by parameters $MinPts$ and ϵ (or DBSCAN's eps). We set $MinPts$ = 5 for all approaches; various density settings can be achieved by just tuning ϵ (or DBSCAN's eps). For instance, a large $MinPts$ has similar effect as a small ϵ (or DBSCAN's eps). By default, parameter ω in the geo-social distance is set to 0.5 to equally weigh the social and spatial distances. By default, parameter τ is set to 0.7, and $maxD$ is set to 100 meters for dataset Gowalla and 120 meters for dataset Brightkite. In the rest of the paper, the length values of the spatial distance are in the unit of meter, and thus we omit "meter" when context is clear.

3.1 Visualization-based Analysis

We first visualize and compare the clusters found by DCPGS and alternative approaches in the area of Manhattan on the Gowalla dataset and in the area of Chicago on the Brightkite dataset. Figures 2(a)-(c) show the clusters by DCPGS, DBSCAN (which disregards the social network behind the places) and PureSocialDistance (which disregards the spatial information). DCPGS finds geo-social clusters with the following features.

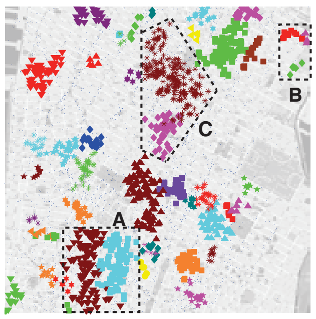
Geo-Social Splitting/Merging Criteria. Geo-social clusters that are very close to each other are split correctly by DCPGS, while DBSCAN may consider them as a single cluster due to their spatial closeness; in other cases, clusters split by DBSCAN due to relatively low spatial density between them are merged by DCPGS because of their strong social ties. For example, consider region A in Figures 2(a) and the corresponding region A′ in Figure 2(b), where DCPGS and DBSCAN detect clusters with totally different layouts. By tuning the parameters of DBSCAN, we are not able to find the clusters found by DCPGS, because the densities of the two clusters in region A are similar and the two clusters are close to each other. Thus, DBSCAN can only consider the places in region A′ as either a single cluster or as several fragmented clusters (Figure 2(b)), under different parameter settings. In certain cases, spatially dense clusters may be split by DCPGS because of some natural barriers, such as rivers, and walls. These barriers make it inconvenient to travel from one side to the other, resulting in a splitting effect. As an example, in Figure 3, a cluster (region D) found by DBSCAN is split into two DCPGS clusters (regions D_1 and D_2) by the river, since the users on different river sides are proved to have weak social connection. While it is possible for DBSCAN to find the two DCPGS clusters by reducing the value of eps, its parameter settings in this case make some existing clusters disappear, resulting in too many outliers.

Spatially Loose Clusters. Some geo-social clusters detected by DCPGS in region B of Figure 2(a) are considered as outliers by DBSCAN in the corresponding region B′ of Figure 2(b). Region B′ is spatially too sparse to satisfy the density requirement of DBSCAN, and thus most places inside it are filtered out as outliers. However, the users who checked in those places have strong social relationships. Hence, geo-social clusters are discovered in region B by DCPGS in Figure 2(a). While it is possible for DBSCAN to discover such spatially loose clusters by reducing the density parameters, this would result in merging too many clusters together, making denser clusters indistinguishable.

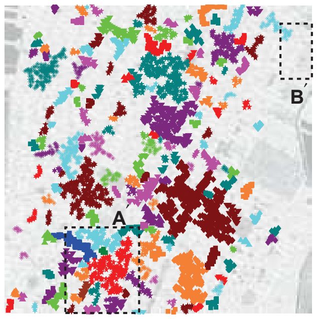
Fuzzy Boundary Clusters. Some DCPGS geo-social clusters have fuzzy boundaries with each other, which is reasonable in the real world, since groups of socially connected users may spatially overlap. On the other hand, DBSCAN produces clusters with strict boundaries. For instance, in Figure 2(a), there is no strict boundary between the two clusters enclosed in region C. Although PureSocialDistance, which is the other extreme method, also produces clusters with fuzzy boundaries (see Figure 2(c)), the clusters are spatially indistinguishable and they are not interesting, i.e., for the applications mentioned in the Introduction.

Alternatives to DCPGS. We visually analyzed the results of the alternatives to DCPGS, i.e., SNN-based clustering model and graph-based clustering models (LinkClustering and Metis), described in Section 2.1.1. LinkClustering and Metis produce similar results; indicatively, we show the clusters produced by LinkClustering in Figure 2(d). LinkClustering produces thousands of small clusters (average size around 3), which are typically not well-separated spatially. Due to the sparsity of geo-social network data, the constructed place network contains many connected components that are disconnected with each other (e.g., the place

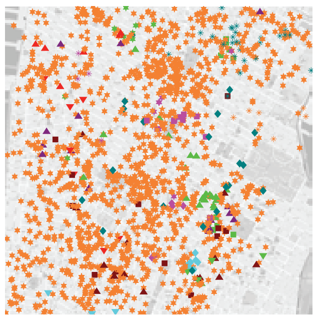
6. downloaded from snap.stanford.edu/data/index.html

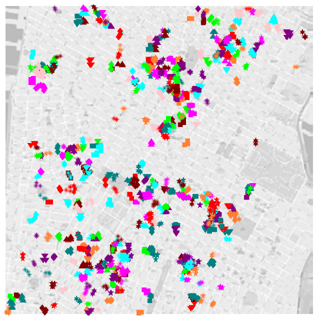


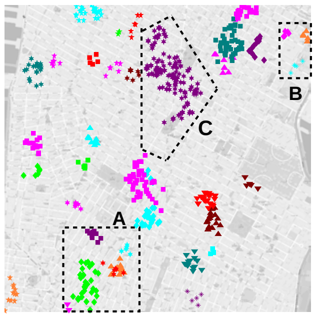
(a) DCPGS: $\epsilon = 0.4$, $\tau = 0.7$, $maxD = 100$



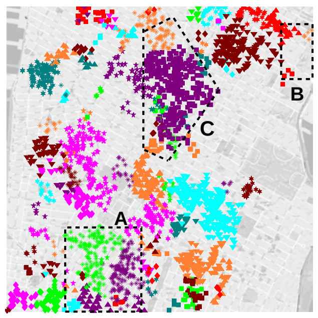
(b) DBSCAN: $eps = 40$


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

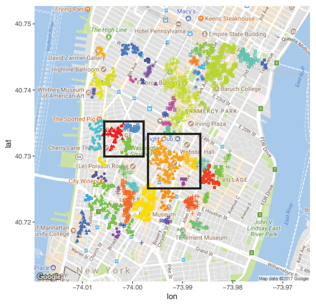

(d) LinkClustering: $\tau = 0.7$, $maxD = 100$



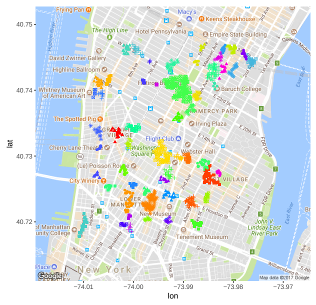
(e) Jaccard: $\epsilon = 0.4$, $\tau = 0.7$, $maxD = 100$



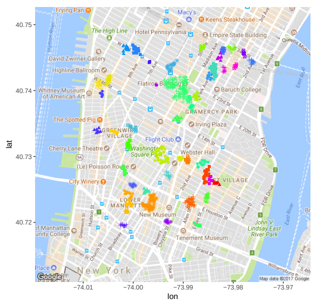
(f) SimRank: $\epsilon = 0.3$, $\tau = 0.7$, $maxD = 100$



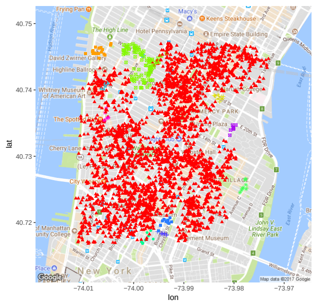
(g) SNN: $eps_{snn} = 1$, $maxD = 60$



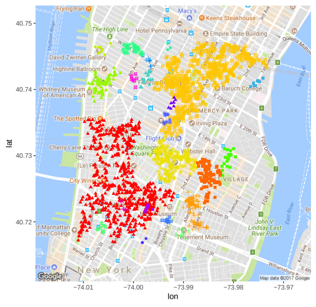
(h) SNN: $eps_{snn} = 2$, $maxD = 60$



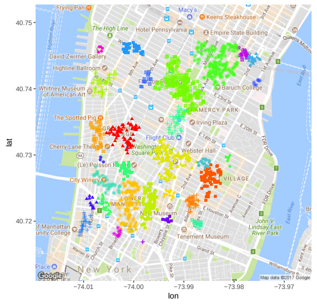
(i) SNN: $eps_{snn} = 3$, $maxD = 60$



(j) SNN: $eps_{snn} = 1$, $maxD = 120$



(k) SNN: $eps_{snn} = 3$, $maxD = 120$

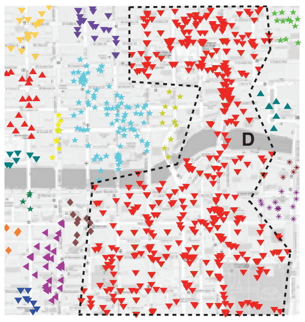


(l) SNN: $eps_{snn} = 5$, $maxD = 120$

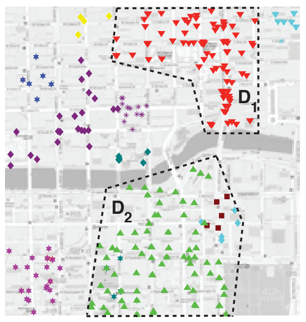
Fig. 2. Place clusters of Gowalla found in Manhattan

network built when $\tau = 0.7$, $maxD = 100$, and $\omega = 0.5$ contains 34,496 connected components with 4.3 nodes and 8.2 edges on average). The clusters found by Metis are fewer and larger, but also spatially indistinguishable. Metis ignores outliers; as a result, places belong to the same cluster may have low spatial proximity and social similarity.

Figures 2(g)–2(l) show the clusters found by the SNN-based clustering model. Since DCPGS and SNN-based model only differ in the way of measuring the closeness/similarity between places, by carefully tuning the parameters of SNN-based model, we try to figure out whether it can discover similar clusters to those of DCPGS. We observe that when $maxD = 60$ and $eps_{snn} = 1$ (Figure 2(g)), the SNN-based model indeed finds similar clusters in the areas highlighted by rectangles. As eps_{snn} increases (Figures 2(h) and 2(i)), the sizes of clusters decrease. This is because large eps_{snn} puts strict constraints on the places to be a member of a cluster. When $maxD = 120$ and $eps_{snn} = 1$ (Figure 2(j)), a big cluster is discovered by the SNN-based model. As eps_{snn} increases (Figures 2(k) and 2(l)), several clusters are identified from this big cluster. Although our geo-social distance can also be



(a) DBSCAN: eps = 60



(b) DCPGS: ϵ = 0.4 s.t. τ = 0.7, $maxD$ = 120

Fig. 3. Clusters of Brightkite found by DBSCAN and DCPGS in Chicago

used in an extension of DBSCAN (DCPGS) or in an extension of SNN for geo-social clustering, In most cases, DCPGS discovers clusters with better social entropy than SNN-based model does (Figure 5).

Alternatives to D_S. Finally, we analyzed the results of DCPGS, if our D_S definition (Definition 2) is replaced by the alternatives described in Section 2.2.1. For D_S^{Katz} and D_S^{ct}, we set $L = 3$; for bigger L values, these measures become extremely expensive. We observed that D_S^{Jac}, D_S^{Katz}, and D_S^{ct} produce similar results to each other. Indicatively, Figure 2(e) shows the clusters found by DCPGS if D_S^{Jac} is used instead of D_S. All these measures produce small clusters and too many outliers since they give large distance values for most pairs of places p_i and p_j. The set of common users for two places in Jaccard (i.e., $U_{p_i} \cap U_{p_j}$) is expected to be small and the decay factor of Katz dampens the effect of long connections between U_{p_i} and U_{p_j}. The expected CommuteTime distance between places is also high due to the effect of normalization.

On the other hand, D_S^{sim} produces clusters of slightly larger sizes compared to D_S. We observed that the probability distribution of D_S^{sim} is skewed towards low values, meaning that many pairs of places have low bipartite minimax SimRank social distance, because SimRank is based on the most similar pair of visiting users. The clustering results of SimRank (Figure 2(f)) and DCPGS are visually similar; it is hard to tell which results are better based on visualization.

3.2 Social Quality Evaluation

In this section, we design and use two measures for assessing the social coherence between places in the discovered clusters. Based on these measures, we assess the quality of DCPGS and the alternative approaches for clustering GeoSN places.

3.2.1 Social Entropy based Evaluation

The first measure, called social entropy, measures the social quality of the clusters based on the network communities that the GeoSN users form. Given a social network $G = (U, E)$, we first partition all the users in U into several disjoint network communities. Let PC be a cluster of GeoSN places and let U_{PC} be the set of users who visit the places in PC, i.e., $U_{PC} = \cup_{p \in PC} U_p$. According to the detected network communities, the visiting users of PC, U_{PC}, can be divided into several disjoint sets in $C_{PC} = \{C_1, C_2, \ldots, C_m\}$, such that each set C_i is a subset of users in U_{PC} belonging to the same network community. We call C_{PC} the community set of PC.

Definition 4. *(Social Entropy) Given a cluster PC, let U_{PC} be the set of users who visit the places in PC, i.e., $U_{PC} = \cup_{p \in PC} U_p$. The social entropy of PC is then defined as:*

$$\mathcal{E} = \sum_{C_i \in C_{PC}} -\frac{|C_i|}{|U_{PC}|} \log \frac{|C_i|}{|U_{PC}|}$$

The social entropy, analogous to the entropy used in decision tree induction, measures the impurity of a cluster PC with respect to the participation of its users into different communities. A low social entropy means that the great majority of the visitors of PC come from the same community (i.e., low impurity), indicating that the places in PC have tighter social relationships between each other, which is favored.

We applied the METIS community detection algorithm [11] to divide the set of users in the GeoSN into k non-overlapping communities[7], providing a baseline for social entropy evaluation. To avoid a comparison that is biased to parameter k, we evaluate the social entropy of clustering results obtained by DCPGS and the competitors for two different values of k. One value of k is chosen based on the following *rule of the thumb* [18], i.e., $k = \sqrt{N/2}$ where N is the number of users in the social network. The other value of k is decided by Dunbar's number that suggests humans can only comfortably maintain 150 stable relationships and the mean community size is around 150. For dataset Gowalla, these values are $k = 313$ and $k = 1310$, respectively.

In Figure 4, we use social entropy to test our method DCPGS with alternative social distance definitions, which are denoted by the name of the social distance measure used in each case. Figures 4(a) and 4(b) show the average social entropy for DCPGS, versions of DCPGS with alternative D_S (SimRank, CommuteTime, Katz, and Jaccard) and PureSocialDistance when varying ϵ. CommuteTime, and Katz have the lowest social entropy; however, these methods produce small clusters and have too many outliers as explained in Section 3.1. Within each small cluster, the places are only visited by few people and this explains the low entropy. Jaccard also has low social entropy for the same reason. PureSocialDistance has low social entropy in most cases; this indicates that our proposed social distance between places is effective in putting places with close social relationships together. When $\epsilon = 0.1$, the social entropies of DCPGS and PureSocialDistance are similar, both good, since only those places with very close social distances are clustered. When $\epsilon = 1$, PureSocialDistance has no social distance constraint τ thus its entropy becomes higher than that of DCPGS. DCPGS outperforms SimRank-based DCPGS, meaning that our proposed social distance is better than the SimRank-based $D_S^{sim}(p_i, p_j)$ distance. When $k = 1310$, the average social entropy of all the methods is larger than in the case where $k = 313$, since a larger number of network communities increases the probability that users in a single cluster belong to diverse communities. The average cluster size increases with ϵ, increasing the probability that the visitors of a cluster belong to different communities; thus, the average social entropy increases. In addition, the clustering result becomes stable at large values of ϵ, thus the social entropy converges. By visualization, we observed that ϵ should be set to a value smaller than 0.5 for the clustering results to be interesting.

Figures 4(c) and 4(d) show the average social entropies of DCPGS, DCPGS based on SimRank, CommuteTime, Katz,

7. This is different from Metis used as a competitor of DCPGS in GeoSN place clustering (discussed in Section 2.1.1).

and Jaccard, and the graph-based clustering methods Metis and LinkClustering, when varying the social distance constraint τ. Similar to the case when ϵ varies, the social entropy increases and then stabilizes as τ increases, except for the entropy of Metis, which keeps increasing due to the network partitioning methodology of Metis with the increase of τ, the constructed place network becomes less connected, however, due to its partitioning nature, Metis puts disconnected places in the constructed place network into same cluster. When τ is less than 0.5, the social entropy of CommuteTime is zero, since with these distance measures the places in each cluster are visited by only one person when $\tau < 0.5$. Jaccard has low social entropy also due to the small sizes of its clusters. For $\tau \leq 0.1$ SimRank-based DCPGS fails to find any clusters, therefore the entropy is 0. After investigation, we found that there is no pair of places p_i, p_j with $D_S^{sim}(p_i, p_j) < 0.2$ because of the decay factor ϕ. When $\tau = 0.2$, SimRank has a low social entropy, since only few (987) clusters of small size are found compared to the 3605 clusters discovered by DCPGS. After the point where the two approaches find a similar number of clusters (e.g., at $\tau = 0.5$, SimRank finds 5880 clusters, while DCPGS finds 6742 clusters), DCPGS has constantly lower entropy than SimRank. In addition, DCPGS is less sensitive to τ compared to SimRank. DCPGS outperforms the two graph-based competitors Metis and LinkClustering. As we observed by visualization, in practice τ should be set to a value higher than 0.5, because a very tight social distance constraint creates too few and too small geo-social clusters.

Figures 4(e) and 4(f) show the average social entropies of the various versions of DCPGS and all the competitor approaches when varying the spatial distance constraint $maxD$ (eps for DBSCAN). DCPGS is superior to SimRank-based DCPGS, DBSCAN, LinkClustering and Metis for all values of $maxD$ (eps). In general, the social entropies of all methods are not very sensitive to $maxD$. For Gowalla, a good value for $maxD$ is around 100; large $maxD$ values result in clusters that are spatially too loose.

Figure 5 compares the social entropy of SNN-based model (when varying eps_{snn}) and DCPGS (when varying ϵ), where SNN−60 and SNN−120 denote SNN-based model with $maxD = 60$ and $maxD = 120$, respectively. Parameter τ is fixed at 0.7 and $MinPts$ is set to 5. The first row of the x-axis represents the values of ϵ in DCPGS, while the second row contains the values of eps_{snn} in SNN-based model. The range of ϵ in DCPGS is $[0, 1]$. Parameter eps_{snn} in SNN is an integer in the range of $[0, +\infty)$. Small ϵ in DCPGS can be translated into large eps_{snn} in SNN-based model, which indicates that the condition of a place being a core is that there should be enough places with small geo-social distances surrounding it. We observe that the social entropy of SNN-based model does not change much with eps_{snn} and it is worse than DCPGS with small ϵ, and comparable with DCPGS with large ϵ.

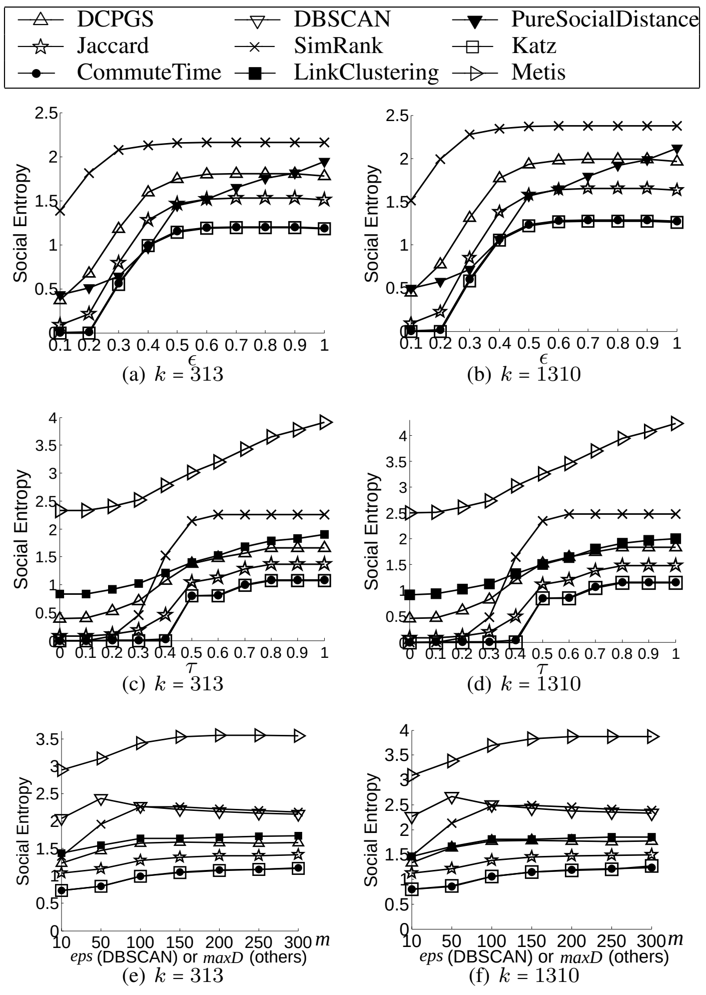


Fig. 4. Social entropy evaluation in Gowalla

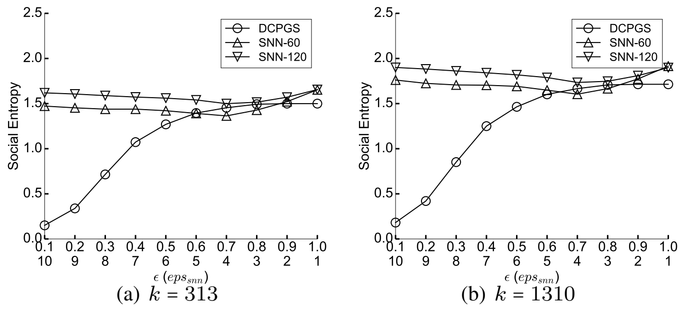
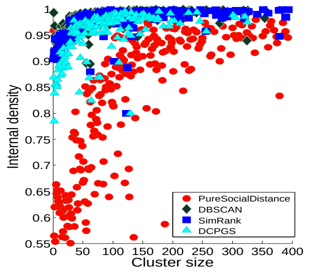
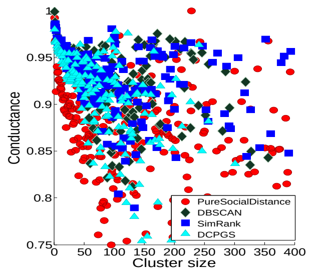


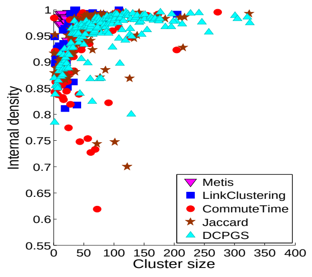
Fig. 5. Social entropy of SNN geo-social clusters in Gowalla

3.2.2 Community Score based Evaluation

Given a GeoSN place cluster PC, let U_{PC} be the set of users who visit the places in PC, i.e., $U_{PC} = \cup_{p \in PC} U_p$. Assume each U_{PC} is a community in the GeoSN. We adopt the eight network community multi-criterion scores surveyed in [19] to compute the community score of U_{PC} for each cluster PC. Figure 6 compares the results of DCPGS and its alternatives on Gowalla (Katz is omitted because its result is quite similar to that of CommuteTime), in terms of the *internal density* and *conductance* scores. We group the clusters discovered by each method by size and compute and plot the average community score (i.e., internal density and conductance) for each cluster size group. The results based on the other six criteria of [19] are similar and we omit them due to lack of space. The internal density of U_{PC} is defined by $1 - m_{U_{PC}}/(|U_{PC}|(|U_{PC}| - 1)/2)$, where $m_{U_{PC}}$ is the number of edges, which belongs to E and whose two endpoints are both in U_{PC}, $m_{U_{PC}} = |\{(u, v)|u \in U_{PC}, v \in U_{PC}, (u, v) \in E\}|$. Conductance is the fraction of edges, which belongs to E, from nodes of U_{PC} that point outside U_{PC}, i.e., $o_{U_{PC}}/(2m_{U_{PC}} + o_{U_{PC}})$, where $o_{U_{PC}} = |\{(u, v)|u \in U_{PC}, v \notin U_{PC}, (u, v) \in E\}|$. Let $f(U_{PC})$ be the community score of PC, based on either internal density or conductance; a smaller value of $f(U_{PC})$ indicates better social quality.


(a) Internal Density


(b) Conductance


(c) Internal Density

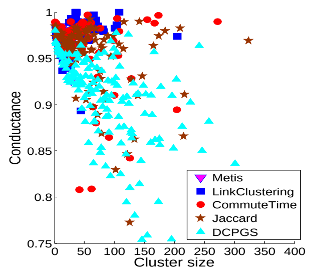

(d) Conductance

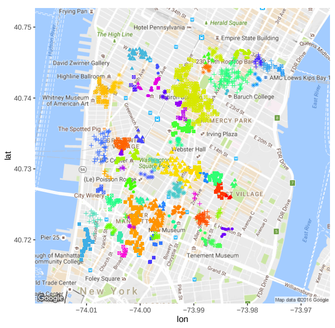
Fig. 6. Community score evaluation in Gowalla

As Figures 6(a) and 6(c) show, the internal density increases with the cluster size. As the size of a place cluster increases, the denominator of the internal density formula increases quadratically while the number of social links between users in the cluster (i.e., the numerator) does not increase at the same pace. On the other hand, Figures 6(b) and 6(d) show that conductance initially decreases as the size of a cluster increases and fluctuates randomly for larger U_{PC} sizes, which is in line with the observations in [19]. In Figure 6, we observe that the geo-social clusters discovered by DCPGS have better community scores (i.e., lower internal density and conductance scores) compared to all competitors, except PureSocialDistance. Since PureSocialDistance uses our social distance in clustering and disregards spatial proximity, its social quality is expected to be better than that of DCPGS; still, as shown in Figure 2(c), its clusters are not distinguishable spatially. DCPGS outperforms DBSCAN and SimRank-based DCPGS. The quality gap between DCPGS and the 3 competitors in Figure 6(a) and 6(b) narrows as the size of clusters increases, since it is more difficult for a larger U_{PC} (usually obtained from a larger PC) to maintain a community-like structure compared to a smaller U_{PC} [19]. This indicates that our social distance is effective in finding geo-social clusters with small or medium size. DCPGS is also generally better than the four competitors in Figures 6(c) and 6(d). CommuteTime has better community scores when the cluster size is around 50. Most community scores of Jaccard, CommuteTime, LinkClustering, and Metis concentrate at the top-left corner of Figures 6(c) and 6(d), which indicates that these competitors have limited ability to discover geo-social clusters of various sizes. Furthermore, the quality gap between DCPGS and the five competitors in Figures 6(c) and 6(d) grows when the cluster size increases. We conclude that DCPGS (paired with our social distance measure) is the most effective method in finding geo-social clusters with both good social quality and identifiable spatial contour.

3.3 Temporal Effects on Geo-Social Clusters

In this section, we illustrate the discovered temporal-geo-social clusters in the area of Manhattan on the Gowalla dataset and in the area of Chicago on the Brightkite dataset using the three methods introduced in Section 2.3.

History-Frame Geo-Social Clustering. Figure 7 shows the temporal-geo-social clusters found using continuous history frame method in periods 01/08/2009–31/01/2010, 01/02/2010–31/07/2010, and 01/08/2010-31/01/2011 in the area of Manhattan. We observe that some clusters evolve over time. For instance, the cluster in region A first expanded and then shrinked. During period 01/02/2010–31/07/2010, there are multiple clusters in region B, while in period 01/08/2010-31/01/2011, those clusters are merged into one big cluster. In region C, before there exists no cluster, while later a new cluster appeared. Figure 8 shows the temporal-geo-social clusters found using the periodic history frames, i.e., on working days and weekends. We observe different place clusters on working days and weekends which is expected, since normally people visit places related to work on working days, while visit entertainment places on weekends.


(a) Working Days

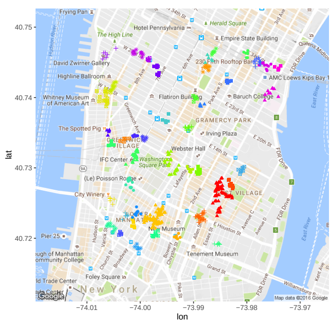
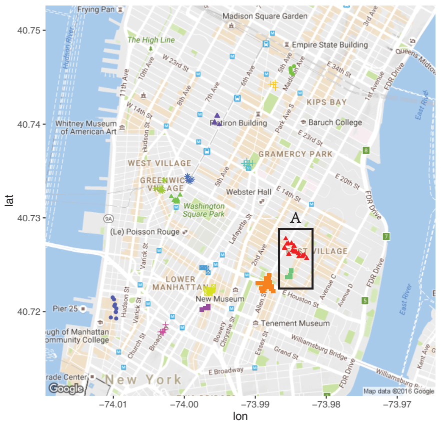
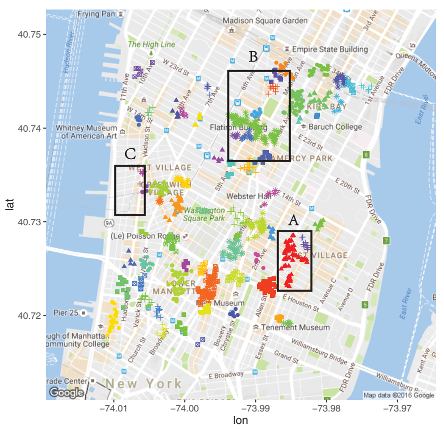

(b) Weekends

Fig. 8. Periodic History Frame Geo-Social Clustering in Manhattan: ϵ = 0.5, $MinPts$ = 5, $maxD$ = 120, τ = 0.7, ω = 0.5.

Temporally Contributing Users. Figure 9 shows the temporal-geo-social clusters found in Manhattan when parameter θ of temporally contributing users is set to 1 week, 1 month, and 6 months, respectively. When θ is a short time interval (e.g., 1 week), the result reveals that the places in the same clusters are revisited by socially connected users within a short period of time. As θ increases, the temporal-social distances between more places decrease, thus as expected (1) some clusters expand, (2) multiple clusters merge (e.g., regions A and B), (3) new clusters are found (e.g., region C). For marketing and management purpose, people may be interested in both the short term and long term clusters.

Damping Window. In Figure 11, we show the temporal-geo-social clusters found by the damping window method in Manhattan, where the starting time is 01/02/2009 and the current time is set to 31/07/2010 and 31/01/2011, respectively. To better understand the effect of the damping window, Figure 10 shows the clusters found by DCPGS on the same data in the same periods as the damping window method. We observe that both the size of the clusters and the number of clusters found by the damping window are smaller than DCPGS. This is expected, since weighing the old data less increases the temporal-social distances between places. Under the same parameter setting, more regions in the damping window are considered as sparse. However, the advantage of the damping window is offering the up-to-date clustering result. As we


(a) 01/08/2009–31/01/2010


(b) 01/02/2010–31/07/2010

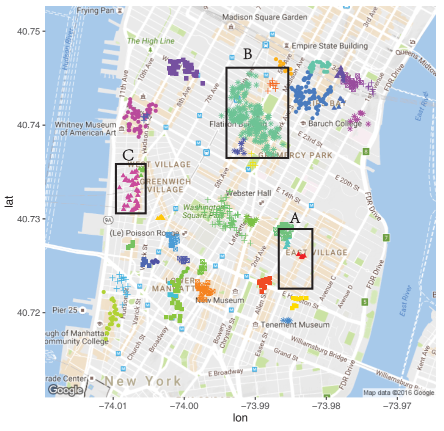
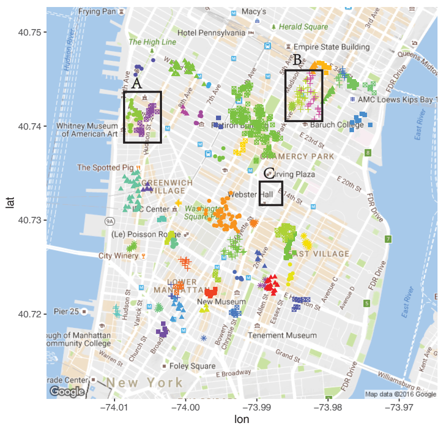
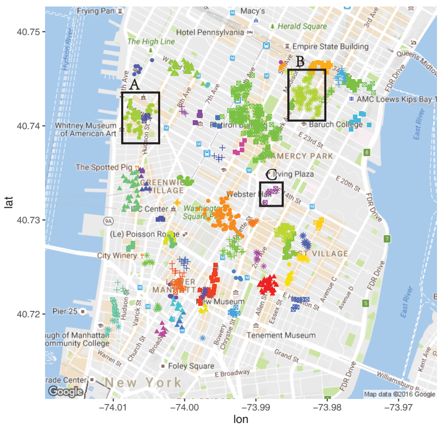

(c) 01/08/2010-31/01/2011

Fig. 7. Continuous History Frame Geo-Social Clustering in Manhattan: $\epsilon = 0.5$, $MinPts = 5$, $maxD = 120$, $\tau = 0.7$, $\omega = 0.5$.


(a) θ = 1 week


(b) θ = 1 month

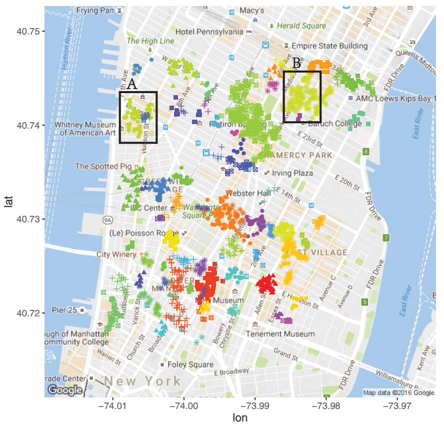
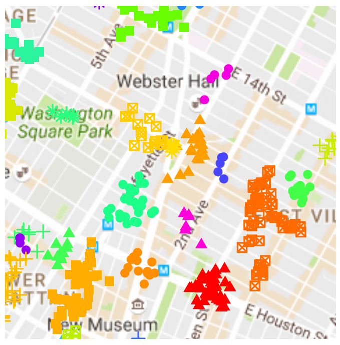

(c) θ = 6 months

Fig. 9. Temporally Contributing Users in Manhattan: $\epsilon = 0.5$, $MinPts = 5$, $maxD = 120$, $\tau = 0.7$, $\omega = 0.5$

can see in Figure 11, the discovered clusters for the two periods are different, which are sensitive to time. For instance, by 31/07/2010, clusters are found on Lafayette St., while by 31/01/2011, clusters are found close to Washington Square Park. Nevertheless, it is not easy to notice these interesting up-to-date small clusters in Figure 10 because of the accumulated old data.

In summary, the three ways of considering the temporal information in the geo-social clustering yield different temporal-geo-social clusters, which may serve various purposes of analysis and investigation. The history-frame method offers the evolution of clusters over time. The damping window method generates the up-to-date clusters. The temporally contributing user method shows the places that are revisited within a period of time. More results from the area of Chicago on the Brightkite data set can be found in Appendix A.

Social Entropy based Evaluation Figure 12 shows the social entropy of the temporal-geo-social clusters discovered by the three methods introduced in Section 2.3 compared with the social entropy of the geo-social clusters found by DCPGS. Recall that a low social entropy means that the great majority of the visitors of a place cluster come from the same community, indicating that the places in the cluster have tighter social relationships between each other. We observe that the clusters found by the damping window method (DCPGSDW-Exp) and the temporally contributing user method (DCPGSTT) have better (lower) social entropy, compared to the result found by DCPGS that does not consider the temporally information. Furthermore, the improvement achieved by damping window method is larger than that of the temporally contributing user method. However, the social entropy of the


(a) 01/02/2009–31/07/2010

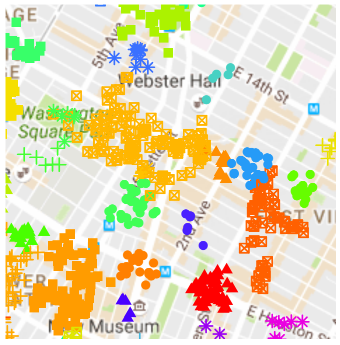
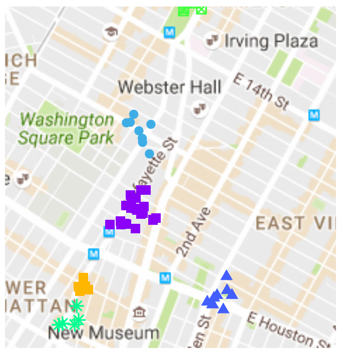
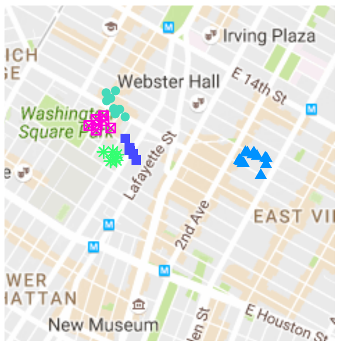

(b) 01/02/2009-31/01/2011

Fig. 10. DCPGS in Manhattan: $\epsilon = 0.5$, $MinPts = 5$, $maxD = 120$, $\tau = 0.7$, $\omega = 0.5$

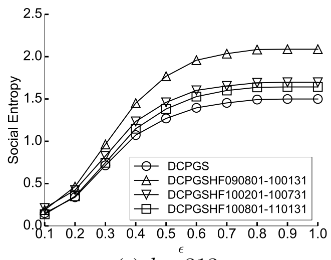


(a) 01/02/2009–31/07/2010

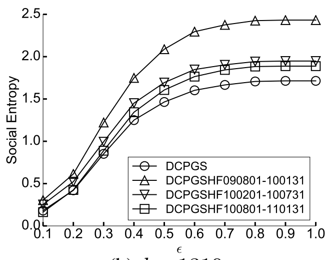


(b) 01/02/2009-31/01/2011

Fig. 11. Damping Window in Manhattan: $\epsilon = 0.5$, $MinPts = 5$, $maxD = 120$, $\tau = 0.7$, $\omega = 0.5$

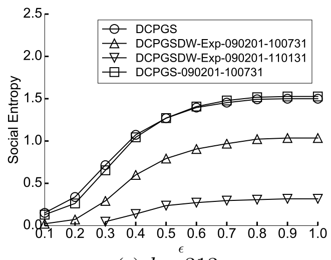


(a) k = 313

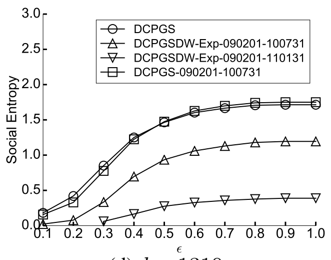


(b) k = 1310

History-Frame Geo-Social Clustering

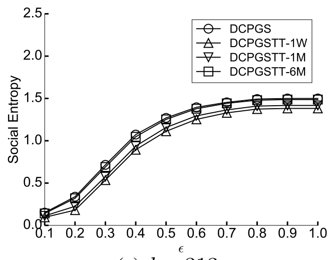


(c) k = 313

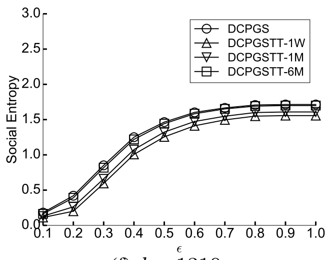


(d) k = 1310

Damping Window



(e) k = 313



(f) k = 1310

Temporally Contributing Users

Fig. 12. Social entropy of temporal-geo-social clusters in Gowalla

clusters found by the history frame method (DCPGSHF) is worse than the result of DCPGS. DCPGSHF in fact splits the whole data into several sub-datasets based on time frames and performs clustering on these sub-datasets. The social entropies of the clusters from these sub-datasets should not necessarily be smaller than the clusters from the whole data. DCPGSTT and DCPGSDW-Exp require more intra temporal closeness among places within the temporal-geo-social cluster. The smaller entropies of these two methods indicates that temporal constraints can enhance the social connections within a cluster. According to the analysis in previously sections, the result of the community score based evaluation is consistent with the result of the social entropy based evaluation, and thus is omitted.

4 Related Work

Our clustering problem is related to various research topics, including traditional spatial clustering, using mobility data to analyze places, clustering using spatial and non-spatial attributes, studying the relationship between spatial and social attributes, community detection, and other work on GeoSNs.

Spatial Clustering. Spatial clustering algorithms, surveyed in [20], are divided into three categories: *partitioning, hierarchical* and *density-based* clustering. Partitioning methods, including k-means, k-medoids, and CLARANS [21], are good at finding spherical-shaped clusters in small and medium-sized datasets. They need a pre-defined parameter k to specify the number of clusters obtained. However, partitioning methods are not able to detect clusters of arbitrary shapes. Hierarchical clustering techniques, such as BIRCH [22], Chameleon [23] and CURE [24], assign objects to clusters in two fashions: *agglomerative* (bottom-up) and *divisive* (top-down). Hierarchical clustering methods do not have well-defined termination criteria and cannot correct the result if some objects are assigned to the wrong clusters at an early stage. Density-based clustering methods, like DBSCAN [1], [25], discover clusters of arbitrary shapes and sizes. Objects in dense regions are grouped as clusters, while objects in sparse regions are labeled as outliers. OPTICS [26] is an extension of DBSCAN, which generates an augmented ordering of the dataset that captures its density-based clustering structure at different granularities. DENCLUE [27] models the overall point density analytically as the sum of influence functions of the data points. Clusters can then be identified by determining density-attractors and clusters of arbitrary shapes can be easily described by a simple equation based on the overall density function. Later, an adaptive method [28] that automatically determines the parameter ϵ of DBSCAN is proposed. GDBSCAN [29] is a generalization of DBSCAN that clusters point objects as well as spatially extended objects according to both their spatial and their non-spatial attributes. A-DBSCAN [30] is an anytime density-based clustering algorithm which is applicable to many complex data such as trajectory and medical data. It uses a sequence of lower bounding functions of the true distance function to produce multiple approximate results of the true density-based clusters. Recently, Gan and Tao [31] discussed hardness of DBSCAN and proposed an efficient approximate version. Our work adopts the density-based clustering framework to find place clusters in a geo-social network, by considering both the spatial distance and the social coherence of the places.

Analysis of Places based on Mobility Data. Brilhante et al. [10] detect "communities" of places of interest (POIs) on a map based on how strongly the places are correlated in sequences of visits by mobile users. Different from our work, the social relationships between the users who visit the places and the spatial distances between places are disregarded. The co-existence of places in the visiting histories of users is the only criterion used for clustering. The proposed solution generates a graph G that connects pairs POIs according to the nature of their co-existence in sequences of user visits and then employs a classic algorithm for community detection on G to identify the place communities. Andrienko et al. [32] present an analysis and visualization tool, which first identifies interesting events of moving objects (e.g., instances of slow car movements), then spatially clusters these events, using density-based clustering to derive a set of significant places (e.g., regions where traffic jams occur), and finally applies visual analytics to aggregate and analyze the events with respect to parameters such as location, time and direction of movement. Noulas et al. [33] perform an empirical analysis of the topological properties of place networks formed by the trajectories of mobile users. They note their resemblance to online social networks in terms of heavy-tailed degree distributions, triadic closure mechanisms and the small world property.

Clustering based on Spatial and Non-Spatial Attributes. Clustering objects based on spatial and non-spatial attributes finds applications in different areas, such as computer vision, GIS, and social networks. Yu et al. [34] cluster pixels considering both the RGB color vectors and spatial proximity that is useful in natural image segmentation. Gennip et al. [35] use spectral clustering to identify communities in a graph where nodes are gang members and weighted edges indicate the gang members' social interactions and geographic locations. Zhang et al. [36] apply clustering by adjusting the spatial distance between two objects according to the non-spatial attribute values between them. EBSCAN [37] clusters georeferenced big data based on not only spatial information but also human behavior derived from geographical features.

Spatial-Social Relationship. The relationship between geography and social structure has been long studied by sociologists. Researchers have found that the likelihood of friendship with a person is decreasing with distance, which has been observed within colleges [38], new housing developments [39], and projects for the elderly [40]. Scellato et al. [4] performed a quantitative study on the socio-spatial properties of users in GeoSNs. By utilizing social and spatial properties of GeoSNs, the same research group proposed a link prediction model [5]. Backstrom et al. [41] predict the location of an individual from a sparse set of known user locations using the relationship between geography and friendship. Wang et al. [15] find that the similarity between the movements of two individuals strongly correlates with their proximity in the social network. This correlation is used as a tool for link prediction in a social network. Pham et al. [42] propose an entropy-based model (EBM) that not only infers social connections but also estimates the strength of social connections by analyzing people's co-occurrences in space and time. Different from existing work, we neither study the spatial-social relationship nor do prediction or recommendation utilizing this relationship. We perform density-based clustering of GeoSN places considering both the spatial distances between them and the social relationships between users that visit the places.

Detecting and Evaluating Communities in Networks. There are many existing works on network community detection and clustering of nodes in a graph using only the network distance between nodes [43], [44], [45], [46]. SCAN [45] is an algorithm that detects clusters, hubs, and outliers in networks. [46] proposed partitioning, hierarchical and density-based algorithms to cluster objects on spatial networks, based on shortest-path distance. [19] summarized and empirically evaluated algorithms for network community detection. In Section 3.2.2, we have used network community quality measures [19] to evaluate the social quality of the place clusters found by our algorithms.

Importing Time into Clustering. Previous works on spatio-temporal data clustering typically include the time concept in the data or algorithms as a threshold or as a parameter of the distance function. Some works transform the spatio-temporal clustering problem to a multi-sequence spatial data clustering problem [47]. The history-frame clustering method that we propose is similar to multi-sequence clustering. Trasarti [48] imported time to data as a new dimension. Thus clustering is performed on high dimensional data using standard distance measure such as the Euclidean distance. In this method, the time effect may be reduced compared with many other dimensions. ST-DBSCAN [49] improves DBSCAN to cluster spatialtemporal data where a time period attached to the spatial data expresses when it was valid or stored in the database. During clustering, spatio-temporal data are filtered by retaining only the temporal neighbors and their corresponding spatial values. Two objects are temporal neighbors if the values of these objects are observed in consecutive time units such as consecutive days in the same year or in the same day in consecutive years. Similarly, [50], [51], [52] integrate the temporal information into the distance function used for clustering. In other words, two objects have to be close in terms of time (e.g., time difference lower than 1 hour) in order to belong to the same cluster. Different from existing works that simply set a time threshold and use it as an additional filtering step when computing distances between objects, the damping window and temporally contributing user methods in the paper take the temporal information and the users' social relationships into account simultaneously.

5 Conclusion

In this paper, we studied for the first time the problem of Density-based Clustering Places in Geo-Social Networks (DCPGS). Our clustering model extends the density-based clustering paradigm to consider both the spatial and social distances between places. We defined a new measure for the social distance between places, considering the social ties between users that visit them. Our measure is shown to be more effective to compute, compared to more complex ones based on node-to-node graph proximity and SNN-based model. We analyzed the effectiveness of DCPGS via case studies and demonstrated that DCPGS can discover clusters with interesting properties (i.e., barrier-based splitting, spatially loose clusters, clusters with fuzzy boundaries), which cannot be found by merely using spatial clustering. To improve the quality of clusters, we incorporate the temporal information of the checkins using three different ways that satisfy different analysis and investigation requirements. Besides, we designed two evaluation measures to quantitatively evaluate the social quality of clusters detected by DCPGS or competitors, called social entropy and community score, which also confirm that DCPGS is more effective than alternative approaches and the temporal dimension further improves the quality of the clusters.

Acknowledgment

This work was supported in part by NSFC grant No. 61502310 and by the European Unions Horizon 2020 research and innovation programme under grant agreement No 657347.

References

[1] 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.

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

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

[4] S. Scellato, A. Noulas, R. Lambiotte, and C. Mascolo, "Socio-spatial properties of online location-based social networks," in *ICWSM*, 2011.

[5] S. Scellato, A. Noulas, and C. Mascolo, "Exploiting place features in link prediction on location-based social networks," in *KDD*, 2011.

[6] 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.

[7] N. Armenatzoglou, S. Papadopoulos, and D. Papadias, "A general framework for geo-social query processing," *PVLDB*, vol. 6, no. 10, pp. 913–924, 2013.

[8] L. Ertöz, M. Steinbach, and V. Kumar, "Finding clusters of different sizes, shapes, and densities in noisy, high dimensional data," in *SDM*, 2003, pp. 47–58.

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

[10] 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.

[11] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM Journal on scientific Computing*, vol. 20, no. 1, pp. 359–392, 1998.

[12] S. Milgram, "The small world problem," *Psychology today*, vol. 2, no. 1, pp. 60–67, 1967.

[13] J. Leskovec, J. M. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *TKDD*, vol. 1, no. 1, 2007.

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

[15] D. Wang, D. Pedreschi, C. Song, F. Giannotti, and A.-L. Barabasi, "Human mobility, social ties, and link prediction," in *KDD*, 2011.

[16] C. Wang, V. Satuluri, and S. Parthasarathy, "Local probabilistic models for link prediction," in *ICDM*, 2007.

[17] P. Sarkar and A. W. Moore, "A tractable approach to finding closest truncated-commute-time neighbors in large graphs," in *UAI*, 2007.

[18] K. V. Mardia, J. Kent, and J. Bibby, *Multivariate analysis (probability and mathematical statistics)*. Academic Press London, 1980.

[19] J. Leskovec, K. J. Lang, and M. W. Mahoney, "Empirical comparison of algorithms for network community detection," in *WWW*, 2010.

[20] J. Han and M. Kamber, *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[21] R. T. Ng and J. Han, "Efficient and effective clustering methods for spatial data mining," in *VLDB*, 1994.

[22] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: An efficient data clustering method for very large databases," in *SIGMOD*, 1996.

[23] G. Karypis, E.-H. Han, and V. Kumar, "Chameleon: Hierarchical clustering using dynamic modeling," *IEEE Computer*, vol. 32, no. 8, pp. 68–75, 1999.

[24] S. Guha, R. Rastogi, and K. Shim, "Cure: An efficient clustering algorithm for large databases," in *SIGMOD*, 1998.

[25] A. Gunawan, "A faster algorithm for dbscan," *Master Thesis*, Technische University Eindhoven, March 2013.

[26] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: Ordering points to identify the clustering structure," in *SIGMOD*, 1999.

[27] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *KDD*, 1998.

[28] S. Jahirabadkar and P. Kulkarni, "Short communication: Algorithm to determine epsilon-distance parameter in density based clustering," *Expert Syst. Appl.*, vol. 41, no. 6, pp. 2939–2946, 2014.

[29] J. Sander, M. Ester, H.-P. Kriegel, and X. Xu, "Density-based clustering in spatial databases: The algorithm gdbscan and its applications," *Data Min. Knowl. Discov.*, vol. 2, no. 2, pp. 169–194, 1998.

[30] S. T. Mai, X. He, J. Feng, C. Plant, and C. Böhm, "Anytime density-based clustering of complex data," *Knowl. Inf. Syst.*, vol. 45, no. 2, pp. 319–355, 2015.

[31] J. Gan and Y. Tao, "Dbscan revisited: Mis-claim, un-fixability, and approximation," in *SIGMOD*, 2015, pp. 519–530.

[32] G. L. Andrienko, N. V. Andrienko, C. Hurter, S. Rinzivillo, and S. Wrobel, "Scalable analysis of movement data for extracting and exploring significant places," *IEEE TVCG*, vol. 19, no. 7, pp. 1078–1094, 2013.

[33] A. Noulas, B. Shaw, R. Lambiotte, and C. Mascolo, "Topological properties and temporal dynamics of place networks in urban environments," in *WWW*, 2015, pp. 431–441.

[34] Z. Yu, O. C. Au, R. Zou, W. Yu, and J. Tian, "An adaptive unsupervised approach toward pixel clustering and color image segmentation," *Pattern Recognition*, vol. 43, no. 5, pp. 1889–1906, 2010.

[35] Y. van Gennip, B. Hunter, R. Ahn, P. Elliott, K. Luh, M. Halvorson, S. Reid, M. Valasik, J. Wo, G. E. Tita, A. L. Bertozzi, and P. J. Brantingham, "Community detection using spectral clustering on sparse geosocial data," *CoRR*, vol. abs/1206.4969, 2012.

[36] B. Zhang, W. J. Yin, M. Xie, and J. Dong, "Geo-spatial clustering with non-spatial attributes and geographic non-overlapping constraint: A penalized spatial distance measure," in *PAKDD*, 2007.

[37] S. Yokoyama, A. Bogárdi-Mészöly, and H. Ishikawa, "Ebscan: An entanglement-based algorithm for discovering dense regions in large geo-social data streams with noise," in *LBSN*, 2015, pp. 7:1–7:10.

[38] J. Q. Stewart, "An inverse distance variation for certain social influence," *Science*, vol. 93, no. 2404, pp. 89–90, 1941.

[39] L. Festinger, S. Schachter, and K. Back, "Social pressures in informal groups: a study of human factors in housing," *Stanford Univ. Pr.*, 1963.

[40] L. Nahemow and M. Lawton, "Similarity and propinquity in friendship formation," *Journal of Personality and Social Psychology*, vol. 32, no. 2, pp. 205–213, 1975.

[41] L. Backstrom, E. Sun, and C. Marlow, "Find me if you can: Improving geographical prediction with social and spatial proximity," in *WWW*, 2010.

[42] H. Pham, C. Shahabi, and Y. Liu, "Ebm: an entropy-based model to infer social strength from spatiotemporal data," in *SIGMOD*, 2013.

[43] C. Tantipathananandh, T. Y. Berger-Wolf, and D. Kempe, "A framework for community identification in dynamic social networks," in *KDD*, 2007.

[44] A. Clauset, M. E. Newman, and C. Moore, "Finding community structure in very large networks," *Physical review E*, vol. 70, no. 6, 2004.

[45] X. Xu, N. Yuruk, Z. Feng, and T. A. J. Schweiger, "Scan: a structural clustering algorithm for networks," in *KDD*, 2007.

[46] M. L. Yiu and N. Mamoulis, "Clustering objects on a spatial network," in *SIGMOD*, 2004.

[47] H. F. Tork, "Spatio-temporal clustering methods classification," in *DSIE*, 2012.

[48] R. Trasarti, "Mastering the spatio-temporal knowledge discovery process," Ph.D. dissertation, Department of Computer Science, University of Pisa, 2010.

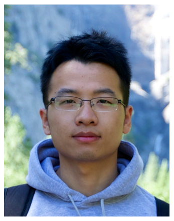
[49] D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatial-temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, 2007.

[50] M. Wang, A. Wang, and A. Li, "Mining spatial-temporal clusters from geo-databases," in *ADMA*, 2006, pp. 263–270.

[51] G. Andrienko and N. Andrienko, "Interactive cluster analysis of diverse types of spatiotemporal data," *SIGKDD Explor. Newsl.*, vol. 11, no. 2, pp. 19–28, 2010.

[52] D. Fitrianah, A. N. Hidayanto, H. Fahmi1, J. L. Gaol, and A. M. Arymurthy, "St-agrid: A spatio temporal grid density based clustering and its application for determining the potential fishing zones," *IJSEIA*, vol. 9, no. 1, 2015.

Dingming Wu is an assistant professor with College of Computer Science & Software Engineering, Shenzhen University, China. She received the Ph.D. degree in Computer Science in 2011 from Aalborg University, Denmark. Her general research area is data management and mining, including data modeling, database design, and query languages, efficient query and update processing, indexing, and mining algorithms.



Jieming Shi received his bachelors degree of science from the department of Computer Science and Technology in Nanjing University, 2011, and his PhD in computer science from the University of Hong Kong. He is now a staff researcher at Lenovo big data lab in Hong Kong. His research interests include geo-social network mining, knowledge graph management, and query processing on spatial-textual data.

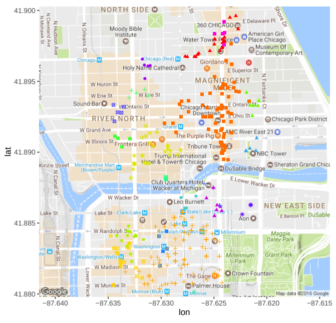
Nikos Mamoulis received the diploma in computer engineering and informatics in 1995 from the University of Patras, Greece, and the PhD degree in computer science in 2000 from HKUST. He is currently a faculty member at the Department of Computer Science and Engineering, University of Ioannina. His research focuses on the management and mining of complex data types, privacy and security in databases, and uncertain data management.

Appendix A
Temporal-Geo-Social Cluster Evaluation

History-Frame Geo-Social Clustering. Figures 17 show the place clusters found using continuous history frame method in periods 01/03/2008–31/08/2008, 01/09/2008–28/02/2009, and 01/03/2009-31/08/2009 in Chicago. Similar to the result of Manhattan, clusters change with time. Take region A as an example. The cluster first shrinks and then becomes multiple clusters. Figure 13 shows the temporal-geo-social clusters found using the periodic history frames, i.e., on working days and weekends. As expected, different place clusters on working days and weekends are discovered.

Temporally Contributing Users. Figure 14 shows the temporal-geo-social clusters found in Chicago when parameter θ of temporally contributing users is set to 1 week, 1 month, and 6 months, respectively. The result of Chicago has similar behaviors as the result of Manhattan does, i.e., expanded cluster in region A, merged clusters in region B, and newly found clusters in region C.

Damping Window. In Figure 16, we show the temporal-geo-social clusters found by the damping window method in Chicago, where the starting time is 01/03/2008 and the current time is set to 28/02/2009 and 28/02/2010, respectively. To better understand the effect of the damping window, Figure 15 shows the clusters found by DCPGS on the same data in the same periods as the damping window method. The result in Chicago is consistent with the result in Manhattan. Less and smaller clusters are found using damping window method. There is a cluster close to W Madison St, which is ignored by DCPGS.


(a) Working Days

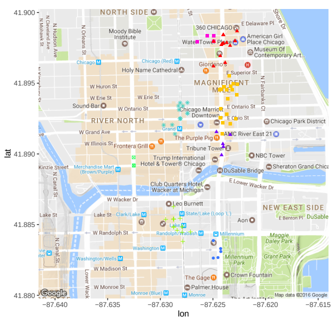

(b) Weekends

Fig. 13. Periodic History Frame Geo-Social Clustering in Chicago: ϵ = 0.5, $MinPts$ = 5, $maxD$ = 120, τ = 0.7, ω = 0.5.

Appendix B
Algorithms of DCPGS

We propose two algorithms for DCPGS. DCPGS-R (Section B.1) is based on the R-tree index, while DCPGS-G (Section B.2) uses a grid partitioning.

B.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 in the GeoSN 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.

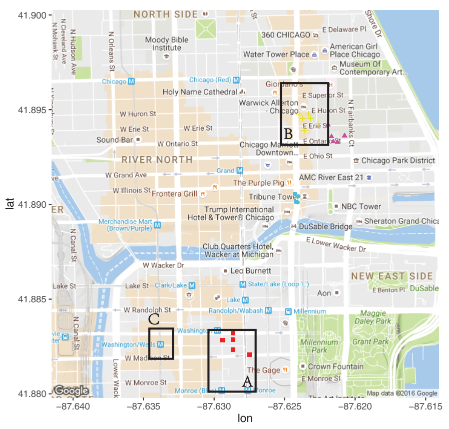
For the sake of efficiency, the social network is stored in a hash table. Specifically, each pair of friends in the social network is recorded as an entry in the hash table, such that checking whether two users are friends or not only incurs constant cost. In addition, for each place p_i, we keep track of its visiting users U_{p_i}. The computation of the social distance (Definition 2) between two places p_i and p_j involves finding the pairs of friends between sets U_{p_i} and U_{p_j} and has insignificant cost compared to the range queries used to compute the set of candidate places.

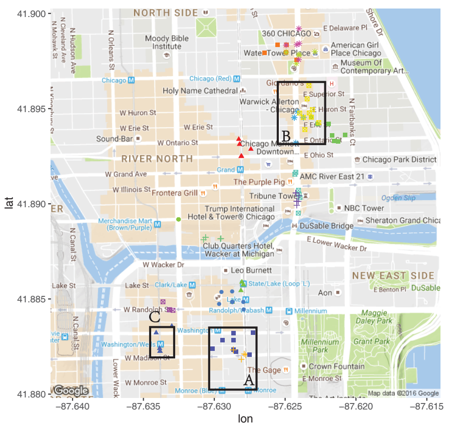
Algorithm 1 is the pseudocode of DCPGS-R. The identity of the current cluster cid is initialized to 1 in line 1. Queue Q (initialized in line 2) stores the places that have the potential to be added to the current cluster. Hash table H records whether the geo-social distances between pairs of places have been computed before (line 3) and its use will be explained later. For each *unprocessed* place p_i, its geo-social ϵ-neighborhood $N_\epsilon(p_i)$ is obtained by calling function GetNeigh(p_i, ϵ, τ, $maxD$, $MinPts$, ω, H), outlined in Algorithm 2. A place is *unprocessed* if its geo-social ϵ-neighborhood has not been computed before. If $N_\epsilon(p_i)$ contains at least $MinPts$ places, then p_i is a *core* place, i.e., p_i belongs to a cluster and all places in $N_\epsilon(p_i)$ should be given the same cid as p_i (lines 6-11). Next, all unprocessed places in $N_\epsilon(p_i)$ are pushed into Q for later processing. Lines 12-20 expand the current cluster cid as much as possible by checking the geo-social ϵ-neighborhood of the unprocessed places in Q. No more places can be included in the current cluster when Q is empty. In this case, the algorithm proceeds to find the next cluster (cid is increased in line 21).

Algorithm 1 DCPGS-R(GeoSN, ϵ, τ, $maxD$, $MinPts$, ω)

```
1: $cid$ = 1
2: $Q$ = $empty$
3: Geo-social distance cache $H$
4: for each $unprocessed$ place $p_i$ in GeoSN do
5:     $N_\epsilon(p_i)$ = GetNeigh($p_i$, $\epsilon$, $\tau$, $maxD$, $MinPts$, $\omega$, $H$)
6:     if $|N_\epsilon(p_i)| \geq MinPts$ then
7:         assign $cid$ to $p_i$
8:         for each place $p_j \in N_\epsilon(p_i)$ do
9:             assign $cid$ to $p_j$
10:            if $p_j$ is $unprocessed$ then
11:                $Q.push(p_j)$
12:    while $!Q.isEmpty()$ do
13:        $p_k$ = $Q.pop()$
14:        if $p_k$ is $unprocessed$ then
15:            $N_\epsilon(p_k)$ = GetNeigh($p_k$, $\epsilon$, $\tau$, $maxD$, $MinPts$, $\omega$, $H$)
16:            if $|N_\epsilon(p_k)| \geq MinPts$ then
17:                for each place $p_m \in N_\epsilon(p_k)$ do
18:                    assign $cid$ to $p_m$
19:                    if $p_m$ is $unprocessed$ then
20:                        $Q.push(p_m)$
21:    $cid$ = $cid$ + 1
```

Function GetNeigh, shown in Algorithm 2, is used to get the geo-social ϵ-neighborhood of place p_i. Initially $N_\epsilon(p_i)$ is empty. Lines 2-4 first perform a spatial range query centered at the current place p_i with radius $maxD$ to get a candidate set $CandSet$ containing the places that may fall in $N_\epsilon(p_i)$. If the size of $CandSet$ is less than $MinPts$, $N_\epsilon(p_i)$ definitely includes less than $MinPts$ places and, therefore, p_i is a non-


(a) θ = 1 week


(b) θ = 1 month

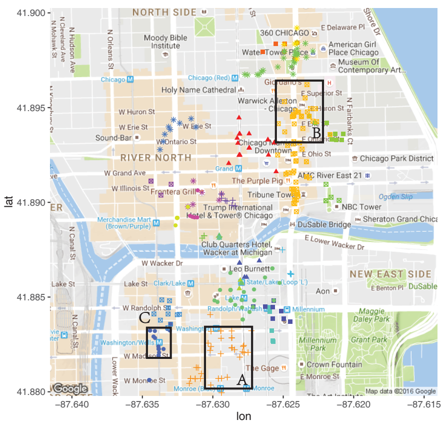
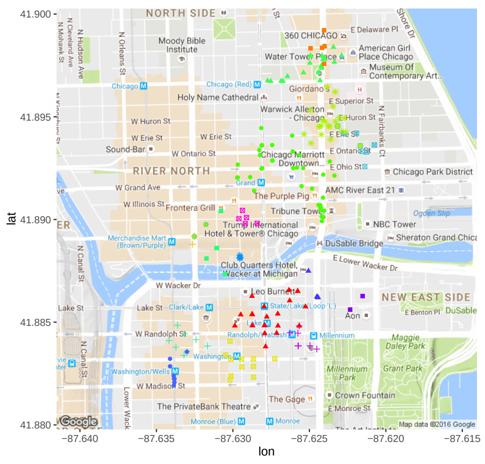

(c) θ = 6 months

Fig. 14. Temporally Contributing Users in Chicago: $\epsilon = 0.5$, $MinPts = 5$, $maxD = 120$, $\tau = 0.7$, $\omega = 0.5$


(a) 01/03/2008–28/02/2009

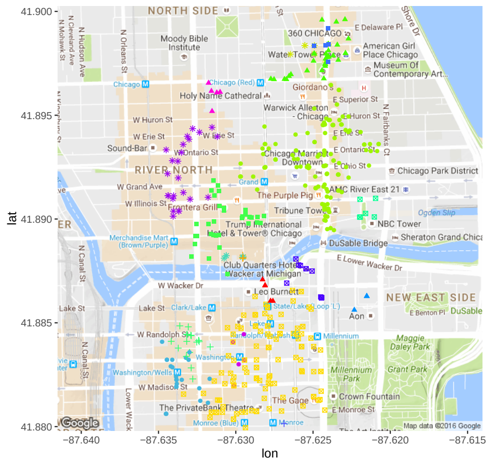
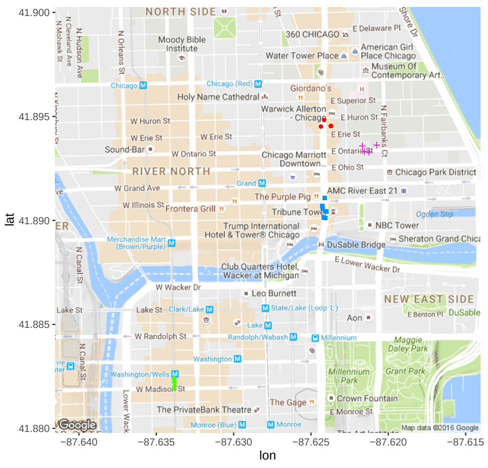

(b) 01/03/2008-28/02/2010

Fig. 15. DCPGS in Chicago: $\epsilon = 0.5$, $MinPts = 5$, $maxD = 120$, $\tau = 0.7$, $\omega = 0.5$


(a) 01/03/2008–28/02/2009

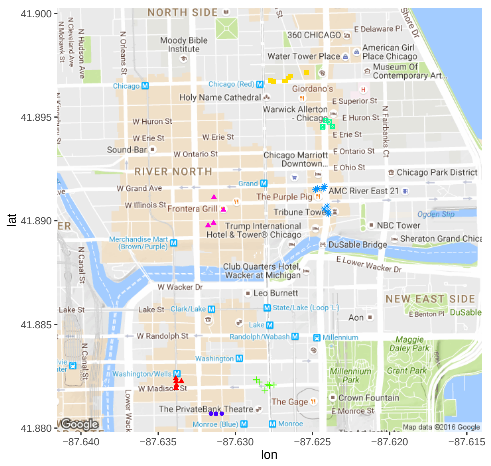
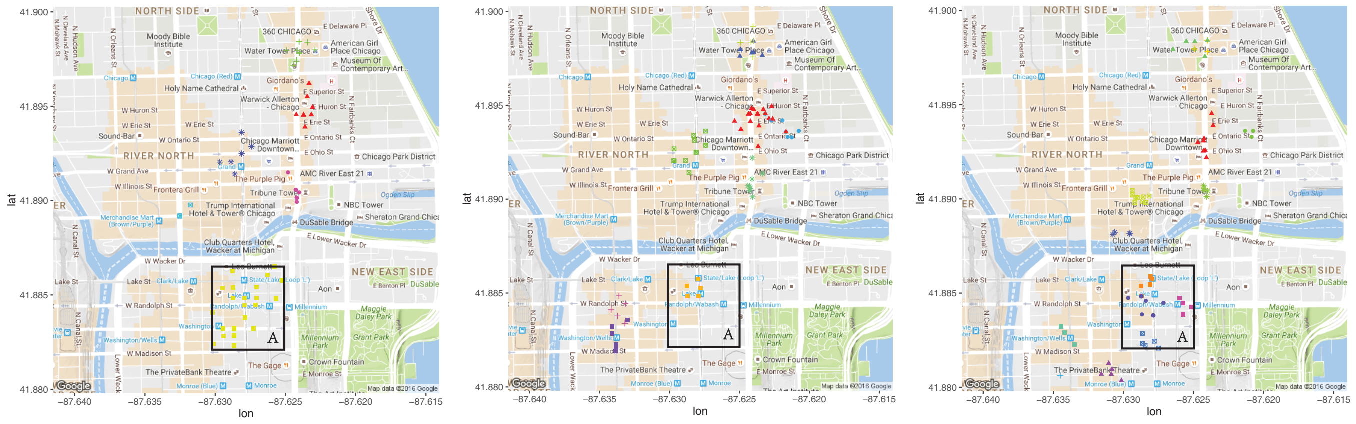

(b) 01/03/2008-28/02/2010

Fig. 16. Damping Window in Chicago: $\epsilon = 0.5$, $MinPts = 5$, $maxD = 120$, $\tau = 0.7$, $\omega = 0.5$

core place. Otherwise, the algorithm tries to compute the social distance between every point $p_j \in CandSet$ and p_i. However, before this, given the fact that the spatial distance between p_i and every candidate place p_j is already obtained in the spatial range query step, a *spatial filter* is employed to avoid unnecessary social distance computations (line 6).

Proposition 1. *Spatial filter. Given two places p_i and p_j with spatial distance $D_P(p_i, p_j)$, if $\omega \cdot D_P(p_i, p_j) > \epsilon$, then $\omega \cdot D_P(p_i, p_j) + (1 - \omega) \cdot D_S(p_i, p_j) > \epsilon$.*

Proof. Since $\omega \in [0, 1]$ and $D_S(p_i, p_j) \in [0, 1]$, $(1 - \omega) \cdot D_S(p_i, p_j) \geq 0$. Consequently, if $\omega \cdot D_P(p_i, p_j) > \epsilon$, then $\omega \cdot D_P(p_i, p_j) + (1 - \omega) \cdot D_S(p_i, p_j) > \epsilon$. □



(a) 01/03/2008–31/08/2008 (b) 01/09/2008–28/02/2009 (c) 01/03/2009-31/08/2009

Fig. 17. Continuous History Frame Geo-Social Clustering in Chicago: $\epsilon = 0.5$, $MinPts = 5$, $maxD = 120$, $\tau = 0.7$, $\omega = 0.5$.

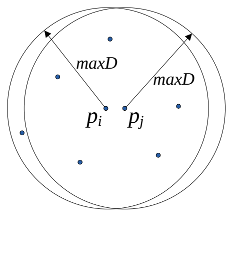
Note that distances D_{gs} and D_S and D_P are all symmetric. Hence, given a place p_i, if p_j is (not) in $N_\epsilon(p_i)$, then p_i is also (not) in $N_\epsilon(p_j)$, and vice versa. Therefore, we keep track in a hash table H (line 14) whether p_i and p_j are in each other's geo-social ϵ-neighborhood once their geo-social distance has been computed. This information is used when computing the geo-social ϵ-neighborhood of p_j later (line 7-9), in order to avoid computing the distance between the same pair of places twice. Lines 11-15 compute distances, verify candidates, and update H if necessary. Line 16 is another filter, called *sufficient filter*, to check whether there are enough remaining candidate places in $CandSet$ to render p_i's geo-social ϵ-neighborhood dense. If no, the algorithm can stop without checking the remaining candidate places and return.

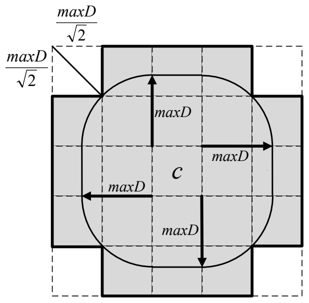
Algorithm 2 GETNEIGH(p_i, ϵ, τ, $maxD$, $MinPts$, ω, H)

```
1:  $N_\epsilon(p_i) \leftarrow \varnothing$
2:  $CandSet$ = RANGEQUERY($p_i, maxD$)
3:  if $|CandSet| < MinPts$ then
4:      return $N_\epsilon(p_i)$
5:  for each place $p_j \in CandSet$ do
6:      if $\omega \cdot D_P(p_i, p_j) \leq \epsilon$ then
7:          if $H.exists((p_i, p_j))$ then
8:              if $H[(p_i, p_j)]$ is $TRUE$ then
9:                  $N_\epsilon(p_i).insert(p_j)$
10:         else
11:             Compute $D_S(p_i, p_j)$ and $D_{gs}(p_i, p_j)$
12:             if $D_S(p_i, p_j) \leq \tau$ && $D_{gs}(p_i, p_j) \leq \epsilon$ then
13:                 $N_\epsilon(p_i).insert(p_j)$
14:                 $H[(p_i, p_j)] \leftarrow TRUE$
15:     $CandSet.erase(p_j)$
16:     if $|CandSet| + |N_\epsilon(p_i)| < MinPts$ then break
17: return $N_\epsilon(p_i)$
```

B.2 Algorithm DCPGS-G: Grid based

DCPGS-R conducts a spatial range query for each place in the GeoSN 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 18(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, to get the candidate places $CandSet$ for each place in Figure 18(a), 8 independent range queries are issued on the R-tree that search 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

Fig. 18. Nearby spatial range queries and grid partitioning

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

Neighbor Cells. We define the neighbor cells $NC(c)$ of cell c as the cells (excluding c) that intersect the Minkowski region of c, using $maxD$ as radius and excluding c itself. In practice, the Minkowski region of c can be defined by the region defined by the four 1/4-circles of the 4 corners, plus the straight lines that are parallel to the 4 edges of the cell and at distance $maxD$. As an example in Figure 18(b), $NC(c)$ contains the 20 gray cells surrounding c.

Cluster Discovery. Algorithm 3 is a pseudocode for DCPGS-G. It includes three phases. First, the algorithm maps all places into grid cells (line 1). Second, it obtains the geo-social ϵ-neighborhoods of all places (lines 2-6). The third phase discovers all geo-social clusters in the GeoSN (line 7). The geo-social ϵ-neighborhoods of places are computed at the grid cell level. Specifically, for each nonempty and *unprocessed* cell c, function GETNEIGHCELLS retrieves its neighbor cells $NC(c)$ (GETNEIGHCELLS is trivial and thus details are omitted). A cell is 'unprocessed' if its neighbor cells have not been retrieved before. Function COMPCELLPAIR (Algorithm 4) first filters out the pairs of places (p_i, p_j) with spatial distance greater than $maxD$, where $p_i \in c, p_j \in NC(c)$ and $p_i \neq p_j$. This step is not needed if p_i and p_j are in the same cell (in this case, the spatial distance between p_i and p_j is certainly at most $maxD$). Next, the pairs of places (p_i, p_j) that satisfy the social distance constraint τ and the geo-social distance threshold ϵ are selected. If p_i and p_j are in each others' geo-social ϵ-neighborhood, their corresponding $N_\epsilon(p_i)$ 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, function GETCLUSTERS is called to discover geo-social clusters following the framework of DCPGS-R (Algorithm 1), except that the $N_\epsilon(p_i)$ of each place p_i has already been computed. Note that an *unprocessed* place p_i in function GETCLUSTERS means that the size of p_i's geo-social ϵ-neighborhood, $|N_\epsilon(p_i)|$, has not been checked.

Algorithm 3 DCPGS-G(GeoSN, $\epsilon, \tau, maxD, MinPts, \omega$)

```
1: Map all places into grid cells
2: for each $nonempty$ && $unprocessed$ cell $c$ do
3:     COMPCELLPAIR$(c, c, \epsilon, \tau, maxD, \omega)$
4:     $NC(c)$=GETNEIGHCELLS$(c)$
5:     for each $nonempty$ && $unprocessed$ cell $c' \in NC(c)$ do
6:         COMPCELLPAIR$(c, c', \epsilon, \tau, maxD, \omega)$
7: GETCLUSTERS($N_\epsilon$, $MinPts$)
```

Algorithm 4 COMPCELLPAIR(cell c, cell c',$\epsilon, \tau, maxD, \omega$)

```
1: for each pair $(p_i, p_j)$ where $p_i \in c, p_j \in c', p_i \neq p_j$ do
2:     if $c = c'$ —— $E(p_i, p_j) \leq maxD$ then
3:         Compute $D_P(p_i, p_j)$
4:         if $\omega \cdot D_P(p_i, p_j) \leq \epsilon$ then
5:             Compute $D_S(p_i, p_j)$ and $D_{gs}(p_i, p_j)$
6:             if $D_S(p_i, p_j) \leq \tau$ && $D_{gs}(p_i, p_j) \leq \epsilon$ then
7:                 $N_\epsilon(p_i).insert(p_j)$
8:                 $N_\epsilon(p_j).insert(p_i)$
```

Algorithm 5 GETCLUSTERS(N_ϵ, $MinPts$)

```
1: $cid = 1$
2: $Q = empty$
3: for each unprocessed place $p_i$ in GeoSN do
4:     if $|N_\epsilon(p_i)| \geq MinPts$ then
5:         assign $cid$ to $p_i$
6:         for each place $p_j \in N_\epsilon(p_i)$ do
7:             assign $cid$ to $p_j$
8:             if $p_j$ is unprocessed then
9:                 $Q.push(p_j)$
10:     while $!Q.isEmpty()$ do
11:         $p_k = Q.pop()$
12:         if $p_k$ is unprocessed then
13:             if $|N_\epsilon(p_k)| \geq MinPts$ then
14:                 for each place $p_m \in N_\epsilon(p_k)$ do
15:                     assign $cid$ to $p_m$
16:                     if $p_m$ is unprocessed then
17:                         $Q.push(p_m)$
18:     $cid = cid + 1$
```

B.3 Complexity Analysis.

Let P be the set of places and $n = |P|$. For a place $p \in P$, let $|U_p|$ be the number of users visited p. The worst-case complexities of both the DCPGS-R and DCPGS-G algorithms are $O(\sum_{p_i, p_j \in P \wedge p_i \neq p_j} |U_{p_i}| \times |U_{p_j}|)$. This is because in the worst-case, both the algorithms have to compute the geo-social distances for $\Omega(n^2)$ pairs of places, where the cost of computing $D_{gs}(p_i, p_j)$ is $O(|U_{p_i}| \times |U_{p_j}|)$. However, if one assumes that the input dataset has a property that the average number of users visited a place is $O(1)$, then the complexity decreases to $O(n^2)$. Moreover, if one further assumes that in the grid-based algorithm, each cell contains at most $O(1)$ places, the complexity of this algorithm becomes $O(n)$ since in this case, there can be at most $O(n)$ pairs of places that will be checked. It should be noted that these two bounds only hold under the above assumptions. Below is the analysis based on the characteristics of the data sets used.

In the two data sets used for experiments, the average number of check-ins per place is 5 in Gowalla and 6 in Brightkite. Hence the cost of computing $D_{gs}(p_i, p_j)$ can be considered as a constant cost. Then we derive that the complexity of DCPGS-R is the same as that of DBSCAN, which is $O(n^2)$.

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. For any cell c, there are at most 20 neighbor cells, $NC(c)$. (i) Among the cells in $NC(c)$, in total there are at most 10 cells that are above c, or in the same row of c but on c's left. For each of such cells, noted as $c_<$, we need to retrieve places in c once when it is processed, specifically when Algorithm 4 is called for cell pair $(c_<, c)$. (ii) When c itself is under processing, all its places need to be retrieved once. (iii) For all the neighbor cells of c that are below c, or in the same row of c but on c's right, noted as $c_>$, when they are under processing, there is no need to call Algorithm 4 for cell pair $(c_>, c)$ since cell pair $(c, c_>)$ has all ready been computed by Algorithm 4 when c is under processing. Therefore, in total, any cell is accessed at most 11 times, which is a constant, during the process of computing all places' geo-social neighborhoods. On average, in our experiment, the number of places per cell is around 1.65 when $maxD = 120$. In addition, a cell contains at most 615 places which is significant smaller than the number of places in the data set. Hence, for each place, the number of comparisons in a cell is a constant. Taking the fact that the average cost of computing the geo-social distance is a constant, the complexity of DCPGS-G is $O(n)$ (as each of its three phases makes one pass over the data).

B.4 Algorithms for Temporal-Geo-Social Clustering

Algorithms DCPGS-R and DCPGS-G can be easily applied to compute temporal-geo-social clusters. Specifically, the results of the history-frame method are obtained by applying one of the two algorithms on the data that belong to each time period. For the damping window and the temporally contributing user method, both the two algorithms can be used with the only modification that substituting the social distance with the temporal-social distance between places.

APPENDIX C EFFICIENCY EVALUATION

Effect of ϵ. Figures 19(a) and 19(b) show the performance of all methods when varying ϵ, except LinkClustering and Metis, which do not use parameter ϵ. We keep $\tau = 0.7$ and $\omega = 0.5$ for both datasets and $maxD = 100$ for Gowalla and $maxD = 120$ for Brightkite. Note that DCPGS-G is faster than DCPGS-R in all cases, which again indicates that grid partitioning greatly improves the efficiency of DCPGS. DCPGS-G (DCPGS-R) is slightly more expensive than Jaccard-G (Jaccard-R). The runtimes of SimRank-G, SimRank-R, Katz-G, and CommuteTime-G increase significantly when ϵ increases from 0.1 to 0.5, because the increase weakens the power of the spatial filter (Proposition 1) and more (expensive) social distances ($D_S^{sim}(p_i, p_j)$, $D_S^{Katz}(p_i, p_j)$, and $D_S^{ct}(p_i, p_j)$) are computed. DCPGS-G and DCPGS-R remain quite stable for all ϵ values since the computational cost of our social distance is insignificant.

Effect of $maxD$ (eps in DBSCAN). Parameter $maxD$ (eps in DBSCAN) decides the spatial ranges of the geo-social (spatial) neighborhoods of places and the size of the constructed place network for LinkClustering and Metis. Thus, $maxD$ (eps) greatly influences the efficiency of all methods. Figures 19(c) and 19(d) display the runtime of all methods when varying $maxD$ (eps). We set τ = 0.7, ϵ = 0.4, and ω = 0.5 for all the methods except DBSCAN on both Gowalla and Brightkite datasets. Note that SimRank-G, SimRank-R, Katz-G and CommuteTime-G are more sensitive to $maxD$ compared to the other methods, due to the high cost of $D_S^{sim}(p_i, p_j)$, $D_S^{Katz}(p_i, p_j)$, and $D_S^{ct}(p_i, p_j)$ distance computations.

The cost of DCPGS-G stays below 100 seconds as $maxD$ ranges from 10 to 300 and remains close to that of DBSCAN-G and Jaccard-G. DCPGS-R is also close to DBSCAN-R and Jaccard-R. DCPGS-G remains faster than DCPGS-R, but their performance gap narrows with $maxD$; the reason is that grid-based computations become more expensive as the grid cell size (i.e., $maxD$) increases. DCPGS-R eventually becomes faster than DCPGS-G for a very large $maxD$, e.g. $maxD$ > 1000; however, as shown in Figure 2(c), geo-social clusters with a very large $maxD$ are not spatially identifiable. At an appropriate $maxD$ value (around 100), DCPGS-G is clearly the best choice. Metis has similar cost to DCPGS-G while LinkClustering is slower.

Effect of ω. Figures 19(e) and 19(f) display the runtimes of the clustering methods when varying ω. We keep τ = 0.7 and ϵ = 0.4 for all methods on both datasets, while setting $maxD$ = 100 in Gowalla and $maxD$ = 120 in Brightkite. The costs of all methods are not much sensitive to ω; an exception is the runtimes of SimRank-G, SimRank-R, Katz-G, and CommuteTime-G which decrease when ω > 0.5. The reason is again the high effect of the spatial filter (Proposition 1), when ω is large, due to which many expensive social distance computations are avoided. As a final note, the social distance constraint τ is only applied as an ultimate filter and has no influence on the runtimes of all methods. It only slightly influences the costs of LinkClustering and Metis because it affects the size of the place network, on which these methods operate.

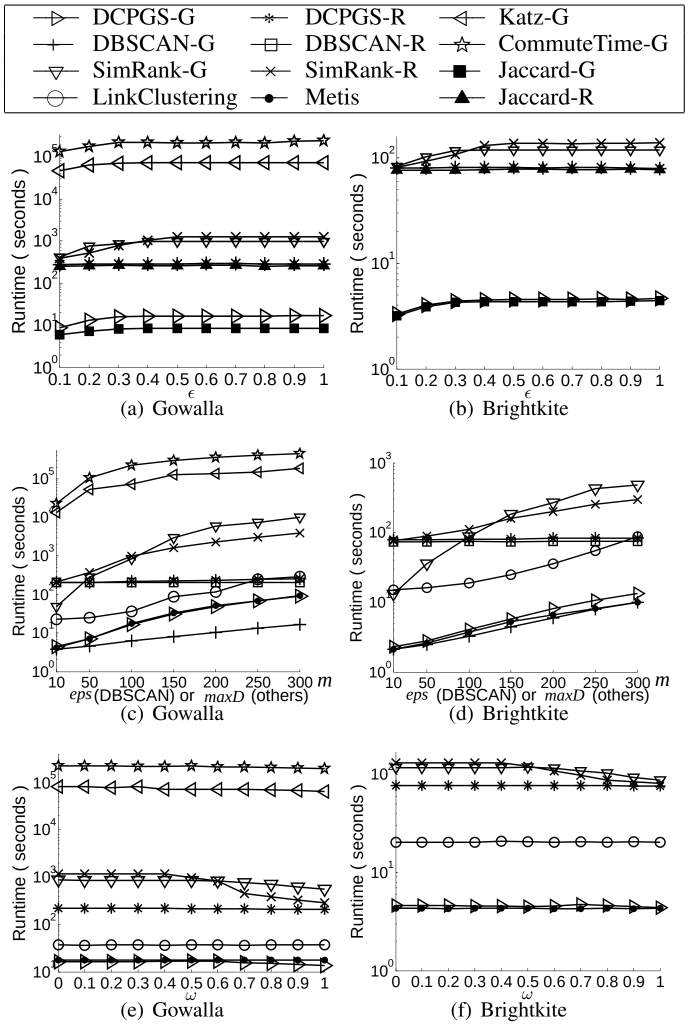


Fig. 19. Runtime experiments