Finding Geo-Social Cohorts in Location-Based Social Networks

Muhammad Aamir Saleem[1], Toon Calders[2], Torben Bach Pedersen[1], and Panagiotis Karras[3]

[1] Aalborg University, Denmark,
[2] University of Antwerp, Belgium,
[3] Aarhus University, Denmark,

Abstract. Given a record of geo-tagged activities, how can we suggest groups, or *cohorts* of likely companions? A brute-force approach is to perform a spatio-temporal join over past activity traces to find groups of users recorded as moving together; yet such an approach is inherently unscalable. In this paper, we propose that we can identify and predict such cohorts by leveraging information on social ties along with past geo-tagged activities, i.e., geo-social information. In particular, we find groups of users that (i) form cliques of friendships and (ii) maximize a function of common *pairwise* activities on record among their members. We show that finding such groups is an **NP**-hard problem, and propose a nontrivial algorithm, COVER, which works as if it were enumerating maximal social cliques, but guides its exploration by a pruning-intensive activity-driven criterion in place of a clique maximality condition. Our experimental study with real-world data demonstrates that COVER outperforms a brute-force baseline in terms of efficiency and surpasses an adaptation of previous work in terms of prediction accuracy regarding groups of companions, including groups that do not appear in the training set, thanks to its use of a social clique constraint.

1 Introduction

Advances in positioning and communication technologies enable sharing geo-tagged content, and hence location-based social networking services. Location-Based Social Networks (LBSNs) such as Weeplaces, Foursquare, Gowalla, GeoLife, and Twinkle are built around positioning capabilities, while Facebook and vKontakte provide a consensual *option* to check-in at visited locations.

Such online LBSNs can utilize user mobility in consensual recommendation services directed to *groups*, as opposed to individuals. For instance, a discount offer for a concert may be recommended to a group of friends that have habitually visited similar concerts and other events together; a car-pooling service may suggest group formations to its customers who may be unaware of their similar activities; or a travel agency may promote an offer for a group travel package to groups of users potentially interested in traveling together. Past research has proposed location-based recommendations of locations, routes, users, activities,

and media [4]; while some works refer to individual predictions [15], others detect groups based on location histories and mobility profiles [10]. Yet these studies rely on clustering trajectories and co-location traces [22] and Bayesian learning so as to predict co-location features [11], hence do not scale well. Further, there has been work on clustering *static locations* using geo-social information [18], and a long line of works on detecting communities of interest [7, 13, 6] Yet to our knowledge, no attempt has been made to suggest groups *of users* using *both social and mobility data*, i.e., geo-social information.

In this paper, we propose a method that suggests *cohorts* of social companions *without* clustering trajectories or traces; instead, it leverages *social* and *co-location* connections, which LBSNs record. We conjecture that recommendable groups of companions are prone to be directly or transitively connected in both the *social domain* — i.e., to form *cliques* in a graph of social ties — and the *activity domain* — i.e., to engage in *common geo-tagged activities* manifested as frequent pairwise location check-ins. Our rationale is that each person may have a large circle of friends and acquaintances, yet may engage in specific activities only with particular ones, with whom they share related interests. For instance, consider a socially linked group of people who have visited museums, at least in pairs; our method can detect such a cohort, so as to propose new interesting locations. We utilize location *categories* to tailor cohorts to particular interests, and test the power of our method to predict cohorts on real-world data.

We build a graph G_A from geo-tagged activities, *distinct* from a graph of social ties, G, yet defined over the same vertex set; in G_A, a pair of nodes is connected by a weighted edge expressing the pairs' history of co-location in common activities. We reason that a group of tightly-connected users engaging in common activities frequently appear in G_A as a connected subgraph with high edge density, i.e., form a *quasi-clique* of high edge weight in G_A, and a *clique* in G. We formally define the problem as retrieving sets of nodes that induce subgraphs maximizing *edge density* in G_A and also induce *cliques* in G.

The rest of this paper is structured as follows. Section 2 discusses related work. In Section 3, we formulate the problem and study its hardness. Section 4 presents COVER, an efficient heuristic drawing from graph mining techniques to address the spatiotemporal cohort discovery challenge. Section 5 presents an extensive experimental study that showcases the effectiveness and predictive power of our technique. Section 6 concludes the paper.

2 Related Work

In our problem, groups are sought after and have no labels. That is different from *group recommendation*, in which we recommend items to a *given group* [1], and from *group discovery*, which retrieves labeled groups of users from collaborative rating data sets [19]. Our problem relates rather to the problems of finding *cliques*, dense subgraph discovery, and multi-layer community detection.

Tomita et al. [20] studied the worst-case complexity of enumerating maximal cliques on a graph. Such cliques can be used to define communities; for example,

one may consider only maximal cliques of size above a threshold, and define communities as the disconnected components of the graph formed by the union of those cliques [8]. Alternatively, one may use cliques of fixed size, k; the *clique percolation* method [16] finds all such k-cliques in a network, and then finds clusters made out pairs of k-cliques sharing $k-1$ nodes.

The *densest subgraph* problem asks for a vertex subset $S \subseteq V$ on graph $G(V, E)$ such that its induced subgraph achieves the maximum average degree; it is solvable in polynomial time with a maximum flow algorithm [9], while a greedy $\frac{1}{2}$-approximation scheme requires linear time [12]. Asahiro et al. [2] study the $k - f(k)$ dense subgraph problem, which calls for finding a k-vertex subgraph of a given graph G that has at least $f(k)$ edges, for different functions $f(k)$. When a restriction is imposed on the size of set S, the problem becomes **NP**-hard [12]. Recently, the problem has been studied in streaming and MapReduce models [3].

Boden et al. [5] mine *multi-layer coherent subgraphs*, i.e., subgraphs that contain vertices densely connected by edges with similar *labels* in a subset of layers in an edge-labeled multi-layer graph; this technique applies the same density criterion on multiple layers. We aim to apply *different density criteria* per layer: a clique constraint vs. quasi-clique optimality, a distinction consequential on predictive power — as we show, *if we relax the clique constraint, we lose predictive power.*

3 Problem Formulation

Let V be a set of LBSN users and $\mathcal{U}$ a universe of categories (i.e., types, based on function and audience) to which locations of interest are associated.

3.1 Preliminary Concepts

Definition 1. *A point of interest (POI) is a geographical location (e.g.,* the Metropolitan Museum of Art*) represented by a quadruple (l, lat, lon, cat), where l is the identifier, lat and lon the latitude and longitude of the GPS coordinates of the center of the POI, and $cat \in \mathcal{U}$ a category that this location belongs to.*

Definition 2. *An activity refers to a visit of a user $u \in V$ at a location l at a* discretized *time interval t, represented as a triplet (u, l, t); when the user u is implied from context, we omit it and represent an activity by the pair (l, t).*

Definition 3. *The activity set of user u, $\mathcal{A}(u)$, is the set of activities user u has engaged in; likewise, the activity set for a category cat, $\mathcal{A}(cat)$, is the set of activities associated with category cat among users in V; last, $\mathcal{A}(u, cat)$ is the set of all activities by user u over POIs of category cat. We overload these notations to also denote ordered* sequences *of activities depending on context.*

Definition 4. *A spatiotemporal join operation among sets of activities $\mathcal{S}$ and $\mathcal{T}$, $\mathcal{S} \bowtie \mathcal{T}$, returns the set of pairs of activities $\{(l, t) \in \mathcal{S}, (l', t') \in \mathcal{T}\}$, where locations and times match, i.e., $l = l'$ and $t = t'$ Furthermore, a consecutive*

spatiotemporal join is defined among two temporal sequences *of activities* $\mathcal{S}$ *and* $\mathcal{T}$, $\mathcal{S} \bowtie^c \mathcal{T}$, *and returns the set of* quadruples *of activities* $\{(l,t),(l_{suc},t_{suc}) \in \mathcal{S},(l',t'),(l'_{suc},t'_{suc}) \in \mathcal{T}\}$, *where* (l_{suc},t_{suc}) *is the successor activity of* (l,t) *in sequence* $\mathcal{S}$ *and* (l'_{suc},t'_{suc}) *the successor activity of* (l',t') *in sequence* $\mathcal{T}$, *such that locations and times match, i.e.,* $(l,t) = (l',t')$ and $(l_{suc},t_{suc}) = (l'_{suc},t'_{suc})$.

The spatiotemporal join between $\mathcal{S}$ and $\mathcal{T}$ returns all pairs of activities occurring in $\mathcal{S}$ and $\mathcal{T}$, by the discretization we employ; a consecutive spatiotemporal join returns all *quadruples* of activities occurring, as two consecutive pairs, in sequences $\mathcal{S}$ and $\mathcal{T}$. We use these concepts to define weights in activity graph G_A.

3.2 Objective

Given a data set of users, their relationships, and records of activities, and a set of categories of interest, $\mathcal{L} \subset \mathcal{U}$, we are interested to identify any group of users $\mathcal{C} \subset V$ that are likely to participate, as a group, in future activities related to $\mathcal{L}$.

We leverage (i) a *social graph* $G(V,E)$, where V is the set of users and E the set of friendship relationships; and (ii) an *activity graph* $G_A^{\mathcal{L}}(V,E')$, coterminous with (i.e., defined over the same set of vertices V as) G, built out of the log of user co-locations associated with $\mathcal{L}$; an edge $(u,v) \in E'$ between users $u,v \in V$ has a non-zero weight $w_{uv} \in (0,1]$, representing the extent to which these two users participate in common activities associated with $\mathcal{L}$, by the following definition.

Definition 5. *The edge weight* w_{uv} *between the pair of users* (u,v) *in* $G_A^{\mathcal{L}}$ *is defined via the (consecutive) spatiotemporal join* $\bigcup_{cat \in \mathcal{L}} \{\mathcal{A}(u,cat) \bowtie \mathcal{A}(v,cat)\}$, *normalized by dividing by the highest value obtained among all pairs of users, as follows:*

$$w_{uv} = \frac{|\bigcup_{cat \in \mathcal{L}} \{\mathcal{A}(u,cat) \bowtie^{(c)} \mathcal{A}(v,cat)\}|}{max_{x,y \in V} |\bigcup_{cat \in \mathcal{L}} \{\mathcal{A}(x,cat) \bowtie^{(c)} \mathcal{A}(y,cat)\}|} \tag{1}$$

We aim to retrieve groups, or *cohorts*, of users that have a track record of pairwise common activities associated with $\mathcal{L}$, and also form a *clique* (i.e., are related to each other) in the social graph. Such cohorts are likely to act together, hence may be used for recommendation, prediction, and social analysis.

Problem 1. [Cohort Retrieval] Given a set of LBSN users V, a set of categories $\mathcal{L}$, a social graph $G(V,E)$ among users in V, and an activity graph $G_A^{\mathcal{L}}(V,E')$ among users in V with edges in E' weighted according to activities in $\mathcal{L}$, and letting $\mathcal{C}$ denote the set of all *cliques* in the social graph G, where $\forall C \in \mathcal{C}, C \subseteq V$, *find the top-k cliques in* $\mathcal{C}$ in terms of an *activity density* function $f^{\mathcal{L}}$ calculated on the subgraphs they induce in $G_A^{\mathcal{L}}$, i.e., $\arg\max_{C \in \mathcal{C}}^{k} \{f^{\mathcal{L}}(C)\}$.

3.3 Maximizing Activity Density

Problem 1 requires an *activity density* function: (i) relying on *edge weights* of $G_A^{\mathcal{L}}$, and (ii) independent of subgraph (i.e., group) size, allowing for comparison among larger and smaller groups. A function that satisfies these properties is the *edge surplus* function f_α, maximized by an Optimal Quasi-Clique (OQC) [21]:

Definition 6. [OQC] *Given a graph $G = (V, E)$ and $\alpha \in (0, 1)$, an* optimal quasi-clique *of G is a subset of vertices $S^* \subseteq V$ such that:*

$$f_\alpha(S^*) = e[S^*] - \alpha\binom{|S^*|}{2} \geq f_\alpha(S), \text{ for all } S \subseteq V. \tag{2}$$

where $e[S]$ is the number of edges in the subgraph of G induced by S.

This *edge surplus* function provides a size-independent measure of edge density without favoring large subgraphs: a subgraph achieves a high value of edge surplus f_α *not merely* by means of a high average degree, as large subgraphs may have, but by coming close to completing a *clique* among its nodes. Besides, rather than imposing some arbitrary threshold on edge weights so as to obtain binary edges, it takes all weights in consideration, regardless of their values, and can be straightforwardly generalized to the weighted edges in an activity graph. We thus define our activity density function based on the edge surplus function:

Definition 7. *Given an activity graph $G_A^{\mathcal{L}} = (V, E')$, a vertex subset $C \subseteq V$, and a parameter $\alpha \in (0, 1)$, the* activity density *on C is:*

$$f_\alpha^{\mathcal{L}}(C) = w[C] - \alpha\binom{|C|}{2} \tag{3}$$

where $w[C]$ is the sum of normalized edge weights for all edges in the subgraph induced by C: $w[\mathcal{C}] = \sum_{u,v \in \mathcal{C}} w_{uv}$.

We aim to retrieve cohorts under the constraint of forming a clique in a social graph and the objective of maximizing edge surplus in the activity graph. To that end, it is useful to investigate the hardness of the problem of finding an OQC, which we henceforward name OQC. After all, in case our social graph is a complete graph, and all non-zero edge weights in the activity graph are equal to 1, then Cohort Retrieval is reduced to OQC, hence it is at least as hard as OQC. Tsourakakis et al. [21] suspect OQC to be **NP**-hard, yet provide no formal proof of hardness. We provide such a proof in the following, starting out with some results regarding the nature of the OQC problem.

Lemma 1. *For $\alpha \in (0, 1)$, any clique in $G = (V, E)$ has positive edge surplus.*

Proof. By definition, the number of edges in a clique $S \subseteq V$ is $e[S] = \binom{|S|}{2}$. Then, the edge surplus of S is $f_\alpha(S) = e[S] - \alpha\binom{|S|}{2} = (1 - \alpha)\binom{|S|}{2} > 0$.

Lemma 2. *For any $\alpha \in (0, 1)$, a maximum clique of a graph $G = (V, E)$ has the maximum edge surplus among all cliques in G.*

Proof. By Lemma 1, the edge surplus of a clique $S \subseteq V$ is $f_\alpha(S) = (1-\alpha)\binom{|S|}{2} > 0$. A *maximum clique* achieves the maximum number of vertices $|S|$ among all cliques in G; therefore, it also has the maximum edge surplus.

Theorem 1. *Given a simple undirected graph $G = (V, E)$, for $\alpha = 1 - \binom{|V|}{2}^{-1}$, a subset of vertices $S \subseteq V$ has positive edge surplus* if and only if *it is a clique.*

Proof. The $\Leftarrow$ direction is provided by Lemma 1. We now prove the $\Rightarrow$ direction: The edge surplus of a subset of vertices $S \subseteq V$ is $f_\alpha(S) = e[S] - \alpha\binom{|S|}{2} = e[S] - (1 - \binom{n}{2}^{-1})\binom{|S|}{2} = e[S] - \binom{|S|}{2} + \frac{|S|(|S|-1)}{|V|(|V|-1)}$. If S has positive edge surplus, then $f_\alpha(S) > 0 \Leftrightarrow e[S] > \binom{|S|}{2} - \frac{|S|(|S|-1)}{|V|(|V|-1)}$. Since $S \subseteq V$, it follows that $\frac{|S|(|S|-1)}{|V|(|V|-1)} \leq 1$. Thus, $f_\alpha(S) > 0 \Rightarrow e[S] > \binom{|S|}{2} - 1$. Yet the only subgraph of $|S|$ vertices that has more than $\binom{|S|}{2} - 1$ edges is a clique.

Theorem 2. *Finding an OQC is* **NP***-hard.*

Proof. We construct our proof by reduction from the **NP**-hard Clique problem, which calls for deciding whether a clique of certain size k exists in a simple undirected graph. Assume we are given a polynomial-time algorithm $\mathcal{A}(G, \alpha)$ that can find an OQC in any simple undirected graph $G(V, E)$ for any parameter $\alpha \in (0, 1)$. Then, given any instance of the Clique problem on a simple undirected graph $G(V, E)$, we invoke $\mathcal{A}(G, \alpha)$ with $\alpha = 1 - \binom{|V|}{2}^{-1}$. We emphasize that an elaborate *reduction* is not necessary, as we use the *same graph* in both problems. By Theorem 1, if the returned optimal quasi-clique (OQC) has non-positive edge surplus, it follows that G has no cliques; otherwise, if the returned OQC has any positive edge surplus, it is a clique. Moreover, by definition, the returned OQC has the maximum edge surplus among all such cliques in G, hence, by Lemma 2, it *is a maximum clique* of G. In effect, an algorithm that finds an OQC in G in polynomial time would also effectively decide whether G contains a clique, and, if so, what the maximum clique size is; thus, it would solve any instance of Clique, effectively deciding that a clique of size k exists if and only if k is no less than the maximum clique size. It follows that finding an OQC is at least as hard as any problem in **NP**.

Our proof resolves a question left open in [21]. Given this result, we should strive for non-optimal solutions to Cohort Retrieval, which is at least as hard as the problem of finding an OQC.

4 COVER Algorithm

We present COVER, our algorithm for the cohort retrieval problem; COVER merges and builds upon techniques for maximal clique enumeration [20] and the OQC problem [21]. It searches for cliques on the social graph, yet, instead of striving to satisfy just a maximality condition upon them, it checks, for any possible clique candidate, the edge surplus of its induced subgraphs on the activity graph, pruning nodes that cannot lead to higher edge surplus than already found. Eventually, it outputs the top-k results by our problem definition.

A cohort should form a social clique and also achieve as high activity density as possible, as outlined in Section 3.3. COVER explores the social graph in order to find cliques, as an algorithm for maximal clique enumeration would do. Yet, for each clique it finds in the social graph G, it searches locally for its subgraphs of high activity density in the activity graph G_A, maintaining a queue of the

top-k results, and, thereby, also a global queue of the top-k cohorts overall. In this process, when considering a new node v, it evaluates its strength, in terms of marginal activity density, that it may bring to any cohort under construction; node v is further considered only if its marginal activity density can bring an advantage compared to the top-k cohorts retrieved so far.

Algorithm 1: COVER: retrieving top-k geo-social cohorts

```
1  Input: G(V, E), G_L(V, E), k, T_Max, α
2  Output: Set of k traveler groups : CO
3  begin
4      C = ∅ /* a clique in G */
5      cohG = ∅ /* queue of top-k cohorts within C */
6      CO = ∅ /* global queue of top-k cohorts */
7      searchCliques(V, V, C) /* recursive function */
```

Algorithm 2: searchCliques($Sub, Cand, C$)

```
1  begin
2      /* Sub: seed set to be searched for cliques */
3      /* Cand: expansion set for building cliques */
4      /* N_G(u): friends of u in G*/
5      if Sub ≠ ∅ then
6          u ← vertex ∈ Sub maximizing |Cand ∩ N_G(u)|
7          foreach v ∈ Cand \ N_G(u) do
8              Sub_v ← Sub ∩ N_G(v)
9              Cand_v ← Cand ∩ N_G(v)
10             cohG_M ← Cand_v ∪ C ∪ {v}
11             /* cohG_M: largest possible clique on v */
12             if (C(|cohG_M|, 2))(1 − α) > min_{S∈CO} (f_α(S)) then
13                 searchCliques(Sub_v, Cand_v, C ∪ {v})
14             Cand ← Cand \ {v}
15     else
16         cohG ← findCohorts(G_A, C, k)
17         CO ← CO ∪ cohG
```

Algorithm 1 is the shell of COVER; it initializes variables and priority queues and finds top-k activity-based cohorts recursively on the back of social cliques. Algorithm 2 searches for promising cliques C in the social graph G. We start with the set of all users V, and recursively explore subgraphs having a clique property in depth-first-search manner. We maintain two set variables: Sub maintains the intersection of the neighbor sets of all nodes already entered in the clique C currently under construction. On the other hand, $Cand$ maintains the intersection of such neighbor sets *minus* any nodes that have already been checked, i.e., included, or considered for inclusion, in C. We use $Cand$ to generate new candidates for checking at each iteration (Line 7). Besides, to accelerate the search, at each iteration we pick up a high-degree *pivot* node $u \in Sub$ having many neighbors $N_G(u)$ in $Cand$ (Line 6). Then, we check candidate nodes $v \in Cand \backslash N_G(u)$ (Lines 7-14) for inclusion in C. We exclude those neighbors from the search (Line 7), since, if we have not already considered them, we consider them recursively later by virtue of them being neighbors of u. A critical check is performed in Line 12: if the largest possible clique that can be built by including v and its neighbors can yield a best-case activity density *among* the current top-k values, then C is recursively expanded with v (Line 13); otherwise, we discard the depth-first search path leading to v, as it cannot bring forth new results, and *thereby we avoid redundant computations*. Lastly, when clique C cannot be ex-

panded further, we search for the top-k cohorts therein by calling Algorithm 3 and update the global priority queue $\mathcal{CO}$ accordingly (Lines 16-17).

Algorithm 3 finds top-k cohorts within G_A and social clique C by local search. A candidate cohort S starts out as the node $u \in C$ of highest ratio of adjacent triangles to degree and its neighbors (Line 3). We iteratively revise S, first by adding nodes, as long as that can bring a benefit in activity density, choosing the best such option (Lines 8–12); then by removing the best node whose removal brings benefit (Lines 13–16); we repeat until we reach a local optimum or the maximum iterations T_{max}. In each iteration, we insert the running S to the priority queue $cohG$, and eventually merge the result in priority queue $\mathcal{CO}$.

Algorithm 3: findCohorts(G_A, C, k)

```
1  begin
2      u : vertex with max #triangles/degree ratio in G_A[C]
3      S ← N(u) ∪ {u} /* u and G_A neighbors */
4      cohG ← {S}
5      b_1 ← True, t ← 1 /* local search begins */
6      while b_1 and t ≤ T_max do
7          b_2 ← True
8          while b_2 do
9              if ∃v ∈ C \ S such that f_α(S ∪ {v}) ≥ f_α(S) then
10                 S ← S ∪ {v}; cohG ← cohG ∪ S
11             else
12                 b_2 ← False /* growth of S stops */
13         if ∃x ∈ S such that f_α(S \ {x}) ≥ f_α(S) then
14             S ← S \ {x}; cohG ← cohG ∪ S
15         else
16             b_1 ← False /* local search stops */
17         t ← t + 1 /* iteration counter */
```

Given the worst-case complexity of clique enumeration [20], which forms the backbone of COVER, the worst-case complexity of COVER is $O(3^{\frac{n}{3}} + cT_{max}m)$, where n is the number of nodes, c the number of enumerated cliques that reach Line 12 of Algorithm 2, m the number of activity graph edges, and T_{max} the maximum number of iterations in Algorithm 3, which touches each edge at most once per iteration [21]. We avoid this worst-case scenario by a *massive* discarding of paths in Line 12 of Algorithm 2. Therefore the algorithm is efficient in practice.

5 Experimental Study

We present an extensive experimental evaluation of COVER, including its capacity to predict convoys of mobile companions, *regardless of whether they form a social clique*. We ran all experiments on a 2.3GHz, 4 AMD Opteron 6376 Linux machine with 512GB of RAM. All algorithms are implemented[4] in Scala.

Datasets. We utilize three real-world datasetsfrom Foursquare, Gowalla, and Weeplaces [14] (The Gowalla and Weeplaces data are available at `https://www.yongliu.org/datasets/`). Table 1 gathers information about the data. Each of the datasets consisted of three parts: the social friendship graph, an ordered list of check-ins, and a collection of Venues. A check-in record contains the user-id, check-in time, GPS coordinates, and a location-id. Venues provide

[4] The code is available at `http://bit.ly/2tUTEuu`.

the details of locations, i.e., city, country, and semantic categories of those locations. **Data Preprocessing.** The data required cleaning, as many locations

	Users	Locations	Checkins	POIs	Friend pairs	Duration	Categories
FourSquare	4K	0.2M	0.47M	0.12M	32K	1322 days	35
Gowalla	77K	2.8M	18M	2M	4M	913 days	363
Wee	16K	0.9M	8M	0.76M	0.1M	2796 days	770

Table 1: Dataset characteristics

were associated with multiple identifiers, each having slightly different GPS coordinates. We applied grid-based spatial clustering on GPS points, with a grid of size 10m $\times$ 10m, as in [17]. We assign a unique location Id to each resulting cluster and use these Ids as POIs in all experiments. All three datasets presented similar multiplicity problems, which we addressed in the same manner. Statistics regarding these new POI Ids are reported in Table 1.

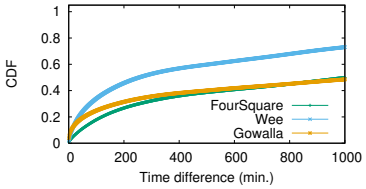
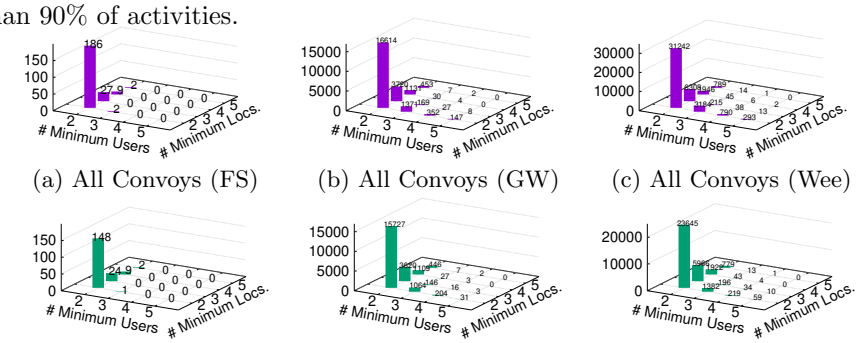


Fig. 1: CDF of time between consecutive activities

5.1 Brute-Force Convoy Retrieval

We first present a brute-force baseline that retrieves *groups of users moving together, i.e., convoys*. We divide all activities into a series of *time intervals*, or snapshots, using a time-stamp threshold t_s, recording *one* activity per interval: the last recorded activity. We can tune t_s so as to strike a fine balance in the tradeoff between time granularity and computation time. With larger t_s values, it becomes likelier to miss activities within a time snapshot. To set a suitable t_s value, we plot the cumulative distribution of time differences between consecutive user activities in Figure 1. By this plot, we set $t_s = 1$ hour, which covers more than 90% of activities.



(a) All Convoys (FS) (b) All Convoys (GW) (c) All Convoys (Wee)

(d) Clique Convoys (FS) (e) Clique Convoys (GW) (f) Clique Convoys (Wee)

Fig. 2: Statistics on convoys: all convoys vs. convoys that also form social cliques

We detect convoys based on recorded activities. A convoy should: (i) contain *at least two users* moving together across locations; and (ii) involve *at least two* consecutively visited locations. We maintain a convoy list L and iterate

over data snapshots in temporal order. In each snapshot, we group activities by location and check each group against L. If a group extends an existing convoy C, we update C accordingly; if a group forms a new convoy, we insert it in L. We store away items in L that are no longer expandable, and go on til the last snapshot. This approach is *more computationally demanding* than COVER's activity density estimation: it detects all groups in the training set, while COVER only considers pairs of users and at most pairs of consecutive activities, and social links. Figure 2 shows statistics on convoys so retrieved vs. those whose members *also* form social cliques. About 76% of all convoys form *social cliques*. This finding validates our conjecture that people are likely to *move in social cliques*, hence justifies our clique constraint.

5.2 Use Case: Convoy Prediction

We surmise that the cohorts COVER retrieves predict *convoys* of mobile companions; we design an experiment to assess that conjecture. We *do not test* an ability to predict *social cliques*; we claim that the clique constraint in our problem helps predict *convoys*. We use a *holdout approach*: we sort activities by timestamp and divide them into training (earlier) and test (later) sets, having an equal number of activities. A group retrieved from the training data thta forms a convoy in the test data is a true positive. We apply two prediction regimes:

Without Input Categories: We find groups likely to form future convoys. COVER ranks groups by activity density; brute-force by appearances.

With Input Categories: In this case, we are given a set of categories of locations of interest $\mathcal{L}$. We find groups deemed likely to form future convoys, moving among locations of the given categories. To identify such groups, we filter the dataset so that we maintain only activities at locations of $\mathcal{L}$. Then, we retrieve groups as explained above; again the brute-force methods ranks convoys by count, while COVER ranks geo-social cohorts by surplus.

We measure a method's capacity to predict future convoys in the test data by averaging an accuracy measure, defined as follows:

$$Acc = \frac{|C \cap T|}{\min\{|C|, |T|\}} \tag{4}$$

where C is the returned set of top-k groups and T the set of convoys in the test data. We *sanitize* the measure, dividing by the minimum of $|C| = k$, and $|T|$. The rationale is that the denominator should exceed neither k, since the top-k results cannot be more than k, nor the number of existing convoys in the test data, which may be less than k, especially when we filter by input categories. In effect, this measure is the maximum of *precision* and *recall*; we think this is a reasonable measure given the sparsity of real-world LBSN data.

5.3 Revalidating the Social Clique Constraint

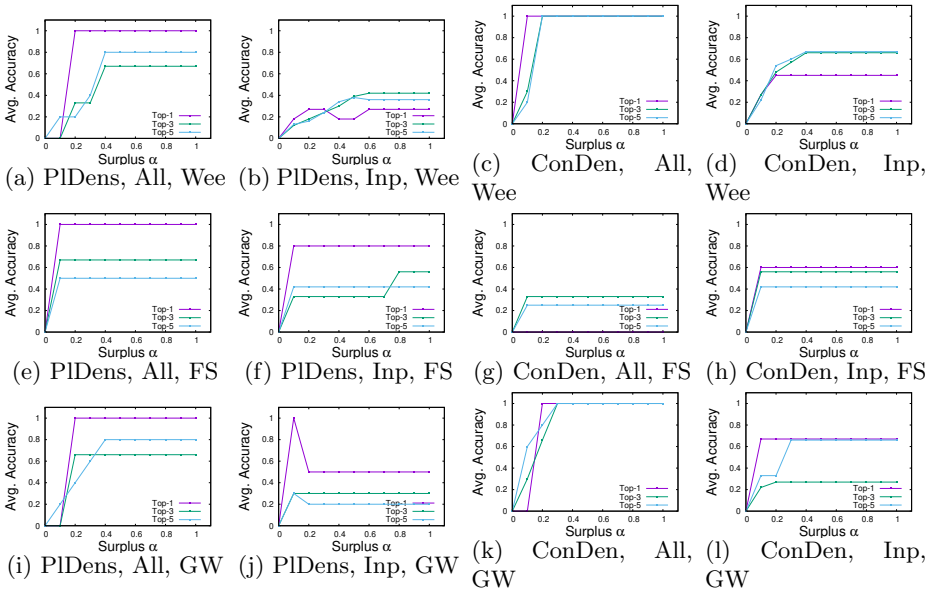
Before we proceed, we revalidate our social clique conjecture. To do so, we test COVER *without considering the social graph*. We extract top-k groups in terms

of activity edge surplus in the training data, for $k = 1, 3, 5$. This way, we only get a positive prediction with the Wee dataset for $k = 3$, which is five times less accurate than what we achieved using the clique constraint. This result reconfirms the logic of that constraint: users who engage in common a activity but do not form a social clique are not likely to do so again. Further, we relaxed the clique constraint to find *quasi-cliques* in the social graph for several α values. Unfortunately, this way we could not predict *any* group of travel companions. We reiterate that the groups (convoys) we aim to *predict* are *not required* to form social cliques. We simply observe that groups of future travel companions *form* social cliques, and *we can well predict them using this constraint.*

5.4 Prediction without Input Categories

We first present prediction results for the case *without* a restrictive set of given categories of interest $\mathcal{L}$.

Brute-force method: We run the brute-force method of Section 5.1 on the training data, retrieving the k most frequently observed convoys, and test its predictions on the test data. The last three columns in Table 2 show top-k convoy prediction accuracy for $k = 1, 3, 5$, reaching 100% in all but two cases.



(a) PlDens, All, Wee (b) PlDens, Inp, Wee (c) ConDen, All, Wee (d) ConDen, Inp, Wee

(e) PlDens, All, FS (f) PlDens, Inp, FS (g) ConDen, All, FS (h) ConDen, Inp, FS

(i) PlDens, All, GW (j) PlDens, Inp, GW (k) ConDen, All, GW (l) ConDen, Inp, GW

Fig. 3: Use Case: COVER's performance on convoy prediction with three datasets

COVER on plain activity density: We test COVER on the same problem, setting weights in the activity graph without considering consecutive activities. Figures 3a, 3e, and 3i present results on top-k group prediction as a function of density surplus α. The top prediction becomes correct at $\alpha = 0.1$ and remains so for larger values of α; for top-3 and top-5 returned groups, the accuracy is lower. This is due to variations among training and test data; similar

divergences appear in Table 2 even with the brute-force method. The fact that COVER does almost as well is remarkable, as it is up to 3 orders of magnitude faster, as we will see in Figure 4b. In most cases, accuracy drops as k increases; it is easier to get a correct top group than the whole group of top 3 or 5.

COVER on consecutive activity density: Next, we set density by consecutive spatiotemporal join. As Figures 3c, 3g, and 3k show, smaller α achieve maximum accuracy. A large α forces the detection of small, tightly connected groups; the consecutivity requirement creates smaller cohorts on its own, rendering the impact of α less significant. COVER performs remarkably well.

5.5 Prediction with Input Categories

Now we examine the case *with* a restrictive set of categories of interest, $\mathcal{L}$. We construct 10 instances of $\mathcal{L}$, each consisting of 2–3 categories that appear together in real-world contexts, and report the *average* accuracy over all queries. For COVER, edge weights in activity graph $G_A^{\mathcal{L}}$ are based solely on locations in $\mathcal{L}$.

Brute-force method: The brute-force method achieves accuracy up to 66% (Table 2), as convoys specific to the input categories may not be met in the training data, but appear in the test data.

	Input Cat			**All Cat**		
k	**FS**	**GW**	**Wee**	**FS**	**GW**	**Wee**
1	0.60	0	0.50	1	1	1
3	0.66	0.17	0.60	1	1	1
5	0.64	0.20	0.70	0.30	0.80	1

Table 2: Accuracy in brute-force convoy prediction

COVER on plain activity density: Figures 3b, 3f, and 3j present the average top-k set prediction accuracy for COVER with plain activity density. While the problem is more challenging due to data sparsity, we still obtain high accuracy in most cases. Less accuracy variation with varying α appears on Foursquare, the smallest of our data sets, as fewer convoys arise in it. Accuracy still drops with increasing k, except for the case of the Wee data, where enlarging the set of results raises the chances they exist in the test data.

COVER on consecutive activity density: Lastly, Figures 3d, 3h, and 3l show the results for COVER with consecutive activity density. Remarkably, accuracy is either similar to or *significantly higher* than that in the non-consecutive case reaching 67% for the top-5 cohorts. This result vindicates the use of consecutive activity density. On Foursquare and Gowalla, accuracy increases sharply as α grows by virtue of smaller returned group sizes, then stabilizes above 60%.

5.6 Effect of Surplus Parameter α

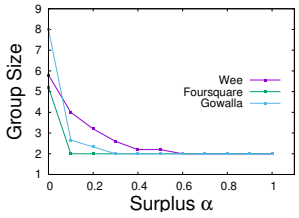
Prediction accuracy grows with α, yet the size of retrieved geo-social cohorts decreases with α, as higher values demand stronger cohesion. To study this tradeoff, we measure the average retrieved cohort size vs. α on the top-5 cohorts with input categories and consecutive activity density. The results in Figure 4a show that cohort size decreases with α up to 0.6, then stabilizes at 2; $\alpha = 0.4$ yields both large size and predictive power. We employ this value in Section 5.7.

5.7 Scalability and Prediction Quality

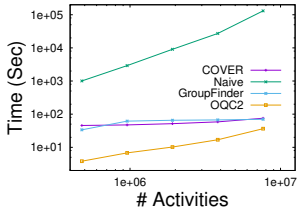
As there is no previous work on the problem we study, we juxtapose COVER with fixed $\alpha = 0.4$ vs. the following methods:

- **BF:** The Brute-Force convoy retrieval method of Section 5.1.
- **GroupFinder:** A method that finds groups of a given size k for a given user u and set categories [6]. We adapt this method to our problem so as to conduct a reasonable comparison: Given a set of categories, we apply GroupFinder to each user in the data set and then choose the best group of size k, where k is the most popular group size returned by COVER (in all cases, 2). We utilize both pairwise user-item relevance measures proposed in [6]: pairwise aggregated voting (PAV) and pairwise least misery (PLM).
- **OQC2**: A COVER variant that relaxes the clique constraint, finding groups of high edge surplus on both graphs, using two α values, one for each surplus component; we try out α values with step 0.1 and weights for the two components in $\{0, 0.25, 0.66, 1, 1.5, 4, \infty\}$, and present best results.

First, we assess all methods in terms of scalability, measuring runtime on 6.25%, 12.5% 25%, 50% and 100% of the Wee data. Figure 4b shows the results (Brute-Force as Naive, OQC2 with $\alpha = 0.6$). Despite an exponential worst-case complexity, the practical runtime of COVER is comparable to that of GroupFinder, and the most scalable of all examined algorithms, while Brute-Force (Naive) does not scale well. Other data produces similar trends.



(a) Average group size vs. α



(b) Runtime vs. data size (Wee)

Fig. 4: Effect of α and scalability in data size

Next, we evaluate the following measures of prediction quality on the Wee data, which has the highest number of observed convoys:

- *Accuracy*, the metric we have used in previous results;
- *Precision@K*, the ratio of the number of true top-k convoys *by frequency* returned, over k or the total number of true convoys, whichever is smaller;
- *Mean Average Precision* (MAP), the mean of Precision@K for all k values up to the examined one; and
- *Normalized Discounted Cumulative Gain* (NDCG), on the *ranking* of returned results vs. their frequency ranking in the test data.

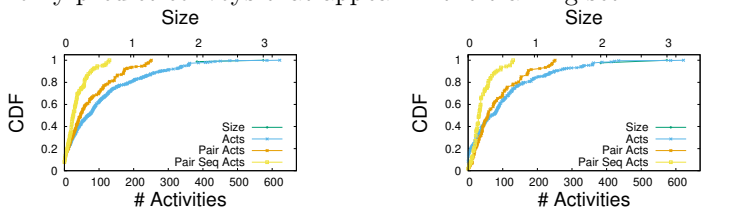
Table 3 shows results on consecutive activity density, with input categories, for five values of k. Brute-Force sometimes outperforms COVER, yet COVER stands its ground in several measures, while avoiding an exhaustive calculation. GF-PAV, GFPLM and OQC perform poorly by all measures; their results are not in the top-20, resulting in 0 value for P@K and MAP. GroupFinder falters as it does not consider activities, but only interests; OQC performs poorly as it abolishes the social clique constraint; this finding *corroborates* that constraint.

	Accuracy					P@K					MAP					NDCG				
K	BF	GF PAV	GF PLM	OQC	COVER	BF	GF-PAV	GF-PLM	OQC	COVER	BF	GF-PAV	GF-PLM	OQC	COVER	BF	GF-PAV	GF-PLM	OQC	COVER
1	0.5	0.01	0.01	0.25	**0.6**	0	0	0	0	0	0	0	0	0	0	**0.5**	0.09	0.1	0.25	0.35
5	**0.7**	0.03	0.03	0.1	0.4	**0.35**	0	0	0	0.15	**0.18**	0	0	0	0.17	0.7	0.25	0.1	0.5	**0.76**
10	0.6	0.03	0.03	0.05	**0.63**	**0.22**	0	0	0	0.15	**0.22**	0	0	0	0.16	0.79	0.25	0.15	0.5	**0.81**
15	0.5	0.08	0.05	0.03	**0.57**	**0.18**	0	0	0	0.15	**0.22**	0	0	0	0.16	0.81	0.1	0.25	0.5	**0.81**
20	0.47	0.08	0.05	0.025	**0.53**	0.14	0	0	0.01	**0.14**	**0.2**	0	0	0	0.16	0.8	0.25	0.25	0.5	**0.82**

Table 3: Comparative Evaluation on Wee (α=0.4 for COVER, best OQC values)

5.8 Analysis on Retrieved Groups

Figure 5 presents the cumulative density function of characteristics for all retrieved geo-social cohorts (5a) and those forming mobility convoys (5b), i.e., their size and number of activities performed by their members (i.e., individuals or pairs). Group size goes from 2 to 3 (green line), while we predict convoys correctly even when they have not performed any activities together on input categories. This capacity of COVER sets it apart from the brute-force baseline that can only predict convoys that appear in the training set.



(a) All Retrieved Cohorts (b) Correctly Predicted Convoys

Fig. 5: Analysis of Retrieved Groups (Wee)

6 Conclusion

We proposed the problem of finding geo-social cohorts of frequent companions in LBSNs, defined in terms of (i) a *selective* clique constraint on a social graph, and (ii) a density objective on a coterminous graph (i.e., defined on the same set of nodes) capturing common (and *consecutive*) pairwise activities. We designed COVER, a nontrivial algorithm for that problem. Our experimental study with real-life data sets showed that COVER is effective, scalable, and efficient; moreover, it predicts future convoys (i.e., groups moving together), including convoys that do not appear in the training set, while neither an adaptation of previous work nor a brute-force approach based on user traces can deliver such a result.

In the future, we intend to (i) expand our techniques so as to include different activity density functions, and (ii) study the robustness of our method, in terms of its predictive power, in the face of missing, incomplete, uncertain, noisy, and privacy-aware data.