Community Detection using Social Relations and Trajectories

Jifeng Wu, Yuanyuan Zhu

June 2022

1 Abstract

Community detection is an essential task in social network analysis. However, many friends on social networks are not close to one another in the real world. In this paper, we explore utilizing trajectories consisting of user check-ins to detect cohesive groups of users within social networks who frequently hang out together. First, we propose an algorithm to efficiently calculate the spatiotemporal similarity between two discrete trajectories in linear time. Then, we propose a community detection algorithm to discover communities where we jointly enforce social cohesion and trajectory similarity. Finally, we evaluate our trajectory similarity algorithm and community detection algorithm on two datasets, which validate the effectiveness and efficiency of our trajectory similarity algorithm and offer valuable insights into our community detection algorithm.

2 Introduction

Community detection, or cohesive subgraph search, is essential in social network analysis. In a social network, cohesive subgraphs are subsets of users among whom there are relatively strong, direct, intense, frequent, or positive ties [63], and finding such subsets is one of the major concerns of social network analysis with many applications.

However, social networks can be rife with casual acquaintances and zombie friends who are only nominally "friends" on the online platform but are not close with one another in the real world. However, through trajectory similarity, it is possible to filter intimate relationships that frequently hang out together offline.

With the proliferation of GPS-enabled smartphones, location-based social networks, where users can share their check-in locations with friends, have rapidly developed. Given such check-ins, we can extract users' trajectories containing large amounts of spatiotemporal information and reflect the users' moving patterns. Furthermore, by assessing the similarities between such tra-

jectories, we can discover communities where social cohesiveness and mutual trajectory similarity are jointly enforced.

This is a flexible and robust paradigm of community detection with many potential applications.

1. Social platforms could utilize users' social relations and trajectories to provide grouping recommendations for travel. In this case, groups containing users cohesive in terms of trajectory similarity coupled with relatively low social distance are beneficial. This allows the users to expand their social circles, as users with similar trajectories have similar moving patterns and are likely to have similar hobbies and ways of life, facilitating friend-making.

2. In an epidemic, a virus could spread to acquaintances with frequent contact. Thus, epidemic prevention and control workers can jointly use social data and trajectory data to find probable cluster cases and take measures to curb the spread of the virus.

3. This approach is also helpful in tracking down clandestine criminal gangs, as such members are often acquaintances with each other and display similar moving patterns when conducting criminal activity, facilitating the need to utilize both data sources jointly.

Existing studies on trajectory mining consider trajectories to be either continuous or discrete. A continuous trajectory records an object's continuous movement with high fidelity, and interpolation on such a trajectory is feasible. In contrast, a discrete trajectory only records a few places an object visits, rendering interpolation infeasible. In our study, the trajectories extracted from check-ins in location-based social networks are discrete, as users often check in when they desire instead of periodically. Thus the polyline formed by connecting a user's check-in points does not accurately reflect the user's actual movement. For example, a user's ith check-in might be recorded at 8:00 AM at home, while the user's $(i+1)$th check-in might be recorded at 10:00 AM the next day in a scenic spot. Although the two check-ins are adjacent, we only know that the user was at home at 8:00 AM on the first day and the scenic spot at 10:00 AM on the second day. It is not a reasonable estimate that the user gradually moved from home to the scenic spot in a straight line during this period. However, we can still assess the similarity between such discrete trajectories through a trajectory similarity measure, which measures the overall similarity between two trajectories.

Existing trajectory similarity measures usually calculate a distance matrix containing the distances from each point in the first trajectory to each point in the second trajectory. Afterward, they would find an alignment between the points of the two trajectories or find a critical match pair minimizing or maximizing a given condition. In addition, Li et al. [35] proposed a linear-time, deep representation learning-based trajectory similarity algorithm that models points

as "words" sampled from a predefined vocabulary of grid locations and trajectories as "sentences" made up of these words. Unfortunately, finding a critical match pair is sensitive to noise [56], while [35] requires a predefined vocabulary of grid locations, so we choose an alignment-based approach to trajectory similarity. However, existing alignment-based trajectory similarity measures have a quadratic time complexity, which is infeasible for calculating pairwise similarities between large numbers of long trajectories.

Furthermore, in our problem, temporal similarity is an essential aspect of trajectory similarity. For example, even if two users' trajectories visit similar locations in similar orders, visiting them at different times still suggests that the two users have different activity modes. Thus, we propose an alignment-based trajectory similarity model considering both spatial and temporal similarity that utilizes temporal constraints to support the efficient calculation of the similarity between two discrete trajectories in linear time. In addition, we propose a new community detection algorithm using social relations and trajectories to find socially cohesive communities where we only retain social connections reinforced with high trajectory similarity.

To sum up, the main contributions of this paper are as follows:

1. We propose a trajectory similarity model enabling us to calculate the spatiotemporal similarity between two discrete trajectories in linear time.
2. We propose a community detection algorithm to discover communities where only users who are friends on the social network and have similar trajectories are connected.
3. We evaluate our trajectory similarity algorithm and community detection algorithm on two datasets.

The rest of the paper is organized as follows. Section 3 presents related work on community detection and trajectory similarity calculation. Section 4 introduces mathematical notations and definitions used throughout this paper and formalizes our problem. Sections 5 and 6 explain in detail how we calculate trajectory similarity and perform community detection respectively. Section 7 evaluates our proposed approaches with an experimental study on two public datasets. Finally, Section 8 concludes.

3 Related Work

3.1 Community Detection

Finding communities, subgraphs in which nodes are densely connected with each other, is an essential topic in graph mining with many applications, and there is a wealth of research on this topic.

Classic community models, such as clique [40], k-core [52], and modularity-based community models [27, 45, 12], often define communities based on edges, the known relations between vertices. Of these models, the k-core model is of

particular interest. Being both efficient and effective, it has various extensions, such as k-truss considering triangles instead of node degrees [19], s-core for weighted graphs [20], D-core for directed graphs [26], and multilayer k-core for multilayer graphs [24]. [42] presents a thorough survey of these models.

Some recent community detection research also considers both the social and spatial constraints in finding communities. Modularity maximization-based algorithms for community detection in spatial graphs were studied in [23, 17]. [69] studies geosocial group queries with minimum acquaintance constraints, including the problems of finding the maximum k-core in a given rectangle containing a query vertex and finding the k-core with strictly (or no less than) c vertices such that the longest distance from these vertices to q is minimized. The (k, r)-core community model [68] uses pairwise similarity (distance) between each pair of vertices to ensure the spatial cohesiveness of communities when computing the maximum k-core or all maximal k-cores. Similarly, the RB-k-core model [62] restricts a k-core model, ensuring social cohesiveness within a radius-bounded circle. Furthermore, [66] and [31] proposed algorithms for detecting socially cohesive communities of users in location-based social networks that are spatially cohesive based on the density of their locations. Beyond spatial coordinates, [34] considered spatial cohesiveness on a road network, and models queries of communities satisfying both social cohesiveness and spatial cohesiveness as skyline queries, where each community cannot be dominated by any other in terms of social cohesiveness and spatial cohesiveness.

3.2 Trajectory Similarity

A fundamental problem in trajectory data mining is determining how similar or distinct two trajectories are. [56] presents a comprehensive survey of trajectory distance measures, which divides these measures into two groups: (1) discrete distance measures in which distance values are only calculated based on sample points, such as Dynamic Time Warping [9], Longest Common Subsequence [49], Edit Distance-based distance measures [16, 67], Spatiotemporal Linear Combine Distance [53], Discrete Fréchet Distance [15], and Hausdorff Distance [21], and (2) continuous distance measures in which distance values are calculated based on both sample points and (interpolated) movement in-between, such as Spatiotemporal Euclidean Distance [44] and Locality in-between Polylines Distance [48].

Classic discrete distance measures can be further divided into two groups: (1) those which find an alignment between the points of two trajectories and calculate trajectory distance based on such alignment, and (2) those which find a "critical match pair" consisting of one point from each trajectory and use the distance between these two points to represent trajectory distance. The former includes Dynamic Time Warping, Longest Common Subsequence, Edit Distance-based distance measures, and Spatiotemporal Linear Combine Distance, while the latter includes Hausdorff Distance and Discrete Fréchet Distance. Apart from these classic discrete distance measures, Li et al. [35] also proposed a linear-time, deep representation learning-based discrete trajectory

Table 1: Mathematical notations used throughout this paper.

Notation	Definition
$G(V, E)$	a social network, an undirected graph with vertex set V and edge set E
$N(G, u)$	the neighbors of vertex u in graph G
$d(G, u, v)$	the distance between two users, u and v, within the social network $G(V, E)$ $(u, v \in V)$
$p(\phi, \lambda, t)$	a check-in point on a trajectory, containing latitude ϕ, longitude λ and timestamp t
T	a trajectory, with $T[i]$ being the ith point within the trajectory
$\mathbf{T}$	a trajectory dataset, with $\mathbf{T}[u]$ being the trajectory of user u
$s(T_1, T_2)$	the similarity between two trajectories, T_1 and T_2, under a trajectory similarity measure s
$NN(\mathbf{T}, T, k)$	the k nearest neighbors of trajectory T in a set of trajectories $\mathbf{T}$

similarity algorithm that models points as "words" sampled from a predefined vocabulary of grid locations and trajectories as "sentences" made up of these words.

As for our problem, interpolation is not an option, and we resort to discrete distance measures. In addition, finding a critical match pair is sensitive to noise and is poorer at reflecting the overall similarity between all points. Furthermore, [35] limits us to calculate the similarities between trajectories visiting a finite set of predefined locations and incurs a training cost. As a result, we take inspiration from alignment-based distance measures in our study.

4 Preliminaries

Mathematical notations used throughout this paper are summarized in Table 1.

In terms of social cohesion, we adopt the k-core model [52].

k-core. Given an undirected graph $G(V, E)$, the k-core $G'(V', E')$ of G is a maximal subgraph of G such that each vertex is adjacent to at least k other vertices. For a designated undirected graph G, given different values of k, the corresponding k-cores form a series of hierarchical subgraphs such that if $k_1 > k_2$, the k_1-core of G is a denser subgraph of the k_2-core of G.

In terms of trajectory similarity, there are two criteria we can use to identify similar trajectories, k-nearest neighbors (k-NN), under which each trajectory is considered to be similar to the k other trajectories with the highest similarity value, as well as ϵ-nearest neighbors, under which pairs of trajectories having similarity values above a threshold ϵ are considered to be similar [37]. However, ϵ-nearest neighbors is very sensitive to the parameter ϵ [10, 11], and when used to filter edges, it may result in networks with many disconnected parts under

an improper value of ϵ. Thus, k-nearest neighbors is a better choice.

However, k-nearest neighbors also has its own problems, such as being a one-way relationship. Consider the situation in which a user is not a sociable person, and his or her trajectory is not very similar to any other user's trajectory. In this situation, k-nearest neighbors would still consider the trajectories of k other users to be similar, but should these k users be socially active, it is likely that for each of these users, the k-nearest neighbors of his or her trajectory does not include the aforementioned introverted user's trajectory. To overcome this limitation, mutuality can be enforced when adapting k-nearest neighbors, such that two trajectories, T_1 and T_2, are considered similar only if T_1 is a k-nearest neighbor of T_2, and that T_2 is a k-nearest neighbor of T_1. In addition, directly calculating the k-nearest neighbors of a trajectory in the entire trajectory dataset overlooks social relations. Thus, we only consider the k-nearest neighbors of a user's trajectory within the trajectories of the user's friends on the social network.

Our problem is formally stated as follows.

Problem Statement. Given (1) a social network $G(V, E)$, (2) a trajectory dataset $\mathbf{T}$ such that $\forall u \in V, \exists! T[u] \in \mathbf{T}$, (3) a coreness requirement k, and (4) a trajectory mutual nearest neighbor requirement m, find the maximal subgraph $G'(V', E')$, such that (1) $\forall (u, v) \in E', \mathbf{T}[u] \in NN(\mathbf{T}[N(G, v)], \mathbf{T}[v], m)$, $\mathbf{T}[v] \in NN(\mathbf{T}[N(G, u)], \mathbf{T}[u], m)$, and (2) $G'(V', E')$ is a k-core.

5 Trajectory Similarity

This section introduces our algorithm to calculate the similarity between two trajectories.

As different users check in at different times, the check-in times of two trajectories are usually asynchronous, facilitating a need to find an alignment between the check-ins of two trajectories.

In many traditional alignment-based trajectory distance measures, such as Dynamic Time Warping [9] and Spatiotemporal Linear Combine Distance [53], this is done by calculating all pairwise distances and minimizing a cost function to find the best alignment between the points of the two trajectories. This implies a time complexity of $O(mn)$, with m, n being the length of the two trajectories, making them impracticable for handling many trajectories.

In our problem, temporal similarity is an essential aspect of trajectory similarity. Even if two users' trajectories visit similar locations in similar orders, visiting them at different times still suggests that the two users have different activity modes. Accordingly, we can implement this temporal requirement during alignment finding. Given two trajectories T_1 and T_2, we first match each point in T_1 with the point in T_2 closest in time, as depicted in Figure 1, before matching each point in T_2 with the point in T_1 closest in time, as depicted in Figure 2.

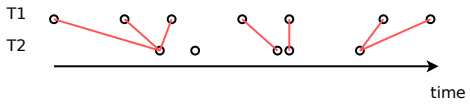


Figure 1: Matching each point in T_1 with the point in T_2 closest in time

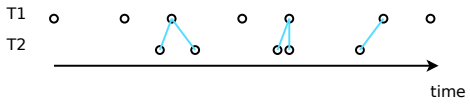


Figure 2: Matching each point in T_2 with the point in T_1 closest in time

5.1 Finding Matching Points Between Trajectories

Matching each point in one trajectory with the point in the other closest in time can be accomplished in linear time. To demonstrate this, we first prove that given two trajectories T_1 and T_2, if the ith point in T_1 matches the jth point in T_2, the index of the next point in T_2, j', that matches the $(i+1)$th point in T_1 must satisfy $j' \geq j$.

Proof.

(1) If $T_2[j].t < T_1[i].t$, it must be the case that $T_2[j'].t \geq T_2[j].t$. Otherwise, given $T_2[j'].t < T_2[j].t$, it would be $T_2[j]$ instead of $T_2[j']$ that matches $T_1[i+1]$, as $T_2[j]$ would be closer in time, as depicted in Figure 3.

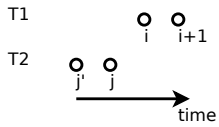


Figure 3: $T_2[j].t < T_1[i].t$, $T_2[j'].t < T_2[j].t$ is impossible

(2) If $T_1[i].t \leq T_2[j].t < T_1[i+1].t$, it must be the case that $T_2[j'].t \geq T_2[j].t$. Otherwise, given $T_2[j'].t < T_2[j].t$, it would be $T_2[j]$ instead of $T_2[j']$ that matches $T_1[i+1]$, as $T_2[j]$ would be closer in time, as depicted in Figure 4.

(3) If $T_1[i+1].t \leq T_2[j].t$, it must be the case that $T_2[j'].t \geq T_2[j].t$. As $T_2[j]$ matches $T_1[i]$, there is no other point in T_2 whose timestamp t satisfies $2T_1[i].t - T_2[j].t < t < T_2[j].t$. Thus, given $T_2[j'].t < T_2[j].t$, it must be the case that $T_2[j'].t \leq 2T_1[i].t - T_2[j].t$, such that it would be $T_2[j]$ instead of $T_2[j']$ that matches $T_1[i+1]$, as $T_2[j]$ would be closer in time, as depicted in Figure 5.

According to (1)(2)(3), it is always the case that $T_2[j'].t \geq T_2[j].t$. Thus, $j' \geq j$.

Furthermore, given a trajectory T, a beginning index b, an ending index e, and a target time t, as $T[i].t$ monotonically increases with the increase of $i \in \{b, b+1 \ldots, e\}$, $|T[i].t - t|$ either monotonically increases, monotonically decreases, or monotonically decreases before monotonically increasing, depending on the value of t. As a result, we can find the index of the next point in T_2, j', that matches the $(i+1)$th point in T_1 using Algorithm 1, which finds

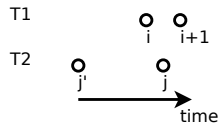


Figure 4: $T_1[i].t \leq T_2[j].t < T_1[i+1].t$, $T_2[j'].t < T_2[j].t$ is impossible

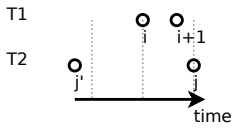


Figure 5: $T_1[i+1].t \leq T_2[j].t$, $T_2[j'].t < T_2[j].t$ is impossible

$\underset{i}{\operatorname{argmin}} |T[i].t - t|, i \in \{b, b+1, \ldots, e\}$.

Algorithm 1: ArgminIndex

Input: trajectory T, beginning index b, ending index e, target time t
Output: $\underset{i}{\operatorname{argmin}} |T[i].t - t|, i \in \{b, b+1 \ldots, e\}$

```
1  begin
2      i ← b;
3      if i = e then
4          return i;
5      |Δt| ← |T[i].t − t|;
6      while true do
7          i′ ← i + 1;
8          if i′ = e then
9              return i;
10         |Δt′| ← |T[i′].t − t|;
11         if |Δt′| > |Δt| then
12             return i;
13         I ← i′; |Δt| ← |Δt′|;
```

With this algorithm, given two trajectories T_1 and T_2, containing m and n points respectively, we can match each point in T_1 with the point in T_2 closest in time, as shown in Algorithm 2.

In Algorithm 2, the loop is iterated m times, once for each point in T_1, while all calls to *ArgminIndex* may result in T_2 being fully traversed as well once the loop finishes. Thus, the total time complexity of Algorithm 2 is $O(m+n)$.

Algorithm 2: MatchingIndices

Input: trajectory T_1 of length m, trajectory T_2 of length n
Output: array J, containing the indices of points in T_2 closest in time to points in T_1

```
1 begin
2     initialize J to an empty array of length m;
3     j ← 1;
4     for i ∈ {1, 2, ..., m} do
5         // Find the index of the point in T_2 closest in time to T_1[i]
          j ← ArgminIndex(T_2, j, n, T_1[i].t);
6         J[i] ← j;
7     return J;
```

5.2 Similarity Between Two Matching Points

The similarity between two matching points can be described from spatial and temporal aspects.

For the spatial aspect, we calculate the geographical distance d between the two points and assume that spatial similarity decays exponentially over distance, with a spatial time constant δ governing decay speed. Thus, if the geographical distance between two points is d, we use $e^{-\frac{d}{\delta}}$ to represent their spatial similarity.

For the temporal aspect, we also assume that temporal similarity decays exponentially over time, with a temporal time constant τ governing decay speed. In other words, if there are two points with a time delta of Δt, we use $e^{\frac{\Delta t}{\tau}}$ to represent their temporal similarity.

To combine the space aspect and the time aspect, we can directly multiply the spatial similarity and the temporal similarity together. This is based on the natural assumption that with time fixed, the similarity between two points decreases with distance and that with distance fixed, the similarity between two points drops if they become further apart in time.

Formally, given the parameters δ, τ, and having calculated d, Δt, the similarity s between two check-ins is calculated as follows:

$$s = e^{-(\frac{d}{\delta} + \frac{\Delta t}{\tau})} \tag{1}$$

5.3 Overall Similarity Between Two Trajectories

Having proposed a model for the similarity between two matching points, we introduce a procedure to calculate the overall similarity between two trajectories.

Given two trajectories T_1 and T_2, matching each point in T_1 with the point in T_2 closest in time and matching each point in T_2 with the point in T_1 closest in time may result in different results, as depicted in Figures 1 and 2. Thus, we

can calculate two one-way similarities, from T_1 to T_2 and from T_2 to T_1, and average them to obtain the overall bidirectional similarity between T_1 and T_2.

The aforementioned one-way similarities can be implemented by first finding the matching points before calculating the similarity of each two matching points averaged over time, as described in Algorithm 3. Finally, we take the minimum two one-way overall similarities in Algorithm 4 to calculate the overall bidirectional similarity between T_1 and T_2.

Algorithm 3: OneWaySimilarity

Input: trajectory T_1 of length m, trajectory T_2 of length n, parameters δ and τ

Output: the one-way similarity from T_1 to T_2

begin
- $TotalTime \leftarrow 0$; $TotalArea \leftarrow 0$;
- $J \leftarrow MatchingIndices(T_1, T_2)$;
- $t_1 \leftarrow T_1[1].t$; $s_1 \leftarrow similarity(T_1[1], T_2[J[1]])$;
- **for** $i \in \{2, 3, \ldots, m-1\}$ **do**
  - $t_2 \leftarrow T_1[i].t$; $s_2 \leftarrow similarity(T_1[i], T_2[J[i]])$;
  - add $t_2 - t_1$ to $TotalTime$;
  - add $\frac{1}{2}(s_1 + s_2)(t_2 - t_1)$ to $TotalArea$;
  - $t_1 \leftarrow t_2$; $s_1 \leftarrow s_2$;
- return $\frac{TotalArea}{TotalTime}$;

Algorithm 4: OverallSimilarity

Input: source trajectory T_1 of length m, target trajectory T_2 of length n, parameters δ and τ

Output: the overall similarity between T_1 and T_2

begin
- $s_{12} = OneWaySimilarity(T_1, T_2, \delta, \tau)$;
- $s_{21} = OneWaySimilarity(T_2, T_1, \delta, \tau)$;
- return $min\{s_{12}, s_{21}\}$;

6 Community Detection

With social network analysis addressing networks of rapidly-increasing size, it is critical to identify community detection methods that are not only effective but also efficient [19]. k-core decomposition, proposed by Seidman [52], presents itself as a viable solution.

The k-core is a maximal subgraph in which each member is adjacent to at least k other members as a cohesive community. It is a time-tested concept that

been applied extensively to real-word graphs in areas as diverse as social network analysis [33, 58], the study of Internet topology [2, 14], complex network modeling [8, 29, 60], anomaly detection [54, 55], influential spreader identification [13, 70, 47, 36, 38, 39, 55], graph similarity [46], large-scale network visualization [6, 5, 2, 1, 4], graph embedding [51], keyword extraction [50, 57, 43], networks of protein interaction [3, 65, 41, 30, 22], and neuroscience [28, 59, 64, 32]. A comprehensive review of its applications can be found in [42].

Besides the fact that it provides an effective manner for finding hierarchical structures of increasing cohesiveness with increasing k within a graph, core decomposition also stands out from more complex and computationally intensive algorithmic techniques with its linear, $O(|V| + |E|)$ time complexity, accomplished using the bin sorting-based algorithm [7], which calculates the coreness of each vertex. The coreness of a vertex v is the maximum value of k such that v is contained in a k-core.

However, directly reducing community detection to finding the k-core within a graph has its caveats, as rather than being sets of high cohesion, Seidman characterizes k-cores as "seedbeds, within which cohesive subsets can precipitate out."[52] Furthermore, in our problem, we must consider trajectory similarity in tandem with social cohesion. Thus, we adopt a community detection scheme of selecting edges based on trajectory similarity before finding the k-core within the selected edges to enforce social cohesiveness.

Specifically, each edge (u, v) is retained if $\mathbf{T}[u] \in NN(\mathbf{T}[N(G, v)], \mathbf{T}[v], m)$ and $\mathbf{T}[v] \in NN(\mathbf{T}[N(G, u)], \mathbf{T}[u], m)$, otherwise, it is filtered out. Such a process can be conducted on the entire graph using the beam search-based algorithm in Algorithm 5.

In this algorithm, every vertex of the social network is visited in a beam search manner and edges (u, v) for which one of $\mathbf{T}[u] \in NN(\mathbf{T}[N(G, v)], \mathbf{T}[v], m)$ and $\mathbf{T}[v] \in NN(\mathbf{T}[N(G, u)], \mathbf{T}[u], m)$ is checked is stored in *SinglyChecked*. At each vertex u (lines 10-32), we examine the m neighboring vertices with trajectories most similar to $\mathbf{T}[u]$. For each such vertex v (lines 16-31), $\mathbf{T}[v] \in NN(\mathbf{T}[N(G, u)], \mathbf{T}[u], m)\}$ is verified. Thus, if $\mathbf{T}[u] \in NN(\mathbf{T}[N(G, v)], \mathbf{T}[v], m)$ has been verified as well (implying $(u, v) \in SinglyChecked$, lines 17-19), the edge (u, v) is moved into *Selected* (line 18). There are two cases if it still needs to be verified (lines 20-30).

1. Whether or not $\mathbf{T}[u] \in NN(\mathbf{T}[N(G, v)], \mathbf{T}[v], m)$ is yet to be confirmed. In this case, (u, v) is added to *SinglyChecked*. If v is in the same layer as u, we will visit v later, and if it is not, we will add it to *Next* to schedule it to be visited later.

2. $\mathbf{T}[u] \in NN(\mathbf{T}[N(G, v)], \mathbf{T}[v], m)$ has been confirmed to be false, implying $v \notin Remaining \wedge (u, v) \notin SinglyChecked$. In this case, (u, v) is filtered out, and nothing is done.

This algorithm requires calculating the trajectory similarities corresponding to each edge within the social network, which incurs a time complexity of

Algorithm 5: SelectEdges

Input: social network $G(V, E)$, trajectory dataset $\mathbf{T}$, δ, τ, m
Output: set of selected edges *Selected*

```
1  begin
2      initialize Selected and SinglyChecked to ∅;
3      initialize Similarities to an empty map;
4      Remaining ← V;
5      while Remaining ≠ ∅ do
6          remove a vertex v from Remaining;
7          Current ← {v};
8          while Current ≠ ∅ do
9              Next ← ∅;
10             for u ∈ Current do
11                 for v ∈ N(G, u) do
12                     if (u, v) ∉ Similarities then
13                         Similarities[(u, v)] ←
                            OverallSimilarity(T[u], T[v], δ, τ);
14                 for v ∈ {v|T[v] ∈ NN(T[N(G, u)], T[u], m)} do
15                     if (u, v) ∈ SinglyChecked then
16                         move (u, v) from SinglyChecked to Selected;
17                     else
18                         if v ∈ Current then
19                             insert (u, v) to SinglyChecked;
20                         else
21                             if v ∈ Remaining then
22                                 insert (u, v) to SinglyChecked;
23                                 insert v to Next;
24             Remaining ← Remaining − Current;
25             Current ← Next;
26     return Selected;
```

$O(L|E|)$, with L being the average trajectory length. Each vertex of the social network is visited in a beam search manner, requiring $O(|V| + |E|)$ time, and for each vertex u, $\{v|\mathbf{T}[v] \in NN(\mathbf{T}[N(G,u)], \mathbf{T}[u], m)\}$ is calculated, which has an $O(deg(u) \log m)$ cost. As $\sum_{u \in V} deg(u) \log m = 2|E| \log m$, the total time complexity of Algorithm 5 is $O(L|E|) + O(|V| + |E|) + O(2|E| \log m) = O(|V| + (L + 2 \log m)|E|)$.

Finally, the entire community detection procedure is encapsulated in Algorithm 6. The time complexity is also $O(|V| + (L + 2 \log m)|E|)$.

Algorithm 6: CommunityDetection

Input: social network $G(V, E)$, trajectory dataset $\mathbf{T}$, δ, τ, m, k
Output: the community, $G'(V', E')$

```
1 begin
2     E' ← SelectEdges(G, T, δ, τ, m);
3     replace edge set of G, E, with E';
4     Corenesses ← CalculateCorenesses(G);
5     calculate the subgraph of G, G'(V', E'), such that
        v ∈ V', Corenesses[v] > k;
6     return G';
```

7 Experimental Study

7.1 Datasets

We conduct experiments on two public datasets.

Brightkite [18]. The Brightkite dataset including a friendship network and check-in data was generated worldwide from April 2008 to October 2010. We filter out those users with fewer than 10 check-in points and those check-in points with fewer than 10 users. The filtered friendship network comprises 1,849 users and 13,065 friendships, while the filtered check-in data contains 257,179 check-ins, with an average of 76 check-ins per trajectory.

Gowalla [18]. The Gowalla dataset including a friendship network and check-in data was generated worldwide from February 2009 to October 2010. We filter out those users with fewer than 15 check-in points and those check-in points with fewer than 10 users. The filtered friendship network comprises 18,737 users and 86,985 friendships, while the filtered check-in data contains 1,278,274 check-ins, with an average of 48 check-ins per trajectory.

7.2 Experimental Setup

In this paper, we have proposed a trajectory similarity algorithm and a community detection algorithm. They are implemented in C++ while the data analysis is conducted in Python. The experiments were conducted on an 8-core,

64-bit Linux server with 32 GB of RAM. In all experiments, we use the spatial time constant $\delta = 1000$m and the temporal time constant $\tau = 3600$s in our trajectory similarity algorithm. All source code, including our algorithms and our data analysis code, is available on GitHub[1].

7.3 Case Study

We conducted a case study on discovering communities where social cohesiveness and mutual trajectory similarity are both enforced. We performed community detection on the Brightkite dataset with $k = 3$ and $m = 5$. Figure 6 presents the detected communities, a subgraph of the 3-core of the Brightkite social network, while Figures 7 and 8 present the locations and times of the check-ins within each user's trajectory for each connected component.

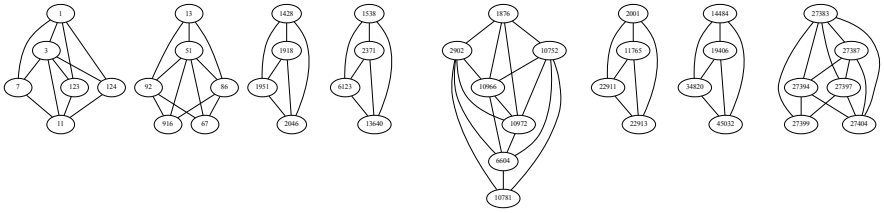
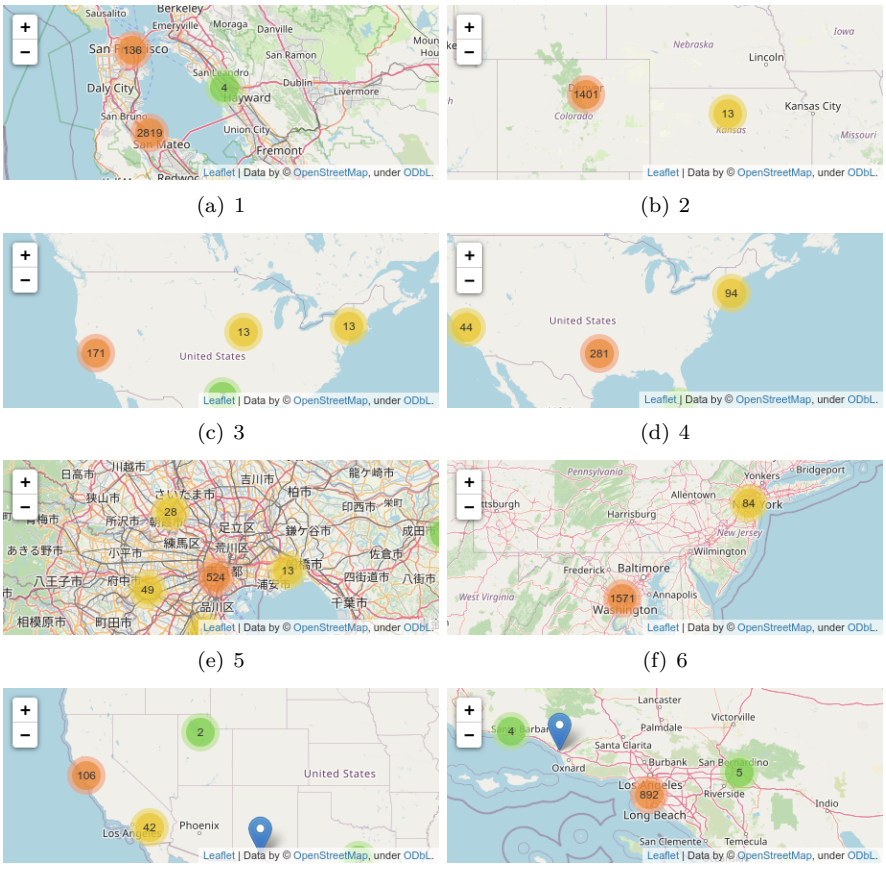


Figure 6: The Detected Communities

[1]https://github.com/abbaswu/community-detection-using-social-relations-and-trajectories



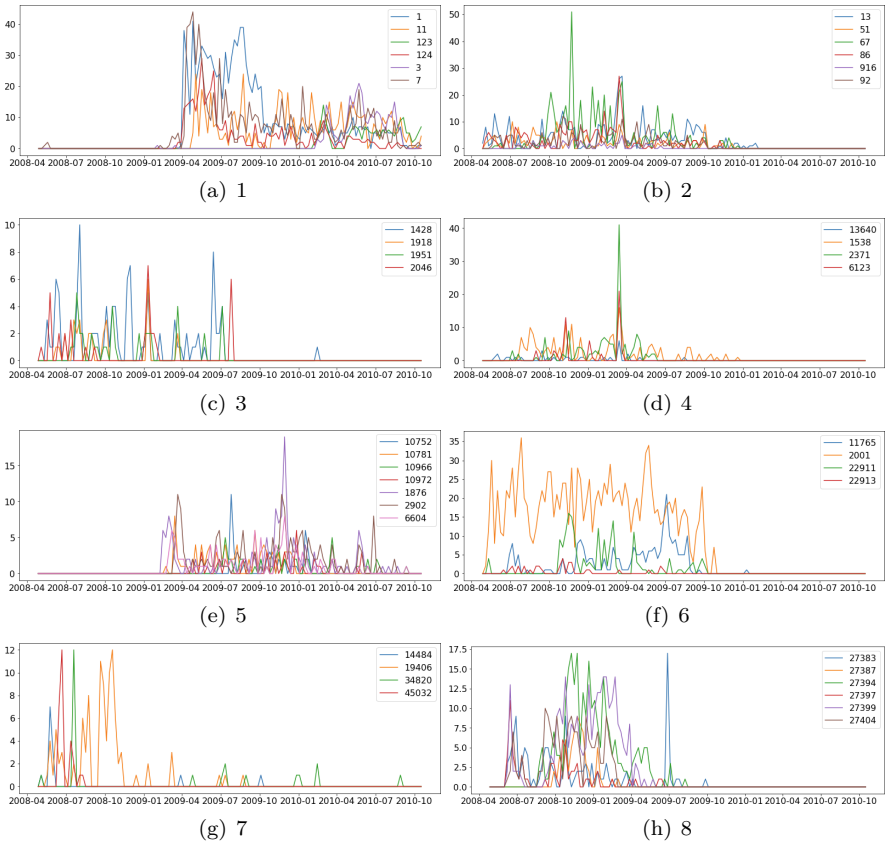
(a) 1 (b) 2

(c) 3 (d) 4

(e) 5 (f) 6

(g) 7 (h) 8

Figure 7: Check-in Locations



(a) 1 (b) 2

(c) 3 (d) 4

(e) 5 (f) 6

(g) 7 (h) 8

Figure 8: Check-in Times

Figure 6 shows that our community detection algorithm detects communities with socially cohesive connected components. Furthermore, Figures 7 and 8 present the spatiotemporal cohesiveness of the trajectories of the connected component's users, such as checking in at similar locations (Figure 7) and at similar times (Figure 8). Furthermore, though all users are in the 3-core of the Brightkite social network, we can observe that users in different connected components exhibit markedly different mobility patterns, as suggested by the geographically distinct check-in locations and separated check-in times. This showcases our community detection algorithm's ability to detect communities with both cohesiveness and trajectory similarity, which is useful in scenarios that require filtering intimate relationships who do frequently hang out together offline from a large social network, such as grouping recommendations for travel based on both social acquaintances and similar moving patterns, detecting probable cluster cases in epidemics, and tracking down clandestine criminal gangs.

7.3.1 Evaluation of Our Trajectory Similarity Algorithm

We have proposed an algorithm that can calculate the spatiotemporal similarity between two discrete trajectories in linear time. Thus, we shall evaluate our trajectory similarity algorithm from the following two aspects:

1. Speed. We compare the running time of our algorithm with those of other spatiotemporal discrete trajectory similarity algorithms.
2. Correlation between matching point spatiotemporal distance and trajectory similarity. For each pair of trajectories involved in trajectory similarity calculation, we calculate the average spatial and temporal distance between matching points detected by our algorithm or other spatiotemporal discrete trajectory similarity algorithms. We then compute the Spearman's rank correlation coefficient [25] between matching point spatiotemporal distance and trajectory similarity for all pairs of trajectories under different trajectory similarity algorithms. The closer to -1 the correlations are, the better the trajectory similarity algorithm encapsulates both spatial and temporal similarity within trajectory similarity.

We perform calculations on our datasets using our algorithm and two other discrete spatiotemporal trajectory similarity algorithms presented in [56], namely Spatiotemporal Longest Common Subsequence (STLCSS) [61], and Spatiotemporal Linear Combine (STLC) [53].

- For STLCSS, we set its parameters, ε and Δ, which control how far in space and time two trajectories can go in order to match a given point from one trajectory to a point in another trajectory, to be multiples of our spatial and temporal time constant (e.g. $\varepsilon = k\delta, \Delta = k\tau, k \in \{1, 2, \dots\}$).
- For STLC, we set its parameter, $\lambda \in [0, 1]$, which controls the relative importance of the spatial and temporal similarities, to the values $\{0.2, 0.4, 0.6, 0.8\}$.

7.3.2 Evaluation of Our Community Detection Algorithm

Compared to traditional community detection algorithms that only consider known social relations between users, our community detection algorithm utilizes both social relations and trajectories extracted from user check-ins. Considering that our goal is to discover communities where social cohesiveness and mutual trajectory similarity are both enforced out of casual acquaintances and zombie friends who are not actually close with one another in the real world, we evaluate our community detection algorithm from the following aspects.

1. Size of the detected communities. We measure the number of users in the detected communities under different values of k and m.
2. Social cohesiveness of the detected communities. We measure the distances between two users within each connected component of the detected communities under different values of k and m.

3. Spatial cohesiveness of the detected communities. We measure the similarities between two users' trajectories within each connected component of the detected communities under different values of k and m.

4. The runtimes of our community detection algorithm under different values of k and m.

7.4 Experimental Results

7.4.1 Evaluation of Our Trajectory Similarity Algorithm

Speed The average time required to calculate the similarity between a pair of trajectories using our trajectory similarity algorithm, OverallSimilarity, as well as STLCSS and STLC, on the Brightkite and Gowalla datasets, are presented in Table 2.

Besides being able to calculate the similarity between a pair of trajectories much faster than STLCSS and especially STLC, our algorithm, being a linear-time algorithm, also outperforms STLCSS and STLC by virtue of scaling well with increased trajectory length, as evidenced by the change in average trajectory similarity calculation time from the Gowalla dataset to the Brightkite dataset.

Table 2: Average Trajectory Similarity Calculation Time (us)

Algorithm	Brightkite, mean length 76	Gowalla, mean length 48
OverallSimilarity	30.79	17.14
STLCSS	109.78	36.52
STLC	6,331.37	1,968.69

Correlation between matching point spatiotemporal distance and trajectory similarity The Spearman's rank correlation coefficients between matching point spatiotemporal distance and trajectory similarity under our trajectory similarity algorithm, OverallSimilarity, as well as STLCSS and STLC, on the Brightkite and Gowalla datastes, are presented in Tables 3.

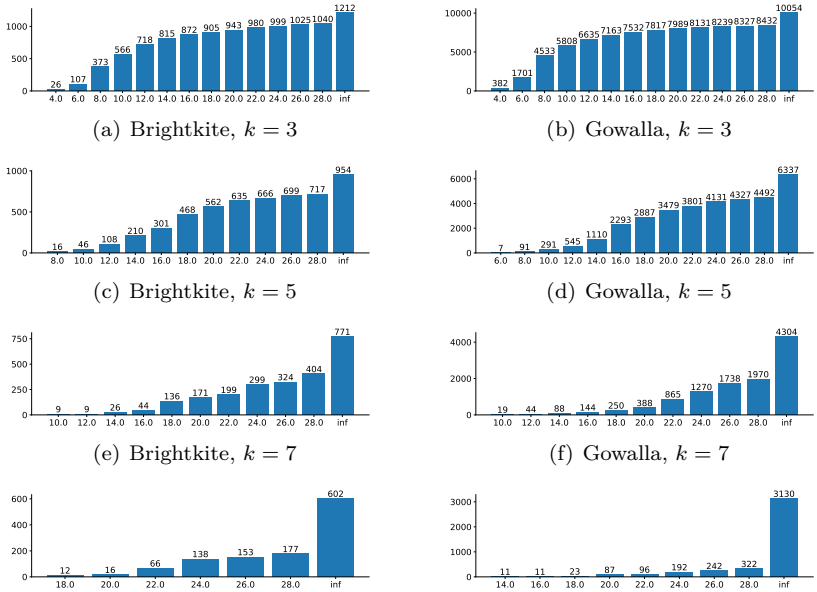
Compared with STLCSS and STLC, our trajectory similarity algorithm, OverallSimilarity, consistently achieves correlation coefficients between matching point spatiotemporal distance and trajectory similarity closer to -1. As a result, our trajectory similarity algorithm better encapsulates both spatial and temporal similarity within trajectory similarity.

7.4.2 Evaluation of Our Community Detection Algorithm

Size of the Detected Communities The number of users in the detected communities under different values of k and m is depicted in Figure 9.

Table 3: Correlations between Spatial and Temporal Distance of Matching Points and Trajectory Similarity

Algorithm	Brightkite	Gowalla
OverallSimilarity	-0.1986, -0.3871	-0.5112, -0.4084
STLCSS, k=1	0.1959, -0.1066	-0.1350, -0.1559
STLCSS, k=2	0.1550, -0.0926	-0.2162, -0.1112
STLCSS, k=3	0.0861, -0.0975	-0.1991, -0.0775
STLCSS, k=4	0.0550, -0.0927	-0.1800, -0.0632
STLCSS, k=5	0.0360, -0.0862	-0.1584, -0.0566
STLC, $\lambda = 0.1$	-0.0527, -0.1680	-0.4789, -0.1198
STLC, $\lambda = 0.2$	-0.0525, -0.1678	-0.4792, -0.1193
STLC, $\lambda = 0.3$	-0.0524, -0.1677	-0.4793, -0.1191
STLC, $\lambda = 0.4$	-0.0524, -0.1676	-0.4793, -0.1190
STLC, $\lambda = 0.5$	-0.0523, -0.1676	-0.4793, -0.1189
STLC, $\lambda = 0.6$	-0.0523, -0.1676	-0.4794, -0.1188
STLC, $\lambda = 0.7$	-0.0523, -0.1676	-0.4794, -0.1188
STLC, $\lambda = 0.8$	-0.0523, -0.1676	-0.4794, -0.1188
STLC, $\lambda = 0.9$	-0.0523, -0.1676	-0.4794, -0.1187



(a) Brightkite, $k = 3$ (b) Gowalla, $k = 3$

(c) Brightkite, $k = 5$ (d) Gowalla, $k = 5$

(e) Brightkite, $k = 7$ (f) Gowalla, $k = 7$

(g) Brightkite, $k = 9$ (h) Gowalla, $k = 9$

Figure 9: The Number of Users in the Detected Communities

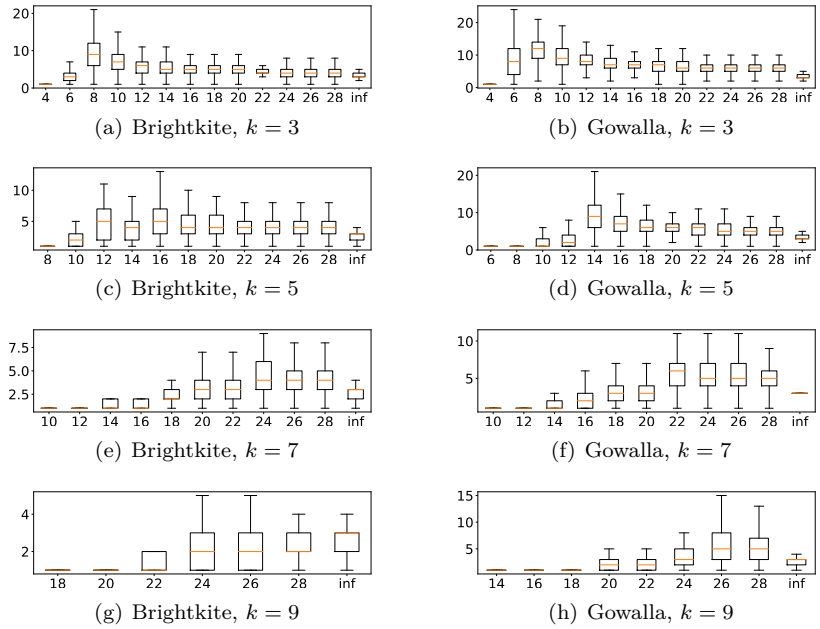
Under all values of k, our detected community only includes a fraction of the users when compared with the k-core. Thus, compared with k-core "seedbeds", our community detection algorithm can effectively find small subsets of users through the lens of trajectory similarity.

In our community detection algorithm, the set of edges selected for coreness calculation depends on the value of m, with more edges in the edge set with a larger value of m. Thus, given a value of k, the size of the detected community increases with m. Furthermore, given a value of m, the size of the community shrinks with an increasing value of k as a higher coreness requirement filters more users from the selected edge set.

Social Cohesiveness of the Detected Communities The distances between two users within each connected component of the detected communities under different values of k and m are depicted in Figure 10.

Given a value of k, there is a general trend for social cohesiveness measured by the median distance between two users within each connected component of the detected communities to first decrease (indicated by an increasing median distance) when $m < 3k$, before slowly reincreasing after $m \geq 3k$. This is because the vast majority of users in the social network have relatively few connections, while a few users have a huge number of connections. Increasing the value of m, more and more edges that function to tie the former into the selected edge set are retained, and as a result, they are more likely to be in the k-core of the edge set, which enlarges the community detected and causes the median distance to increase. However, when m becomes sufficiently large, many peripheral users have already been secured into the edge set, and the increase in m mainly leads to tighter interconnections within the edge set. As a result, the enlargement of the k-core slows, while the k-core becomes dense, leading to reducing median distances.

In addition, given a value of m, with the increase of k, not only does the detected community become smaller with fewer users within the selected edge set satisfying the coreness requirement, but the users that do satisfy the coreness requirement are also more closely knit. These two factors lead to a reduced median distance, which signifies an increased social cohesiveness.



(a) Brightkite, $k = 3$ (b) Gowalla, $k = 3$

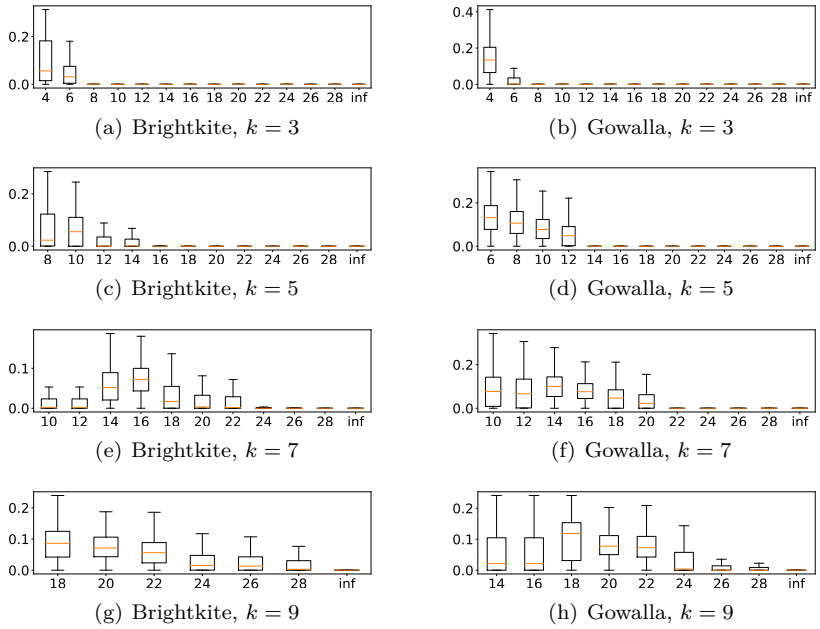
(c) Brightkite, $k = 5$ (d) Gowalla, $k = 5$

(e) Brightkite, $k = 7$ (f) Gowalla, $k = 7$

(g) Brightkite, $k = 9$ (h) Gowalla, $k = 9$

Figure 10: The Distances Between Two Users Within Each Connected Component of the Detected Communities Under Different Values of k and m

Spatial Cohesiveness of the Detected Communities The similarities between two users' trajectories within each connected component of the detected communities under different values of k and m are depicted in Figure 11.



(a) Brightkite, $k = 3$ (b) Gowalla, $k = 3$

(c) Brightkite, $k = 5$ (d) Gowalla, $k = 5$

(e) Brightkite, $k = 7$ (f) Gowalla, $k = 7$

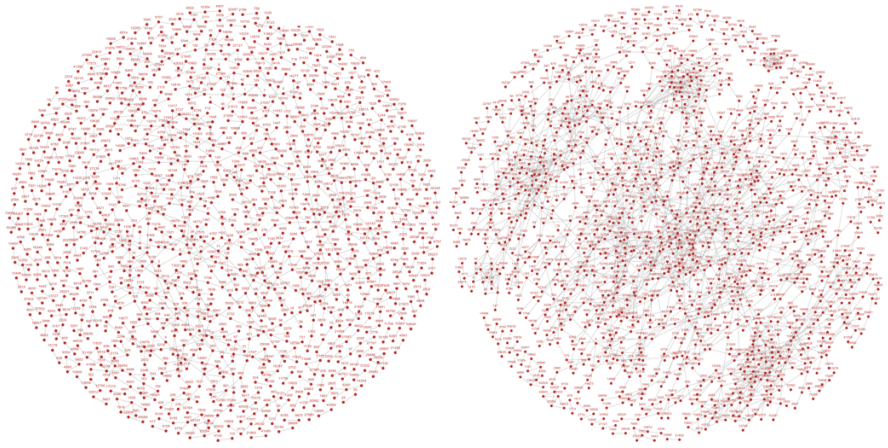
(g) Brightkite, $k = 9$ (h) Gowalla, $k = 9$

Figure 11: The Similarities Between Two Users' Trajectories Within Each Connected Component of the Detected Communities Under Different Values of k and m

Given a value of k, we can observe that the median similarities between two users' trajectories increases to a maxima around $m = 1.75k$, before decreasing to insignificant around $m = 3k$.

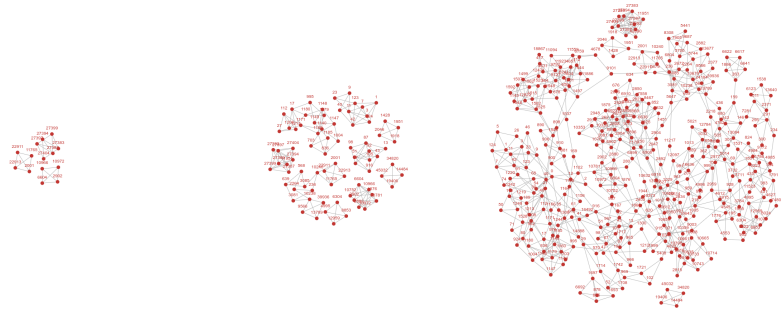
Given a value of k, under low values of m ($m < 1.75k$), only edges corresponding to high localized trajectory similarity are retained in the selected edge set. Although the resulting edge set is large, containing almost all users in the social network, it is sparse, with only a few densely connected regions resulting from a high mutual degree of trajectory similarity, as depicted on the left in Figure 12. However, not all users within these regions make it to the detected community, as some users may not satisfy coreness requirements. When the value of m is slightly raised, more users from the aforementioned dense regions would appear in the detected community owning to newly added edges in the edge set. Although these edges correspond to slightly lower trajectory similarity values, the high degree of interconnectivity within a connected component (which results from it coming from a densely connected region in the selected edge set) still guarantees spatial cohesiveness among its users, as depicted in the middle in Figure 13. Furthermore, the increase in the size and number of such connected components would mean that there would be more pairwise trajectory similarities taken into consideration. Such an increase would allow the median pairwise trajectory similarity to increase.

However, when the value of m is further raised ($m > 1.75k$), edges newly added to the edge set would not only enlarge and enhance original dense regions but also bridge them, resulting in larger connected components that are not cohesive in terms of trajectory similarity, as depicted on the right in Figure 13. At this point, the abundance of trajectory pairs with low similarity would quickly drag down the median pairwise trajectory similarity, resulting in what we see in Figure 11.



(a) $m = 4$ (b) $m = 8$

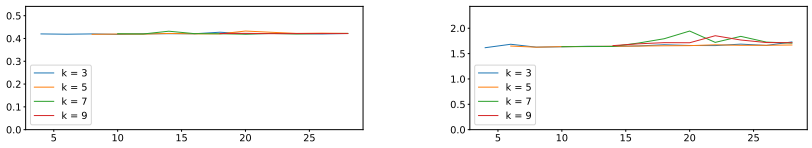
Figure 12: Filtered Edge Sets of the Brightkite Dataset



(a) $m = 4$ (b) $m = 6$ (c) $m = 8$

Figure 13: Communities Detected from the Brightkite Dataset, $k = 3$

Runtimes of Our Community Detection Algorithm The runtimes of our community detection algorithm under different values of k and m are plotted in Figure 14.



(a) Brightkite (b) Gowalla

Figure 14: Runtimes (in seconds) of Our Community Detection Algorithm

The effects of k and m are limited on the runtimes of our community detection algorithm. This is consistent with our analyzed time complexity of $O(|V| + (L + 2\log m)|E|)$, where k is not a factor and the logarithm of m is usually negligible when compared with $|V|$, $|E|$, and L.

Furthermore, the Brightkite dataset boasts $|V_1| = 1849, |E_1| = 13065, L_1 = 76$, while the Gowalla dataset boasts $|V_2| = 18737, |E_2| = 86985, L_2 = 48$. $\frac{|V_1|+L_1|E_1|}{|V_2|+L_2|E_2|}$ is close to the ratio of the median community detection time on the Brightkite and Gowalla datasets, $\frac{0.42}{1.75}$, which verifies our analyzed time complexity.

Being linear in terms of each of $|V|$, $|E|$, and L, the time complexity of our community detection algorithm, $O(|V| + (L + 2\log m)|E|)$, allows our algorithm to scale efficiently to large graphs and long trajectories.

Summary From the aforementioned experiment results, we can summarize the following rules of thumb for using our community detection algorithm: given a value of k, we can detect small, close-knit communities with both social cohesiveness and mutual trajectory similarity when $m < 3k$, with the median similarities between two users' trajectories peaking at around m = 1.75k. Larger values of m increase the community size but reduce social cohesiveness and trajectory similarity.

8 Conclusions

In this paper, we propose a community detection algorithm utilizing social relations and trajectories consisting of user check-in points to discover communities within social networks with both social cohesiveness and mutual trajectory similarity. To efficiently accomplish this goal, we propose an algorithm that can calculate the spatiotemporal similarity between two discrete trajectories in linear time and an efficient community detection algorithm that selects an edge set based on trajectory similarity before running k-core detection. Evaluating our trajectory similarity algorithm and community detection algorithm on two real-world datasets, we conclude that our trajectory similarity algorithm is both efficient and effective when compared with other discrete spatiotemporal trajectory similarity algorithms, and we uncover how and why each parameter affects the results of community detection. In the future, we plan to further

improve the efficiency of our community detection algorithm, as well as study the problem of community search utilizing social relations and trajectories.

References

[1] J Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. Large scale networks fingerprinting and visualization using the k-core decomposition. *Advances in neural information processing systems*, 18, 2005.

[2] José Ignacio Alvarez-Hamelin, Luca Dall'Asta, Alain Barrat, and Alessandro Vespignani. K-core decomposition of internet graphs: hierarchies, self-similarity and measurement biases. *arXiv preprint cs/0511007*, 2005.

[3] Gary D Bader and Christopher WV Hogue. An automated method for finding molecular complexes in large protein interaction networks. *BMC bioinformatics*, 4(1):1–27, 2003.

[4] Mathieu Bastian, Sebastien Heymann, and Mathieu Jacomy. Gephi: an open source software for exploring and manipulating networks. In *Proceedings of the international AAAI conference on web and social media*, volume 3, pages 361–362, 2009.

[5] Vladimir Batagelj and Andrej Mrvar. Pajek—analysis and visualization of large networks. In *Graph drawing software*, pages 77–103. Springer, 2004.

[6] Vladimir Batagelj, Andrej Mrvar, and Matjaž Zaveršnik. Partitioning approach to visualization of large graphs. In *International Symposium on Graph Drawing*, pages 90–97. Springer, 1999.

[7] Vladimir Batagelj and Matjaz Zaversnik. An o (m) algorithm for cores decomposition of networks. *arXiv preprint cs/0310049*, 2003.

[8] Michael Baur, Marco Gaertler, Robert Görke, Marcus Krug, and Dorothea Wagner. Generating graphs with predefined k-core structure. In *Proceedings of the European Conference of Complex Systems*, volume 1166, 2007.

[9] Donald J Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, volume 10, pages 359–370. Seattle, WA, USA:, 1994.

[10] Lilian Berton and Alneu de Andrade Lopes. Graph construction for semi-supervised learning. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[11] Lilian Berton, Alneu de Andrade Lopes, and Didier A Vega-Oliveros. A comparison of graph construction methods for semi-supervised learning. In *2018 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2018.

[12] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of statistical mechanics: theory and experiment*, 2008(10):P10008, 2008.

[13] Phil Brown and Junlan Feng. Measuring user influence on twitter using modified k-shell decomposition. In *Proceedings of the International AAAI Conference on Web and Social Media*, volume 5, pages 18–23, 2011.

[14] Shai Carmi, Shlomo Havlin, Scott Kirkpatrick, Yuval Shavitt, and Eran Shir. A model of internet topology using k-shell decomposition. *Proceedings of the National Academy of Sciences*, 104(27):11150–11154, 2007.

[15] Jinyang Chen, Rangding Wang, Liangxu Liu, and Jiatao Song. Clustering of trajectories based on hausdorff distance. In *2011 International Conference on Electronics, Communications and Control (ICECC)*, pages 1940–1944. IEEE, 2011.

[16] Lei Chen, M Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 491–502, 2005.

[17] Yu Chen, Jun Xu, and Minzheng Xu. Finding community structure in spatially constrained complex networks. *International Journal of Geographical Information Science*, 29(6):889–911, 2015.

[18] Eunjoon Cho, Seth A Myers, and Jure Leskovec. Friendship and mobility: user movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1082–1090, 2011.

[19] Jonathan Cohen. Trusses: Cohesive subgraphs for social network analysis. *National security agency technical report*, 16(3.1), 2008.

[20] Marius Eidsaa and Eivind Almaas. S-core network decomposition: A generalization of k-core analysis to weighted networks. *Physical Review E*, 88(6):062819, 2013.

[21] Thomas Eiter and Heikki Mannila. Computing discrete fréchet distance. Technical report, Citeseer, 1994.

[22] Arnold I Emerson, Simeon Andrews, Ikhlak Ahmed, Thasni KA Azis, and Joel A Malek. K-core decomposition of a protein domain co-occurrence network reveals lower cancer mutation rates for interior cores. *Journal of clinical bioinformatics*, 5(1):1–11, 2015.

[23] Paul Expert, Tim S Evans, Vincent D Blondel, and Renaud Lambiotte. Uncovering space-independent communities in spatial networks. *Proceedings of the National Academy of Sciences*, 108(19):7663–7668, 2011.

[24] Edoardo Galimberti, Francesco Bonchi, and Francesco Gullo. Core decomposition and densest subgraph in multilayer networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1807–1816, 2017.

[25] Thomas D Gauthier. Detecting trends using spearman's rank correlation coefficient. *Environmental forensics*, 2(4):359–362, 2001.

[26] Christos Giatsidis, Dimitrios M Thilikos, and Michalis Vazirgiannis. D-cores: measuring collaboration of directed graphs based on degeneracy. *Knowledge and information systems*, 35(2):311–343, 2013.

[27] Michelle Girvan and Mark EJ Newman. Community structure in social and biological networks. *Proceedings of the national academy of sciences*, 99(12):7821–7826, 2002.

[28] Patric Hagmann, Leila Cammoun, Xavier Gigandet, Reto Meuli, Christopher J Honey, Van J Wedeen, and Olaf Sporns. Mapping the structural core of human cerebral cortex. *PLoS biology*, 6(7):e159, 2008.

[29] Laurent Hébert-Dufresne, Antoine Allard, Jean-Gabriel Young, and Louis J Dubé. Percolation on random networks with arbitrary k-core structure. *Physical Review E*, 88(6):062820, 2013.

[30] Arnold Emerson Isaac and Sitabhra Sinha. Analysis of core–periphery organization in protein contact networks reveals groups of structurally and functionally critical residues. *Journal of biosciences*, 40(4):683–699, 2015.

[31] Junghoon Kim, Tao Guo, Kaiyu Feng, Gao Cong, Arijit Khan, and Farhana M Choudhury. Densely connected user community and location cluster search in location-based social networks. In *Proceedings of the 2020 ACM SIGMOD international conference on management of data*, pages 2199–2209, 2020.

[32] Nir Lahav, Baruch Ksherim, Eti Ben-Simon, Adi Maron-Katz, Reuven Cohen, and Shlomo Havlin. K-shell decomposition reveals hierarchical cortical organization of the human brain. *New Journal of Physics*, 18(8):083013, 2016.

[33] Jure Leskovec and Eric Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th international conference on World Wide Web*, pages 915–924, 2008.

[34] Qiyan Li, Yuanyuan Zhu, and Jeffrey Xu Yu. Skyline cohesive group queries in large road-social networks. In *2020 IEEE 36th International Conference on Data Engineering (ICDE)*, pages 397–408. IEEE, 2020.

[35] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S Jensen, and Wei Wei. Deep representation learning for trajectory similarity computation. In *2018 IEEE 34th international conference on data engineering (ICDE)*, pages 617–628. IEEE, 2018.

[36] Jian-Hong Lin, Qiang Guo, Wen-Zhao Dong, Li-Ying Tang, and Jian-Guo Liu. Identifying the node spreading influence with largest k-core values. *Physics Letters A*, 378(45):3279–3284, 2014.

[37] Caihong Liu and Chonghui Guo. Stccd: Semantic trajectory clustering based on community detection in networks. *Expert Systems with Applications*, 162:113689, 2020.

[38] Linyuan Lü, Duanbing Chen, Xiao-Long Ren, Qian-Ming Zhang, Yi-Cheng Zhang, and Tao Zhou. Vital nodes identification in complex networks. *Physics Reports*, 650:1–63, 2016.

[39] Linyuan Lü, Tao Zhou, Qian-Ming Zhang, and H Eugene Stanley. The h-index of a network node and its relation to degree and coreness. *Nature communications*, 7(1):1–7, 2016.

[40] R Duncan Luce and Albert D Perry. A method of matrix analysis of group structure. *Psychometrika*, 14(2):95–116, 1949.

[41] Feng Luo, Bo Li, Xiu-Feng Wan, and Richard H Scheuermann. Core and periphery structures in protein interaction networks. In *BMC bioinformatics*, volume 10, pages 1–11. BioMed Central, 2009.

[42] Fragkiskos D Malliaros, Christos Giatsidis, Apostolos N Papadopoulos, and Michalis Vazirgiannis. The core decomposition of networks: Theory, algorithms and applications. *The VLDB Journal*, 29(1):61–92, 2020.

[43] Polykarpos Meladianos, Antoine Tixier, Ioannis Nikolentzos, and Michalis Vazirgiannis. Real-time keyword extraction from conversations. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 462–467, 2017.

[44] Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006.

[45] Mark EJ Newman and Michelle Girvan. Finding and evaluating community structure in networks. *Physical review E*, 69(2):026113, 2004.

[46] Giannis Nikolentzos, Polykarpos Meladianos, Stratis Limnios, and Michalis Vazirgiannis. A degeneracy framework for graph similarity. In *IJCAI*, pages 2595–2601, 2018.

[47] Sen Pei, Lev Muchnik, José S Andrade Jr, Zhiming Zheng, and Hernán A Makse. Searching for superspreaders of information in real-world social media. *Scientific reports*, 4(1):1–12, 2014.

[48] Nikos Pelekis, Ioannis Kopanakis, Gerasimos Marketos, Irene Ntoutsi, Gennady Andrienko, and Yannis Theodoridis. Similarity search in trajectory databases. In *14th International Symposium on Temporal Representation and Reasoning (TIME'07)*, pages 129–140. IEEE, 2007.

[49] Mark T Robinson. The temporal development of collision cascades in the binary-collision approximation. *Nuclear Instruments and Methods in Physics Research Section B: Beam Interactions with Materials and Atoms*, 48(1-4):408–413, 1990.

[50] François Rousseau and Michalis Vazirgiannis. Main core retention on graph-of-words for single-document keyword extraction. In *European Conference on Information Retrieval*, pages 382–393. Springer, 2015.

[51] Soumya Sarkar, Aditya Bhagwat, and Animesh Mukherjee. Core2vec: A core-preserving feature learning framework for networks. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 487–490. IEEE, 2018.

[52] Stephen B Seidman. Network structure and minimum degree. *Social networks*, 5(3):269–287, 1983.

[53] Shuo Shang, Lisi Chen, Zhewei Wei, Christian Søndergaard Jensen, Kai Zheng, and Panos Kalnis. Trajectory similarity join in spatial networks. *Proceedings of the VLDB Endowment*, 10(11), 2017.

[54] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. Corescope: Graph mining using k-core analysis—patterns, anomalies and algorithms. In *2016 IEEE 16th international conference on data mining (ICDM)*, pages 469–478. IEEE, 2016.

[55] Kijung Shin, Tina Eliassi-Rad, and Christos Faloutsos. Patterns and anomalies in k-cores of real-world graphs with applications. *Knowledge and Information Systems*, 54(3):677–710, 2018.

[56] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. A survey of trajectory distance measures and performance evaluation. *The VLDB Journal*, 29(1):3–32, 2020.

[57] Antoine Tixier, Fragkiskos Malliaros, and Michalis Vazirgiannis. A graph degeneracy-based approach to keyword extraction. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1860–1870, 2016.

[58] Johan Ugander, Brian Karrer, Lars Backstrom, and Cameron Marlow. The anatomy of the facebook social graph. *arXiv preprint arXiv:1111.4503*, 2011.

[59] Martijn P Van Den Heuvel and Olaf Sporns. Rich-club organization of the human connectome. *Journal of Neuroscience*, 31(44):15775–15786, 2011.

[60] Trivik Verma, F Russmann, Nuno AM Araújo, Jan Nagler, and Hans J Herrmann. Emergence of core–peripheries in networks. *Nature communications*, 7(1):1–7, 2016.

[61] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Proceedings 18th international conference on data engineering*, pages 673–684. IEEE, 2002.

[62] Kai Wang, Xin Cao, Xuemin Lin, Wenjie Zhang, and Lu Qin. Efficient computing of radius-bounded k-cores. In *2018 IEEE 34th international conference on data engineering (ICDE)*, pages 233–244. IEEE, 2018.

[63] Stanley Wasserman, Katherine Faust, et al. Social network analysis: Methods and applications. 1994.

[64] Cynthia I Wood and Illya V Hicks. The minimal k-core problem for modeling k-assemblies. *The Journal of Mathematical Neuroscience (JMN)*, 5(1):1–19, 2015.

[65] Stefan Wuchty and Eivind Almaas. Peeling the yeast protein network. *Proteomics*, 5(2):444–449, 2005.

[66] Kai Yao, Dimitris Papadias, and Spiridon Bakiras. Density-based community detection in geo-social networks. In *Proceedings of the 16th international symposium on spatial and temporal databases*, pages 110–119, 2019.

[67] Yihong Yuan and Martin Raubal. Measuring similarity of mobile phone user trajectories–a spatio-temporal edit distance method. *International Journal of Geographical Information Science*, 28(3):496–520, 2014.

[68] Fan Zhang, Ying Zhang, Lu Qin, Wenjie Zhang, and Xuemin Lin. When engagement meets similarity: efficient (k, r)-core computation on social networks. *arXiv preprint arXiv:1611.03254*, 2016.

[69] Qijun Zhu, Haibo Hu, Cheng Xu, Jianliang Xu, and Wang-Chien Lee. Geo-social group queries with minimum acquaintance constraints. *The VLDB Journal*, 26(5):709–727, 2017.

[70] Ren Zhuo-Ming, Liu Jian-Guo, Shao Feng, Hu Zhao-Long, and Guo Qiang. Analysis of the spreading influence of the nodes with minimum k-shell value in complex networks. *Acta Physica Sinica*, 62(10), 2013.