GeoScop: A Scalable Approach to Geo-social Clustering of Places

Shivam Srivastava, Shiladitya Pande, Jithin Vachery, Sayan Ranu

Abstract—Spatial clustering of places is a well-researched area. However, when it comes to considering the semantics of these places, the work done is rather sparse. In this paper, we develop an algorithm to cluster places not only based on their locations but also their semantics. Specifically, two places are considered similar if they are spatially close and visited by people of similar communities. With the explosion in the availability of location-tracking technologies, it has become easy to track locations and movements of users through user "check-ins". These check-ins provide insights into the community structure of people visiting a place, which is leveraged and integrated into the proposed *geo-social* clustering framework called *GeoScop*. While community detection is typically done on social networks, in our problem, we lack any network data. Rather, two people belong to the same community if they visit similar geo-social clusters. We tackle this chicken-and-egg problem through an iterative procedure of *expectation maximization* and *DBSCAN*. Extensive experiments on real check-in data demonstrate that GeoScop mines semantically meaningful clusters that cannot be found by using any of the existing clustering techniques. Furthermore, GeoScop is up to 6 times more pure in social quality than the state-of-the-art technique in this space.

Index Terms—clustering, check-ins, DBSCAN, expectation maximization.

1 Introduction

Many third parties collect, or have the ability to collect, user location data. Credit card companies record every purchase, network providers record every connection to a cell tower, transport providers record journeys and reviewer websites record every destination. This data comprises many distinct locations and the associated user. The technique of spatial clustering involves grouping spatially close locations into *clusters*. Spatial clustering is an unsupervised learning procedure to group places into clusters, such that those within a cluster are similar, and places across clusters are dissimilar. Spatial clustering is widely used for urban planning [1], marketing [2], and traffic congestion modeling [3]. One of the most popular approaches to spatial clustering is density-based clustering [4], [5], which groups places into sets of densely populated regions. The popularity of density-based clustering is primarily driven by the fact that it can automatically learn the number of clusters and that the clusters can be of arbitrary shapes and sizes. Traditionally, the distance function between two places quantifies the spatial distance between them and ignores any semantic similarity. In certain applications, such a distance function leads to spurious clusters.

To illustrate, consider Fig. 1. Each dot denotes a "check-in" by a user. Check-ins are essentially records of users visiting a certain location and are denoted by tuples $\langle u, x, y, t \rangle$, where u is the ID of the user, x and y encode the spatial coordinates (such as latitude and longitude), and t denotes the timestamp at which the visit occurred. In this figure, we ignore the timestamp and u for brevity. Let the blue check-ins represent visits by students and the green check-ins represent tourists. If the check-ins are clustered using traditional spatial-clustering algorithms, then all check-ins in the zoo, university, book store, and the library would form a single cluster, and the two white check-ins would be outliers. However, it is easy to see that while the university, library and book store are mostly visited by students, the zoo is frequented by tourists. In other words, although there is one spatial cluster, there are two *geo-social* clusters. A geo-social cluster is a group of places that are similar in geographical locations as well the communities of users visiting them. GeoScop is built on the intuition that people belonging to the same community have similar tastes.

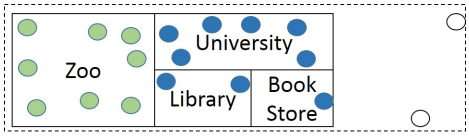


Fig. 1. A scenario where spatial clustering leads to spurious results.

Applications of geo-social clusters lie in multiple areas.

• **Resource Management:** Consider a telecom operator who wants to install k 4G base-stations in a city. To maximize the impact, traditional wisdom tells us to install them in the k largest clusters. However, since 4G services are more expensive than 3G, it is typically used by business class users. Thus, a better solution is to install the base-stations in the k largest clusters visited by business class users.

•**Marketing:** Advertising through billboards is one of the most popular ways to gain eyeballs. Choosing the correct billboards is an important decision in this procedure. Consider an advertising firm that wants to market a new smartphone that is

- *Shivam Srivastava and Shiladitya Pande are with Samsung Research, Bangalore, India. However, the work was done at IIT Madras, India. E-mail: {shivam.srivastava511,shilpande2010}@gmail.com*
- *Jithin Vachery is affiliated to the Department of Computer Science and Engineering, IIT Madras, India. E-mail: jvachery@cse.iitm.ac.in*
- *Sayan Ranu is affiliated to the Department of Computer Science and Engineering, IIT Madras, India. E-mail: sayanranu@cse.iitd.ac.in*

Manuscript received April 19, 2005; revised August 26, 2015.

priced and targeted primarily for the student community. In this scenario, it is imperative that the billboards chosen are located in geo-social clusters popular among students.

• **Collaborative Campaigns:** Mining geo-social clusters allows advertising firms to engage people in collaborative campaigns. For example, users visiting restaurant X would be offered discounts if they visit restaurants Y or Z within the next one month. Collaborative campaigns are attractive to both users for the discounts, and to the restaurants since it drives up their business. Naturally, for a collaborative campaign to be successful, the participating business entities need to belong to the same geo-social cluster. More specifically, visitors of restaurant X would be inclined to visit Y, if they are located in the same neighborhood and serve people having similar tastes. Indeed, we need to identify communities of people who have similar tastes in their check-ins.

In this paper, we develop a technique called *GeoScop (GEO-Social Clustering Of Places)* to mine geo-social clusters from check-in data. Due to the ubiquity of GPS-enabled phones, there has been a spurt of services, such as Yelp and Zomato, which are built on check-in data. In addition, websites such as Foursquare, Facebook, and Google+ encourage users to share their locations publicly. Various forums such as TripAdvisor and Lonely Planet collect check-in data to provide better recommendations. Check-in data is also inferred through involuntary means such as geo-tagged photo albums, credit card transactions, and practically any smartphone application that tracks users through GPS sensors. These check-ins provide a rich form of geographical data, the analysis of which provides new and interesting insights, compared to raw spatial data.

While clustering is a well-studied area, the work done in geo-social clustering is limited. The closest work to our problem is DCPGS [6]. Although the formulation of DCPGS is different, it has the same goal of grouping places based on both spatial distance and social distance. GeoScop differs from DCPGS in the following three fronts.

1. Reliance on a social network: To compute the social distance between places, DCPGS relies on a social network $G = (V, E)$. More specifically, the social distance $sdist(p_i, p_j)$ between two places p_i, p_j, which have been visited by the set of users U_i and U_j respectively, is defined as follows:

$$sdist(p_i, p_j) = 1 - \frac{\|C_{ij}\|}{\|U_i \cup U_j\|} \tag{1}$$

where C_{ij}, the set of *contributing users*.

$$C_{ij} = \{u_a \in U_i | u_a \in U_j \text{ or } u_b \in U_j, (u_a, u_b) \in E\} \cup \{u_a \in U_j | u_a \in U_i \text{ or } u_b \in U_i, (u_a, u_b) \in E\}$$

More simply, if a user visits both the places then it is one of the contributing users. Furthermore, if user u_a visits p_i and u_b visits p_j, where u_a and u_b are friends in the social network, then they are both contributing users as well. Thus, two places are socially close, if they have significant overlap in their users or their 1-hop social circle.

In our problem, we have access to only the check-in data and lack the social network of users. While it is always useful to have additional information in the form of the social network, in many scenarios, this assumption is not realistic. It is well known that only a few companies, such as Facebook and Twitter, have access to large scale social networks. For the majority of the companies like Yelp, Zomato, TripAdvisor, and credit card agencies, the only source of data are the check-ins. Social network owners are known to not share their data due to competitive business advantage, its inestimable value [7], as well as privacy concerns. Therefore, the question that arises is the following: *How do we perform geo-social clustering without knowledge of the explicit network among users?*

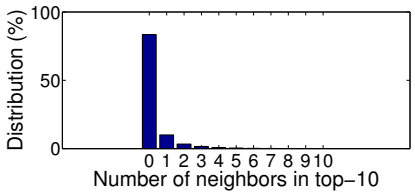


Fig. 2. The distribution of number of friends in the top-10 closest neighbors.

2. Formulation: As visible in Eq. 1, DCPGS assumes that two people have common tastes only if they are within the 1-hop neighborhood of each other. This assumption is too restrictive in real life scenarios. Consider the student population in a university who regularly visits the campus gym. Clearly, they all form a community of health conscious individuals even though direct friendship links across all of them may be missing. If a large section of this population also play tennis in the adjacent tennis courts, then the gym and tennis courts should be part of the same cluster. This community level analysis of users is missing in DCPGS.

To establish the intuition mentioned above, we study the Gowalla geo-social dataset [8]. Gowalla contains both user check-ins and the social network among the users. In this dataset, we take each user, and find its top-10 closest users in terms of places visited. Two check-ins are considered to be the same location if they are within 100 meters of each other. Each user is represented as frequency vector over all locations in the dataset and the similarity between two users is quantified using the MinMax distance measure (Eq. 2), which we discuss in further detail in Sec. 2. Once the top-10 closest neighbors are computed, for each user, we check how many of these top-10 users are also their friends in the social network. Fig. 2 presents the results. As can be seen, a huge majority of the users have no friends in their top-10 closest users. Certainly, one can move to a x-hop neighborhood, $x > 1$, to better capture the social distance. However, as the authors of DCPGS themselves argue and show [6], due to the small-world phenomenon [9], moving beyond 1-hop does not produce good results. What we need is a community-level analysis.

3. Merging multiple check-in data sources: Consider two sources of check-in data that are accompanied by their corresponding social networks. DCPGS cannot mine geo-social clusters from the merged dataset since the social distance can be computed only among users of the same network. In our approach, we do not have this limitation. While we do not perform entity resolution across datasets, our technique is robust enough to group similar users into the same community. Thus, if the same user is indeed present in both datasets, they will be correctly grouped together.

The above analysis highlights the need to de-couple the dependence on a social network. To overcome the consequent challenges, we develop *GeoScop*. The pipeline of GeoScop is outlined in Fig. 3. GeoScop ingests check-in data, which could potentially come from multiple sources. Following some initialization of parameters, the data is fed to an iterative procedure

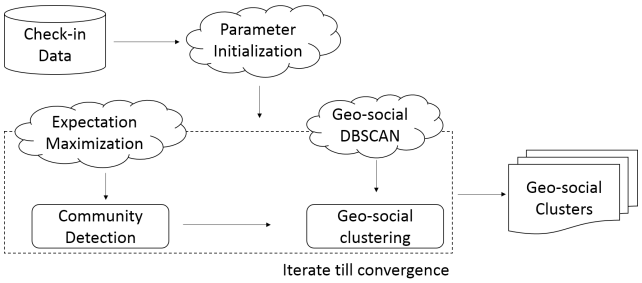


Fig. 3. The pipeline of GeoScop.

of community detection and geo-social clustering. In geo-social clustering, we face a chicken-and-egg problem. Two users are part of the same community if they visit similar geo-social clusters. On the other hand, two places are similar if they are visited by people of similar communities. Although there are connections to *bi-clustering* [10], the problem is not identical. Two users are similar not only if they visit similar places, but only if they are part of the same community. A community consists of users who influence each other, and we learn these communities by modeling the underlying information diffusion process in the invisible social network. Following community detection, geo-social clustering is performed and this process is iterated over till convergence.

In our earlier work [11], we had proposed the idea of GeoScop. In this work, we enhance our study through the following aspects.

- **Scalability:** The existing framework does not scale to large datasets. In this work, we develop techniques to scale geo-social clustering for real life datasets. The scalability is achieved by developing a *sliding-window* based pruning algorithm and by exploiting parallelism.
- **Experiments:** We perform extensive benchmarking that showcases (i) the impact of the pruning algorithm on the scalability of GeoScop, and (ii) the ability of GeoScop to run on heterogeneous data originating from multiple sources.
- **Theory:** A significantly more detailed description of the theory behind GeoScop.

To summarize, the contributions of our work are as follows.

- We formulate the problem of *geo-social clustering* from check-in data. As a by-product of our problem, we also mine *geo-social* communities, which itself is another unique contribution.
- We develop a technique called *GeoScop* to mine geo-social clusters from check-in data without relying on a social network. GeoScop derives its power by employing the unique combination of expectation maximization with DBSCAN in an iterative manner.
- Due to the expensive computation costs of expectation maximization, we develop a *sliding-window* based pruning algorithm to scale GeoScop. In addition, we exploit parallelism to further lower the running time.
- Extensive experiments on real check-in data establish that GeoScop is able to mine clusters that are semantically meaningful and would not be found by any of the existing clustering algorithms. Furthermore, the social quality of the clusters mined by GeoScop is up to 6 times more pure than the state-of-the-art techniques.
- We establish the impact of the proposed sliding-window based pruning algorithm on GeoScop in real datasets. We show that the proposed framework is more than an order of magnitude faster than the naive implementation.

The rest of the paper is organized as follows. In Sec. 2, we formulate the problem. In Sec. 3, we develop the GeoScop clustering algorithm. Owing to the non-scalability of the naive implementation, in Sec. 4, we develop data structures to search more efficiently and in a parallelized manner. We establish the utility of GeoScop on real datasets in Sec. 5. We discuss the related work in Sec. 6 and finally conclude in Sec. 7.

2 Problem Formulation

The data input to GeoScop is a set of *check-ins*.

Definition 1. CHECK-IN. *A check-in is a tuple $\langle u, x, y, t\rangle$, where u represents the user id of the person generating the visit, x,y are the spatial coordinates of the location, and t is the timestamp at which the visit occurred.*

Given a dataset of check-ins $\mathbb{D}$, the set of places $\mathcal{S}$ contains all the unique locations in $\mathbb{D}$ in terms of their spatial co-ordinates. Thus, alternatively, a check-in can be represented as $\langle u, p, t\rangle$, where $p \in \mathcal{S}$. Hereon, we use this shortened notation for check-ins. Similar to places, we use the notation $\mathcal{U}$ to denote the set of unique users in $\mathbb{D}$.

Since our goal is to perform geo-social clustering among places, we define a distance function for the geographical world, denoted as $gdist(p_1, p_2)$ and the social world, denoted as $sdist(p_1, p_2)$. $gdist(p_1, p_2)$ is simply the Euclidean distance between p_1 and p_2. To define the social distance, assume there is some oracle that partitions all users in the check-in dataset into disjoint sets of *communities* $\mathbb{C} = \{C_1, \cdots, C_m\}$, where $C_i \in \mathbb{C}$ is a set of users, and $C_i \cap C_j = \emptyset$. Each community contains users of similar tastes. We now construct a community profile $pr(p)$ for each place $p \in \mathcal{S}$.

Definition 2. COMMUNITY PROFILE. *The community profile $pr(p)$ of a place p is a vector of dimension $\|\mathbb{C}\|$. The i^{th} dimension, denoted as $pr(p)[i]$, contains the number of visits to p from community $C_i \in \mathbb{C}$. Mathematically, $pr(p)[i] = \|\{u \in \mathcal{U} | \exists \langle u, p, t\rangle \in \mathbb{D},\ u \in C_i\}\|$.*

Based on the community profile, we next define the social distance between places.

Definition 3. SOCIAL DISTANCE. *Given, the community profiles of places p_1 and p_2, the social distance $sdist(p_1, p_2)$ is defined as the following.*

$$sdist(p_1, p_2) = 1 - \frac{\sum_{\forall C_i \in \mathbb{C}} \min\{pr(p_1)[i], pr(p_2)[i]\}}{\sum_{\forall C_i \in \mathbb{C}} \max\{pr(p_1)[i], pr(p_2)[i]\}} \quad (2)$$

Intuitively, two places are socially close, if they are visited by users from similar communities. Notice that we do not rely on direct friendships as required in DCPGS [6]. Additionally, the communities, representing people of similar tastes, are learned from the check-in data itself.

With both the geographical and social distance defined, we now define a geo-social cluster. GeoScop is modeled on DBSCAN [4]. A place p is a *core* point if it contains at least $minPts$ points in its *geo-social* neighborhood $\mathcal{N}_{\delta_g, \delta_s}(p)$.

Definition 4. GEO-SOCIAL NEIGHBORHOOD. *Given a geographical distance threshold δ_g and social distance threshold δ_s, the geo-social neighborhood* $\mathcal{N}_{\delta_g,\delta_s}(p) = \{p' \in \mathcal{S} |\ gdist(p,p') \leq \delta_g,\ sdist(p,p') \leq \delta_s\}$.

With the definition of geo-social neighborhood, the definition of a cluster in GeoScop follows as in DBSCAN. Specifically, p' is *directly density reachable* from p if $p' \in \mathcal{N}_{\delta_g,\delta_s}(p)$ and p is a core point. p' is *density reachable* from p if there exists a chain of places $p_1, \cdots, p_n$, such that $p_1 = p$, $p_n = p'$ and each p_{i+1} in the chain is directly density reachable from p_i. Finally, p' and p'' are *density connected* if $\exists p \in \mathcal{S}$, such that both p' and p'' are density reachable from p. Given the set of places $\mathcal{S}$, our goal is to partition it into a set of *geo-social clusters* $\mathbb{P} = \{P_1, \cdots, P_m\}$ with the following properties.

1) $\forall P_i \in \mathbb{P}, \forall p, p' \in P_i$, p is density connected to p'.
2) If $p \in P_i, P_i \in \mathbb{P}$, and p' is density reachable from p, then $p' \in P_i$.

Note that property 2 is violated in some rare cases. For more information, we point readers to [4]. The set of points $\mathcal{S} \setminus \cup_{\forall P_i \in \mathbb{P}} P_i$ are the *outliers*.

Parameters: There are three input parameters in the formulation: $minPts$, δ_g, δ_s. Setting $minPts$ and δ_g have been extensively studied in the clustering literature owing to the popularity of DBSCAN. In city-like conditions, setting $minPts \approx 5$, $\delta_g \approx 100$ meters (m), have been shown to produce good intuitive results [6]. For δ_s, which is introduced in our model, the threshold is automatically learned from the data. Specifically, a random subset of places are selected, and δ_s is set to 80% percentile of all social distances. A deeper analysis of the parameters are performed in Sec. 5.

3 GeoScop

As evident from our formulation, geo-social clustering is straightforward if the set of communities is known. Generally, a community in a social network is a set of users who influence each other, either directly or indirectly, due to shared interests. For our problem, a community is a set of users who have similar interests in visiting places. In other words, if user u visiting a geo-social cluster C increases the chances of user u' also visiting C, then they are part of the same community. u can influence u' directly or indirectly. However, we do not have access to the underlying social network and mine the communities. We need to mine communities from just the check-ins. Towards that goal, we base our inference procedure on the following assumptions.

- There is an unobserved social network with well-defined communities of users. Connectivity among users in the community is dense and across communities is sparse.
- Each user u influences all other members of his/her community C with a certain intensity.
- Users' check-ins are governed by the *independent cascade* model [12]. Specifically, the chances of u visiting a cluster P depends on the influence exerted by the *active* members of C on u. A user in C gets active with respect to cluster P after he/she visits P for the first time.

A recent work called *C-IC* exists in mining communities from user activity data [13]. Each activity is represented by a tuple $\langle u, i, t \rangle$, where i denotes a certain activity such as purchasing a product, sharing a photo, etc. C-IC builds on the assumptions that two users u_1, u_2 are likely to be part of the same community if u_1 performing an activity i increases the chances of u_2 performing i as well. It is easy to see that our problem maps to this same exercise where i represents the place $p \in \mathcal{S}$ visited. However, there is one key difference, which makes a direct application of C-IC inadequate. In our problem, the influence exerted on a user is at the cluster level and not at the place level. This is necessary for two reasons.

1) The granularity of a place, which are the unique $\langle x, y \rangle$ co-ordinates, is extremely small. Two users visiting a zoo will not check-in at the exact same coordinates. The spatial extent of the zoo can be identified by clustering all check-ins from the zoo-loving community.
2) Consider a user who is visiting a mall to shop for clothes and notices a new restaurant in the neighborhood. The user goes back and recommends the restaurant to his/her friends. This is a common scenario in real life. Mathematically, it means that a user u visiting a place p not only influences u's community C to visit p, but also other places that are in p's neighborhood and matches the common interests of C. In other words, all places in the geo-social cluster of P could be affected.

The above analysis highlights the circular dependency between communities and geo-social clusters. Without knowing the communities, we cannot compute the social distance between places and without knowing the clusters, we cannot infer communities. We therefore adapt the C-IC model for our problem of geo-social clustering. As outlined in Fig. 3, GeoScop iterates over community detection and clustering till convergence. Initially, we assign each place to a cluster by performing DBSCAN using only the geographical distance. Next, we mine communities through *Expectation Maximization (EM)*, which allows us to incorporate social distance in the next round of DBSCAN and subsequently, better community detection in future rounds of EM.

3.1 Mining Communities

Let user u visit a geo-social cluster P, and soon after, we observe user u' also visiting P. Due to the small time difference in their visits, we hypothesize that u may have influenced u' and therefore they may belong to the same community. Now, if this event happens multiple times across various clusters, then the probability of these two users lying in the same community should further increase. In the description that follows, we see how it is modeled to arrive at communities.

We hypothesize that actions of users are governed by a set of parameters Θ. The likelihood of the check-in data $\mathbb{D}$ is therefore expressed as a function of Θ.

$$\mathcal{L}(\Theta; \mathbb{D}) = \prod_{\forall u \in \mathcal{U}} P(u|\Theta) \tag{3}$$

where $P(u|\Theta)$ is the likelihood of u's check-ins relative to the parameters. Our learning problem is to find $\hat{\Theta} = \arg\max_{\Theta}\{\mathcal{L}(\Theta; \mathbb{D})\}$.

We assume u's check-ins are dependent on u's community membership. To model this, we have a hidden binary variable $m_{u,i}$ denoting users u's membership in community i with the constraint that $\sum_{i=1}^{K} m_{u,i} = 1$, where K is the total number of communities. The parameter set Θ can therefore be partitioned

into $\{\pi_1, \cdots, \pi_K, \Theta_1, \cdots, \Theta_K\}$, where $\pi_i = \frac{\sum_{\forall u \in \mathcal{U}} m_{u,i}}{\|\mathcal{U}\|}$ represents the prior probability $P(m_{u,i}) = 1$ and Θ_i is the parameter set for community C_i. Eq. 3 is therefore expressed as follows.

$$\mathcal{L}(\Theta; \mathbb{D}) = \prod_{\forall u \in \mathcal{U}} \sum_{i=1}^{K} P(u|\Theta_i)\pi_i \tag{4}$$

We optimize this likelihood through the standard EM algorithm. The data likelihood can be expressed as

$$P(\mathbb{D}, \mathbf{M}, \Theta) = P(\mathbb{D}|\mathbf{M}, \Theta).P(\mathbf{M}|\Theta).P(\Theta) \tag{5}$$

where

$$P(\mathbb{D}|\mathbf{M}, \Theta) = \prod_{\forall u \in \mathcal{U}} \prod_{i=1}^{K} P(u|\Theta_i)^{m_{u,i}} \tag{6}$$

$$P(\mathbf{M}|\Theta) = \prod_{\forall u \in \mathcal{U}} \prod_{i=1}^{K} \pi_i^{m_{u,i}} \tag{7}$$

Following the same approach as in [13], we model $P(\Theta)$, which represents the prior relative to the parameter set Θ, as

$$P(\Theta) \propto \prod_{i}^{K} \pi_i^{-\frac{1}{2}\sqrt{\|\Theta_i\|}} \tag{8}$$

The interpretation for this model is that when K is fixed, which is true in our case, the parameters π_k allow a Dirichlet-type prior. Now, following standard EM manipulation, the *Q-function* is expressed as the following.

$$\begin{aligned} \mathcal{Q}(\Theta|\Theta') &= E[\log P(\mathbb{D}, \mathbf{M}, \Theta|\mathbb{D}, \Theta')] \\ &\propto \sum_{\forall u \in \mathcal{U}} \sum_{i=1}^{K} \gamma_{u,i}\{\log P(u|\Theta_i) + \log \pi_i\} \\ &\quad - \sum_{i=1}^{K} \frac{\sqrt{\|\Theta_i\|}}{2} \log \pi_i \end{aligned} \tag{9}$$

where

$$\gamma_{u,i} = P(m_{u,i} = 1|u, \Theta') \tag{10}$$

$$= \frac{p(u|\Theta_i)\pi_i}{\sum_{j=1}^{K} P(u|\Theta_j)\pi_j} \tag{11}$$

We optimize $\gamma_{u,i}$ using the standard EM procedure. The parameter set Θ is first initialized and the *E* and M steps are performed recursively till convergence.

E Step: Given Θ, estimate $\gamma_{u,i}$ for each u, i.

M Step: Given $\gamma_{u,i}$, maximize the $\mathcal{Q}$-function (Eq.9).

While Θ can be initialized randomly, better solutions and faster convergence is achieved if the initialization is performed in a more supervised manner. We discuss this in Sec. 3.2.

Our task is now to model $P(u|\Theta_i)$. We simulate $P(u|\Theta_i)$ based on the Independent Cascade model (IC). We assume time unfolds in discrete timestamps (in our case 1 second). A user u becomes active with respect to a cluster P after visiting a place in P for the first time. Now, u has a single shot at influencing all other inactive members of u's community in visiting P. When the social network is visible, the pairwise influence of u on another user v can be computed from the edge weights in the networks. Since, we do not have access to the network, we assume that u exerts influence globally on all members of his/her community C_i with a weight $p_u^i \in [0, 1]$. The core idea is that the community-level influence of u is highest in the community u belongs to.

Now, given that u has visited cluster P, we identify the potential users who could have influenced u. These potential influencers are those who visited P within a short time span prior to u. Mathematically, the influencers $F_{u,P}^{+}$ is the following.

$$\begin{aligned} F_{u,P}^{+} = \{v \in \mathcal{U} | \exists \langle u, p_u, t_u \rangle, \langle v, p_v, t_v \rangle \in \mathbb{D}, \\ 0 \leq t_u - t_v \leq \Delta,\ p_u \in P,\ p_v \in P\} \end{aligned} \tag{12}$$

In the same manner, $F_{u,P}^{-}$ defines the users who visited P, but potentially failed to influence u.

$$\begin{aligned} F_{u,P}^{-} = \{v \in \mathcal{U} | \exists \langle u, p_u, t_u \rangle, \langle v, p_v, t_v \rangle \in \mathbb{D}, \\ t_u - t_v \rangle \Delta,\ p_u \in P,\ p_v \in P\} \end{aligned} \tag{13}$$

Thus, $P(u|\Theta_i)$ is defined as:

$$P(u|\Theta_i) = \prod_{\forall P \in \mathbb{P}_u} P_+(P|u, \Theta_k) P_-(P|u, \Theta_k) \tag{14}$$

where $\mathbb{P}_u = \{P \in \mathbb{P} | \exists \langle u, p, t \rangle \in \mathbb{D}, p \in P\}$ is the set of clusters visited by u, $P_+(P|u, \Theta_i)$ is the probability that some of the potential influencers actually influenced u in visiting P, and $P_-(P|u, \Theta_i)$ is the probability that the users in $F_{u,P}^{-}$ did not influence u.

$$P_+(P|u, \Theta_i) = 1 - \prod_{v \in F_{u,P}^{+}} (1 - p_v^i) \tag{15}$$

$$P_-(P|u, \Theta_i) = \prod_{v \in F_{u,P}^{-}} (1 - p_v^i) \tag{16}$$

With the formalization of $P(u|\Theta_i)$, we can now compute $\gamma_{u,i}$ for the E step. The next task is therefore to perform the M step, where we update the set of variables p_u^i for all u and i to maximize the $\mathcal{Q}$-function (Eq. 9). Since the derivation for the updated p_u^i is identical to the C-IC model [13], we skip the proof and directly present the result and the intuition behind it.

$$p_u^i = \frac{\sum_{\{\langle v,P \rangle | u \in F_{v,P}^{+}\}} \gamma_{v,i} * \eta_{P,u,v,i}}{S_{u,i}^{+} + S_{u,i}^{-}} \tag{17}$$

where

$$S_{u,i}^{+} = \sum_{\{\langle v,P \rangle | u \in F_{v,P}^{+}\}} \gamma_{v,i} \tag{18}$$

$$S_{u,i}^{-} = \sum_{\{\langle v,P \rangle | u \in F_{v,P}^{-}\}} \gamma_{v,i} \tag{19}$$

$$\eta_{P,u,v,i} = \frac{p_u^i}{1 - \prod_{w \in F_{v,P}^{+}} (1 - p_w^i)} \tag{20}$$

Here, $\{\langle v, P \rangle | u \in F_{v,P}^{+}\}$ denotes the set of users who have been potentially influenced by u over some cluster P.

We now explain the intuition behind this proof. $\eta_{P,u,v,i}$ stands for the "responsibility" of user u in triggering v's visit to cluster P in the context of the community C_i. Hence, the numerator in Eq. 17 is proportional to the probability of all users who have been influenced by u in community i. The denominator factors in the probabilities that the users who have been influenced by u and those whom u failed to influence truly belong to community i. Thus, to summarize, p_u^i increases if u is responsible for influencing many people in community C_i to visit a cluster already visited by u. Finally, we set the mixture probabilities $\pi_i = \frac{\sum_{\forall u \in \mathcal{U}} \gamma_{u,i}}{\|\mathcal{U}\|}$ and restart the E step.

3.2 Completing the GeoScop pipeline

Revisiting Fig. 3, to complete the pipeline for GeoScop, we need to formulate the initialization module and the clustering module. Alg. 1 lays out the pseudocode for the entire GeoScop algorithm and how these two modules fit in. Initialization is performed once at the start (lines 3-8). We explain the initialization algorithm later in the section. After initialization and EM (E-Step: lines 12-16, M-Step: lines (17-19), we perform geo-social clustering on the places (lines 21-24). Geo-social clustering is straightforward since it simply performs DBSCAN with the additional constraint of social distance on the community profiles of places. Note that following EM, instead of hard communities, we have a membership distribution over the communities for each user. This distribution is captured in $\gamma_{u,i}$. Thus, the community profile (Def. 2) for a place pl is computed as $pr(pl)[i] = \sum_{\forall \langle u,pl,t\rangle \in \mathbb{D},} \gamma_{u,i}$, where i is the dimension representing community C_i (lines 21-23).

The more challenging task is to initialize the parameters. Recall, that EM assumes that clusters are already known. In addition, for EM, we need to initialize the number of communities, the initial values of p_u^i and π_i. While theoretically all these parameters can be randomly assigned, that makes EM prone to converge at a local optima that is much worse than the global optima. We therefore develop a more supervised initialization procedure.

From the formulation of our geo-social clusters, we can guarantee the following.

Theorem 1. *If two points in a geo-social cluster are density connected, then they are also density connected in the geographical world. In other words, a geo-social cluster must be a subset of a geographical cluster.*

Theorem 1 lies at the core of our initialization procedure. Alg. 2 presents the pseudocode. For the initial set of clusters, we simply perform DBSCAN without considering the social aspect (line 3 in Alg. 1) and feed these clusters to the community initialization algorithm (Alg. 2). Next, we perform a crude community detection based on the geographical clusters. From Theorem 1, a community of people who visit similar geo-social clusters is also likely to be visiting similar geographical clusters. We leverage this property. Specifically, for each user u, we construct a *cluster profile*, denoted as $cr(u)$ (line 3). A cluster profile is identical to the idea of community profile. It is a feature vector of dimension size equal to the number of geographical clusters. The i^{th} dimension in a cluster profile contains the number of visits to the i^{th} cluster. The similarity between two users, $usim(u_1, u_2)$ is now quantified using the same MinMax measure.

$$usim(u_1, u_2) = \frac{\sum_{\forall P_i \in \mathbb{P}} \min\{cr(u_1)[i], cr(u_2)[i]\}}{\sum_{\forall P_i \in \mathbb{P}} \max\{cr(u_1)[i], cr(u_2)[i]\}} \quad (21)$$

Now, if we cluster users based on their cluster profiles, we would have a rough estimate of the underlying communities. Following this intuition, we cluster users using Eq. 21 as the similarity function in a manner analogous to DBSCAN. A random user, u, is picked (line 7) and the user's top-k similar users are found (line 13). If the k-th user has a similarity greater than a certain threshold $minSim$, then user u and the top-k users are grouped together into a community (lines 14-15). The intuition here is that in a community, a user must have at least k users who are similar ($k = 20$, $minSim = 0.4$ in our experiments). If not, u must be an outlier. The top-k users are then pushed into a queue (line 16) and one by one, each of them is picked for further expansion of the community (lines 10-16). This is repeated until the queue gets empty. A new user who has not yet been processed is then picked and the procedure is repeated (lines 6-18). Finally, the set of communities is returned (line 19).

Algorithm 1 GeoScop($\mathbb{D}$)

Ensure: return the set of geo-social clusters $\mathbb{P}$
1: $\mathcal{U} \leftarrow$ unique users in $\mathbb{D}$
2: $\mathcal{S} \leftarrow$ unique places in $\mathbb{D}$
3: $\mathbb{P} \leftarrow DBSCAN(\mathcal{S})$
4: $\mathbb{C} \leftarrow CommunityInitialization(\mathbb{D}, \mathbb{P})$
5: **for** $\forall C_i \in \mathbb{C}$ **do**
6: **for** $\forall u \in C_i$ **do**
7: $\gamma_{u,i} = 1$
8: $p_u^i = usim(centroid(C_i, u))$
9: **repeat**
10: $\forall C_i \in \mathbb{C}, \pi_i = \frac{\sum_{\forall u \in \mathcal{U}} \gamma_{u,i}}{\|\mathcal{U}\|}$
11: **repeat**
12: **for** $\forall C_i \in \mathbb{C}$ **do**
13: **for** $\forall u \in \mathcal{S}$ **do**
14: $F_{u,i}^{+} \leftarrow$ positive influencers of u in C_i
15: $F_{u,i}^{-} \leftarrow$ members of C_i who failed to influence u
16: Compute $\gamma_{u,i} \backslash\backslash$ (Eq. 10)
17: **for** $\forall C_i \in \mathbb{C}$ **do**
18: **for** $\forall u \in C_i$ **do**
19: Compute $p_u^i \backslash\backslash$ (Eq. 17)
20: **until** Convergence of parameter set Θ
21: **for** $\forall p \in \mathcal{S}$ **do**
22: **for** $\forall C_i \in \mathcal{C}$ **do**
23: $pr(pl)[i] = \sum_{\forall \langle u,pl,t\rangle \in \mathbb{D},} \gamma_{u,i}$
24: $\mathbb{P} \leftarrow$ geo-social DBSCAN using $pr(pl)$
25: **until** Convergence of $\mathbb{P}$
26: **return** $\mathbb{P}$

Once the initial communities are computed, we initialize p_u^i and π_i. π_i is simply $\frac{\|C_i\|}{\|\mathcal{U}\|}$, where C_i is the i^{th} community (line 7, line 10 in Alg. 1). We use a centroid based approach to initialize p_u^i. For each community C, we calculate the mean cluster profile vector where $cr(centroid)[j] = \sum_{\forall u \in C} \frac{cr[j]}{\|C\|}$. Conceptually, the centroid is assumed to be the most influential a user can get. Thus, p_u^i, for user u and community C_i, is set to $p_u^i = usim(centroid, u)$ (line 8 in Alg. 1). Owing to the MinMax formulation, $usim$ lies in the range $[0, 1]$. In other words, users close to the centroid are set as influential and those lying on the borders do not exert much influence on the community members. As we show later in Sec. 5, the proposed initialization procedure has a drastic impact on the convergence rate.

The formalization of the initialization module completes the GeoScop algorithm. To summarize, first, users are assigned to communities based on their correlation of visits to various geographical clusters. Then, the parameters for the EM model are initialized and EM is run until convergence of its parameter set Θ. Finally, the geo-social clustering is performed. The iteration of EM and geo-social clustering continues till geo-social clusters converge in terms of place membership.

4 Scaling GeoScop

GeoScop has two components: EM and DBSCAN. When supported by index structures, DBSCAN has $O(n \log n)$ complexity where n is the number of unique places [4], [6]. Recently, a linear

Algorithm 2 CommunityInitialization($\mathbb{D}, \mathbb{P}$)

Require: $\mathbb{P}$ is a set of clusters over places
Ensure: return the set of communities $\mathbb{C}$

$\mathcal{U} \leftarrow$ unique users in $\mathbb{D}$
$\mathcal{S} \leftarrow$ unique places in $\mathbb{D}$
compute cluster profile of all users in $\mathcal{U}$
$\mathbb{C} \leftarrow \emptyset$
$Q \leftarrow$ empty queue
while $\exists u \in \mathcal{U}$ **do**
  $u \leftarrow$ random unprocessed user
  $Q \leftarrow Q.queue(u)$
  $C \leftarrow \emptyset$
  while $Q \neq \emptyset$ **do**
    $u \leftarrow Q.dequeue()$
    $\mathcal{U} \leftarrow \mathcal{U} \setminus u$
    $T \leftarrow topk(u, k, \mathcal{U})$ \\(Eq. 21)
    if k^{th} closest user to u has similarity above $minSim$ **then**
      $C \leftarrow C \cup T \cup \{u\}$
      $\forall u' \in T\ Q.queue(u')$
  if $C \neq \emptyset$ **then**
    $\mathbb{C} \leftarrow \mathbb{C} \cup C$
return $\mathbb{C}$

time approximate DBSCAN algorithm has been proposed [14]. For our problem, we utilize the same grid-based index structure used in DCPGS to scale DBSCAN. In this section, we develop a technique to scale the EM component, which is unique to our problem.

4.1 Sliding window

In EM, the most expensive step is computing the influence sets F^+, F^-. To compute $F^+_{u,P}$ for a user u, we need to iterate over all check-ins, and identify those that visit cluster P within Δ time of u's first check-in at P. This results in $O(\|\mathcal{U}\|\|\mathbb{D}\|)$ complexity when computed over all users. The cost is same for F^-. This computation cost is too high for city-wide datasets where both the user set and the number of check-ins is large. We therefore develop a more efficient approach.

For each geo-social cluster P, we sort all the check-ins at P by their timestamps. Let $\langle u_0, p_0, t_0 \rangle$ be the first check-in at P. Clearly, $F^-_{u_0,P} = \emptyset$. In fact, all unique users with a check-in within Δ time from t_0 would have u_0 in its $F^+_{u,P}$ and an empty $F^-_{u,P}$. We generalize this intuition by sliding a window of width Δ across the sorted list of check-ins. Fig. 4 illustrates the idea. A sliding window partitions all check-ins in cluster P into three sets: *within window (WW)*, *before window (BW)*, and *after window (AW)*. Now, if the siding window ends at check-in $\langle u, p, t \rangle$, then $F^-_{u,P} = \{u' | \langle u', p', t' \rangle \in BW\}$ and $F^+_{u,P} = \{u' | \langle u', p', t' \rangle \in WW\}$. This property lies at the core of our algorithm. We first position the window on the earliest check-in and construct the F^+ and F^- sets. This takes constant time. Next, we slide the window to the next check-in, which results in all check-ins that are more than Δ away going out. If w check-ins go out of the window, then accordingly updating the F^+ and F^- for the current user takes $O(w)$ time. In this manner, we continue sliding the window till the last check-in. Thus, overall, for users visiting cluster P, their $F^-_{u,P}$ and $F^+_{u,P}$ can be computed in $O(\{CI_P|)$, where $CI = \{\langle u, p, t \rangle \in \mathbb{D} | p \in P\}$ is the set of check-ins in P. Since this operation needs to be repeated across all clusters $\mathbb{P}$, the total complexity is $O(\| \cup_{\forall P \in \mathbb{P}} CI_P \|) = O(\|\mathbb{D}\|)$. Consequently, there is a drastic reduction from the complexity of $O(\|\mathcal{U}\|\|\mathbb{D}\|)$ in the naïve approach.

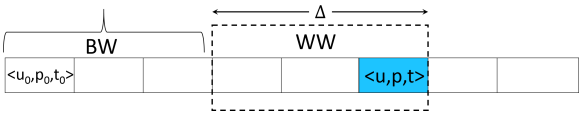


Fig. 4. Illustration of the sliding window idea.

City	Code	Source	‖Users‖	‖Places‖	‖Check-ins‖
Chicago	C	Gowalla	4697	16721	95154
Philadelphia	P	Gowalla	2574	11857	47538
Boston	B	Gowalla	3142	15977	74708
San Francisco	S	Brightkite	3598	14042	85758
New York City	N	Brightkite	4029	21448	128397
Washington	W	Brightkite	2668	20129	117298
San Jose	J	Brightkite	2163	9838	63893

TABLE 1
Statistics of various city check-ins.

5 Experiments

In this section, we show that

- GeoScop is more effective in mining geo-social clusters than the state-of-the-art technique.
- GeoScop is able to overcome the bottleneck of not having knowledge of the underlying social network of users.
- GeoScop scales to large city-wide datasets.

5.1 Datasets

We use two datasets for benchmarking purposes: *Gowalla* [8] and *Brightkite* [8]. Gowalla contains 196,591 users with 6,442,892 check-ins performed on 1,280,969 places over a period from February 2009 to October 2010. Brightkite contains 58,228 users with 4,491,143 check-ins from April 2008 to October 2010 across 772,783 places. Both these datasets are accompanied with a social network connecting their users. Gowalla contains 950,327 edges and Brightkite has 214,078 edges. Although, GeoScop does not use the network, its availability allows us to evaluate the performance against a technique like DCPGS [6], which relies on the network to perform geo-social clustering.

5.2 Experimental Setup

All our experiments are performed on a 3.4 GHz quad-core i7 machine running Ubuntu 14.04 with 8GB of main memory.

As discussed in Sec. 1, the only other technique for geo-social clustering is DCPGS [6]. We therefore compare GeoScop's performance with DCPGS. DCPGS has already explored various other obvious approaches to geo-social clustering such as SimRank [15], the Katz measure [16] and Jaccard coefficient, and outperforms them by a significant margin. In addition to DCPGS, we also include DBSCAN in our benchmarking studies to showcase the need to capture semantics.

Since clusters across two different cities are always independent, we perform our analysis on a city-by-city basis. Specifically,

we extract all check-ins within a city and analyze the clusters obtained. The same approach is also used in DCPGS. Table 1 summarizes the check-ins in cities used for our experiments. We intentionally distribute the city datasets evenly across Brightkite and Gowalla.

Parameters: The parameters that are common to GeoScop, DCPGS, and DBSCAN are $minPts$ and the geographical radius (δ_g in Defn. 4). We use the default parameter values recommended in DCPGS since the same datasets are used in their evaluation as well. These values are $minPts = 5$ and $\delta_g = 120m$. DCPGS requires three other parameters. DCPGS performs DBSCAN with a weighted combination (weight w) of geographical distance and social distance with two additional thresholds on the weighted distance (ϵ) and the social distance (τ). We set these parameters as recommended by the authors in [6], which are $w = 0.5, \epsilon = 0.4, \tau = 0.7$. GeoScop needs to know a social distance threshold δ_s (Defn. 4), which we learn automatically from the dataset. More specifically, following the initialization step, we sample 10% of the users and compute the social distance between them. δ_s is set to 80 percentile, which is the value where 80% of the social distances are larger. Note that since the social distance function is different for GeoScop and DCPGS, the same threshold value cannot be used for both techniques. In EM, we construct F^+ and F^- using the time threshold Δ set to 30 days.

5.3 Visualization-based Analysis

We initiate the empirical evaluation through a visualization-based analysis. Here, we present real-life geo-social clusters that GeoScop mines, while DBSCAN groups them together into a single geographical cluster. In addition, we also show the results of DCPGS and demonstrate why a community level analysis to compute social distance is critical. To keep the analysis focused, we extract the check-ins in Philadelphia from Gowalla dataset and the check-ins in San Jose from the Brightkite dataset.

Figs. 5(a) presents an example from Philadelphia where two geo-social clusters are found (color coded in blue and orange). Notice in Fig. 5(c) that DBSCAN merges the two into one single cluster. On closer inspection, we find that while the larger blue cluster generally corresponds to the downtown area, the orange cluster represents the zoo. This is a perfect example demonstrating the need to incorporate social information and capture the semantics. Zoo is frequented by tourists compared to the community of residents who routinely visit the downtown areas. GeoScop is able to make this distinction due to mining communities through the proposed iterative framework. Now, notice in Fig. 5(b), that DCPGS fails to detect any geo-social cluster in the zoo. It also classifies most of the check-ins in the downtown area as outliers. This results due to the restrictive 1-hop based social distance measure.

Figs. 5(d)-5(e) show another set of results from San Jose. Here, GeoScop identifies two geo-social clusters, which corresponds to a single cluster in DBSCAN. We find that these two geo-social clusters correspond to a Walmart store (blue check-ins) and a Target store (orange check-ins). It has been shown that Target's customer base comes from a higher income group than those of Walmart [17]. Once again, DCPGS fails to discriminate these two geo-social clusters. In fact, most of the check-ins in this area are classified as outliers. Note that although there are 5 places in the DCPGS cluster in Fig. 5(e), due to overlap, it appears that there are only 4 places.

Finally, we present another set of results from Philadelphia, where GeoScop picks out Walmart (blue), Macy's (red) and Kohl's (pink) as three different geo-social clusters (Fig. 5(g)) from a single cluster as identified by DBSCAN (Fig. 5(i)). These three are well known stores in USA. Both Macy's and Kohl's sell different kind of products than Walmart. Between Macy's and Kohl's, Macy's caters to the upper income group, whereas Kohl's core customer base is the middle-class section [18], [19]. This separation in the communities of people visiting the same neighborhood is picked by GeoScop and appropriately segregated. DCPGS (Fig. 5(h)) classifies all check-ins in this area as outliers.

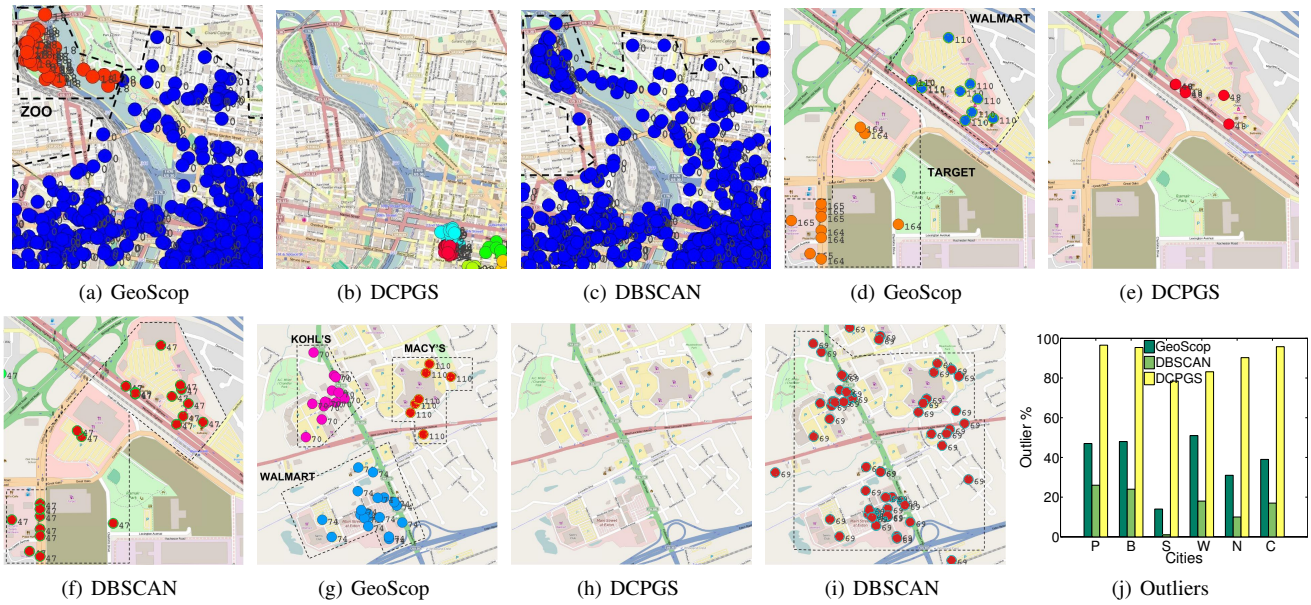
One consistent theme that we observe across all these results is that in DCPGS, most places get classified as outliers. Intrigued by this observation, we study deeper and quantify what percentage of the places in each of these cities are outliers. Fig. 5(j) presents the results for all three clustering techniques across 6 major US cities. We use the city codes shown in Table 1 as x-axis labels. It is natural that both GeoScop and DCPGS would have more outliers than DBSCAN since they enforce social constraints in addition to the geographical constraints of DBSCAN. At the same time, the percentage of outliers in each of the techniques are correlated across all cities. The most striking observation in this study though is the high outlier percentage in DCPGS. This result stems from the low overlap in 1-hop neighborhoods of users visiting places, which translates to high social distance. GeoScop, on the other hand, finds much less outliers due to community level analysis. As already analyzed through visualization, these clusters are semantically meaningful as well.

5.4 Quantitative Analysis

After establishing the utility and superiority of GeoScop through a visualization-based analysis, we next focus on a more quantitative analysis.

In the first set of experiments, we continue evaluating the quality of clusters, but from a quantitative view point. In the previous section, we look at few specific instances of geo-clusters. Our goal in the following experiments is to measure the social quality across all clusters in a city. Note that the geographical quality is already bounded by DBSCAN. Furthermore, it also guaranteed that places in a cluster are visited by similar communities since this condition is explicitly enforced in the definition of a geo-social cluster. What is unknown is the following question: *Do geo-social clusters exist that attract an entire community?* We answer this question in a systematic and quantitative manner with respect to the communities that exist in the actual social network rather than those mined by GeoScop. Thus, the following experiments also evaluate to what extent GeoScop is able to overcome the inaccessibility to the social network.

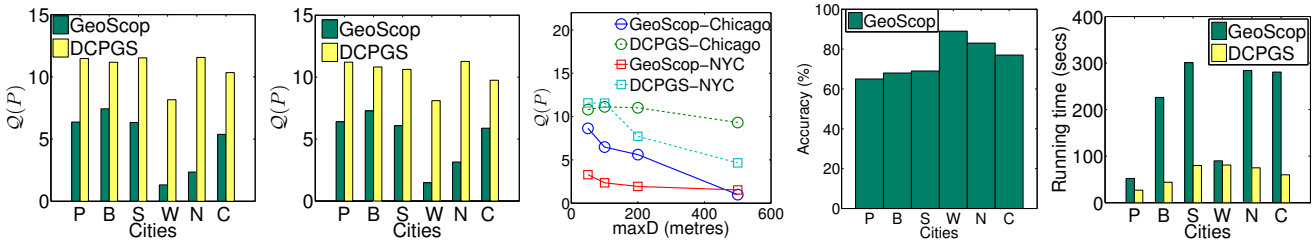
Since the problem of geo-social clustering itself is new, we propose a new metric to evaluate the social quality of a cluster. Intuitively, a cluster P is good if a person u visiting P indicates a high likelihood of the remaining members of u's community also visiting P. We formalize this property. Let $\mathbb{C}$ be the set of all communities visiting a cluster P and $\mathbb{P}$ be the set of all geo-social clusters. For any community $C \in \mathbb{C}$, $visits(C, P) = \{\langle u, p, t\rangle \in \mathbb{D} | u \in C, p \in P\}$ denotes the set of visits from C to P. Therefore, higher the value of $\frac{\|visits(C,P)\|}{\sum_{\forall P' \in \mathbb{P}} visits(C,P')}$, better is the social quality of P with respect to community C. A high value indicates that there is a strong affinity of visiting P in community C. However, this ratio ignores the fact that a high number of visits



(a) GeoScop (b) DCPGS (c) DBSCAN (d) GeoScop (e) DCPGS

(f) DBSCAN (g) GeoScop (h) DCPGS (i) DBSCAN (j) Outliers

Fig. 5. (a-i) Comparison of the clusters obtained using GeoScop, DCPGS, and DBSCAN. Colored viewing is recommended for these plots. (a-c) GeoScop correctly separates zoo from downtown Philadelphia, which DCPGS and DBSCAN fails to detect. Similar results are obtained in (d-f) where GeoScop separates the check-ins at Walmart and Target into two clusters and in (g-i) where GeoScop is able to detect three communities going to Macy's, Walmart, and Kohl's. (j) The percentage of outliers for the three clustering techniques.



(a) Dunbar (b) Automatic (c) (d) (e)

Fig. 6. (a-b) The social quality of the clusters obtained from GeoScop and DCPGS. (c) The variation in the social quality with the maxD parameter. (d) Accuracy of data merging. (e)Running times of GeoScop and DCPGS in the various cities.

from a single member of the community could bias the overall ratio if the other members are not as outgoing. Hence, we compute $people(C,P) = \{u \in \mathbb{C} | \langle u,p,t \rangle \in \mathbb{D}, p \in P\}$, which is the set of members who visited P. For a "good" geo-social cluster, we would also like $\frac{\|people(C,P)\|}{\|C\|}$ to be high. We combine these intuitions into the following metric:

$$\mathcal{Q}(P) = -\sum_{\forall C \in \mathbb{C}} log(p(C)) \times \frac{\|C\|}{\sum_{\forall C' \in \mathbb{C}} \|C'\|} \tag{22}$$

where,

$$p(C) = \frac{\|visits(C,P)\|}{\sum_{\forall P' \in \mathbb{P}} visits(C,P')} \times \frac{\|people(C,P)\|}{\|C\|} \tag{23}$$

$p(C)$, which lies in the range $[0,1]$, is proportional to the representation of a community in cluster P and also to the proportion of visits paid by the community to P. We take the negative log of $p(C)$ for each community visiting cluster P and weight it based on its size so that a larger community has a higher say. Note that this term can take a value of 0 even if several communities visit the cluster C. This happens when all the communities visiting this cluster, visit only this cluster. The lower the value of $\mathcal{Q}(P)$, the better is the social quality. Given the entire set of geo-social clusters $\mathbb{P}$, the quality is summarized as $\mathcal{Q}(\mathbb{P}) = \sum_{\forall P \in \mathbb{P}} \mathcal{Q}(P) \frac{\|P\|}{\sum_{\forall P' \in \mathbb{P}} \|P'\|}$.

We compute $\mathcal{Q}(\mathbb{P})$ with respect to the actual communities that exist in the social networks of Gowalla and Brightkite. The communities are mined using *Metis* [20]. Metis requires the number of communities as an input. We use two approaches to determine this parameter. GeoScop proposes an automated way of detecting the number of communities in the initialization step. We use this number as the input parameter for Metis. In the second approach, we use the heuristic proposed by DCPGS. The authors of DCPGS use the *Dunbar number* to set the number of communities, which is simply $\frac{\|\mathcal{U}\|}{150}$. The number 150 is called the Dunbar number in social sciences and was arrived at through research that showed humans can maintain at most 150 stable relationships, and the mean community size is 150. $\mathcal{Q}(\mathbb{P})$ is computed for both scenarios.

Figs. 6(a) and 6(b) demonstrate the results in 6 major cities of USA. Regardless of the number of communities provided to Metis, GeoScop performs better. Recall, the lower the value of $\mathcal{Q}(\mathbb{P})$, the better is the social quality. The difference in performance is most evident in San Francisco and New York City, where GeoScop is 6 times better than DCPGS. This result brings out two key conclusions. First, it is indeed possible to perform geo-

social clustering even when the network is not available. Second, computing social similarity based on community produces better results than the one-hop neighborhood.

After a thorough evaluation of cluster quality at the default parameter values, we next investigate the impact of the parameter δ_g, which is common to both GeoScop and DCPGS. Recall, δ_g defines the geographical neighborhood of a place. Fig. 6(c) studies how the social quality changes with δ_g in Chicago and NYC, which are two of the largest cities in USA. As can be seen, the social quality improves with increase in δ_g. This is natural since as δ_g grows, the geographical connectivity constraint gets relaxed leading to far-away, but socially similar places getting grouped into the same geo-social cluster. In a way, this is similar to clustering places based only on their social similarity, which obviously improves social quality. Notice that regardless of the specific value of δ_g, GeoScop performs better than DCPGS, and thereby establishing better robustness and stability.

In Sec. 1, we argued that GeoScop is more flexible than DCPGS in allowing us to merge check-in datasets from multiple sources. This is not possible in DCPGS since it relies on a social network and users across social networks cannot be linked. In the next experiment, we establish this property of GeoScop empirically. First, we pick the top 100 users with highest check-ins from each city, and then partition their check-ins evenly into two sets. Now, we create two users corresponding to each check-in set. In other words, we artificially partition users into two. All remaining users are kept unaltered. Following this, we perform GeoScop on this dataset and verify what portions of these 100 users are grouped into the same community. Fig. 6(d) presents the results. The accuracy is at least 65% in all cities with Washington showing an accuracy as high as 90%. To put this accuracy numbers in context, if there are K communities, a random assignment would be correct with probability $1/K$ and K is typically in the range $[20, 50]$. Overall, this result brings out the stability and flexibility of GeoScop.

5.5 Scalability

In this section, we evaluate the scalability of GeoScop. Fig. 6(e) shows the running times of GeoScop and DCPGS. DCPGS is indexed using the structure proposed by the authors [6]. As can be seen, DCPGS is about 2 to 4 times faster in general. However, it must be noted that the problem of geo-social clustering is significantly simpler for DCPGS since the social network is known. DCPGS performs a single instance of DBSCAN with a geo-social distance function. In our problem, DBSCAN and EM is run iteratively since we have the additional task of inferring the communities from the unobserved social network. Despite this challenging task, GeoScop finishes within 5 minutes across all cities. In other words, a two to four minutes overhead per city in exchange of better quality and independence from social network data. Furthermore, clustering is an off-line procedure where few extra minutes can be afforded in exchange of better quality.

Next, we evaluate the growth of the running time against the number of check-ins. In addition, we also analyze the impact of the sliding window based indexing technique on GeoScop. To construct the dataset for this experiment, we could randomly sample check-ins till a target dataset size is reached. However, that would result in check-ins being distributed all across the world, which is not meaningful for cluster identification. Thus, we sample check-ins in a city-by-city basis. More specifically, if

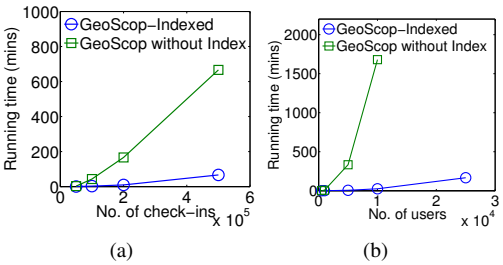


Fig. 7. Growth of running time with number of (a) check-ins and (b) users.

we want to construct a database of 100,000 check-ins, we sample check-ins from the largest city (NYC) in Table 1 till the target value is reached. Now, if we want to know the running time at 200,000 check-ins, since no city has as many check-ins, we start adding check-ins from the next largest city, and this process continues by moving through cities one-by-one. Fig. 7(a) shows the results. We vary the number of check-ins till half a million. To put this number in context, even if five cities from Table 1 are put together, the number of check-ins is less. Despite that high load, the indexed version of GeoScop finishes in around an hour for 0.5 million points. On the other hand, without indexing, the running time growth rate is much higher and it takes more than 12 hours to finish at 0.5 million check-ins. Overall, sliding window results in more than an order of magnitude speed-up.

Fig. 7(b) evaluates the running time against the number of users in the dataset. We vary the number of users till 25,000. To put this number in context, the seven cities in Table 1 put together do not have as many users. We thus needed to add users from Dallas to reach 25,000 users. The trend observed is similar to Fig. 7(a), although the difference in running times is even more drastic. This is consistent with the theoretical analysis since the proposed index structure changes the $O(\|\mathcal{U}\|\|\mathbb{D}\|)$ complexity of influence set construction to $O(\|\mathbb{D}\|)$. The running time for the without-index version is missing at 25,000 users since it did not finish even after 3 days. With index, we finish in under three hours, which is two orders of magnitude faster than the naïve implementation. Overall, the last two experiments showcase the impact of the proposed sliding window based algorithm in scaling GeoScop to real city-wide check-in datasets.

6 Related Work

A detailed comparison with DCPGS, which is the only other technique to perform geo-social clustering, has been done theoretically in Sec. 1, and empirically in Sec. 5. Here, we overview the existing works that overlap with our problem.

6.0.1 *Spatial clustering*

Spatial clustering approaches can broadly be grouped into three categories: *density-based, hierarchical* and *partitioning*. We use the density-based approach due to the advantages discussed in Sec. 1. While we use DBSCAN [4] as the base for GeoScop, our technique can easily be adapted to use Optics [5] instead. OPTICS improves on DBSCAN to cluster objects of non-uniform densities. DENCLUE, [21], is another density-based algorithm, which determines the point density analytically and uses the concept of density-attractors to form clusters of arbitrary shapes.

6.0.2 Using mobility data to analyze places

The algorithm ComeTogether, [22], finds clusters of points-of-interests (POIs) based on the correlation of the sequence of visits by mobile users. In contrast to our work, this algorithm ignores the social relationship between users. Rather, the focus is on connecting places that are visited in same sequence by users.

6.0.3 Geo-social relationships

A significant volume of work has been done on the correlation between geographical and social relationships. Scellato et al. study this correlation in Geo-Social networks [23] and use it for link prediction [24]. Backstrom et al. use geo-social relations to predict user locations [25]. Along similar lines, Pham et al. [26] predict social connections from geographical data, and Cho et al. [27] explain human movements from social relationships. Although related to our work, none of these techniques solve the problem of geo-social clustering of places.

7 Conclusion

In this paper, we showed how user check-ins can lead to useful insights about communities and places. Specifically, we formulated the problem of mining geo-social clusters from check-in data in the absence of access to a social network. Our problem is motivated from practical real-life scenarios where the social network is inaccessible to majority of the companies that collect check-in data. To solve this problem, we developed a technique called *GeoScop (GEO-Social Clustering Of Places)*, which uses an iterative framework of community detection and clustering. To scale GeoScop for city-wide check-ins, we designed a sliding window based indexing technique and obtained up to 2 orders of magnitude speed-up over the naïve approach. We conducted extensive experiments on check-ins across 7 major cities in USA. The empirical results demonstrated that even in the absence of a social network, GeoScop is up to 6 times better than the state-of-the-art technique. Overall, GeoScop unleashes the capability to capture semantics in place clustering, which till now, was handicapped due to the reliance on an observable social network.

References

[1] S. Mitra, S. Ranu, V. Kolar, A. Telang, A. Bhattacharya, R. Kokku, and S. Raghavan, "Trajectory aware macro-cell planning for mobile users." in *INFOCOM*, 2015.

[2] D. Pennerstorfer and C. Weiss, "Spatial clustering and market power: Evidence from the retail gasoline market," *Regional Science and Urban Economics*, vol. 43, no. 4, pp. 661–675, 2013.

[3] A. Diker and E. Nasibov, "Estimation of traffic congestion level via fn-dbscan algorithm by using gps data," in *Problems of Cybernetics and Informatics*, 2012, pp. 1–4.

[4] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." in *SIGKDD*, vol. 96, no. 34, 1996, pp. 226–231.

[5] M. Ankerst, M. M. Breunig, H.-P. Kriegel, and J. Sander, "Optics: ordering points to identify the clustering structure," in *ACM Sigmod Record*, vol. 28, no. 2, 1999, pp. 49–60.

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

[7] "http://techcrunch.com/2013/01/24/my-precious-social-graph/."

[8] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *SIGKDD*, 2011.

[9] J. Kleinberg, "The small-world phenomenon: An algorithmic perspective," in *STOC*, 2000, pp. 163–170.

[10] Y. Cheng and G. M. Church, "Biclustering of expression data." 2000.

[11] S. Srivastava, S. Pande, and S. Ranu, "Geo-social clustering of places from check-in data," in *Data Mining (ICDM), 2015 IEEE International Conference on*. IEEE, 2015, pp. 985–990.

[12] D. Kempe, J. Kleinberg, and E. Tardos, "Maximizing the spread of influence through a social network," in *SIGKDD*, 2003, pp. 137–146.

[13] N. Barbieri, F. Bonchi, and G. Manco, "Influence-based network-oblivious community detection," in *ICDM*, 2013, pp. 955–960.

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

[15] G. Jeh and J. Widom, "Simrank: a measure of structural-context similarity," in *SIGKDD*, 2002, pp. 538–543.

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

[17] "http://www.cbsnews.com/news/walmart-and-target-a-tale-of-two-discount-chains/."

[18] "http://www.bloomberg.com/news/articles/2015-03-30/kohl-s-backs-big-name-brands-to-boost-shares-faster-than-macy-s."

[19] "http://www.washingtonpost.com/blogs/wonkblog/wp/2013/11/14/how-wal-mart-and-macys-explain-the-economy/."

[20] G. Karypis and V. Kumar, "A fast and high quality multilevel scheme for partitioning irregular graphs," *SIAM J. Sci. Comput.*, vol. 20, pp. 359–392, 1998.

[21] A. Hinneburg and D. A. Keim, "An efficient approach to clustering in large multimedia databases with noise," in *SIGKDD*, 1998, pp. 58–65.

[22] I. Ramalho Brilhante, M. Berlingerio, R. Trasarti, C. Renso, J. A. F. de Macedo, and M. A. Casanova, "Cometogether: Discovering communities of places in mobility data," in *MDM*, 2012, pp. 268–273.

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

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

[25] L. Backstrom, E. Sun, and C. Marlow, "Find me if you can: Improving geographical prediction with social and spatial proximity," in *WWW*, 2010, pp. 61–70.

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

[27] E. Cho, S. A. Myers, and J. Leskovec, "Friendship and mobility: User movement in location-based social networks," in *SIGKDD*, 2011.

PLACE PHOTO HERE

Michael Shell Biography text here.

John Doe Biography text here.

Jane Doe Biography text here.