# Trajectory-matching Prediction for Friend Recommendation in Anonymous Social Networks

Yichun Duan[1], Yuanxing Zhang[1], Chengliang Gao[1], Meng Tong[1], Yuanyuan Zhang[2], Kaigui Bian[1], Wei Yan[1,3]
[1]School of EECS, Peking University, Beijing, China
[2]Ledongli Co. LTD, Beijing, China
[3]Institute of Big Data Technologies, School of SECE, Peking University, Shenzhen, China
Email: {duanyichun, longo, gaochengliang, tongmeng, bkg, w}@pku.edu.cn

***Abstract*—People connect to each other over conventional online social networks (OSNs) based on many parameters (common interests, experiences, locations), while the anonymous social networks (ASNs) recommend candidate friends to a user mainly by the location proximity at a coarse-granularity. In this paper, we formulate a fine-grained trajectory-matching prediction problem for friend recommendation in ASNs—what is the likelihood for two users to encounter with each other in the future based on their historical trajectory data? We define the *serendipity* of two trajectories in both spatial and temporal domains to quantify the similarity between two users' trajectories, and propose an algorithm that recommends candidate friends to a user by determining the similarity of their trajectories. Experiments show that our proposed algorithm can predict the encounter of users quite accurately and it outperforms the conventional algorithms in terms of the precision and consumed time.**

## I. Introduction

Online social relation is the key of online social networks (OSNs). People connect to each other as friends in conventional OSNs based on many parameters, including their common interests, shared experiences, location proximity, or their social relationship (families, colleagues) in the real world [1], [2]. Specifically, the OSNs suggest candidate friends to a user by looking at the common keywords in their profiles, e.g., whether they have the same college, work experience, common friends [1], or whether they have visited similar PoIs (points of interests) or similar locations [3].

Anonymous social networking (ASN) apps on smartphones become popular in recent years as it facilitates the offline connection (e.g., dating) among people who are strangers but located close to each other—location proximity is the main factor when considering the friend recommendation in ASNs. ASN apps borrow the location based services (LBS) for OSNs that only capture the coarse-grained locations of a user: (1) the iOS or Android platforms may shut down a service if it is detected to continuously monitor the user's location information; (2) many OSN services require a user to explicitly "check-in" at certain PoIs to showcase their locations (Facebook check-in, or adding location to a new tweet) [4], which fails to continuously follow one's mobile trajectory.

To facilitate tracking one's fine-grained trajectory, iOS has released a unified solution of providing the location information to the app developers: it acts as a blackbox that outputs the phone's location by employing a hybrid localization approach based on GPS, WiFi, cellular signals, etc; a new location record will be captured whenever the phone moves [5]. This systematic support on iOS devices greatly relieves the efforts of app developers for building their own localization techniques with high precision requirements.

Owing to the iOS support, many apps started adding anonymous social networking function (building the location-based friend recommendation system) for promoting the number of daily active users (DAU). For example, ASN apps (e.g., Tinder [6]) can leverage the users' location information to introduce two people that have similar trajectories to start a conversation directly. This leads to an interesting yet challenging problem: is that possible for two users with similar trajectories to encounter in the future?

In this paper, we formulate the preceding problem as a fine-grained trajectory-matching prediction problem for friend recommendation in ASNs—what is the likelihood for two users to encounter with each other in the future based on their historical trajectory data? By converting a trajectory to a series of discrete geographical location points, we define the *serendipity* of two trajectories in temporal and spatial domains to quantify the similarity between two users' mobility patterns, and formulate a serendipity maximization problem that seeks to maximize the expected serendipity value between a user and others that may have a similar trajectory.

We then present an algorithm to solve the problem: we first calculate the ESI (expectation of serendipity index) for determining the similarity of location points in two trajectories by simultaneously considering temporal and spatial domains; then by using time-slicing technique, we propose the TS-ESI (time-sliced ESI) algorithm that recommends the candidate friends to a given user by looking at their time-domain similarity first and then at the space-domain. We collect the real-world user trajectories from a social sports app "Ledongli" in China (Ledongli is the best iOS app 2013 on App Store in China)[1], and the dataset involves 3,000 active users for a six-day time period in July 2016. Experimental results show that our TS-ESI algorithm can predict the probability that two users with similar historical trajectories may encounter

[1]Social Sports apps seek to quantify people's health by capturing users' motion data, provide them feedbacks on their sports and health conditions, and help users with similar behavior patterns to create offline activities, like running, yoga for weight loss. http://www.ledongli.cn/

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

with each other, and performs better than existing approches. By solving this problem, we can design a trajectory-matching based recommendation strategy for ASNs.

The rest of this paper is organized as follows: We briefly review the related work in Section II. We formulate the friend recommendation problem in ASNs as a serendipity maximization problem in Section III, and we propose the algorithm that calculates the TS-ESI for addressing the formulated problem in Section IV. We evaluate the performance by comparing it with existing approaches in Section V. Finally, we conclude our work in Section VI and give our acknowledgement in Section VII.

## II. Related Work

### A. Friend Recommendation in OSNs

Friend recommendation is one of the most popular research topic in conventional OSNs. Various strategies have been proposed to recommend candidate friends for a user based on the knowledge implied in the network structure. Typically, there have been four categories of strategies:

- **Graph-based** strategies are based on the analysis of the network graph formed by the user's friends and friends-of-friends (FoF) relationships [7];
- **Context-based** strategies are determined by users' physical or social context [8];
- **Location-based** strategies recommend friends that attach to similar Points of Interest (PoIs) or locations [9];
- **Semantic-based** strategies are appeared in recent years, which collect data from mutiple sensors and resources, and then infer the group of candidate friends that have similar behaviors in their daily life (visited locations, text messaging behaviors, etc)[10].

### B. Friend Recommendation in ASNs

In ASNs, the concepts of identifiable identities and strong social ties among users are missing. Many ASN apps such as Whisper simply do the recommendations by a random-choosing strategy, in which case users tend to interact with strangers instead of anyone that may share common interests, leading to weak social ties and challenges in the potential of engagement [11]. Other ASN apps such as Tinder recommend nearby persons who have the same friends in Facebook to users, which partly relys on conventional OSNs[6]. In contrary to these methods, we seek to design a trajectory-matching based recommendation strategy for ASNs.

### C. Trajectory Similarity

There are many approaches for measuring the similarity between two trajectories. Yi et al. propose the *Dynamic Time Wrapping* (DTW) method which reflects the distance between two time-sequences by DTW distance [12]. Vlachos et al. develop the concept of *Longest Commons Subsequence* (LCSS) scores between two trajectories [13]. Later, Chen et al. adopt the *Edit Distance on Real Sequence* (EDR) which is more concise than LCSS while keeping similar performance [14].

However, all these methods fail to consider the time-domain similarity for two trajectories. In our trajectory-matching based recommendation, we predict the likelihood that two users will encounter in the future based on their historical data of past trajectories. That is, our methods evaluate the similarity between trajectories both on spatial and temporal dimensions.

## III. Problem Formulation

Let $\mathcal{A} = (\mathcal{U}, \mathcal{D}, \mathcal{G})$ represents an ASN, where $\mathcal{U}$ represents the user set of size $n$, i.e. $\mathcal{U} = \{u_i\}_{i=1}^{n}$; $\mathcal{D}$ denotes the set of $m$ days being recorded, i.e. $\mathcal{D} = \{d_j\}_{j=1}^{m}$; and $\mathcal{G}$ represents the set of trajectories of all users. Specifically, $\mathcal{G} = \{G_i^{\mathcal{D}}\}_{i=1}^{n}$, where $G_i^{\mathcal{D}}$ marks all the trajectories of user $u_i$, whose trajectory on a specific day $d_j$ is denoted as $G_i^j(t)$. A single trajectory is defined as a location-time function. For every time-stamp $t_k$ in $d_j$, $G_i^j(t_k) = \vec{s_k}$, where $\vec{s_k}$ is the user's location in $t_k$, and we define $p_k = \langle \vec{s_k}, t_k \rangle$ as a space-time pair. Besides, we are only interested in the user-active time period during which a user is moving. Thus, we set $T(G_i^j)$ to represent the user-active time period of trajectory $G_i^j(\cdot)$.

**Definition 1.** *For two space-time pairs $p_k$ and $p_{k'}$, if $|\vec{s_k}, \vec{s_{k'}}| < \epsilon$ and $|t_k - t_{k'}| < \delta$, then $< p_k, p_{k'} >$ is called as a pair of **time-space cross point**, where $\epsilon$ and $\delta$ are space and time restriction parameters, respectively. If there exists $p_k = \langle G_i^j(t_k), t_k \rangle$ and $p_{k'} = \langle G_{i'}^j(t_{k'}), t_{k'} \rangle$, where $< p_k, p_{k'} >$ becomes a pair of time-space cross points, we define user $u_i$ and $u_{i'}$ encounter on the day $d_j$.*

Here, we use the spherical distance as the distance between two points because the location of space-time pairs are recorded as longitude and latitude coordinates. According to the definition, we can write an indicator function to describe the encounter of two users:

$$E_e(u_i, u_{i'}, d_j) = \begin{cases} 1 & u_i \text{ and } u_{i'} \text{ encounter on day } d_j, \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

Then we define the *serendipity* of two trajectories given the above definition of time-space cross points.

**Definition 2.** *For a specific day $d_j$, if there exist two start times $t_i^j$ and $t_{i'}^j$ for $G_i^j(\cdot)$ and $G_{i'}^j(\cdot)$ satisfying $|t_i^j - t_{i'}^j| < \delta$ and for any moment $t \in [0, T_p], T_p > 0$, we can ensure that $\langle G_i^j(t_i^j + t), t_i^j + t \rangle$ and $\langle G_{i'}^j(t_{i'}^j + t), t_{i'}^j + t \rangle$ become a pair of time-space cross points, then:*

- *$\langle G_i^j(\cdot), G_{i'}^j(\cdot) \rangle$ becomes a pair of serendipity trajectories;*
- *$T_p$ is called **cross intervals**, and the **serendipity time** is defined as the maximum of $T_p$, denoted as $ST(\cdot, \cdot)$:*

$$ST(G_i^j(\cdot), G_{i'}^j(\cdot)) = \max_{T_p}\{T_p\}$$

- *The **serendipity index** of $\langle G_i^j(\cdot), G_{i'}^j(\cdot) \rangle$ is*

$$SI(G_i^j(\cdot), G_{i'}^j(\cdot)) = \frac{ST(G_i^j(\cdot), G_{i'}^j(\cdot))}{|T(G_i^j(\cdot)) \bigcup T(G_{i'}^j(\cdot))|} \quad (2)$$

Figure 1 illustrates an example of a pair of serendipity trajectories, where the red dash circle marks the time-space

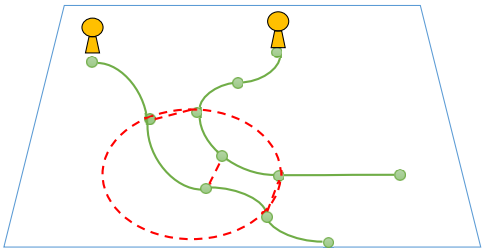

Fig. 1: Given two users' trajectories, we define the time-space cross points and time-space cross intervals.

cross intervals. According to (2), if the serendipity index value $SI(G_i^j(t), G_{i'}^j(t))$ is large, then two users $u_i$ and $u_{i'}$ may stay or move close to each other on day $d_j$ with a high probability. That is, they may have passed by each other for many times without a chance to make friends, and in this case, it is better to recommend them to know each other over the social network platform if the serendipity index tells that they may meet on the next day evidently.

Since two users may have different mobility patterns on different days, the event of encountering each other on the next day cannot be 100% guaranteed. Hence, we want to predict the probability of this event based on users' historical trajectory data. Define $ESI(u_i, u_{i'}, d_j)$ as the expectation of the serendipity index (ESI) of $\langle G_i^j(t), G_{i'}^j(t)\rangle$, then we have:

$$\begin{aligned} ESI(u_i, u_{i'}, d_j) = \int\int & SI(G_i^j(t), G_{i'}^j(t)) \\ & \cdot P(G_i^j | G_i^1, G_i^2, \ldots, G_i^{j-1}) \\ & \cdot P(G_{i'}^j | G_{i'}^1, G_{i'}^2, \ldots, G_{i'}^{j-1}), \end{aligned} \tag{3}$$

where

$$\begin{aligned} P(G_i^j | G_i^1, G_i^2, \ldots, G_i^{j-1}) = \\ \int_{t \in G_i} P(\langle G_i^j(t), t\rangle | G_i^1, \ldots, G_i^{j-1}). \end{aligned} \tag{4}$$

According to (3) and (4), we can calculate the serendipity index of $u_i$ and $u_{i'}$ using the users' historical trajectory data, which gives us evidence to recommend potential friends for any user. Therefore, our goal is to list several candidate friends in the following day for a specific user where the group of candidate friends would have the greatest total ESI value with the given user. Then, we define the following *serendipity maximization* problem for friend recommendation.

**Problem 1.** *For an ASN $\mathcal{A} = (\mathcal{U}, \mathcal{D}, \mathcal{G}_{\mathcal{U}}^{\mathcal{D}})$, given space and time restriction parameters $\epsilon$, $\delta$ and an expected number of candidate friends $K$, for any user $u_i \in \mathcal{U}$, find a subset $U_i \subset \mathcal{U}$ of size $K$ with the greatest total expectation of serendipity index on day $d$, i.e.*

$$\begin{aligned} &\textit{Maximize} \quad \sum_{u_{i'} \in U_i} ESI(u_i, u_{i'}, d) \\ &\textit{subject to} \quad U_i \subset \mathcal{U}, |U_i| = K. \end{aligned}$$

## IV. Algorithms for Serendipity Maximization

In the real world, a user's trajectory during a day is not a continuous curve, but a series of location points, known as $\hat{G}_i^j(\hat{t}), \hat{t} = \hat{t}_1, \hat{t}_2, ..., \hat{t}_l$, where $l$ is the length of the series. We re-denote it as a vector:

$$\hat{G}_i^j = (\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_l)$$

The distance and time interval between two adjacent space-time pairs are different, and we assume that the trajectory between each two adjacent space-time pair is approximated to be a straight segment.

### A. Interpolation to a Trajectory

The points in the original series are not distributed uniformly, and one possible solution is to do sampling on the entire trajectory. As the trajectory can be divided into several segments, we can use interpolation to get the series of points where we can draw a probablity distribution. The space-time cross point $p_x =< \vec{s_x}, t_x >$ where $\exists u \in \mathcal{U}$, $t_x \in [t_u, t_{u+1})$ can then be calculated by

$$\vec{s_x} - \vec{s_v} = \frac{t_x - t_v}{t_{v+1} - t_v} \cdot (\vec{s_{v+1}} - \vec{s_v}).$$

The algorithm is shown in Algorithm 1.

**Algorithm 1** Interpolation for a given trajectory of discrete location points.

**Function**: Interpolation()
**Input**: $\hat{G}_i^j$, $\delta$
**Output**: $\widetilde{G}_i^j$

```
p_0 ← p̂_1
for k ← 2 to l do
    Δp ← p̂_k − p_0
    if Δp.t < (δ/2) then
        continue
    end if
    frac ← ⌈Δp.t/(δ/2)⌉
    for w ← 1 to frac do
        p̃ ← (p̂_k − p̂_{k−1}) · w/frac
        insert p̃ to G̃_i^j
    end for
    p_0 ← p̃
end for
```

### B. Critical Points in A Trajectory

With the interpolation, the trajectory is converted to a set of space-time points, named as *trace-set* and denoted as $\widetilde{G}_i^j$. Then we could rewrite Equation 3 in the following version:

$$\begin{aligned} ESI(u_i, u_{i'}, d_j) = \sum_{\widetilde{G}_i^j} \sum_{\widetilde{G}_{i'}^j} & SI(\widetilde{G}_i^j(t), \widetilde{G}_{i'}^j(t)) \\ & \cdot P(\widetilde{G}_i^j | \widetilde{G}_i^1, \widetilde{G}_i^2, \ldots, \widetilde{G}_i^{j-1}) \\ & \cdot P(\widetilde{G}_{i'}^j | \widetilde{G}_{i'}^1, \widetilde{G}_{i'}^2, \ldots, \widetilde{G}_{i'}^{j-1}). \end{aligned} \tag{5}$$

Given two trace-sets $\widetilde{G}_i^j$ and $\widetilde{G}_{i'}^j$ where $|\widetilde{G}_i^j| = l_i, |\widetilde{G}_{i'}^j| = l_{i'}$, according to (2), the time complexity to calculate $SI(\widetilde{G}_i^j(t), \widetilde{G}_{i'}^j(t))$ is at least $O(l_i \cdot l_{i'})$. Considering that we have to sample the possible trajectories, the time complexity to compute $ESI(\cdot,\cdot,\cdot)$ can be quite enormous, especially in reality where the scale of data may be fairly large.

Since the probability that two users will encounter with each other is not distributed uniformly in the whole geographic space, we can focus on those points with the maximal probability, named as *critical points*.

**Definition 3.** *A space-time pair $p_x =< \vec{s_x}, t_x >$ is a **possible point** of $u_i$ if $\exists j$ that contains $p_x \in \widetilde{G}_i^j$ . And for $u_i$, we call $p_x$ as a **critical point** if $\exists i' \in [1, n]$ and $i \neq i'$, there lies $p_{x'}$ that is a possible point of $u_{i'}$ while $\langle p_x, p_{x'} \rangle$ is a pair of time-space cross point. Let $C_i$ denotes the set of all critical points of $u_i$.*

Critical points help significantly reduce the amount of candidate space-time pairs. Implementing it in Equation 5, the time complexity is $O(|C|^2)$. And considering we can use the time dimension information to do dynamic pruning, the complexity is lower in practical use. Therefore, the solution to Problem 1 can promise to finish for at most $O(n(|C|^2 n + n \log K))$.

### C. Time-sliced ESI Calculation

Since $|C_i|$ can be more than $10^5$ in reality, we propose the time-sliced algorithm to compute *the time-sliced ESI* (TS-ESI) to improve the time complexity. The algorithm requires us to set a time-sliced parameter $\lambda$, which is used to segment the trace-set into several small trajectory slices. During each time slice, we only evaluate whether the encounter occurs between users with similar trajectories. The procedure is listed in Algorithm 2, where $\tau$ represents the length of daytime.

There are some optimization strategies that can be applied in Algorithm 2. When trying to find the time-space cross-points, as shown in line 19, we can use binary-search to accelerate the procedure. Besides, pruning methods to avoid irrelevant points can also contribute to the algorithm's efficiency. By utilizing these two strategies, the time complexity of the $TS-ESI$ is lower than the origin $ESI$. Based on the above procedures, the complete solution to Problem 1 is shown as a top-$K$ friend recommendation method in Algorithm 3, which predicts a list of candidate friends that have similar trajectories with user $u_i \in \mathcal{U}$.

## V. Experimental Evaluation

In this section, we evaluate the performance of the proposed TS-ESI algorithm by examining whether the recommended friends will have a chance of encountering a given user in the future, over the real-world trajectory dataset of the social sports app.

### A. Dataset

The dataset used in our experiments is provided by a popular social sports app in China called Ledongli. Once users install this app on their mobile phones, the app will record their

**Algorithm 2** Calculate TS-ESI between two trace-sets.

**Function**: TSESI()
**Input**: $C_i$,$C_{i'}$, $\lambda$,$\tau$
**Output**: TS-ESI

1: $\chi \leftarrow \lceil \tau/\lambda \rceil$
2: $res \leftarrow 0$
3: **for** $k \leftarrow 1$ to $\chi$ **do**
4: initialize $times(k)$ as empty list
5: initialize $times'(k)$ as empty list
6: **end for**
7: **for** $p_x = \langle t_x, \vec{s}_x \rangle$ in $C_i$ **do**
8: insert $p_x$ into $times(\lceil t_x/\lambda \rceil)$
9: **end for**
10: **for** $p_{x'} = \langle t_{x'}, \vec{s}_{x'} \rangle$ in $C_{i'}$ **do**
11: insert $p_{x'}$ into $times(\lceil t_{x'}/\lambda \rceil)$
12: **end for**
13: **for** $k \leftarrow 1$ to $\chi$ **do**
14: sort $times(k)$ by location
15: sort $times'(k)$ by location
16: **end for**
17: **for** $k \leftarrow 1$ to $\chi$ **do**
18: **for** $p_t$ in $times(k)$ **do**
19: **if** exists $p_{t'} \in times'(k)$ that $\langle p_t, p_{t'} \rangle$ become time-space cross points **then**
20: $res \leftarrow res + 1$
21: **end if**
22: **end for**
23: **end for**
24: TS-ESI = $res/\tau$

**Algorithm 3** Top-$K$ friend recommendation for user $u_g$

**Function**: $\mathcal{A} = (\mathcal{U}, \mathcal{D}, \mathcal{G}_{\mathcal{U}}^{\mathcal{D}})$, $u_g$,K
**Output**: $U_g$

1: **for** $u_i$ in $\mathcal{U}$ **do**
2: initialize $C_i$ as empty set
3: **for** $d_j$ in $\mathcal{D}$ **do**
4: $\widetilde{G}_i^j = \text{Interpolation}(\hat{G}_i^j)$
5: $C_i \leftarrow C_i \bigcup \widetilde{G}_i^j$
6: **end for**
7: **end for**
8: Let $h$ be a minimum heap
9: **for** $u_i$ in $\mathcal{U}$ **do**
10: **if** i == g **then**
11: continue
12: **end if**
13: calculate $esi \leftarrow \text{TSESI}(C_i, C_g)$
14: **if** $esi > 0$ **then**
15: insert $(i, esi)$ into $h$
16: **if** $h.size > K$ **then**
17: pop the front element of $h$
18: **end if**
19: **end if**
20: **end for**
21: Fill $U_g$ with all elements in $h$

motion data every day, such as location, step count, etc. The app records motion data by sensors in smartphone, and if a user stays in the same place for a long time, the app will report the user's location regularly, which makes sure our interpolation algorithm do not introduce large bias. Our dataset includes the anonymized data of 3000 active and representative users' complete trajectories from July 20, 2016 to July 25, 2016 in Beijing, China.

- **Prediction step**: We use data in first five days to calculate TS-ESI and predict the list of friends to every user;
- **Evaluation step**: We use data in the last day as the test set (or the ground truth) to see whether the predicted list of friends to a given user will have a great chance of encountering the specific user, and how strong the serendipity index between them is.

### B. Metrics

Before analyzing results of the experiment, we introduce our metrics at first.

- **Precision and F1-score**: Precision is the fraction of retrieved instances that are relevant, and recall is the fraction of relevant instances that are retrieved, while F1-score considers both precision and recall, which is more convenient in comparison.
- **Mean Average Precision:** Mean average precision (mAP) is a metric in recommendation systems [15]. Average precision (AP) is defined as the average of the precisions of hits. In our case, if we give a list of recommend users with a size of $K$, and take $top-K$ person $u_i$ meet in $d_j$ as the relevant set, then the AP of a $top-K$ query will be:

$$AP = \frac{1}{K} \cdot \sum_{i=1}^{i=K} H(u_i) \frac{\sum_{w=1}^{w=i} \cdot H(u_i)}{i} \tag{6}$$

where $H(u_i) = 1$ when $u_i \in \mathcal{U}_{true\_positive}$, otherwise $H(u_i) = 0$. And mAP is mean AP for all the nodes,which is a rank-related metric.

### C. Main Results

In the experiment, we set $\delta = 1800$ second, and considering Beijing is a big city with $1.641 * 10^4$ square kilometers, where we set $\epsilon = 2$ kilometer. We compare the following three algorithms on our dataset:

- LCSS ALGORITHM. This is the proposed algorithm in [13] using LCSS which does not consider the time-domain similarity of two trajectories, which serves as the baseline algorithm in the experiment;
- ESI CALCULATION ALGORITHM. This algorithm calculates the ESI value without adding the time-slicing technique;
- TS-ESI CALCULATION ALGORITHM. The proposed algorithm uses time-slicing technique to accelerate the calculation of ESI.

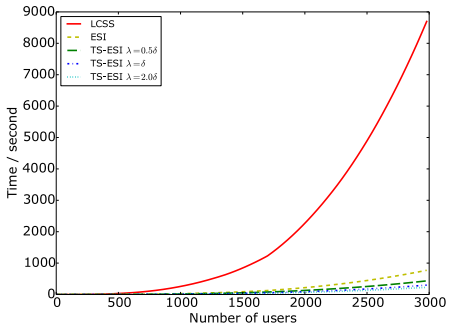


Fig. 2: The running time vs. the number of users.

*1) Runtime:* Figure 2 shows the runtime of $ESI$ and $TS-ESI(\lambda = 0.5\delta,\ \delta$ and $2\delta)$ when generating friend recommendation lists for all users. From this figure, we can see that $ESI$ has good performance, and $TS-ESI$ is the fastest algorithm owing to the time-slicing technique. Because the $LCSS$ algorithm ignores the time-domain constraints, it consumes far more time than other methods. Considering that friend recommendation algorithms can be used in offline scenarios, the time complexity of $TS-ESI$ is applicable.

*2) Precision and F1-score:* Based on the data of first five days, given an algorithm, we can obtain a list of recommended friends that have a chance of encountering any user $u_i$ on the 6-th day.

Table I shows average F1-score of the results, which are satisfying in consideration that users may move around the city randomly. We can observe that the $TS-ESI$ algorithm has best performance when $\lambda = \delta$, in which case both the $TS-ESI$ and $ESI$ have much better performance than $LCSS$.

TABLE I: F1-score of compared algorithms.

| | LCSS | ESI | TS-ESI ($\lambda = 0.5\delta$) | TS-ESI ($\lambda = \delta$) | TS-ESI ($\lambda = 2\delta$) |
|---|---|---|---|---|---|
| F1-score | 0.196 | 0.428 | 0.152 | 0.452 | 0.167 |

In practical scenarios, the ASN platform may recommend a list of candidate friends to a user, and thus examine the accuracy of the top-$K$ recommendation. Figure 3 shows the average precision of top-$K$ recommendation among all 3000 users in the dataset, and from the results we can see that when we require a recommendation list with less than 30 candidates, the average precision of $ESI$ and $TS-ESI(\lambda = \delta)$ are both over 50%, while the $LCSS$ algorithm produces a precision around 13%. When we require a recommendation with only top-5 candidates, the precision under $ESI$ and $TS-ESI(\lambda = \delta)$ can be up to more than 60%.

*3) Mean Average Precision:* Figure 4 presents the mAP of top-$K$ friend recommendation. We can observe that when $K < 12$, the $ESI$ has a better performance in mAP, otherwise the $TS-ESI(\lambda = \delta)$ performs better. This may be attributed to the fact that when $K$ increases, the precision of $ESI$ can be easily tampered by outliers. Figure 5 shows the distribution of mAP when $K = 15$, where both ESI and TS-ESI methods

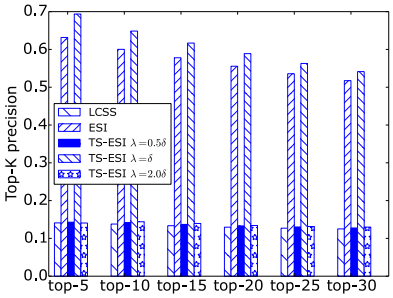


Fig. 3: Precision for top-$K$ friend recommendation.

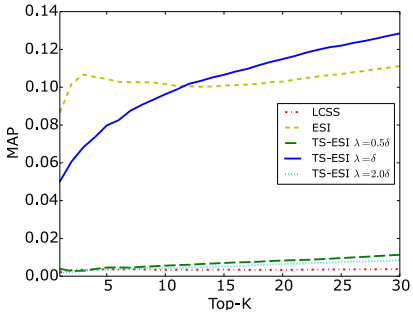


Fig. 4: MAP vs. Top-$K$ friend recommendation.

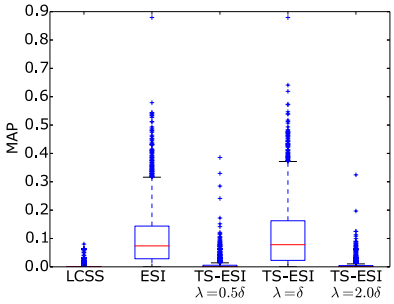


Fig. 5: The mAP distribution for top-15 friend recommendation.

have a good performance in mAP, while $TS-ESI(\lambda=\delta)$ performs better.

In sum, the $TS-ESI(\lambda=\delta)$ performs better both in runtime and precision, because $TS-ESI$ reduces its sensitivity to outliers and can better characterize the statistics of users' mobility patterns. And $TS-ESI(\lambda=\delta)$ performs much better than $TS-ESI(\lambda=0.5\delta)$ and $TS-ESI(\lambda=2\delta)$, because when $\lambda=0.5\delta$, too much eligible pairs have been ignored while too much ineligible pairs have been included when $\lambda=2\delta$.

## VI. Conclusion

In this paper, in order to design a better friend recommendation algorithm in ASNs, we investigate the problem of predicting whether two users will encounter in the future by analyzing their historical fine-grained trajectory data. We formulate a serendipity maximization problem that seeks to maximize the likelihood for two users to encounter with each other. Specifically, we define the *serendipity* of two trajectories in both spatial and temporal domains to quantify the similarity between two users' trajectories, and propose a time-sliced expected serendipity index algorithm for calculating the similarity of the trajectories between a user and the list of recommended candidate friends. Experiments over a real-world dataset collected from a social sports app show that the proposed algorithm outperforms existing approaches the precision and the runtime.

## VII. Acknowledgement

This work was supported by Shenzhen Key Fundamental Research Projects JCYJ20160330095313861 and by NSFC under grants 61572051, 61632017.

## References

[1] N. B. Ellison *et al.*, "Social network sites: Definition, history, and scholarship," *Journal of Computer-Mediated Communication*, vol. 13, no. 1, pp. 210–230, 2007.

[2] Y. Zhang, Y. Bai, L. Chen, K. Bian, and X. Li, "Influence maximization in messenger-based social networks," in *Global Communications Conference (GLOBECOM), 2016 IEEE*. IEEE, 2016, pp. 1–6.

[3] Y. Zheng, L. Zhang, Z. Ma, X. Xie, and W.-Y. Ma, "Recommending friends and locations based on individual location history," *ACM Transactions on the Web (TWEB)*, vol. 5, no. 1, p. 5, 2011.

[4] S.-P. Ma, W.-T. Lee, and C.-H. Kuo, "Location explorer with information services: A mobile application to deliver location-based web services," in *Next-Generation Electronics (ISNE), 2013 IEEE International Symposium on*. IEEE, 2013, pp. 283–286.

[5] "Apple developers' guide." [Online]. Available: https://developer.apple.com/reference/corelocation/cllocationmanager

[6] J. L. James, "Mobile dating in the digital age: Computer-mediated communication and relationship building on tinder," Ph.D. dissertation, Texas State University, 2015.

[7] N. B. Silva, R. Tsang, G. D. Cavalcanti, and J. Tsang, "A graph-based friend recommendation system using genetic algorithm," in *IEEE Congress on Evolutionary Computation*. IEEE, 2010, pp. 1–7.

[8] J. Kwon and S. Kim, "Friend recommendation method using physical and social context," *International Journal of Computer Science and Network Security*, vol. 10, no. 11, pp. 116–120, 2010.

[9] M. Wu, Z. Wang, H. Sun, and H. Hu, "Friend recommendation algorithm for online social networks based on location preference," in *Information Science and Control Engineering (ICISCE), 2016 3rd International Conference on*. IEEE, 2016, pp. 379–385.

[10] Z. Wang, J. Liao, Q. Cao, H. Qi, and Z. Wang, "Friendbook: a semantic-based friend recommendation system for social networks," *IEEE Transactions on Mobile Computing*, vol. 14, no. 3, pp. 538–551, 2015.

[11] G. Wang, B. Wang, T. Wang, A. Nika, H. Zheng, and B. Y. Zhao, "Whispers in the dark: analysis of an anonymous social network," in *Proceedings of the 2014 Conference on Internet Measurement Conference*. ACM, 2014, pp. 137–150.

[12] B.-K. Yi, H. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Data Engineering, 1998. Proceedings., 14th International Conference on*. IEEE, 1998, pp. 201–208.

[13] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Data Engineering, 2002. Proceedings. 18th International Conference on*. IEEE, 2002, pp. 673–684.

[14] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*. ACM, 2005, pp. 491–502.

[15] K. Kishida, *Property of average precision and its generalization: An examination of evaluation indicator for information retrieval experiments*. National Institute of Informatics Tokyo, Japan, 2005.