Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.067470

ARTICLE

A Virtual Probe Deployment Method Based on User Behavioral Feature Analysis

Bing Zhang and Wenqi Shi*

Key Laboratory of Cyberspace Situation Awareness of Henan Province, Zhengzhou, 450001, China

*Corresponding Author: Wenqi Shi. Email: shiwenqi1606@163.com

Received: 05 May 2025; Accepted: 09 October 2025; Published: 09 December 2025

ABSTRACT: To address the challenge of low survival rates and limited data collection efficiency in current virtual probe deployments, which results from anomaly detection mechanisms in location-based service (LBS) applications, this paper proposes a novel virtual probe deployment method based on user behavioral feature analysis. The core idea is to circumvent LBS anomaly detection by mimicking real-user behavior patterns. First, we design an automated data extraction algorithm that recognizes graphical user interface (GUI) elements to collect spatio-temporal behavior data. Then, by analyzing the automatically collected user data, we identify normal users' spatio-temporal patterns and extract their features such as high-activity time windows and spatial clustering characteristics. Subsequently, an anti-detection scheduling strategy is developed, integrating spatial clustering optimization, load-balanced allocation, and time window control to generate probe scheduling schemes. Additionally, a self-correction mechanism based on an exponential backoff strategy is implemented to rectify anomalous behaviors and maintain system stability. Experiments in real-world environments demonstrate that the proposed method significantly outperforms baseline methods in terms of both probe ban rate and task completion rate, while maintaining high time efficiency. This study provides a more reliable and clandestine solution for geosocial data collection and lays the foundation for building more robust virtual probe systems.

KEYWORDS: Virtual probe; behavior feature analysis; anomaly detection; scheduling strategy; geosocial data collection

1 Introduction

With the deep integration of mobile internet and positioning technologies, mobile applications based on LBS have experienced rapid development [1]. These applications provide Location-Based Social Discovery (LBSD) services, which enable users to explore other nearby users and their activities. This process results in the generation of valuable geosocial data. This data holds significant application potential across various fields such as business decision-making, software development and testing, public safety, and privacy research [2]. For instance, in marketing, it can assist enterprises in achieving regionalized precision advertising [3,4]; in development and testing, it helps identify potential issues of applications in different geographical scenarios; in public safety, it can provide data support for criminal geolocation and law enforcement forensics [5]; in location privacy research, it aids in designing more effective privacy protection mechanisms [6]. However, current mainstream data acquisition methods, such as open application programming interface (API) calls and public datasets, commonly face limitations including tightened interface permissions, incomplete data coverage, and delayed updates, making it difficult to obtain large-scale, up-to-date geosocial data.

To overcome these limitations, virtual probe deployment technology has emerged as a novel technical approach for acquiring geosocial data. Its core concept involves simulating mobile devices, adjusting device latitude and longitude coordinates via virtual positioning technology [6–8], accessing LBSD services, and traversing target coordinates in bulk to collect publicly visible user data based on different virtual probe scheduling strategies. Here, a virtual probe is essentially a real user account of an LBS application, which has the ability to arbitrarily modify the geographic location of the account; the virtual positioning refers to technologies that simulate global positioning system (GPS) location data for various applications; the scheduling strategy, which plans the specific rules for probes to execute collection tasks, is the core component of this method.

Existing virtual probe scheduling strategies can be categorized into random path traversal and fixed path traversal based on path planning, and single-probe traversal vs. multi-probe collaborative traversal based on task allocation. Compared to official APIs and public datasets, the virtual probe deployment method offers significant advantages: firstly, strong real-time capability, enabling the acquisition of the latest data within a specified time window; secondly, high flexibility, supporting on-demand setting of collection areas and cycles. Prior research has demonstrated the preliminary application potential of this technology in fields like cybersecurity. For example, a research team used virtual probes to collect user distance information from the "People Nearby" function, constructed a distance notification model, achieved localization of malicious users, and validated the feasibility of this method [9].

However, the large-scale application of virtual probes faces severe challenges from the multi-layer anomaly detection mechanisms employed by LBS applications. To protect platform ecosystem health and user privacy, mainstream LBS applications widely implement detection mechanisms based on spatio-temporal features [10–12]: these include judging whether location switching speed exceeds reasonable limits and detecting whether active time periods are abnormal (e.g., high-frequency activity during early morning hours). When account behavior triggers these rules, it may be flagged as anomalous and subjected to bans. Existing virtual probe methods, in pursuit of traversal efficiency, often frequently switch locations and ignore path rationality, making their behavioral patterns highly prone to triggering platform risk controls and resulting in account bans. Therefore, the "contradiction between the demand for large-scale coordinate traversal and account survival" has become the main bottleneck hindering the practical application of virtual probe technology.

Addressing this contradiction, this paper proposes a virtual probe deployment algorithm based on user behavior features analysis. The algorithm leverages the spatio-temporal behavior features of normal users to guide the construction of an anti-detection scheduling strategy for virtual probes. The core idea of this method is to "simulate human location behavior," enabling the probes to align their coordinate switching processes as closely as possible with the movement feature of real users, thereby evading platform detection and achieving a balance between "traversal efficiency" and "account security." The main contributions of this paper include:

- The extraction of spatio-temporal behavior features of normal users.
- The proposal of an anti-detection virtual probe scheduling strategy based on behavior simulation.
- Experimental validation of the method's effectiveness on the "Joyrun" platform.

The structure of this paper is as follows: Section 2 reviews related research work. Section 3 describes the basic principles and main steps of the proposed method. Section 4 introduces automated architecture for experimental data collection. Section 5 explains the method for extracting user spatio-temporal behavior feature; Section 6 describes in detail the design of the virtual probe scheduling strategy. Section 7 describes the experimental results and analysis. Section 8 summarizes the whole paper and outlines future research directions.

2 Related Work

To address the issue where existing virtual probe deployment methods trigger the anomaly detection mechanisms of LBS applications due to their behavioral patterns deviating from normal users in terms of temporal distribution and spatial movement trajectories—a consequence of lacking understanding of application user behavior patterns—this paper proposes a virtual probe deployment method based on user behavior features analysis. Its fundamental principle is to mine the spatiotemporal behavior patterns of normal users, extract spatiotemporal behavior features, and use these to design an anti-detection virtual probe scheduling strategy that makes probe behavior resemble that of real user groups. This method is structured into three layers: data collection, extraction of user spatiotemporal behavior feature, and design of the virtual probe scheduling strategy.

2.1 Geosocial Data Acquisition Methods

Current geosocial data acquisition technologies primarily include three categories: official API access, utilization of public datasets, and virtual probe deployment. Each has its own features, with significant differences in acquisition efficiency, data quality, and applicable scenarios.

Official API access provides compliant data interfaces for developers, typically requiring qualification reviews and permission applications. Its main advantages are data legitimacy and format standardization. For example, Khan et al. [13] used Sina Weibo's official API to obtain user check-in data, analyzing the correlation between check-in time, frequency, and venue categories, revealing the spatio-temporal patterns of urban population activities in Shanghai. Dokuz [12] utilized the Twitter Streaming API and representational state transfer (REST) API to collect historical user tweet data, proposing a "social velocity measurement" method to detect anomalous daily activities based on the spatial distance and time difference between consecutive posts. However, with the tightening of privacy protection policies, many platforms have raised the bar for API applications, limited call frequency and content scope, or even closed some interfaces. These restrictions make it difficult to support refined, large-scale data acquisition.

Public datasets are often released by research institutions or companies after desensitization, offering advantages of low cost and large sample size. Table 1 lists several widely used geosocial datasets and their statistical information. The Gowalla check-in dataset [14] originates from the defunct platform of the same name and boasts the largest social network among public datasets, with high user activity. It remains irreplaceable in social influence analysis and trajectory prediction studies due to its complete social relationship structure. The Brightkite dataset [14] also comes from a discontinued platform; its dense social network structure has made it a benchmark resource for geosocial research. The Foursquare dataset [15] is the largest, containing two years of global check-in records. With rich point of interest (POI) tags and relatively recent data, it is widely used in urban computing and personalized recommendation research. However, the collection period of these datasets is concentrated in 2010–2013, unable to reflect current user behavior feature. Their geographical coverage also shows obvious bias, focusing on European and American cities while lacking data from emerging market regions, thus suffering from limitations of timeliness lag and insufficient sample representativeness.

Table 1: Publicly available geosocial datasets

Dataset	Users	Links	Check-Ins	Period
Gowalla	196.6 K	950.3 K	6.4 M	2009.2–2010.10
Brightkite	58.2 K	214.1 K	4.5 M	2008.4–2010.10
Foursquare	114.3 K	607.3 K	22.8 M	2012.4–2014.1

The virtual probe deployment method involves simulating devices equipped with virtual positioning technology to masquerade as normal users. These probes then systematically collect public data from LBSD services based on preset scheduling strategies. Device simulation typically relies on emulators (e.g., NoxPlayer) or multi-instance tools for physical devices, modifying hardware parameters like international mobile equipment identity (IMEI), Android identity document (ID), and device model to evade device fingerprinting. Virtual positioning often uses Hook technology (e.g., Xposed framework) or system-level tools (e.g., Fake GPS) to dynamically modify device latitude and longitude, achieving location spoofing and area coverage.

2.2 Virtual Probe Scheduling Strategies

Virtual probe scheduling strategies directly impact data acquisition efficiency and account security. Existing strategies can be classified along two dimensions: by task allocation into single-probe traversal and multi-probe collaborative traversal; by path planning method into random traversal and fixed-path traversal. Single-probe strategies are simple to implement but suffer from low traversal efficiency and longtime consumption. Multi-probe strategies shorten traversal time through parallel collaboration but involve higher complex coordination. Random traversal requires no path planning, is easy to develop, and is suitable for small-scale scenarios. Fixed-path traversal accesses coordinates in a preset sequence, which helps improve the systematic nature of data collection but still fails to address the issue of behavioral rationality.

Qin et al. [16] employed a single-probe gradual approximation strategy in user localization research, achieving precise targeting of users through continuous location adjustments to bypass distance ambiguity. Shi et al. [17] generated location sequences based on triple measurements. A single probe was used to obtain the distance notifications between users, enabling the experiments on user localization. Ding et al. [18], while studying the feasibility of weak adversary tracking, divided the area of interest into honeycomb cells, randomly assigned coordinates to multiple probes, and set fixed sleep times for cyclic traversal to find target users.

2.3 Anomaly Detection Mechanisms in LBS Applications

Anomaly detection operates on the principle that LBS applications pre-establish a baseline of legitimate user behavior; any deviation from this baseline is flagged as anomalous. To maintain platform ecology and user privacy, mainstream LBS applications widely deploy multi-level anomaly detection mechanisms. Existing detection methods can be summarized into the following three categories:

Context-based anomaly detection methods focus on analyzing the text and contextual information of user-generated content. Takahashi et al. [19] proposed a probabilistic generative model for user mention behavior, detecting anomalies that significantly deviate from the model to identify emerging topics. Ruan et al. [20] constructed a set of features describing user social behavior for analyzing and identifying compromised social media accounts. Network-oriented anomaly detection primarily identifies anomalies based on user social relationships and interaction patterns. Dutta et al. [21] proposed 64 features to distinguish normal users from malicious customers targeting black market collusive retweeting behavior. Eswaran et al. [22] proposed the SpotLight method, which quickly identifies graph structural anomalies by monitoring the sudden appearance or disappearance of dense directed subgraphs within the graph. Spatio-temporal behavior-based anomaly detection identifies anomalies based on the time and spatial features of user behavior. This category of detection is directly related to virtual probe scheduling behavior. Dokuz [11] extracted two features from user daily mobility data: the total number of visited locations per day and the total distance moved, applying the density-based spatial clustering of applications with noise (DBSCAN) clustering algorithm to identify anomalous activities. Dokuz [12] further proposed the concept of "social velocity," calculating movement speed from the positional difference and time difference between consecutive user posts to detect spatio-temporal anomalies.

This paper primarily focuses on spatio-temporal behavior anomalies caused by virtual probe scheduling strategies. In contrast, context-based and network-oriented anomaly detection methods involve more account construction and content camouflage, which are not the focus of this study and therefore will not be discussed in depth.

3 Virtual Probe Deployment Based on User Behavioral Feature Analysis

To address the problem that the existing virtual probe deployment method is very easy to trigger the anomaly detection mechanism of LBS applications due to their behavioral patterns deviating from normal users in terms of temporal distribution and spatial movement trajectories—this paper proposes a virtual probe deployment method based on user behavioral features, and its basic principle is to mine the spatio-temporal behavioral patterns of normal users of the application, extract spatio-temporal behavior features, and use these to design an anti-detection virtual probe scheduling strategy that makes probe behavior resemble that of real user groups. This method is structured into three layers: data collection, extraction of user spatio-temporal behavior features, and design of the virtual probe scheduling strategy.

The method first realizes the automated collection of user information by combining GUI element recognition and Appium automated testing tools with the deployment of probes. Then, the user's online time data and location information are statistically analyzed to extract temporal and spatial behavior features. Finally, based on the above spatio-temporal behavioral features, the virtual probe scheduling strategy is designed. The target coordinates are clustered and assigned to probes, and a moving path is planned for each one. This process ensures that each probe conform to the spatio-temporal behavioral patterns of real users, thereby circumventing application monitoring. Fig. 1 illustrates the principle and architecture of this method. The main steps of the method are as follows:

Part I: User Data Collection

We design an automated user information acquisition algorithm based on GUI element recognition. This algorithm aims to simulate user operations. It extracts user data directly from the client interface by parsing mobile application interface elements.

Step 1: GUI Element Recognition

Identify identifiers of the desired GUI elements (e.g., resource-id, XPath, etc.). Obtain information such as the application package name and the version number of the device through decompilation and connect the device using the Appium tool to call its API for accessing UI elements.

Step 2: Automated User Data Acquisition

Automation scripts are written to simulate real user behaviors, including clicking, swiping, text input, and other operations. Navigate to the target interface and extract user data. Then, run the automation scripts with the help of the Appium tool to save the acquired user data to the database.

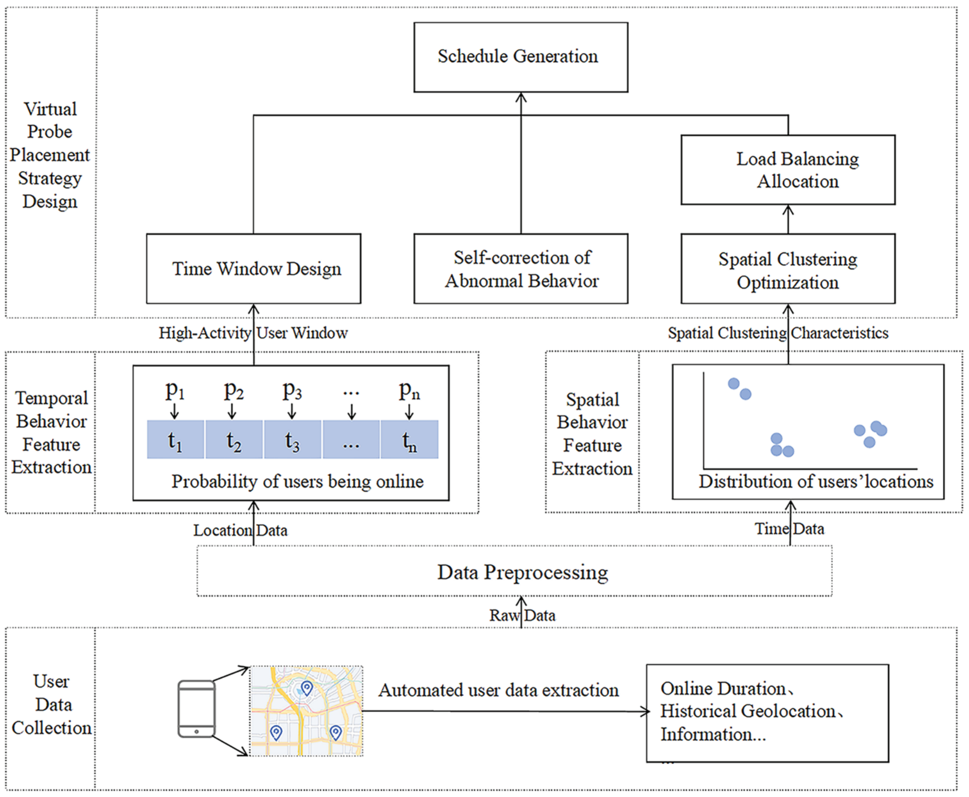

Virtual Probe Placement Strategy Design
Schedule Generation
Load Balancing Allocation
Time Window Design
Self-correction of Abnormal Behavior
Spatial Clustering Optimization
High-Activity User Window
Spatial Clustering Characteristics
Temporal Behavior Feature Extraction
p1 p2 p3 … pn
t1 t2 t3 … tn
Probability of users being online
Spatial Behavior Feature Extraction
Distribution of users' locations
Location Data
Time Data
Data Preprocessing
Raw Data
User Data Collection
Automated user data extraction
Online Duration、Historical Geolocation、Information...

Figure 1: The principal architecture of the virtual probe deployment method based on the analysis of user behavior features

Part II: Feature Extraction of User Spatio-Temporal Behavior

Features are extracted from the cleaned and formatted user data across both temporal and spatial dimensions.

Step 3: User Temporal Behavior Feature Extraction

Based on the regular fluctuations of user behavior over time, a group behavior aggregation analysis method is adopted. The probabilities of users being online in different time periods on different dates are calculated to constrain probes to perform coordinate switching and information collection during high-activity periods.

Step 4: User Spatial Behavior Feature Extraction

Heatmaps of historical user coordinate distributions reveal a "multi-center aggregation" pattern. Spatial clustering analysis is performed using the DBSCAN clustering algorithm.

Part III: Design of Virtual Probe Scheduling Strategy

Based on the spatio-temporal behavioral features of users, the virtual probe scheduling strategy is designed by spatial clustering and time windows.

Step 5: Spatial Clustering Optimization

Use a clustering algorithm to group the target simulated coordinates into clusters. Merge small-scale clusters into the nearest large-scale cluster to optimize the clustering results.

Step 6: Load Balancing Allocation

To minimize the traversal time for probes, a load-balancing algorithm is employed: This algorithm first quantifies the traversal load of each cluster (using the traveling salesman problem (TSP) algorithm to generate the shortest traversal path within the cluster and calculate the traversal time). It then iteratively optimizes the load allocation based on a greedy algorithm to achieve proximity matching between clusters and probes.

Step 7: Probe Schedule Generation

Adopt a hierarchical scheme of "local optimization + global coordination": Calculate the inter-cluster sequence and time intervals for each probe and integrate them with the intra-cluster paths and time intervals to form a complete probe schedule.

Step 8: Time Window Design

Based on the user's historical dynamic release time distribution, extract the time periods with a high probability of user activity. Probes are scheduled during users' active time windows for efficient data collection.

Step 9: Self-Correction of Abnormal Behavior

To cope with the possible abnormal behavior of probes, a self-correction mechanism for abnormal probe behavior is designed. This mechanism adopts an exponential backoff strategy for self-correction. It aims to avoid wasting system resources due to over-frequent attempts and maintain the stable operation of the system.

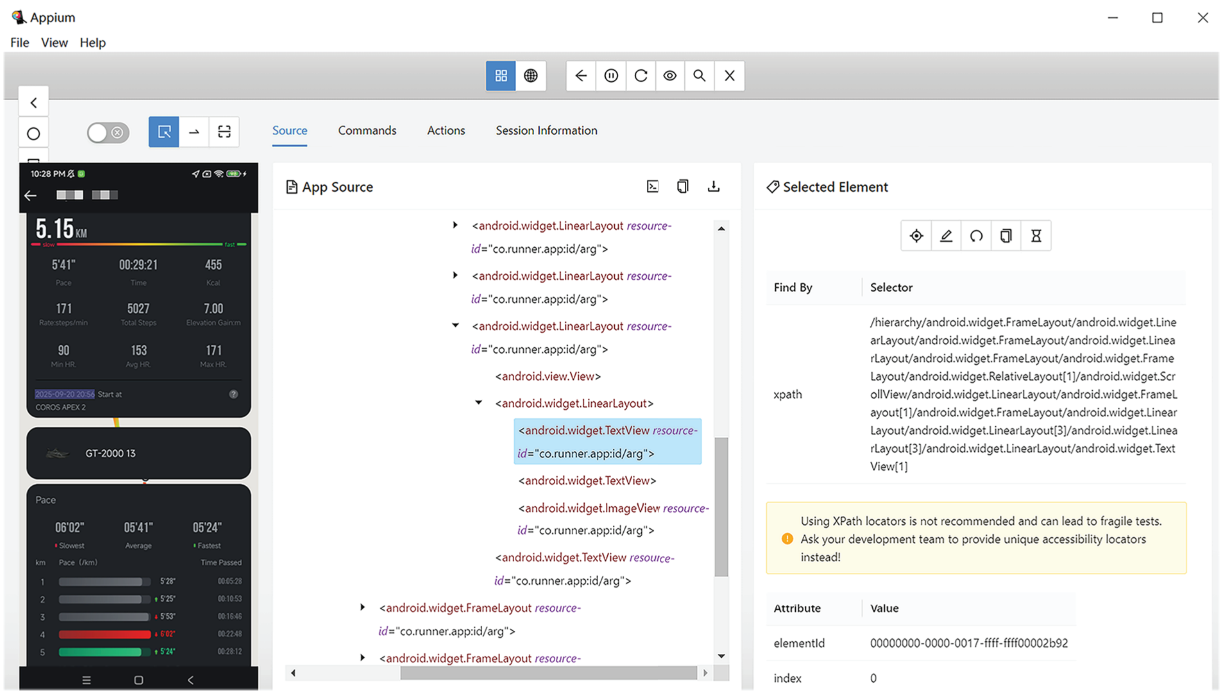
The following is a detailed description of the above steps.

4 Automated User Data Extraction Based on GUI Element Identification

This section details the GUI element recognition and automated user data extraction techniques employed in the data acquisition process, the latter of which is implemented using the Appium tool.

4.1 GUI Element Recognition

The interface of a mobile application is structured as a "view hierarchy" or "UI tree", which represents the logical, tree-like relationships and nesting of interface elements. In Android, each visual element (e.g., button, text box, list item) exists as a node in the UI tree and has a unique set of attributes to support precise positioning. Appium uses UIAutomator (Android) to obtain the application's UI hierarchy to identify the elements. The attributes of the user interface elements in Android are resource ID, text, class, and XPath. In automated script design, the element positioning strategy needs to balance efficiency and robustness. According to the features of the Android platform, the following priority strategy is recommended: First, use resource ID, text, and class as unique identifiers to locate the elements because of their fast-positioning speed and high accuracy. If an element lacks unique identifiers and cannot be located by combining attributes, XPath expressions can be used to describe its hierarchical path within the UI tree. XPath locates elements by traversing parent-child and sibling node relationships. For example, as shown in Fig. 2, the time information of the "Joyrun" lacks other unique identifiers and can be localized using the XPath expression.



```
<android.widget.LinearLayout resource-id="co.runner.app:id/arg">
<android.widget.LinearLayout resource-id="co.runner.app:id/arg">
<android.widget.LinearLayout resource-id="co.runner.app:id/arg">
    <android.view.View>
    <android.widget.LinearLayout>
        <android.widget.TextView resource-id="co.runner.app:id/arg">
        <android.widget.TextView>
        <android.widget.ImageView resource-id="co.runner.app:id/arg">
    <android.widget.TextView resource-id="co.runner.app:id/arg">
<android.widget.FrameLayout resource-id="co.runner.app:id/arg">
<android.widget.FrameLayout resource-
```

Find By	Selector
xpath	/hierarchy/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.FrameLayout/android.widget.FrameLayout/android.widget.RelativeLayout[1]/android.widget.ScrollView/android.widget.LinearLayout/android.widget.FrameLayout[1]/android.widget.FrameLayout/android.widget.LinearLayout/android.widget.LinearLayout[3]/android.widget.LinearLayout[3]/android.widget.LinearLayout/android.widget.TextView[1]

Using XPath locators is not recommended and can lead to fragile tests. Ask your development team to provide unique accessibility locators instead!

Attribute	Value
elementId	00000000-0000-0017-ffff-ffff00002b92
index	0

Figure 2: Schematic diagram of the GUI element identification and search process

4.2 Automated Extraction of User Data

This study implemented automated user data extraction based on the Appium tool. For Android, Appium is built upon the WebDriver protocol and utilizes Bootstrap.jar to conduct automation tests by issuing commands to UIAutomator. Appium is a mobile application automation testing tool based on the C/S (client-server) architecture. Its core function is to send the commands of automation scripts to the mobile device and return the results of device-side operations to the client. Appium's architecture follows a client-server model and comprises three core components: the client, the server (both residing on the PC), and the Bootstrap toolkit running on the mobile device. The process of automated user data acquisition is shown in Fig. 3 and is divided into the following steps:

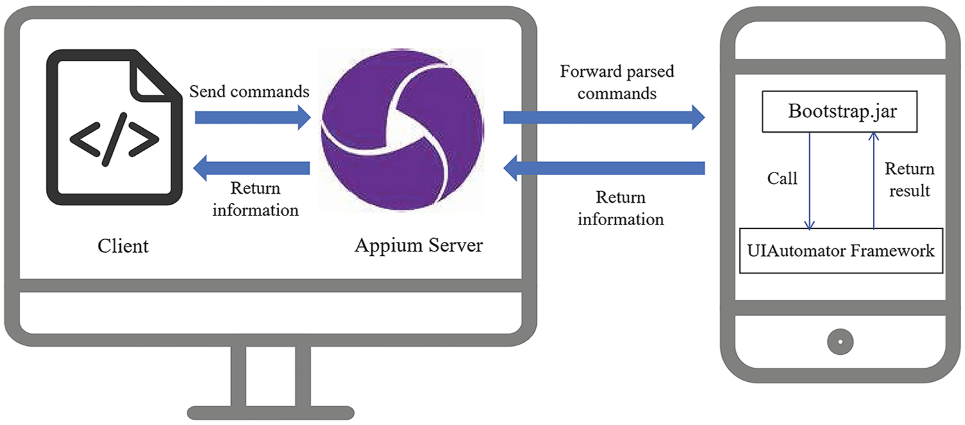


Figure 3: Schematic diagram of the automated extraction process of user data

(1) Client-side script execution and command sending: Write an automation test script on the client side, defining the user operations to be simulated (e.g., clicking, swiping, reading text) and the target data (e.g., username, gender, distance, etc.).
(2) Server-side command forwarding and device-side execution: The Appium server listens to the client's REST request, parses the request content (e.g., operation type, target element, parameters, etc.), and forwards the parsed commands to the server-side Bootstrap toolkit, which receives the commands and then starts to call the automation framework on the phone's UI Automator to execute the operation.
(3) The Appium server receives the operation results returned by Bootstrap and encapsulates them into a REST response. The client receives the data returned by the server and saves it to the database.

5 User Behavior Feature Extraction Based on Spatio-Temporal Data

This section analyzes the collected user dataset to uncover user activity patterns across temporal and spatial dimensions, thereby extracting their spatio-temporal features.

5.1 User Temporal Behavior Feature Extraction

Due to the significant temporal regularity of users' living habits and activity patterns (e.g., fixed workout periods such as morning or night running will trigger higher online behaviors), their application usage behaviors show regular fluctuations in different time periods. To investigate this pattern in depth, this section conducts a fine-grained analysis of user behavior along the temporal dimension. Considering the difference in the life rhythm between weekdays and weekends (e.g., commuting, leisure arrangements, etc.), the data is first divided into two categories of weekdays (Monday to Friday) and weekends (Saturday to Sunday) according to the time attributes, and the differences in the activity patterns of the users during the two time periods are explored separately.

Specifically, we quantify users' fine-grained online preferences by their probability of being active in different periods. To mitigate the limitations of individual-level analysis due to sparse daily data per user, we employ a group behavior aggregation approach. This method first aggregates the active time data of all users to obtain the distribution of user counts across periods. Then, the percentage of activity in each period relative to the total daily activity is calculated, forming a probability distribution. The procedure is as follows: First, data is categorized by date type (weekday or weekend). The day is divided into 24 one-hour periods. Next, for each category, the total number of posts (e.g., "dynamic releases") in each period is counted. Finally, the probability of a user being online in a given period is calculated as the ratio of posts in that period to the total posts of the day, and the formula is:

$$P(t) = \frac{N_t}{\sum_{i=0}^{23} N_i} \times 100\% \tag{1}$$

where N_t denotes the number of online users in the i-th period, and $\sum N_i$ is the total number of online users throughout the day for the corresponding date type (weekday or weekend). Then active time period filtering is performed, a threshold φ is set, and the time period of $P(t) \geq \varphi$ is extracted as the probe operation window $T_{peak} = \{t | P(t) \geq \varphi\}$. The value of the threshold φ is determined in the experimental section.

5.2 User Spatial Behavior Feature Extraction

Users' spatial behavior also demonstrates significant regularity. As shown in Fig. 4 (plotted based on historical check-in data from June 2024 to September 2024 for users of a specific LBS application), user activities are not randomly dispersed but rather cluster in a typical "multi-center aggregation" pattern. This pattern is characterized by dense concentrations of points in a few core areas (e.g., residences, workplaces), with sparse points scattered as low-density noise (e.g., from occasional travel). This pronounced spatial clustering forms the basis for extracting user spatial behavior features.

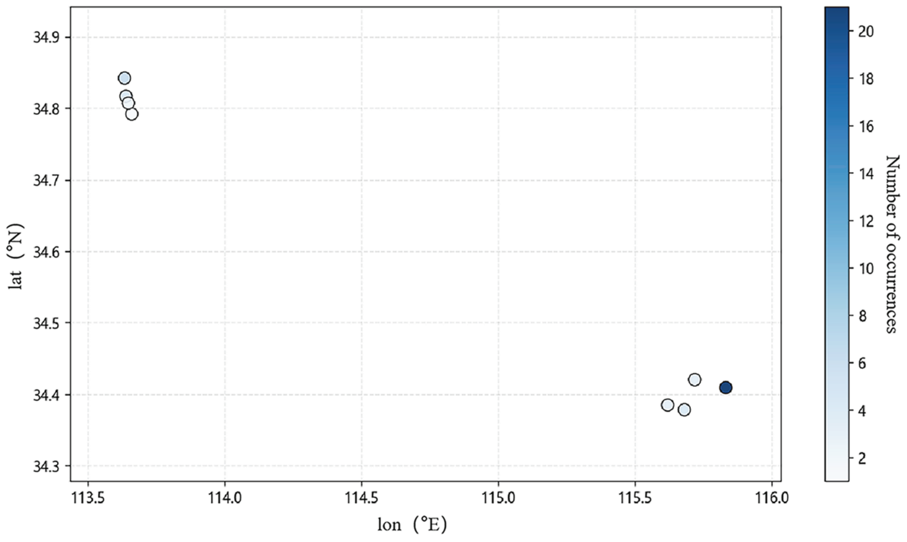


Figure 4: Heat map of the user's historical geographic coordinate distribution

To automatically identify these aggregation patterns from coordinate data, this paper employs the DBSCAN algorithm to cluster user historical coordinates. The values of the algorithm's two key parameters—the neighborhood radius ε and the minimum number of samples required for a core point $MinPTs$—will be determined in the subsequent experimental section.

6 Design of Virtual Probe Scheduling Strategy

Building upon the spatio-temporal behavioral features of normal users extracted in Section 5, this section details the design of an efficient and stealthy virtual probe scheduling strategy. The strategy encompasses three core mechanisms: spatial allocation optimization, time window control, and abnormal behavior self-correction.

6.1 Spatial Allocation Optimization

(1) Spatial Clustering Optimization

To generate coordinate clusters that reflect the spatial patterns of real user activity areas, all target coordinates to be simulated are first projected from the latitude and longitude coordinate system to a Cartesian coordinate system, forming the target coordinate set $P = (p_1, p_2, \ldots, p_m)$, where p_i represents a single coordinate point and m is the total number of coordinates. The DBSCAN algorithm is used to cluster the set P, with its neighborhood radius ε and minimum points threshold $MinPTs$ set according to the analysis results from Section 5.

The clustering results are subsequently optimized: Clusters containing fewer than $MinPTs$ coordinate points (small clusters) are merged into the nearest large cluster. The merger criterion is based on the inter-cluster centroid distance. Let the centroid of a small cluster $\hat{C}_j$ to be merged be $\mu_t = (x_t, y_t)$, and the centroid of a large cluster C_j be$\mu_j = (x_j, y_j)$. The distance between them is:

$$dist(\mu_t, \mu_j) = \sqrt{(x_t - x_j)^2 + (y_t - y_j)^2} \quad (2)$$

Cluster $\hat{C}_j$ is merged into the cluster C_j for which this distance is minimized. The final output is the optimized cluster set $C = (C_1, C_2, \ldots, C_n)$, where C_j represents a single cluster and n is the final total number of clusters.

(2) Load-Balanced Allocation

To achieve efficient matching between clusters and probes, a load-balanced allocation algorithm is proposed, as shown in Algorithm 1. Its goal is to minimize the load difference among probes while also considering the spatial proximity between clusters and probes. The algorithm consists of two stages: cluster load quantification and dynamic allocation.

Algorithm 1: Load balancing allocation algorithm

```
Input:  Cluster set $C = (C_1, C_2, \ldots, C_K)$,
        Set of probes $R = (r_1, r_2, \ldots, r_N)$,
        Movement speed $v$ (m/s),
        Position switching time $t_s$ (s)
Output: Probe-cluster allocation mapping $M$
1   $\delta = 1.2$; //the maximum imbalance factor
2   //Calculate load for each cluster
3   For each $C_k \in C$ do
4       $P_k = [p_1, p_2, \ldots, p_S] = TSP(\text{Points}(C_k))$
5       $\Delta t(C_k) \leftarrow \max\{distance(p_i, p_{i+1})/v, t_s\}$
6       $L_k \leftarrow \sum_{i=1}^{S-1} \Delta t(C_k)$
7   End For
8   //Load balancing allocation process
9   $L_{\text{total}} \leftarrow \sum_{k=1}^{K} L_k$
10  $\overline{L} \leftarrow L_{\text{total}}/|R|$ //Theoretical average load
11  Initialize $L[r_i] \leftarrow 0, \forall r \in R$
12  $U \leftarrow C$ //Set of unallocated clusters
13  While $U \neq \varnothing$ do
14      $r^* \leftarrow argmin_{r \in R} L[r]$ //Compute distances to $U$
15      Calculate distance between $r^*$ and each cluster in $U$
16      $C\_candidate \leftarrow \{C \in U | L[r^*] + L_C \leq \delta \times \overline{L}\}$
17      If $C\_candidate = \varnothing$ then
18          $C\_candidate \leftarrow U$;
19      End If
20      $C^* \leftarrow argmin_{\{C \in C\_candidate\}} distance(r, C)$
21      $M[r] \leftarrow M[r] \cup C^*$
22      $L[r] \leftarrow L[r] + L[C^*]$
23      $U \leftarrow U \setminus \{C^*\}$
24  End While
25  Return $M$
```

In the cluster load quantification stage, the following operations are performed for each cluster C_k:

- Use the TSP path planning algorithm to calculate the optimal traversal sequence $P_k = [p_1, p_2, \ldots, p_S]$ within the cluster (where $s = |C_k|$ is the number of target points in the cluster).
- Calculate the Euclidean distance $d_i = distance(p_i, p_{i+1})$ between consecutive points in the sequence.

- The cluster load L_k is defined as the total time required to traverse the cluster:

$$L_k \leftarrow \sum_{i=1}^{S-1} \max\left\{d_i/v, t_s\right\} \tag{3}$$

where v is the movement speed of the probe (unit: m/s), and t_s represents the location switching time (unit: seconds). The time consumed per segment is the maximum of the movement time d_i/v and the switching time t_s, as these operations can overlap only if the movement time is longer. By accumulating the actual time consumed for each segment, the comprehensive load L_k for cluster C_k is obtained.

In the dynamic allocation stage, the algorithm assigns clusters to probes iteratively. The core idea is to achieve proximity matching between clusters and probes under the constraint of load balancing. In each iteration, the probe r^* with the current smallest load is selected from the probe set. Then, from the pool of unallocated clusters, the cluster that satisfies the load constraint $L\left[r^*\right] + L_C \leq \delta \times \overline{L}$ (where $\overline{L}$ is the current average load per probe) and is closest to C^* is selected for assignment. This process continues until all clusters are assigned. The resulting mapping $M\left[r\right] = \left\{C_1, C_2, \ldots, C_m\right\}$ represents the set of clusters assigned to each probe.

(3) Probe Schedule Generation

Based on the allocation results, a detailed schedule is generated for each probe, including the global path and time sequence.

For a probe r and its assigned cluster set $M\left[r\right] = \left\{C_1, C_2, \ldots, C_m\right\}$:

- Using the centroids of each cluster as nodes, solve a TSP problem to obtain the inter-cluster visitation order.
- Concatenate the intra-cluster paths according to this order to obtain the global path sequence S_{path}.
- Calculate the corresponding time intervals for each segment in the path based on the movement speed v and switching time t_s, generating the global time sequence S_{time}.

Finally, the tuple $\left(S_{\text{path}}, S_{\text{time}}\right)$ constitutes the complete schedule for probe r.

6.2 *Time Window Control*

To enhance behavioral stealth, probe activities are confined to the high-activity periods of real users, as identified in Section 5.1. Specifically, for each coordinate point in a probe's path, its access time is constrained within the local time's high-activity time window. This mechanism significantly reduces the risk of detection due to anomalous activity during low-activity periods and improves the stealth of probe behavior.

6.3 *Self-Correction of Abnormal Behavior*

To handle runtime exceptions (e.g., page timeouts), an automatic retry strategy with an exponential backoff mechanism is implemented to improve system robustness. Waiting time after the n th failure t_n:

$$t_n = t_{base} \cdot \alpha^{n-1} \tag{4}$$

where t_{base} is the initial waiting time, set $t_{base} = 60$ s; α is the backoff coefficient, set $\alpha = 2$; and n is the number of failures, set $n_{\max} = 5$. According to the above formula, the waiting time is doubled whenever an error message is received. The waiting time is reset to the initial waiting time whenever the probe successfully checks in again. When the probe retries more than 5 times, the probe is marked as failed. This mechanism effectively prevents task interruptions caused by temporary network or interface anomalies.

7 Experimental Setup and Result Analysis

This section evaluates the proposed user behavior feature-based virtual probe deployment algorithm. The experiments are specifically divided into two parts: one for extracting spatio-temporal behavior features of normal users, and another for testing the anti-detection probe scheduling strategy.

7.1 Experimental Setup

The hardware and software configuration of the experiment, as well as the selection of the experimental area, are shown in Table 2 below. The coordinates of the probes to be set up and simulated are randomly generated in the experimental area.

Table 2: Experiment settings

Item	Setting
Automation Tool	Appium
Development Tool	Python
Data Storage Tool	SQLite
Simulation Location Tool	Fake location
Mobile Devices & Versions	4 real phones (Android 10) 6 phone emulators (Android 9)
Application Software & Versions	Joyrun5.41.0
Experimental Area	Zhengzhou, Hangzhou, Beijing
Number of Experimental Points	5000
Number of Probes Used	20
Maximum Probe Travel Speed	60 km/h

(1) Data Collection

Most of the users of Joyrun are mainland users, and the number of users outside the country is relatively small. To ensure an adequate amount of data collection, Zhengzhou, Hangzhou, and Beijing were selected for the data collection experiments. The data collection is mainly focused on the historical dynamic data of Joyrun users and the dynamic data of neighboring users in the nearby dynamic service. The data are stored in two data tables: the user history table and the neighborhood table. The user history table mainly collects information such as IP address, POI, and postdate of each user's history; the neighborhood table mainly collects information about the post time of users in the "Nearby" service within the last day.

For temporal behavior data: Since Joyrun does not directly provide user online status, we used the publication time of user posts as a proxy. A user was considered online at the time of posting. The time elapsed since the user's posting is displayed only when the user's posting is made within the same day in the 'Nearby News' service. Within 1 h, the time is displayed to the minute, such as '13 min ago', and between 1 h and 0:00 on the same day, the time is displayed to the hour, such as '4 h ago'. User updates outside of the same day are only displayed up to the date of publication.

For spatial behavior data, publication times in the 'Nearby' feed were stored with hourly precision to standardize time units. The format of the neighborhood table is as follows: (probe number, probe coordinates, user nickname, publish time, fetch date). We filtered the data by removing entries without POI information. The POI locations from historical posts were then converted to latitude and longitude coordinates using the Amap API. The format of the user history dynamic table is as follows: (probe number, probe coordinates, user ID, user nickname, publication date, IP address, POI information, coordinates, acquisition date).

(2) Evaluation Indicators

This paper primarily uses the following three metrics to evaluate the performance of the virtual probe deployment method:

- Probe Ban Rate (BR): The proportion of probes detected and banned by the platform during a complete coordinate traversal task. This indicator directly reflects the stealth and anti-detection capability of the method.

$$BR = \left(\frac{N_{\text{banned}}}{N_{\text{probe\_total}}}\right) \times 100\% \tag{5}$$

where N_{banned} is the number of probes banned within the statistical period, and $N_{\text{probe\_total}}$ is the total number of probes participating in the task.

- Task Completion Rate (CR): Refers to the proportion of successfully traversed target coordinates to the total number of target coordinates after completing one coordinate traversal task. This indicator is core to measuring the execution efficiency and reliability of the virtual probe deployment method.

$$CR = \left(\frac{N_{\text{completed}}}{N_{\text{target\_total}}}\right) \times 100\% \tag{6}$$

where $N_{\text{completed}}$ is the number of coordinates successfully traversed and verified, and $N_{\text{target\_total}}$ is the total number of target coordinates planned for traversal.

- Traversal Time (TT): Refers to the total time spent by all probes to complete one coordinate traversal task. This indicator is key for measuring the time efficiency of the virtual probe deployment method.

$$TT = T_1 + T_2 + \ldots + T_n \tag{7}$$

where T_i is the time required for the i-th probe to traverse its assigned coordinates.

(3) Control Group Experiment

- Baseline-RR (Random Reference): This method randomly and uniformly distributes target coordinates to all probes. Each probe traverses its assigned coordinates in a random order, operating 24/7. This is the most primitive strategy and serves as the baseline for evaluating the performance gains of all other methods.
- Baseline-RA (Random Allocation with TSP Traversal): This method randomly and uniformly distributes target coordinates to all probes. Each probe traverses its assigned coordinates following the TSP shortest path, operating 24/7.
- Ours-S (Our Spatial Strategy): This method uses the complete spatial clustering strategy proposed in this paper to allocate target coordinates to all probes. Each probe traverses its assigned coordinates according to the generated schedule, operating 24/7.
- Ours-BSA (Our Complete Method–Behavior and Spatial Analysis): This is the complete method proposed in this paper. It uses the complete spatial clustering strategy proposed herein to allocate target coordinates to all probes. Each probe traverses its assigned coordinates according to the generated schedule but only within the high-activity time windows.

7.2 Experiments for Determining User Behavior Feature Parameters

(1) Determining the Temporal Feature Extraction Threshold

The core of the temporal feature lies in determining the users' high-activity periods. Fig. 5 illustrates the proportion of the number of users online in each hour to the total online users, including activity patterns for both workdays and weekends. In Fig. 5, it can be observed that the temporal behavioral patterns of users do differ between weekdays and weekends. To convert the continuous probability distribution into discrete high-activity periods, a threshold φ needs to be set. Fig. 6 shows the impact of different φ values on the high-activity periods. Fig. 6a shows the change curve of high-activity period coverage rate (the proportion of data volume within the high-activity periods). As shown, the coverage rate decreases monotonically as φ increases. When φ is small, the coverage is high but introduces many low-activity periods; when φ is large, user activity is highly concentrated within the periods but the coverage is low. Fig. 6b shows the change curve of the total length of high-activity periods. As φ increases, the periods deemed "high activity" continuously decrease, and the total length shortens.

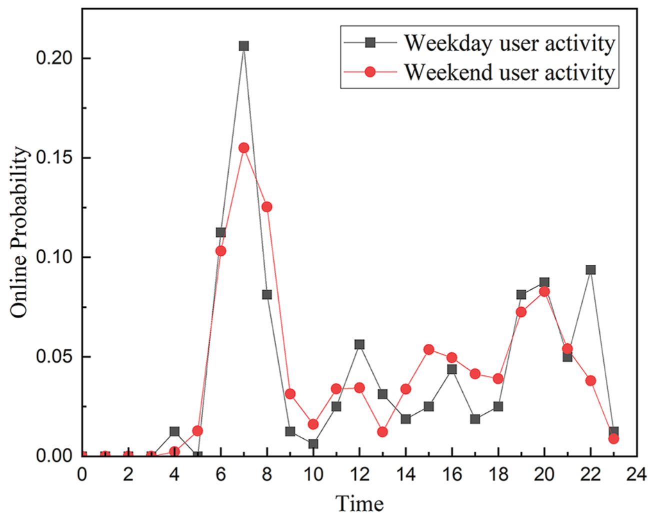


Figure 5: User activity statistics

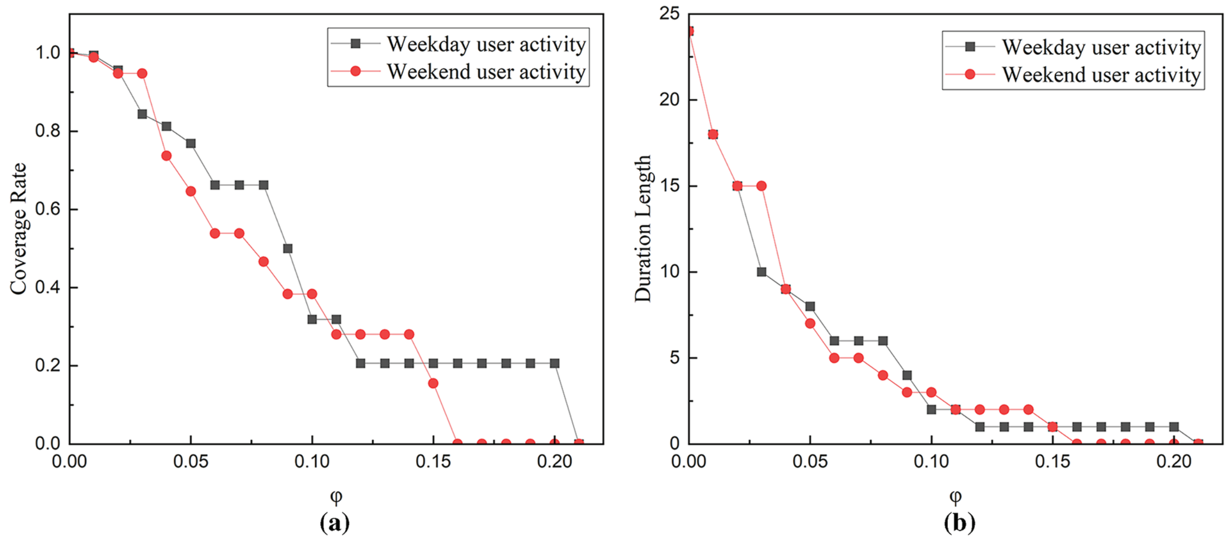


Figure 6: Determination of the temporal feature threshold φ. (**a**) Coverage rate change curve of high-activity periods; (**b**) Duration length change curve of high-activity periods

A threshold that balances high coverage rate and reasonable total length of high-activity periods is needed. Observing Fig. 6, when φ is set to 0.04, the high-activity period coverage for workdays remains at a high level of 81.25%, with a total length of 9 h. Similarly, for weekends, the coverage remains at a high level of 73.73%, with a total length of 9 h. This threshold captures most user activity while keeping the probes' effective working time within a reasonable range, avoiding the risk of introducing low-activity periods due to excessively long windows. Therefore, subsequent experiments use $\varphi = 0.04$ to define the high-activity time window. Thus, the high-activity periods for workdays are $T_{peak} = \{t_6, t_7, t_8, t_{12}, t_{16}, t_{19}, t_{20}, t_{21}, t_{22}\}$, and for weekends are $T_{peak} = \{t_6, t_7, t_8, t_{15}, t_{16}, t_{17}, t_{19}, t_{20}, t_{21}\}$.

(2) Determining Spatial Clustering Parameters

The core of spatial feature extraction is determining the parameters for the DBSCAN algorithm: the neighborhood radius ε and the minimum number of samples $MinPTs$ required for a core point. The value of ε should be close to the radius of the user's core activity areas. We used the k-distance graph to determine the ε value. By calculating the distance from each coordinate point to its k-th nearest neighbor, sorting these distances, and plotting the curve, the distance value at the curve's elbow point serves as a reference for ε. Setting $k = 4$, the resulting k-distance graph is shown in Fig. 7.

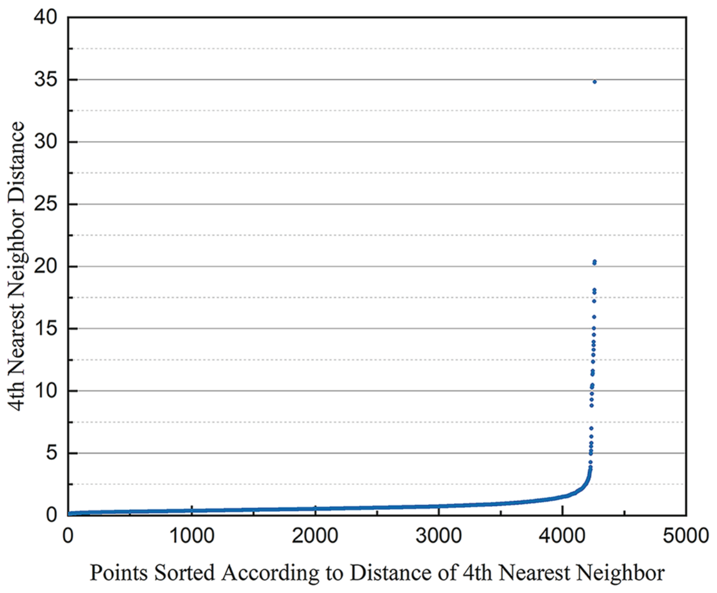


Figure 7: Determination of the spatial clustering parameter ε

Observing the k-distance graph, ε was ultimately determined to be 2.5 km. The parameter $MinPTs$ is used to distinguish core areas from noise. To avoid misclassifying valid small areas as noise, and referencing common check-in habits, $MinPTs$ was set to 4. This means a location must be visited no less than 4 times to be considered a valid area. Under these parameters, DBSCAN can effectively remove noise points and generate clusters with clear geographical meaning.

7.3 Comparison of Methods and Analysis of Results

This section compares and analyzes the proposed virtual probe deployment method against the Random Allocation with Random Traversal (Baseline-RR) and Random Allocation with Shortest Path Traversal (Baseline-RA) methods. The performance of the four methods (including the two baselines and the two proposed variants: Ours-S and Ours-BSA) is compared in terms of Probe Ban Rate (BR), Task Completion Rate (CR), and Traversal Time (TT).

For reliable results, each method was independently executed 10 times. Each experiment used 20 virtual probes and 5000 target coordinates (randomly generated within the 3 experimental areas). The probe movement speed was set to 60 km/h, and the single location switching time t_s was set to 10 s. The experimental results are presented as the average ± standard deviation of the 10 runs, as shown in Table 3 below.

Table 3: Comparison results of virtual probe deployment methods

Method	Probe ban rate (BR)	Task completion rate (CR)	Traversal time (TT)
Baseline-RR	30.50% ± 4.72%	69.20% ± 5.15%	72.52 ± 4.21 h
Baseline-RA	29.50% ± 4.93%	72.51% ± 4.37%	65.87 ± 4.07 h
Our-S	16.50% ± 3.73%	88.32% ± 2.92%	39.75 ± 2.15 h
Ours-BSA	11.50% ± 3.32%	93.20% ± 1.85%	43.82 ± 1.87 h

The Probe Ban Rate (BR) and Task Completion Rate (CR) directly reflect the method's stealth and reliability. As shown in Table 3, the two baseline methods, Baseline-RR and Baseline-RA, exhibit probe behavior patterns significantly different from real users due to their random allocation strategy, resulting in higher probe ban rates and lower task completion rates. Comparing Baseline-RR and Baseline-RA shows that local path optimization (TSP) has little effect on improving anti-detection capability. In contrast, our spatial clustering and load-balancing strategy (Ours-S) significantly optimizes probe movement patterns, reducing the ban rate to 16.50% and increasing the completion rate to 88.32%. This result confirms that mimicking user spatial aggregation is key to evading detection. Finally, the completely proposed method (Ours-BSA), which incorporates the time window mechanism, achieves the best performance, further reducing the ban rate to 11.50% and increasing the task completion rate to 93.20%. This shows that aligning probe activity with user temporal patterns, on top of spatial optimization, synergistically enhances behavioral stealth, achieving the highest task reliability.

The Traversal Time (TT) measures the overall time efficiency of the method. Baseline-RR is the least efficient (72.52 h). Baseline-RA reduces the time by about 9% (65.87 h) through local path optimization. The spatial strategy proposed herein (Ours-S) improves efficiency by about 45% through globally optimized path planning and load balancing, reducing the time to 39.75 h. Notably, the complete method (Ours-BSA) has a slightly longer total time (43.82 h) than Ours-S. This is expected, as the added time window constraints reduce daily effective working hours to enhance stealth, at the cost of a modest increase in total duration.

In summary, our spatial strategy yields substantial gains in both security and efficiency. The additional temporal behavioral camouflage further enhances stealth. Together, the spatio-temporal joint optimization strategy successfully achieves efficient, secure, and stable data collection.

8 Conclusion

To address the conflict between "efficient traversal" and "account security" in virtual probe deployment, this paper presents a method based on user behavior feature analysis. The core of this method is to quantitatively simulate normal users' spatio-temporal behavior. This simulation guides the design of an anti-detection scheduling strategy that aligns virtual probe behavior with real user logic, thereby evading anomaly detection. Experimental results demonstrate that the proposed method significantly outperforms the two baseline methods, Baseline-RR and Baseline-RA, across key metrics such as probe ban rate, task completion rate, and traversal time, providing a more reliable solution for virtual probe deployment. Despite its promising results, this study has limitations. For instance, the feature extraction is tailored to a specific application, and its generalizability requires further validation. Furthermore, the method's long-term adaptability must be tested against the dynamic evolution of platform detection strategies. Future work will explore cross-platform universal deployment frameworks and integrate adaptive mechanisms like reinforcement learning. This would enable the scheduling strategy to dynamically perceive and respond to changes in platform defenses, ensuring long-term stability and stealth.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the National Natural Science Foundation of China (No. U23A20305), and National Key Research and Development Program of China (No. 2022YFB3102900), Innovation Scientists and Technicians Troop Construction Projects of Henan Province, China (No. 254000510007), Key Research and Development Project of Henan Province (No. 221111321200).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Wenqi Shi, Bing Zhang; data collection: Bing Zhang; analysis and interpretation of results: Wenqi Shi, Bing Zhang; draft manuscript preparation: Bing Zhang. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data available on request from Bing Zhang.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Jiang B, Yao X. Location-based services and GIS in perspective. Comput Environ Urban Syst. 2006;30(6):712–25. doi:10.1016/j.compenvurbsys.2006.02.003.
2. Daraio E, Cagliero L, Chiusano S, Garza P. Complementing location-based social network data with mobility data: a pattern-based approach. IEEE Trans Intell Transp Syst. 2022;23(11):21216–27. doi:10.1109/TITS.2022.3182569.
3. Xiong X, Qiao S, Han N, Xiong F, Bu Z, Li RH, et al. Where to go: an effective point-of-interest recommendation framework for heterogeneous social networks. Neurocomputing. 2020;373(4):56–69. doi:10.1016/j.neucom.2019.09.060.
4. Canturk D, Karagoz P, Kim SW, Toroslu IH. Trust-aware location recommendation in location-based social networks: a graph-based approach. Expert Syst Appl. 2023;213(11):119048. doi:10.1016/j.eswa.2022.119048.
5. Shi W, Luo X, Guo J, Liu C, Liu F. Where are WeChat users: a geolocation method based on user missequence state analysis. IEEE Trans Comput Soc Syst. 2021;8(2):319–31. doi:10.1109/TCSS.2021.3049120.
6. Jiang H, Li J, Zhao P, Zeng F, Xiao Z, Iyengar A. Location privacy-preserving mechanisms in location-based services: a comprehensive survey. ACM Comput Surv. 2021;54(1):1–36. doi:10.1145/3423165.
7. Arshad A, Abbas H, Shahid WB, Azhar A. Deceiving eavesdroppers by real time persistent spoofing of android users' location coordinates for privacy enhancement. In: Proceedings of the 2020 IEEE 29th International Conference on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE); 2020 Sep 10–13; Bayonne, France. Piscataway, NJ, USA: IEEE; 2020. p. 107–12.
8. Bethi P, Dept E, Surathkal N. Stealthy GPS spoofing: spoofer systems, spoofing techniques and strategies. In: Proceedings of the 2020 IEEE 17th India Council International Conference; 2020 Dec 10–13; New Delhi, India. Piscataway, NJ, USA: IEEE; 2020. p. 1–7.
9. Shi Y, Luo X, Shi W. From crowds to coordinates: a user density-distance dynamic transformation method for telegram users geolocalization. Comput Netw. 2025;266(1):111316. doi:10.1016/j.comnet.2025.111316.

10. He W, Liu X, Ren M. Location cheating: a security challenge to location-based social network services. In: Proceedings of the 2011 31st International Conference on Distributed Computing Systems; 2011 Jun 20–24; Minneapolis, MN, USA. Piscataway, NJ, USA: IEEE; 2011. p. 740–9.
11. Dokuz AŞ. Anomalous activity detection from daily social media user mobility data. Omer Halisdemir Univ J Eng Sci. 2019;8(2):638–51. doi:10.28948/ngumuh.535232.
12. Dokuz AS. Social velocity based spatio-temporal anomalous daily activity discovery of social media users. Appl Intell. 2022;52(3):2745–62. doi:10.1007/s10489-021-02535-8.
13. Khan NU, Wan W, Yu S. Location-based social network's data analysis and spatio-temporal modeling for the mega city of Shanghai, China. ISPRS Int J Geo-Inf. 2020;9(2):76. doi:10.3390/ijgi9020076.
14. Cho E, Myers SA, Leskovec J. 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; 2011 Aug 21–24; San Diego, CA, USA: ACM; 2011. p. 1082–90.
15. Yang D, Qu B, Yang J, Cudre-Mauroux P. Revisiting user mobility and social relationships in LBSNs: a hypergraph embedding approach. In: Proceedings of the World Wide Web Conference; 2019 May 13–17; San Francisco, CA, USA: ACM; 2019. p. 2147–57.
16. Qin G, Patsakis C, Bouroche M. Playing hide and seek with mobile dating applications. In: Proceedings of the 29th IFIP TC 11 International Conference; 2014 Jun 2–4; Berlin/Heidelberg, Germany: Springer; 2014. p. 185–96.
17. Shi W, Luo X, Guo J, Liu F, Li L. An efficient geolocation method for malicious LBSD users based on dynamic adjustment of probes. Wirel Commun Mob Comput. 2022;2022(1):1–15. doi:10.1155/2022/4147498.
18. Ding Y, Peddinti ST, Ross KW. Stalking Beijing from Timbuktu: a generic measurement approach for exploiting location-based social discovery. In: Proceedings of the 4th ACM Workshop on Security and Privacy in Smartphones & Mobile Devices; 2014 Nov 7; Scottsdale, AZ, USA: ACM; 2014. p. 75–80.
19. Takahashi T, Tomioka R, Yamanishi K. Discovering emerging topics in social streams via link-anomaly detection. IEEE Trans Knowl Data Eng. 2014;26(1):120–30. doi:10.1109/TKDE.2012.239.
20. Ruan X, Wu Z, Wang H, Jajodia S. Profiling online social behaviors for compromised account detection. IEEE Trans Inf Forensics Secur. 2016;11(1):176–87. doi:10.1109/TIFS.2015.2482465.
21. Dutta HS, Chetan A, Joshi B, Chakraborty T. Retweet us, we will retweet you: spotting collusive retweeters involved in blackmarket services. In: Proceedings of the 2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM); 2018 Aug 28–31; Barcelona, Spain. Piscataway, NJ, USA: IEEE; 2018. p. 242–9.
22. Eswaran D, Faloutsos C, Guha S, Mishra N. SpotLight: detecting anomalies in streaming graphs. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018 Aug 19–23; London, UK. New York, NY, USA: ACM; 2018. p. 1378–86.