Towards Recurring Co-traveling Pattern Detection: A Summary of Results

Shuai An
an000033@umn.edu
University of Minnesota-Twin Cities
Minneapolis, Minnesota, USA

Chris Stanford
cstanford@novateur.ai
Novateur Research Solutions
Ashburn, Virginia, USA

Shashi Shekhar
shekhar@umn.edu
University of Minnesota-Twin Cities
Minneapolis, Minnesota, USA

Abstract

Given trajectories or location traces and user-specified thresholds, we investigate algorithms to detect recurring co-traveling patterns. For example, a school bus transports students between a neighborhood and a school. The problem is important for its societal applications in anomaly detection, synthetic trajectory and location trace data evaluation, and transportation planning. For example, deviations from recurring co-traveling groups naturally highlight anomalies, such as unexpected disruptions in commuting flows or rare co-traveling events. The problem is challenging due to the need to model recurring co-traveling routes and process an exponentially large number of candidate groups. Existing spatiotemporal data mining methods primarily focus on detecting co-occurrence relationships, but do not identify recurring co-traveling routes with specific travel areas. To overcome these limitations, we propose a novel recurring co-traveling group interest measure and Recurring Co-traveling Pattern Detection (RCPD) algorithms. We employ a divide-and-conquer method and spatial indices to improve computation efficiency. We also provide theoretical proofs that the proposed interest measure has the anti-monotone property, allowing early pruning, and the proposed algorithm is correct and complete. We evaluate our methods using real and synthetic trajectory/location trace data, as well as a case study on anomaly detection.

CCS Concepts

• **Information systems** → **Data mining**; **Geographic information systems**.

Keywords

geo-anomaly detection, spatiotemporal data mining, recurring co-traveling pattern, trajectory, location trace

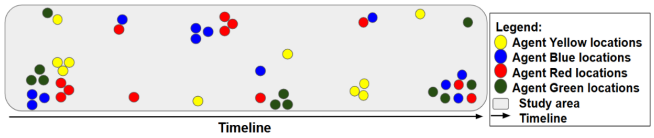
ACM Reference Format:
Shuai An, Chris Stanford, and Shashi Shekhar. 2025. Towards Recurring Co-traveling Pattern Detection: A Summary of Results. In *The 2nd ACM SIGSPATIAL International Workshop on Geospatial Anomaly Detection (GeoAnomalies '25), November 3–6, 2025, Minneapolis, MN, USA.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3764914.3770594

GeoAnomalies '25, Minneapolis, MN, USA

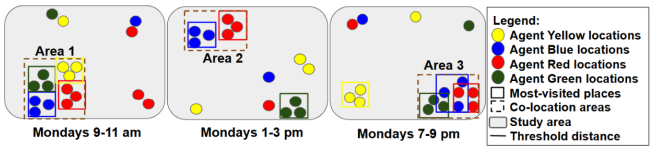
ACM ISBN 979-8-4007-2260-8/2025/11
https://doi.org/10.1145/3764914.3770594

1 Introduction

Given trajectory or location trace[1] data and user-specified thresholds, the recurring co-traveling pattern detection problem aims to find groups of agents who travel from one area to another during the same time windows and identify their recurring co-traveling routes. Each recurring co-traveling group co-travels in at least a threshold number of time windows, and the co-located area is small enough to be geographically meaningful. For example, a co-location in a city is less interesting than one in a business area. The recurring co-traveling agents do not necessarily know each other or travel simultaneously to the exact location. For example, college students travel by campus bus to various parts of the campus. Additionally, "recurring" means that the co-traveling occurs repeatedly, but not necessarily at fixed intervals. Figure 1 shows an example input of location traces (Figure 1a) and the corresponding output of Recurring Co-traveling Groups (RCGs) (Figure 1b). The agents' locations in Figure 1a are grouped by three time windows (Mondays 9-11 am, Mondays 1-3 pm, and Mondays 7-9 pm). For each time window, we find each agent's most-visited place (MVP) and group agents based on whether their MVPs are close to each other in an area. If we require an RCG to co-travel in at least two time windows, then agents Blue, Red, and Green in Figure 1b form an RCG that travels between (Mondays 9-11 am, Area1) and (Mondays 7-9 pm, Area3). If Area1 is a business area and Area3 is a residential area, this group could be a work-home recurring co-traveling group on Mondays.



(a) Input: Location Traces



(b) Output: RCGs

Figure 1: Example input and output of RCG detection

Detecting recurring co-traveling patterns has important societal applications, including but not limited to: 1) Anomaly detection.

[1]In this paper, a location trace is a sequence of logged raw location points with timestamps, such as logged GPS locations from cell phones. Data points of location traces are discrete and often sparse. A trajectory, by contrast, is a continuous/smooth movement path, which may contain other information like speed.

Recurring co-traveling patterns can serve as baseline normal expectations. By contrast, rare co-traveling groups may signal anomalies worth investigating, such as those that suddenly emerge, disappear, or change, co-traveling groups with irregular sizes or routes, or groups having unusual co-traveling frequencies/gatherings. Furthermore, the proposed algorithms can serve as baseline models of normal mobility, against which anomalies can be detected. Section 5.4 gives a detailed case study; 2) Synthetic trajectory/location trace data evaluation. RCG patterns can serve as evaluation metrics for synthetic trajectory/location trace data regarding trajectory/location trace interactions and agents' patterns of life, which are important properties for synthetic trajectory/location trace data in anomaly detection, privacy research, and location-based services [1, 2, 22]; 3) Transportation planning. RCG patterns can support efficient investment in transportation networks, such as directing funds into areas with high travel demands and designing public transit systems based on recurring co-traveling patterns.

Detecting RCGs has several challenges. First, modeling recurring co-traveling routes requires specifying small enough co-location areas, such as a neighborhood or a block, and short time windows, such as an hour, to be useful for societal applications. Also, an RCG can break into multiple subgroups following different recurring co-traveling routes, or different RCGs can merge into the same recurring co-traveling route in some time window. A second challenge is computation complexity. Trajectory/location trace data is usually massive due to its spatiotemporal nature. Also, we need to generate and process an exponentially large number of candidate groups, which results from multiple grouping dimensions, including agents, locations, and time windows. For example, detecting a seven-agent RCG that co-travels in every time window requires generating and processing 127 sub-group candidates (see Appendix A for details). Lastly, grouping agents based on their MVPs and time windows requires an extra spatial clustering step, such as running DBSCAN on every agent's visited locations in each time window.

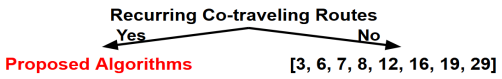


Figure 2: Comparison of Related Work

Most of the literature on spatiotemporal data mining has focused on detecting co-occurrence relationships, but it does not identify recurring co-traveling routes with specific travel areas, which are crucial for anomaly detection. Figure 2 shows a comparison with related work. The most similar methods we are aware of are MDCOP and MDCOPfast [8]. MDCOP algorithms divide spatiotemporal data into time windows and detect co-location patterns using the participation index in each time window. As the participation index is based on the neighbor relationships between instances in the study area as a whole, it only identifies co-location relationships but not specific common origins, destinations, or schedules. In addition, MDCOP/MDCOPfast does not require a minimum number of participating instances (which are recurring visits to a place in our problem). Thus, MDCOP/MDCOPfast is not capable of detecting recurring co-traveling routes. Other spatiotemporal co-occurrence mining methods [3, 6, 7, 12, 16, 19, 29] have the same limitations in identifying recurring co-traveling routes.

To overcome these limitations, we propose a novel recurring co-traveling group interest measure to model recurring co-traveling routes. We show that the interest measure has an anti-monotone property, allowing early pruning. We propose a Recurring Co-traveling Pattern Detection (RCPD) algorithm to correctly and completely detect recurring co-traveling groups. The algorithm divides timestamps into time windows. For each agent in each time window, the algorithm identifies the agent's MVP. Then, it detects agents' co-locations based on MVPs and detects recurring co-traveling routes based on the co-locations across time windows. Additionally, we employ a divide-and-conquer method and spatial indices to improve computation efficiency and propose a Recurring Co-traveling Pattern Detection Advanced (RCPD-Adv) algorithm.

We are also aware of the sensitive nature of trajectory/location trace data and incorporate additional privacy features into the proposed methods. We employ differential privacy [11] to add noise to input trajectories/location traces in both locations and timestamps. We utilize k-anonymity [23] on the output to require that a recurring co-traveling group contains at least K agents. Additionally, if the co-located area is too small, such as a building, it will be spatially cloaked by a larger area for privacy. Lastly, the algorithms provide options to convert latitude/longitude into relative (x, y) coordinates with a randomly picked reference point and de-identify agent and group IDs. Grouping agents' noise-added spatiotemporal data by time window (e.g., Saturday mornings) and region (e.g., a business area), and requiring a minimum number of agents in a group, increases the difficulty in identifying individual trajectories/location traces from the recurring co-traveling groups. Due to the privacy-utility tradeoff [15], these privacy features are optional, and users can decide on the level of privacy protection they require.

This paper makes the following contributions:

- We formalize the problem of recurring co-traveling pattern detection.
- We introduce a novel recurring co-traveling group interest measure and show its anti-monotone property.
- We propose a Recurring Co-traveling Pattern Detection (RCPD) algorithm and an advanced RCPD algorithm (RCPD-Adv) that uses a divide-and-conquer approach and spatial indices to reduce the candidate space.
- We prove the correctness and completeness of the proposed methods and evaluate them through experiments on real and synthetic data and an anomaly detection case study.

Scope: The algorithm outputs groups with recurring co-traveling routes containing time windows and co-located areas defined by minimum bounding rectangles. Interpretation of recurring co-traveling routes based on location types is not considered. We define user-specified thresholds and explore their effects on the output. We leave the choice of threshold values to domain experts and applications. This paper uses existing privacy protection methods, including k-anonymity and differential privacy, in the algorithm to protect individual privacy. Advancing privacy protection methods falls outside the scope of this paper. We do not have permission from the vendor (Veraset) to share the real data, but the synthetic data is provided.

2 Basic Concepts and Problem Formulation

In this section, we define basic concepts, formulate the interest measure, and present the formal problem statement.

2.1 Basic Concepts

We use the following basic concepts in our recurring co-traveling pattern detection formulation: 1) A ***visited location*** of an agent is a (x, y) point where the agent visits at a particular time. For example, the colored dots in Figure 1a are visited locations of those agents at the specific times on the timeline; 2) A ***threshold-largest-cluster-points (T_{LP})*** is the minimum number of visited locations the largest visited-location cluster needs to have to be kept after DBSCAN. This threshold ensures recurring visits of a place; 3) A ***threshold-area-upper (T_{AU})*** is the threshold minimum bounding rectangle (MBR) area that decides whether an MBR is small enough to be geographically meaningful and specific; 4) A ***threshold-area-lower (T_{AL})*** is the threshold MBR area that decides whether an MBR needs to be spatially cloaked for privacy. If an MBR is smaller than T_{AL}, we cover it with another MBR of random size in [T_{AL}, T_{AU}]. This threshold prevents an exact location from being inferred if agents are co-located in a tiny area, such as a building; 5) A ***time window (TW)*** is a partition of time within which an agent's visited locations are grouped, such as Mondays between 9 am and 11 am in Figure 1b; 6) A ***most-visited place (MVP)*** is the MBR of the largest cluster of an agent's visited locations in a time window. An MVP contains greater than or equal to T_{LP} visited locations, ensuring that the place is recurringly visited, and by extension, the co-traveling route is recurringly traveled. An MVP also needs to have an area in [T_{AL}, T_{AU}]. For example, the blue MBR in the first time window in Figure 1b is the MVP of agent Blue on Mondays between 9 am and 11 am. We only consider the MVP in a time window to find the most frequent patterns; 7) A ***threshold-distance (T_D)*** is the distance between two MVPs, deciding whether they co-locate; 8) A ***threshold-count-time-window (T_{CT})*** is the minimum number of time windows during which a group must co-travel repeatedly to be a recurring co-traveling group; 9) A ***gridded study area*** is a study area that is divided into grids of the same size. The number of grids is decided by the user-specified grid length and width and the data inferred study area. Figure 4a shows an example of a gridded study area; 10) ***Buffered grids*** are grids of a gridded study area to which x and y buffers have been added. The buffer sizes in meters of x and y coordinates are determined by T_D and the study area's x and y. Examples of buffered grids are shown in Figure 4b; 11) ***K*** is the minimum number of agents in a recurring co-traveling group to satisfy k-anonymity; 12) ***Differential privacy parameters***: ϵ is the privacy loss parameter. A smaller ϵ means stronger privacy and more noise being added. Sen_S is the spatial sensitivity deciding how far a location can be moved in meters. Sen_T is the temporal sensitivity deciding how many seconds a timestamp can be changed.

2.2 Interest Measure

Definition 1 Group co-located area (GCA): Given a time window t (TW_t) and a group (Gi) of agents, a GCA in TW_t is an MBR of agents' most-visited places (MVPs) in TW_t where distances between agents' MVPs are less than or equal to T_D, and the area of the MBR is less than or equal to T_{AU}, i.e., ($Distance(GCA_{Gi}, TW_t) <= T_D, Area(GCA_{Gi}, TW_t) <= T_{AU}$), where $Distance(GCA_{Gi}, TW_t) \equiv max(distance(MVP_a, MVP_b | a, b \in Gi))$. For example, Area1 in Figure 1b is a GCA of agents Blue, Red, Green, and Yellow.

Definition 2 Group recurring co-traveling route: A group recurring co-traveling route is represented as $\{(TW_t, GCA_t | TW_t \in time\ windows)\}$. For example, the group recurring co-traveling route of agents Blue, Green, and Red in Figure 1b is {(Mondays 9-11am, Area1), (Mondays 7-9pm, Area3)}.

Definition 3 Recurring co-traveling group (RCG): Given thresholds K, T_D, T_{AL}, T_{AU}, and T_{CT}, a group of agents (Gi) is an RCG if the group's cardinality is greater than or equal to K, each GCA is within the range [T_{AL}, T_{AU}] and the length of the group recurring co-traveling route is greater than or equal to T_{CT}, i.e.,

$$\text{Count}(t \in T)\left(\begin{array}{l}\text{Distance}(GCA_{Gi}, TW_t) \le T_D, \\ \text{Area}(GCA_{Gi}, TW_t) \in [T_{AL}, T_{AU}], \\ \text{Cardinality}(Gi) \ge K\end{array}\right) \ge T_{CT}$$

For example, in Figure 1b, if K=3, T_D = 10m, T_{AL} = 10% of the study area, T_{AU} = 25% of the study area, and T_{CT} = 2, then agents Blue, Green, and Red form a recurring co-traveling group of three agents and two recurring co-traveling time windows.

Definition 4 Maximal RCG: A maximal RCG is an RCG that is not contained in any larger RCGs. An RCG A is contained in another RCG B when the agents of A are a subset of agents of B, and for each recurring co-traveling time window, agents of A co-locate in a sub-area of B's GCA.

2.3 Formal Problem Statement

Given trajectory/location trace data and user-specified thresholds, the recurring co-traveling pattern detection problem aims to correctly, completely, and efficiently detect groups of agents who recurringly travel from one area to another in the same time windows. Differential privacy and k-anonymity are incorporated for privacy considerations.

Inputs: 1) Trajectories/location traces: (id, latitude, longitude, timestamp); 2) A list of interested time windows (TW); 3) User-specified thresholds: T_D, T_{AU}, T_{AL}, T_{CT}, T_{LP}, K, ϵ, Sen_S, Sen_T.

Output: Anonymized co-travelling groups which meet Definition 3: {groupID, cardinality, recurring co-traveling route: ([TW1, Area1], [TW2, Area2], ...)}.

Objectives: Computational efficiency.

Constraints: Completeness, Correctness, Privacy consideration.

Example: Suppose that we set K = 3, T_D = 10m, T_{AL} = 10% of the study area, T_{AU} = 25% of the study area, T_{CT} = 2, and T_{LP} = 3, then agents Blue, Green, and Red in Figure 1b form an RCG. This group recurringly co-travels from (Mondays 9-11 am, Area1) to (Mondays 7-9 pm, Area3). In each recurring co-traveling time window, the group's GCA is between 10% and 25% of the study area. Each agent's MVP has three or more visited locations. The pairwise distance between any two agents' MVPs is less than or equal to T_D.

Unlike traditional anomaly detection methods that focus on individual anomalies [10, 17], this problem identifies normal co-traveling groups and allows the definition of new group anomalies as deviations. Users can control thresholds to distinguish between normal and abnormal recurring co-traveling groups. For example, groups co-traveling in an abnormally high number of time windows (T_{CT}) or abnormally high density of appearance in a location (T_{LP})

would be signs of anomalies. Also, unusual time windows and locations of recurring co-traveling routes from the output are other signals of geo-anomalies.

3 Proposed Approach

In this section, we describe our two novel algorithms for detecting recurring co-traveling patterns, RCPD and RCPD-Adv. We also give execution traces of these algorithms. A detailed introduction of helper functions can be found in Appendix B.

3.1 RCPD

There are five steps to detect recurring co-traveling groups from trajectory/ location trace data: 1) Process raw trajectories/location traces and detect recurring one-agent groups (RCG_1); 2) Detect two-agent recurring co-traveling groups (RCG_2) from RCG_1; 3) Detect k-agent recurring co-traveling groups (RCG_k, k>=3) from RCG_2; 4) Refine RCGs to keep only maximal RCGs; 5) Output RCGs that have at least K agents in the group. Only randomly generated group IDs will be reported.

3.1.1 ***Detect RCG_1.*** The pseudocode of this step is in Algorithm 1. This step first processes raw trajectories/location traces by adding spatial and temporal noise at a user-specified level and converting them to a user-specified coordinate system (line 2). Then, it detects agents who have recurring traveling patterns. For each agent, we group visited locations into specified time windows. For each time window of each agent, we run DBSCAN on visited locations to obtain the largest cluster. If the largest cluster has at least T_{LP} visited locations (ensuring recurring visits on a co-traveling route), and the area of the largest cluster's MBR is less than or equal to T_{AU} (ensuring location specificity), then the largest cluster's MBR is the most-visited place (MVP) of the agent in that time window. If an agent has at least T_{CT} time windows that have MVPs, then this agent has a recurring traveling pattern and is an RCG_1.

Algorithm 1 Detect Recurring One-agent Groups (RCG_1)

Input: agents-trajectories/location traces, **TWs**, DBSCAN-eps (DE), DBSCAN-min-samples (DS), T_{LP}, T_{CT}, T_{AU}, ϵ, Sen_S, Sen_T, coordinates, algorithm-choice
Output: RCG_1: {agent-id: [(TW1, MVP1), (TW2, MVP2), …]}
1: **Algorithm:** Initiate **RCG_1** = empty dictionary
2: **agents-trajectories** = Process(**agents-trajectories**, ϵ, Sen_S, Sen_T, coordinates)
3: **if** algorithm-choice is "RCPD-Adv" **then**
4: **agents-trajectories** = *GetActiveAgents*(**agents-trajectories**, T_{LP}, T_{CT})
5: **for** (agent, visited-locations (VLs)) pair in **agents-trajectories do**
6: Initiate **agentMVPs** = empty list
7: **TW-VLs** = Group VLs by **TW**
8: **for** (TW, VLs) pair in **TW-VLs do**
9: MVP-mbr, MVP-mbr-area = *MostVisitedPlace*(VLs, DE, DS, T_{LP})
10: **if** MVP-mbr-area ≤ T_{AU} **then**: Add (TW, MVP-mbr) to **agentMVPs**
11: **if** *Size*(agentMVPs) >= T_{CT} **then**: Add (agent, agentMVPs) to **RCG_1**
12: **Return RCG_1**

3.1.2 ***Detect RCG_2.*** This step detects two-agent recurring co-traveling groups from RCG_1. By the anti-monotone property of the RCG measurement (which we define and prove in Appendix C), we only need to use RCG_1 groups to generate RCG_2 candidates as any two-agent group containing an agent who is not an RCG_1 is not an RCG_2. The pseudocode of this step is in Algorithm 2, lines 1-6 and 15. For each time window, the function *TwoAgentColocations* first generates two-agent combinations of RCG_1 groups that have MVPs in that time window. *TwoAgentColocations* then prunes two-agent combinations if the distance between the two agents' MVPs is greater than T_D or the MBR area of their MVPs is greater than T_{AU}. The remaining two-agent groups have group co-located areas (GCAs) in the given time window (line 5). Finally, two-agent groups having at least T_{CT} (time window, GCA) pairs in their recurring co-traveling routes are RCG_2 groups (line 15).

Algorithm 2 Detect Two-agent Groups (RCG_2)

Input: RCG_1, time windows (TWs), T_D, T_{AU}, T_{CT}, $step_y$, $step_x$
Output: RCG_2: {two-agent-group (TAG): [(time window (TW)1, co-location-area (CA)1), (TW2, CA2), …]}
1: **Algo: Initiate RCG_2 = empty dictionary**; **RCG_1** = *TransformRCG*(**RCG_1**, **TWs**)
2: **for** each (TW, **agentsMBRs**) pair in **RCG_1 do**
3: **if** algorithm-choice is "RCPD" **then**
4: **agents-ids** = get keys from the **agentsMBRs** dictionary
5: **co-locations** = *TwoAgentColocations*(TW, **RCG_1**, **agents-ids**, T_D, T_{AU})
6: **for** each (TAG, CA) in **co-locations do**: Add {TAG: (TW, CA)} to **RCG_2**
7: **if** algorithm-choice is "RCPD-Adv" **then**
8: Rtree-index, Rtree-bound = *RTree*(**agentsMBRs**)
9: **buffered-grids** = *GenerateBufferedGrids*(Rtree-bound, $step_y$, $step_x$, T_D)
10: **for** each buffered-grid in **buffered-grids do**
11: **agents** = *Query*(buffered-grid, Rtree-index)
12: **co-locations** = *TwoAgentColocations*(TW, **RCG_1**, **agents**, T_D, T_{AU})
13: **for** each (two-agent-group (TAG), CA) pair in **co-locations do**
14: Update {TAG: (TW, CA)} to **RCG_2** if not already in **RCG_2**
15: Return **RCG_2** = Keep TAG in **RCG_2** with *Size*(group co-traveling route) >= T_{CT}

Algorithm 3 Detect and Refine K-agent Groups (RCG_K)

Input: RCG_2, K, refine-RCG, T_{AL}, T_{AU}
Output: RCG_K and **$RefinedRCG_K$**
1: **Algorithm:** Initiate k=2; **RCG_K**=empty dictionary; currentRCG$_k$=RCG_2
2: **while** k > 0 **do**
3: RCG_{k+1} = *DetectNextK*(currentRCG$_k$) ▷ Algorithm 4
4: **if** RCG_{k+1} contains zero recurring co-traveling groups **then**
5: **RCG_K**: Keep RCG_k in **RCG_K** with k >= K
6: **Return RCG_K** = *Cloak*(**RCG_K**, T_{AL}, T_{AU})
7: **else**
8: Add RCG_{k+1} to **RCG_K**
9: k = k + 1; currentRCG$_k$ = RCG_{k+1}
10: **if** refine-RCG is "Yes": **then** Initiate **$RefinedRCG_K$** = empty dictionary
11: **for** each RCG_k in **RCG_K do**
12: containedRCG$_k$ = *FindContainedRCG*(RCG_k, RCG_{k+1})
13: maximalRCG$_k$ = remove containedRCG$_k$ from RCG_k
14: Add maximalRCG$_k$ to **$RefinedRCG_K$**
15: **Return $RefinedRCG_K$** = *Cloak*(**$RefinedRCG_K$**, T_{AL}, T_{AU})

3.1.3 ***Detect RCG_k.*** For RCG_k groups with k greater than or equal to three, we input RCG_2 into Algorithm 3, lines 1-9, to detect RCG_{k+1} groups iteratively (line 3) until no larger RCGs can be found (line 4). Line 6 cloaks any GCAs that are smaller than T_{AL} and returns RCG_K. The function *DetectNextK* in line 3 uses Algorithm 4, lines 1-9 and 19, to detect RCG_{k+1} groups from RCG_k groups. In Algorithm 4, for each time window, *GenerateCandidates* generates (k+1)-agent group candidates from RCG_k groups by matching the first (k-1) agents (using the interest measure's anti-monotone property). Then, *PruneCandidates* prunes the (k+1)-agent groups whose GCAs are greater than T_{AU}. The area thresholds ensure the group co-location in a time window is geographically meaningful and spatially cloaked for privacy. Finally, the remaining (k+1)-agent groups that have at least T_{CT} (time window, group GCA) pairs in their recurring co-traveling routes are RCG_{k+1} groups (line 19).

3.1.4 ***Refine and output RCG$_K$.*** After detecting all sizes of RCGs, we use Algorithm 3, lines 10-15, to keep only maximal RCGs. *FindContainedRCG* in line 12 finds RCG_k groups contained in some RCG_{k+1} groups by checking whether an RCG_k's group members are a subset of an RCG_{k+1}'s group members and whether the GCA of the RCG_k group is a sub-area of the GCA of the RCG_{k+1} group in each co-traveling time window. Line 13 removes the contained RCGs at all k levels, leaving only maximal RCGs. Finally, RefinedRCG_K groups having at least K agents in each group are returned.

Algorithm 4 Detect (k+1)-agent Groups (RCG_{k+1})

Input: RCG_k, time windows (TWs), T_{CT}, T_{AU}, algorithm-choice
Output: RCG_{k+1}: {(k+1)-agent-group: [(TW1, Area1), (TW2, Area2), ...]}
1: **Algorithm: Initiate RCG_k** = *TransformRCG*(**RCG_k**, **TWs**); i = 0; N_{TW} = *Size*(**TWs**); **RCG_{k+1}** = empty dictionary
2: **for** each (TW, **groupsMBRs**) pair in **RCG_k do**
3: i = i + 1
4: **if** algorithm-choice is "RCG" **then**
5: **if** i <= N_{TW} **then**
6: **generatedCan** = *GenerateCandidates*(**groupsMBRs**)
7: **prunedCan** = *PruneCandidates*(**generatedCan**, **groupsMBRs**, T_{AU})
8: **for** each (group, locationMBR) pair in **prunedCan do**
9: Update {group: (TW, locationMBR)} to **RCG_{k+1}**
10: **if** algorithm-choice is "RCPD-Adv" **then**
11: **generatedCan** = *GenerateCandidatesFast*(**groupsMBRs**)
12: **if** i <= (N_{TW} - T_{CT} + 1) **then**
13: **prunedCan** = *PruneCandidates*(**generatedCan**, **groupsMBRs**, T_{AU})
14: **if** (N_{TW} - T_{CT} + 1) < i <= N_{TW} **then**
15: **prunedCan** = *PruneFilterCan*(**generatedCan**, **groupsMBRs**, **R_{k+1}**, T_{AU})
16: **for** each (group, locationMBR) pair in **prunedCan do**
17: Update {group: (TW, locationMBR)} to **R_{k+1}**
18: **R_{k+1}** = Keep groups if *Size*(group co-traveling route) >= i - (N_{TW} - T_{CT})
19: Return **R_{k+1}** = Keep (k+1)-agent groups with *Size*(group co-traveling route)>=T_{CT}

3.1.5 ***An execution trace of RCPD.*** An example execution trace is shown in Figure 3 and Table 1. In this toy example, no trajectory pre-processing is needed. We use K = 3, T_{LP} = 3, T_{CT} = 2, T_{AU} = 20% of the study area, T_{AL} = 5% of the study area. There are five agents: Black (BK), Blue(B), Green (G), Red (R), and Yellow (Y). The agents' trajectories are first divided into four time windows. Each point in Figure 3 is an agent's visited location in that time window.

Step 1: Detecting RCG_1. For each time window (TW), we run DBSCAN on each agent's visited locations and keep each agent's largest cluster. For each agent, if the largest cluster has at least three visited locations, and the area of the largest cluster's MBR is less than or equal to T_{AU}, then the largest cluster's MBR is the most-visited place (MVP) of the agent in that time window. In Figure 3, the MVPs of BK, R, G, B, and Y are labeled using rectangles with their corresponding colors. In TW2, Y's largest cluster is pruned because it has only two visited locations. In TW3, Y's largest cluster is also pruned because the MBR area exceeds T_{AU}. Table 1a shows the MVPs of each agent in each time window. T (True) means an MVP was found. F (False) means an MVP was NOT found. N (None) refers to cases that do not exist or were pruned earlier. Labels T/N and F/N reflect procedures that are implemented in RCPD but are not needed in RCPD-Adv. BK is pruned as BK has only one time window (TW2) having an MVP, which does not satisfy the T_{CT} requirement. Agents B, G, R, and Y have MVPs in two or more time windows, and thus qualify as RCG_1 groups.

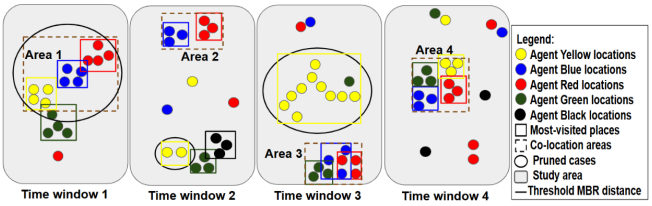


Figure 3: Execution trace

Step 2: Detecting RCG_2. In Table 1a TW1, four RCG_1 groups B, G, R, and Y have MVPs. By combination, we get six candidate two-agent groups: (B, G), (B, R), (B, Y), (G, R), (G, Y), and (R, Y). Group (B, G) is pruned because the distance between B's MVP and G's MVP is greater than T_D, the threshold MBR distance. Similarly, (G, R) and (R, Y) are pruned. Since the areas of (B, R), (B, Y), and (G, Y) are less than or equal to T_{AU}, we find three two-agent co-locations in TW1. Following the same procedure for the rest of the time windows, we get Table 1b. T means a co-location is found between the agents in the time window. F means no co-location is found. Labels N, T/N, and F/N have the same meaning as in Step 1. "Yes/RE" means this group is an RCG_2 but will be pruned after the refinement (discussed in Step 4). Table 1b shows that there are five RCG_2 groups. (R, Y) is pruned because it only co-locates in one time window (TW4).

Table 1: Results of execution traces. TW: time window. Qual. #: Qualified number. T (True): an MVP/co-location was found. F (False): no MVP/co-location was found. N (None): a case that does not exist or was pruned earlier. Labels T/N and F/N reflect procedures that are implemented in RCPD but are not needed in RCPD-Adv. RE: being refined.

(a) Results of one-agent group detection

Groups	TW1	TW2	TW3	TW4	Qual. # of TWs	RCG_1
Black (BK)	F/N	T/N	F/N	F/N	1	No
Blue (B)	T	T	T	T	4	Yes
Green (G)	T	T	T	T	4	Yes
Red (R)	T	T	T	T	4	Yes
Yellow (Y)	T	F	F	T	2	Yes

(b) Results of two-agent group detection

Groups	TW1	TW2	TW3	TW4	Qual. # of TWs	RCG_2
(B, G)	F/N	F/N	T	T	2	Yes/RE
(B, R)	T	T	T	T	4	Yes
(B, Y)	T	N	N	T	2	Yes
(G, R)	F/N	F/N	T	T	2	Yes
(G, Y)	T	N	N	T	2	Yes
(R, Y)	F/N	N	N	T/N	1	No

(c) Results of three-agent group detection

Groups	TW1	TW2	TW3	TW4	Qual. # of TWs	RCG_3
(B, G, R)	N	N	T	T	2	Yes
(B, G, Y)	N	N	N	T/N	1	No
(B, R, Y)	F	N	N	T/N	1	No
(G, R, Y)	N	N	N	T/N	1	No

Step 3: Detecting RCG_3. RCG_3 candidates are generated by merging RCG_2 groups that have the same first agent. For example, (B, G) and (B, R) are merged to generate candidate (B, G, R) in TW3, as they share the same B as the first agent. Since (B, G, R)'s GCA is less than or equal to T_{AU}, (B, G, R) forms a three-agent co-location in

TW3. Note that the distance requirement between any two agents' MVPs is automatically satisfied because any two agents of (B, G, R) form an RCG_2. Following the same procedure for the other time windows, we obtain Table 1c. Note that (B, R, Y) in TW1 is pruned because the area of the merged group exceeds T_{AU}. From Table 1c, we find that (B, G, R) is the only RCG_3 that travels from Area3 in TW3 to Area4 in TW4.

Step 4: Refinement. Since K=3, RCG_3 is the only RCG and the largest group found in this example; thus, no refinement is needed. However, if we choose K=2, (B, G) is a qualified RCG_2 but will be refined because it is contained in (B, G, R). Group (B, G) is a subset of (B, G, R) with the same recurring co-traveling time windows (TW3 and TW4). For each recurring co-traveling time window, (B, G)'s GCA is a subset of (B, G, R)'s GCA. Thus, (B, G) is contained in (B, G, R) and is labeled "RE" in Table 1b.

Step 5: Output. Since K=3, we only output groups with at least three agents. The output will only contain randomly generated group IDs. In this example, the output is {GroupX, three agents, [(TW3, Area3), (TW4, Area4)]}.

3.2 RCPD-Adv

RCPD-Adv follows the same steps as RCPD while incorporating extra efficiency strategies in Steps 1, 2, and 3.

3.2.1 ***Efficiency strategy for detecting RCG_1.*** Before running DBSCAN on each agent's visited locations in each time window to find MVPs, we only keep agents who have at least T_{CT} time windows containing at least T_{LP} visited locations. These are weaker constraints than the requirements of RCG_1, which prune unqualified agents earlier to reduce the number of expensive DBSCAN executions. This strategy is implemented by the *GetActiveAgents* function in line 4 Algorithm 1.

3.2.2 ***Efficiency strategy for detecting RCG_2.*** We noticed that finding two-agent co-locations by combinations of all agents in the whole study area (Algorithm 2, lines 1-6) generates an exponential number of candidates $\binom{N}{2}$, where N is the total number of RCG_1, and many of them have agents far from each other. For example, in Figure 4a, generating the (Blue, Green) candidate co-location and checking the agents' distance is unnecessary because Blue and Green are not even in the same area. Thus, we propose a divide-and-conquer method (Algorithm 2, lines 7-14) to divide the study area into buffered grids and generate two-agent co-locations within each buffered grid. We also generate an R-tree index for the minimum bounding rectangles (MBRs) of agents in each time window, allowing for faster queries of agents within each buffered grid (line 11). The *GenerateBufferedGrids* function divides the study area into grids with specified x and y coordinate step sizes. It then adds a buffer to each grid by (2 * T_D). Using buffered grids ensures that co-locations at the boundary between two grids are not missed. For example, in Figure 4a, co-location (Blue, Red) will be missed if regular grids are used. However, using buffered grids (Figure 4b) with a buffer size larger than T_D will include the edge cases. Using buffered grids may generate duplicate candidates, as seen in the example in Figure 4b, for (Blue, Red) in buffered-grids 1 and 2. Line 14 of Algorithm 2 guarantees that each co-location will only be counted once.

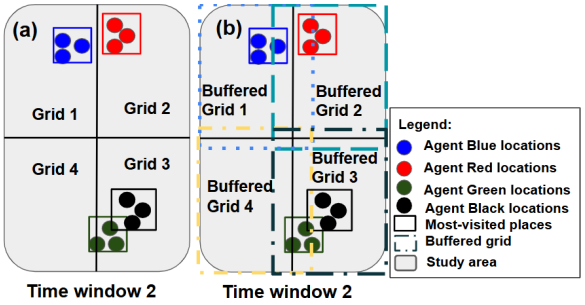


Figure 4: Examples of grids and buffered grids

3.2.3 ***Efficiency strategy for detecting RCG_k.*** Algorithm 4 lines 10-19 show the more efficient way to detect RCG_{k+1} from RCG_k groups. *GenerateCandidateFast* generates (k+1)-agent RCG candidates using the anti-monotone property and a divide-and-conquer strategy. Specifically, the function divides k-agent groups into partitions by their first (k-1) agents, generating (k+1)-agent candidates within each partition. We also adopted and improved a strategy from [8] to prune unqualified candidates as early as possible: After processing the $(N_{TW} - T_{CT} + 1)$th time window (lines 10-13), *PruneFilterCan* (line 15) will filter out newly appearing candidates because the remaining time windows are insufficient for them to qualify. Also, at each remaining iteration i, line 18 will prune candidates who currently do not have at least $(i - (N_{TW} - T_{CT}))$ recurring co-traveling time windows, which means their recurring co-traveling time windows will not meet the T_{CT} threshold, even if the unprocessed time windows would all be qualified (a numerical example is in Appendix D).

3.2.4 ***RCPD-Adv efficiency strategy execution traces.*** In Step 1, BK is pruned as it only has TW2 with at least three visited locations. Thus, RCPD-Adv eliminates the need for BK to go through the DBSCAN procedure, and the BK row in Table 1a is labeled as "/N." In Step 2, the study area in each time window is divided into four buffered grids. Take TW2 in Figure 4b as an example: querying on buffered grid 1 gives us B and R, generating the RCG_2 candidate (B, R). Similarly, querying on buffered grid 2 generates a candidate (B, R). Buffered grids 3 and 4 do not find candidate RCG_2 groups (BK is pruned in Step 1). After collecting candidates from all buffered grids and removing duplicates, only (B, R) is left. Thus, using the RCPD-Adv algorithm in TW2 eliminates the need to generate and process (B, G) and (G, R) from the start. These groups are now labeled "/N" in Table 1b, column TW2. The same buffered-grid procedure eliminates the need to generate and process (B, G), (G, R), and (R, Y) in TW1 and (R, Y) in TW4. In Step 3, groups (B, G, Y) and (G, R, Y) will not be generated in the first place because they do not appear before TW4, and their remaining time windows (TW4) are insufficient for them to qualify. Group (B, R, Y) will be generated in TW1 but will be pruned in TW3 because there is only one time window left, and the candidate currently has zero qualified time windows. Even if the last time window would qualify, this group would not meet the requirement of T_{CT}. Thus, only (B, G, R) will be processed and investigated in TW4. Group (B, G, R) is an RCG_3. Steps 4 and 5 are the same as for RCPD.

In the example we have presented, RCPD-Adv saves 20% of procedures in Step 1, 33% of procedures in Step 2, and 50% of procedures

in Step 3. As the dataset size increases, RCPD-Adv becomes more efficient by orders of magnitude.

4 Theoretical Evaluation

Theorem 1. *Without noise distortion, the RCPD and RCPD-Adv algorithms are complete.*

Proof. The RCPD and RCPD-Adv algorithms are complete if they find all RCGs that satisfy the threshold conditions. When no noise is added, no true patterns will be distorted and missed. We prove this by showing that all steps of the algorithms do not miss any RCGs. *Step 1 does not miss any RCG_1:* Only agents who do not meet the threshold requirements (T_{LP}, T_{AU}, T_{CT}) are removed. Also, *GetActiveAgents* prunes agents early using less strict constraints than the RCPD requirements. *Step 2 does not miss any RCG_2:* RCPD uses the anti-monotone property to generate combinations of all RCG_1 agents and prunes each candidate by the thresholds. No more candidates can exist. RCPD-Adv uses buffered grids with a buffer size of (2 * T_D) to include all edge co-locations. The candidates are only pruned by the RCPD thresholds. *Step 3 does not miss any RCG_k:* RCPD and RCPD-Adv use the anti-monotone property to generate candidates, which includes all cases. RCPD identifies qualified time windows for each agent and prunes agents based on T_{CT}. Thus, no group is missed. RCPD-Adv prunes candidates early only when they do not have enough qualified time windows, even if the remaining time windows all qualify. Thus, no group is missed. *Steps 4 and 5 do not miss any RCGs:* Step 4 simply removes RCGs that are contained in larger RCGs. Step 5 simply removes groups with less than K agents and cloaks GCAs that are smaller than T_{AL}.

Theorem 2. *Without noise distortion, the RCPD and RCPD-Adv algorithms are correct, i.e., any returned RCG satisfies Definition 3.*

Proof. In Step 1, T_{LP}, T_{AU}, and T_{CT} are checked for each candidate to detect RCG_1. In Steps 2 and 3, T_D, T_{AU}, and T_{CT} are checked for each candidate to detect RCG_k, k >= 2. In Step 5, K and T_{AL} are checked for each candidate to output RCG_K. Thus, the returned RCGs meet the requirements of Definition 3.

5 Experimental Evaluation

Experiment goals: We validated the proposed methods with four types of analysis: 1) Algorithm Testing. We generated synthetic trajectory data to test the accuracy, recall, and precision of the proposed methods. We also tested how the added noise level ϵ influences the model performance. 2) Comparative analysis. We compared our proposed methods with existing co-occurrence mining algorithms to highlight the efficiency advantages of RCPD and RCPD-Adv, as well as the fundamental differences in the patterns they detect. We also compared the runtimes between RCPD and RCPD-Adv to show the effectiveness of our efficiency strategies. 3) Sensitivity analysis. We performed a sensitivity analysis on our proposed methods to evaluate the influence of the number of agents, threshold-count-time-window (T_{CT}), and threshold-largest-cluster-points (T_{LP}) on runtime, number of recurring co-traveling groups detected, and size of the largest recurring co-traveling group. 4) Case study. We conducted a case study using real location traces from Los Angeles County in the US to demonstrate how the proposed algorithms can be applied to geo-anomaly detection. The implementation of these evaluations is available online[2].

Dataset: We used a subset of the Veraset mobility dataset within the MBR of Los Angeles County in the US between February 5 and February 26, 2019, for the comparative analysis, sensitivity analysis, and the case study. This data contained anonymized IDs and location information (latitude, longitude, timestamp) from mobile devices. To protect privacy, we randomly selected a reference point within the MBR and converted the (longitude, latitude) coordinates into relative coordinates (x, y) in meters. The data contained 20K agents, and each agent had more than 30 (x, y, timestamp) pairs within this period and region. There were a total of 15.5 million location-time pairs. On average, each agent had 775 location-time pairs. The median number of location-time pairs was 161. These location traces were divided by weekend and weekday, and by 24 hours, resulting in 48 time windows. We received approval from our university's Institutional Review Board to conduct this research using anonymized real trajectory data, including the specific anonymity methods employed to protect participant privacy.

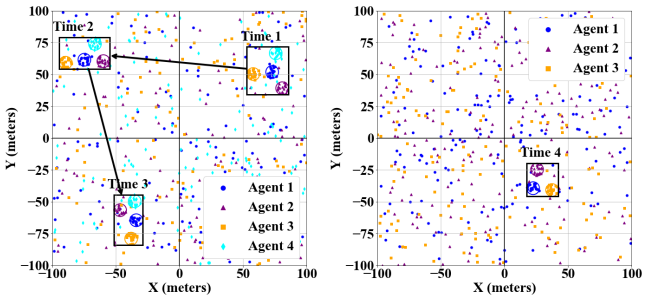
5.1 Algorithm Testing

We generated synthetic trajectories to test the accuracy, recall, and precision of the proposed algorithms. We used four time windows and three group sizes (2, 3, and 4-agent groups). Thus, for each group size, a group could co-locate in zero to four time windows, resulting in 15 patterns, such as a 3-agent group co-locating in one time window or a 4-agent group co-locating in three time windows. For each pattern, we generated 100 groups, resulting in 1500 test cases, 500 cases for each group size. This gave us $2 * 500 + 3 * 500 + 4 * 500 = 4500$ agents in total. Each test case was generated in a 200m X 200m area. We set the threshold-count-time-window (T_{CT}) equal to two. Thus, only groups co-locating in two to four time windows were considered recurring co-traveling groups (900 positive cases). Figure 5 shows two examples of these test cases. In Figure 5a, a synthetic four-agent group recurringly co-travels from Time1-Area1 to Time2-Area2 to Time3-Area3. By contrast, the three-agent group in Figure 5b only co-locates in Time4-Area4, making it a non-co-traveling group. To generate a group co-located area (GCA), we randomly generated agents' anchor points within 15m. For each anchor point, we generated a dense circle with 5m radius as the agent's most-visited place (MVP). The MBR of these dense areas is the group's GCA. Varying the number of agents and time windows to generate GCAs resulted in 1500 test cases.

Feeding each test case to RCPD returned one of four possible results: 1) A true positive (TP), where the test case contained a recurring co-traveling group and the detected pattern had the same agents co-traveling in the same time windows. Also, each MVP detected was less than five meters away from the true MVP; 2) A true negative (TN), where no recurring co-traveling patterns were detected and the test case was a negative case; 3) A false positive (FP), where one or more recurring co-traveling groups were found when the test case was a negative case; 4) A false negative (FN), where a true recurring co-traveling group in a test case was not detected. We used accuracy ($(TP + TN)/(TP + TN + FP + FN)$), precision ($TP/(TP+FP)$), and recall ($TP/(TP+FN)$) to measure the testing results. We also varied the differential privacy noise level ϵ

[2]Our code: https://github.com/ShuaiAn7/Recurring-Co-traveling-Pattern-Detection

to show how noise affects model performance. The results in Figure 6 show that the model is complete and correct when no noise is added. The model achieves high precision scores across various noise levels, indicating that the probability of detecting spurious patterns is low. This is due to the pattern complexity, which makes it difficult to pass all threshold controls for spurious patterns. As the noise level increases, the accuracy and recall scores decrease, driven by the increasing number of false negatives. This is because noise dissolves MVPs into the background. Thus, users can decide the noise level based on the accuracy and recall requirements of their projects. Also, the runtime increases when noise is added.



(a) A Synthetic RCG_4 with the Recurring Co-taveling Route **(b) A Synthetic Non-traveling 3-agent Co-location**

Figure 5: Examples of Synthetic Patterns

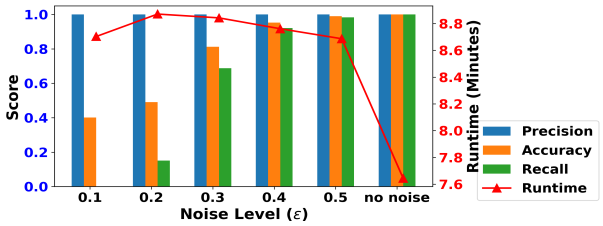


Figure 6: Synthetic Data Testing Results

5.2 Comparative Analysis

As there is a lack of existing methods that detect recurring co-traveling routes like ours, we compared our approaches with the most related existing spatiotemporal co-occurrence mining methods we are aware of. We chose MDCOP and MDCOPfast [8] because they detect subsets of features/agents that co-occur across time windows in a way most comparable to ours, and their limitations are representative of those found in other related methods [6, 16, 29]. Agent groups detected by MDCOP/MDCOPfast may partially overlap with agent groups from our methods if some agents' visited locations are concentrated in a small area. To implement MDCOP/MDCOPfast, we treated each agent in our data as a feature and each visited location as an instance to calculate participation indices in each time window. During the experiment, we observed that MDCOP/MDCOPfast could not process location traces of 20K agents within a reasonable time. To make the results comparable, we randomly selected 1,000 agents for this analysis. Shared thresholds were set the same. The results are shown in Table 2. The MDCOP approach detected six two-agent co-occurrence groups. We examined those groups and found that they did not have MVPs in each time window. Thus, no co-traveling routes existed, and they were not recurring co-traveling groups, as defined in our study. Consistently, the RCPD approach did not detect any recurring co-traveling groups either. In addition, runtimes of RCPD/RCPD-Adv are roughly 0.07% of MDCOP/MDCOPfast's runtimes, showing better scalability of the proposed approaches. We also noted that MDCOPfast actually ran slightly slower than MDCOP, which may have been due to the higher computation overhead in location trace data. By contrast, RCPD-Adv ran faster than RCPD even with more overhead computation from this small dataset.

Table 2: Comparison of Different Algorithms

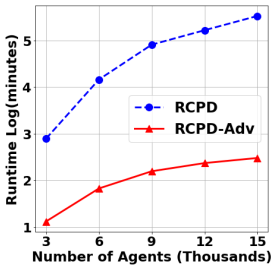
Algo	Runtime (minute)	Results
MDCOP	1124.6853	Six 2-agent groups; No co-traveling groups
MDCOPfast	1141.656	Six 2-agent groups; No co-traveling groups
RCPD	0.8833	No co-traveling groups
RCPD-Adv	0.6821	No co-traveling groups

Essentially, RCPD/RCPD-Adv detected patterns differently from MDCOP/MDCOPfast. In the co-location detection step, MDCOP/MDCOPfast considers all instances within a study area in a time window. They focus on whether two visited locations are co-located and ignore where in the study area these locations co-locate. By contrast, RCPD/RCPD-Adv focuses on the concentration of visited locations in a specific area and ignores the other scattered locations. Also, the T_{LP} threshold in RCPD requires a minimum number of visited locations participating in co-locations, while the MDCOP approach does not have this requirement. Thus, RCPD/RCPD-Adv patterns are recurring co-traveling groups containing both co-located times and specific GCAs (specified by MBRs), while MDCOP/MDCOPfast patterns are global co-occurrences in the study area as a whole with no required minimum number of participating visited locations. Considering runtime and output, MDCOP/MDCOPfast is more suitable for global spatiotemporal co-occurrences with a small number of features and instances and without the need to identify recurring co-traveling routes. RCPD/RCPD-Adv is more suitable for processing extensive trajectory/location trace data and a large feature set to detect recurring regional patterns. A more detailed discussion of the differences between the two methods is provided in Appendix E using illustrative examples.

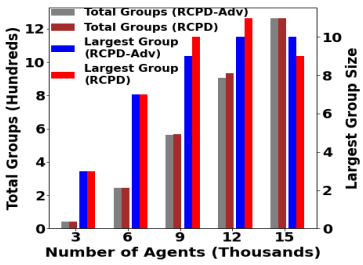
5.3 Sensitivity Analysis

We performed a sensitivity analysis on our proposed methods to evaluate the influence of the number of agents, threshold-count-time-window (T_{CT}), and threshold-largest-cluster-points (T_{LP}) on runtime, number of recurring co-traveling groups (RCGs) detected, and size of the largest RCG. We plotted the runtime in its natural log form because the runtime difference between RCPD and RCPD-Adv is by orders of magnitude. Taking natural logs makes comparing two curves in the same figure easier. Also, in all experiments, RCPD and RCPD-Adv output exactly the same RCGs when no noise was added (Appendix F). Here, we provide results with the noise level at $\epsilon = 0.5$, Sen_S=5m, and Sen_T=100s. The noise slightly reduced accuracy and caused slight discrepancies in results between RCPD and RCPD-Adv, but the sensitivity patterns remained the same.

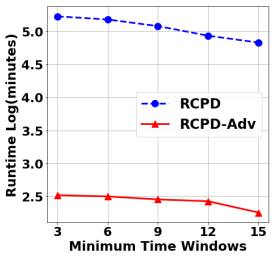
In the first experiment, we fixed $T_{CT} = 3$ and $T_{LP} = 5$ and varied the number of agents from 3K to 15K. Figure 7a shows that the



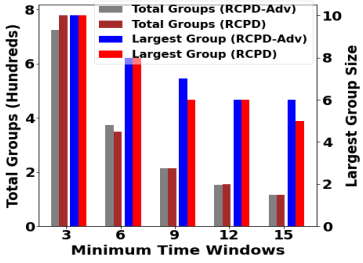
(a) Runtime (varying number of agents)



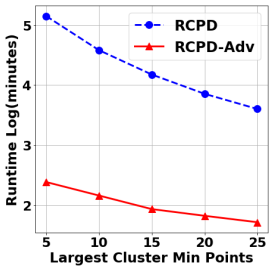
(b) Groups Found (varying number of agents)



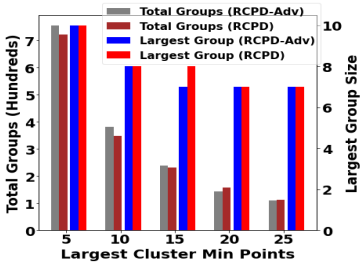
(c) Runtime (varying T_{CT})



(d) Groups Found (varying T_{CT})



(e) Runtime (varying T_{LP})



(f) Groups Found (varying T_{LP})

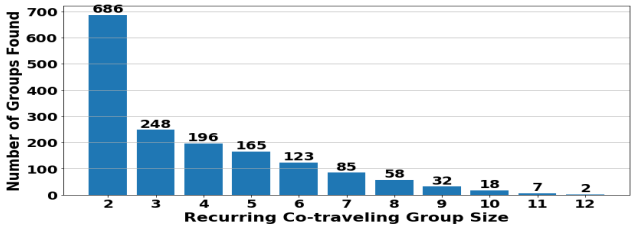
Figure 7: Sensitivity analysis on runtime and RCG count under varying conditions: number of agents, threshold count time window (T_{CT}), and threshold largest cluster points (T_{LP}).

runtimes of RCPD and RCPD-Adv increase at a slower rate as the number of agents increases. This indicates that the algorithms have a good scaling property and can handle larger datasets without an exponential runtime increase. In addition, the margin between the two lines is increasing, indicating that RCPD-Adv gets progressively more efficient than RCPD as the data size increases. Figure 7b shows that as the number of agents increases, the total number of RCGs increases linearly while the size of the largest RCG increases at a slower rate.

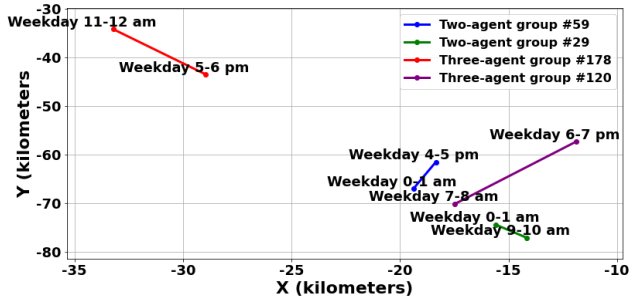
For the effect of T_{CT}, we fixed the number of agents = 10K and T_{LP} = 5, and varied T_{CT} from 3 to 15. Figure 7c shows that both algorithms have almost linearly decreasing log runtime, meaning that the runtime decreases exponentially as the T_{CT} increases. RCPD-Adv remains significantly more efficient than RCPD by orders of magnitude and exhibits good scalability. T_{CT} is the minimum number of required qualified time windows of an RCG. A higher T_{CT} means a stricter requirement for detecting RCGs. Thus, as T_{CT} increases, the number of generated candidates decreases at an accelerating rate, while the number of pruned candidates increases at an accelerating rate, resulting in an exponentially shorter runtime. The number of detected RCGs and the largest RCG size also decrease at a slower rate, as shown in Figure 7d.

Lastly, we tested the effect of T_{LP}. We fixed the number of agents = 10K and T_{CT} = 3, and varied T_{LP} from 5 to 25. Figure 7e shows two convex decreasing lines, meaning the runtime decreases super-exponentially as the threshold increases. RCPD-Adv remains significantly more efficient than RCPD by orders of magnitude and exhibits good scalability. A larger T_{LP} value means a stricter requirement for a cluster to be an MVP. Thus, fewer candidates will be generated, and more candidates will be pruned with exponentially shorter runtime. The total number of detected RCGs and the largest RCG size decrease at a slower rate, as shown in Figure 7f.

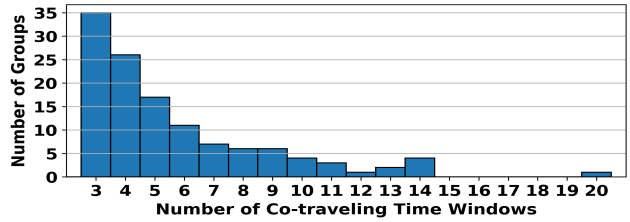
5.4 Case Study



(a) Recurring Co-traveling Groups (RCGs) Found



(b) Examples of Recurring Co-traveling Routes



(c) Distribution of Co-traveling Time-windows of 6-agent RCGs

Figure 8: Case Study in Los Angeles, USA

Our case study had two goals: The first goal was to show how recurring co-traveling patterns can be detected from real-world location traces. The second goal was to show how anomalies naturally arise as rare cases, deviations from the norms, or by controlling user-specified thresholds. We used location traces of 20K agents from the Veraset mobility dataset within the MBR of Los Angeles County in the US between February 5 and February 26, 2019. For

further privacy protection, we used relative coordinates with a randomly selected reference point. For a simpler demonstration, we did not add noise to these location traces and did not restrict the number of agents in a group. We set T_{CT}=3 and T_{LP}=5.

We ran RCPD-Adv on the data and got the RCGs shown in Figure 8a. The total runtime was 13.5 minutes. There were 1620 RCGs found, and the largest RCGs had 12 agents. The number of groups found decreases at a slower rate as the group size increases. Figure 8b plots examples of the routes co-traveled by two and three-agent groups. For example, we can see that the three-agent group #120 co-travels on weekdays between 7-8 am and 6-7 pm in a recurring pattern. If we use latitude-longitude and location type (e.g., residential area, business area), we can infer the type of RCG, such as a work-home commuting group.

Geo-anomalies can be identified as rare cases or deviations from the norms. For example, Figure 8c shows the distribution of the number of co-traveling time windows of 123 six-agent RCGs. Among the 123 groups, 122 have less than 15 co-traveling time windows, and only one group has 20 co-traveling time windows. In addition, the co-traveling route indicates that during 20 time windows, the six agents co-located in the same area of 382 m^2, which is unusual. Thus, this group is a potential anomaly for further investigation. Users can also detect anomalies by controlling thresholds. For example, we can set T_{LP}=50 and T_{CT}=20 to detect co-traveling groups that appear abnormally frequently in group co-located areas and co-travel in a very high number of time-windows. Considering the sparsity of location trace data, these groups are unusual. After rerunning the algorithm with the anomaly thresholds, we identified four 2-agent RCGs and two 3-agent RCGs as potential geo-anomalies.

6 Related Work

Shekhar and Huang [21] introduced the concept of spatial co-location and its interest measures, the participation ratio and participation index, to find spatial features that frequently locate together. Later refinements, e.g., [27, 30], improved the computational efficiency of co-location mining. Today, regional co-location algorithms [9, 18] detect subsets of features that co-locate in certain localities in a study area. Statistically significant co-location algorithms were proposed to reduce chance patterns [4, 14]. However, these spatial co-location methods focus on features that do not move frequently, such as business locations, or that occur in a single time period, such as a day. Thus, they are not applicable to detect moving patterns. One recently proposed method [5] to discover co-location patterns in flow data was able to detect agent groups co-located in both trip origins and destinations. However, this method also does not incorporate time into its mining process and cannot detect recurring co-traveling routes.

A number of co-occurrence mining algorithms have been developed to detect co-locations across time in spatiotemporal data. Of these, MDCOP algorithms [8] are the most related to our problem. The MDCOP methods divide spatiotemporal data into time windows and detect co-location patterns using the participation index in each time window. In other words, they consider all instances occurring in the entire study area for each time window without requiring a minimum number of participating instances. Essentially, the methods indicate whether two agents co-locate. However, they cannot specify where in the study area co-locations occur in a recurring manner. Indeed, an extensive review of the literature [3, 6, 7, 12, 13, 16, 19, 20, 24–26, 28, 29] revealed no methods of spatiotemporal co-occurence mining without this limitation. For example, another recent study [29] proposed an incremental spatiotemporal flow co-location quotient to detect spatiotemporal associations between two types of flow. The method treats each origin-destination (OD) flow as an instance and calculates the spatiotemporal distance between two flows. As OD flows from different features can co-occur across the entire study area, this method still has the same limitation as MDCOP regarding recurring co-traveling pattern detection. To our knowledge, ours is the only method that can detect concentrations of visited places in a study area, and by extension, the specific routes recurringly traveled by agents.

7 Conclusion and Future Work

We formalized the problem of recurring co-traveling pattern detection and proposed a recurring co-traveling group interest measure, along with two accompanying algorithms – RCPD and RCPD-Adv. The algorithms provide options for incorporating differential privacy and k-anonymity. We theoretically proved the correctness and completeness of the proposed algorithms. Experiments on real-world location trace data show that RCPD runs faster than the baseline and can identify co-traveling routes that the baseline cannot. RCPD-Adv also runs faster than RCPD by orders of magnitude. Case study results demonstrate that our approach can be applied to geo-anomaly detection.

In future work, we plan to further improve computation efficiency by exploring new ideas, such as dynamically breaking buffered grids with a large number of agents into smaller ones. We also plan to use sliding time windows and dynamic time-window length to improve temporal precision. We plan to do more thorough experiments using more realistic synthetic data. As there is a lack of literature on recurring co-traveling route detection, we will explore new evaluation metrics for comparative analysis and compare our approach with more co-occurrence mining methods. Additionally, we plan to provide more optimized and user-friendly parameter settings, along with detailed guidance on parameters.

Acknowledgments

This study is supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Interior/Interior Business Center (DOI/IBC) contract number 140D0423C0033. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DOI/IBC, or the U.S. Government. We thank Novateur Research Solutions for providing the Veraset data. We thank Dr. Khurram Shafique from Novateur Research Solutions for his helpful suggestions. We also thank Kim Koffolt and the Spatial Computing Research Group at the University of Minnesota for their helpful comments and refinements. We thank the Minnesota Supercomputing Institute (MSI) for computing resources.

References

[1] Hossein Amiri, Will Kohn, Shiyang Ruan, Joon-Seok Kim, Hamdi Kavak, Andrew Crooks, Dieter Pfoser, Carola Wenk, and Andreas Züfle. 2024. The Patterns of Life Human Mobility Simulation. In *Proceedings of the 32nd ACM International Conference on Advances in Geographic Information Systems*. 653–656.

[2] Hossein Amiri, Ruochen Kong, and Andreas Züfle. 2024. Urban Anomalies: A Simulated Human Mobility Dataset with Injected Anomalies. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Geospatial Anomaly Detection*. 1–11.

[3] Berkay Aydin, Dustin Kempton, Vijay Akkineni, Shaktidhar Reddy Gopavaram, Karthik Ganesan Pillai, and Rafal Angryk. 2014. Spatiotemporal indexing techniques for efficiently mining spatiotemporal co-occurrence patterns. In *2014 IEEE international conference on big data (Big Data)*. IEEE, 1–10.

[4] Sajib Barua and Jörg Sander. 2013. Mining statistically significant co-location and segregation patterns. *IEEE Transactions on Knowledge and Data Engineering* 26, 5 (2013), 1185–1199.

[5] Jiannan Cai and Mei-Po Kwan. 2022. Discovering co-location patterns in multivariate spatial flow data. *International Journal of Geographical Information Science* 36, 4 (2022), 720–748.

[6] Mete Celik. 2011. Discovering partial spatio-temporal co-occurrence patterns. In *Proceedings 2011 IEEE International Conference on Spatial Data Mining and Geographical Knowledge Services*. IEEE, 116–120.

[7] Mete Celik, Nuh Azginoglu, and Ramazan Terzi. 2012. Mining periodic spatiotemporal co-occurrence patterns: a summary of results. In *2012 International Symposium on Innovations in Intelligent Systems and Applications*. IEEE, 1–5.

[8] Mete Celik, Shashi Shekhar, James P Rogers, and James A Shine. 2008. Mixed-drove spatiotemporal co-occurrence pattern mining. *IEEE Transactions on Knowledge and Data Engineering* 20, 10 (2008), 1322–1335.

[9] Min Deng, Jiannan Cai, Qiliang Liu, Zhanjun He, and Jianbo Tang. 2017. Multi-level method for discovery of regional co-location patterns. *International Journal of Geographical Information Science* 31, 9 (2017), 1846–1870.

[10] Minxuan Duan, Yinlong Qian, Lingyi Zhao, Zihao Zhou, Zeeshan Rasheed, Rose Yu, and Khurram Shafique. 2024. Back to Bayesics: Uncovering Human Mobility Distributions and Anomalies with an Integrated Statistical and Neural Framework. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Geospatial Anomaly Detection*. 56–67.

[11] Cynthia Dwork. 2006. Differential privacy. In *International colloquium on automata, languages, and programming*. Springer, 1–12.

[12] Chengxu Feng, Jianghu Xu, Jianqiang Zhang, and Houpu Li. 2024. A spatiotemporal co-occurrence pattern mining algorithm based on ship trajectory data. *Advances in Mechanical Engineering* 16, 9 (2024), 16878132241274449.

[13] Matteo Francia, Enrico Gallinucci, and Matteo Golfarelli. 2024. Colossal trajectory mining: a unifying approach to mine behavioral mobility patterns. *Expert Systems with Applications* 238 (2024), 122055.

[14] Subhankar Ghosh, Jayant Gupta, Arun Sharma, Shuai An, and Shashi Shekhar. 2023. Reducing False Discoveries in Statistically-Significant Regional-Colocation Mining: A Summary of Results. In *12th International Conference on Geographic Information Science (GIScience 2023)*, Vol. 277. 3.

[15] Patricia Guerra-Balboa, Alex Miranda Pascual, Javier Parra-Arnau, Jordi Forné, and Thorsten Strufe. 2022. Anonymizing trajectory data: Limitations and opportunities. In *Proceddings of the Third AAAI Workshop on Privacy-Preserving Artificial Intelligence (PPAI-22), Virtual*, Vol. 28.

[16] Shah Muhammad Hamdi, Berkay Aydin, and Rafal A Angryk. 2016. A pattern growth-based approach for mining spatiotemporal co-occurrence patterns. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE, 1125–1132.

[17] Haoji Hu, Jina Kim, Jinwei Zhou, Sofia Kirsanova, JangHyeon Lee, and YaoYi Chiang. 2024. Context-Aware Trajectory Anomaly Detection. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Geospatial Anomaly Detection*. 12–15.

[18] Pradeep Mohan, Shashi Shekhar, James A Shine, James P Rogers, Zhe Jiang, and Nicole Wayant. 2011. A neighborhood graph based approach to regional co-location pattern discovery: A summary of results. In *Proceedings of the 19th ACM SIGSPATIAL international conference on advances in geographic information systems*. 122–132.

[19] Karthik Ganesan Pillai, Rafal A Angryk, and Berkay Aydin. 2013. A filter-and-refine approach to mine spatiotemporal co-occurrences. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 104–113.

[20] Karthik Ganesan Pillai, Rafal A Angryk, Juan M Banda, Michael A Schuh, and Tim Wylie. 2012. Spatio-temporal co-occurrence pattern mining in data sets with evolving regions. In *2012 IEEE 12th international conference on data mining workshops*. IEEE, 805–812.

[21] Shashi Shekhar and Yan Huang. 2001. Discovering spatial co-location patterns: A summary of results. In *International symposium on spatial and temporal databases*. Springer, 236–256.

[22] Chris Stanford, Suman Adari, Xishun Liao, Yueshuai He, Qinhua Jiang, Chenchen Kuai, Jiaqi Ma, Emmanuel Tung, Yinlong Qian, Lingyi Zhao, et al. 2024. NUMOSIM: A Synthetic Mobility Dataset with Anomaly Detection Benchmarks. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Geospatial Anomaly Detection*. 68–78.

[23] Latanya Sweeney. 2002. Achieving k-anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 10, 05 (2002), 571–588.

[24] Andreas Tritsarolis, George-Stylianos Theodoropoulos, and Yannis Theodoridis. 2021. Online discovery of co-movement patterns in mobility data. *International Journal of Geographical Information Science* 35, 4 (2021), 819–845.

[25] Ulanbek Turdukulov, Andres Oswaldo Calderon Romero, Otto Huisman, and Vasilios Retsios. 2014. Visual mining of moving flock patterns in large spatio-temporal data sets using a frequent pattern approach. *International Journal of Geographical Information Science* 28, 10 (2014), 2013–2029.

[26] Yingcai Wu, Di Weng, Zikun Deng, Jie Bao, Mingliang Xu, Zhangye Wang, Yu Zheng, Zhiyu Ding, and Wei Chen. 2020. Towards better detection and analysis of massive spatiotemporal co-occurrence patterns. *IEEE Transactions on Intelligent Transportation Systems* 22, 6 (2020), 3387–3402.

[27] Xiangye Xiao, Xing Xie, Qiong Luo, and Wei-Ying Ma. 2008. Density based co-location pattern discovery. In *Proceedings of the 16th ACM SIGSPATIAL international conference on advances in geographic information systems*. 1–10.

[28] Lu Yang and Lizhen Wang. 2020. Mining traffic congestion propagation patterns based on spatio-temporal co-location patterns. *Evolutionary Intelligence* 13, 2 (2020), 221–233.

[29] Mengjie Yang, Mengjie Zhou, Xinguang He, Yuhui Wang, Zhe Chen, and Jizhe Xia. 2025. Incremental spatiotemporal flow colocation quotient: a new spatiotemporal association analysis method for geographical flows. *International Journal of Geographical Information Science* (2025), 1–24.

[30] Jin Soung Yoo, Shashi Shekhar, John Smith, and Julius P Kumquat. 2004. A partial join approach for mining co-location patterns. In *Proceedings of the 12th annual ACM international workshop on Geographic information systems*. 241–249.

A Example of an Exponential Number of Candidate Groups

To detect a seven-agent recurring co-traveling group [A, B, C, D, E, F, G] that co-travels in every time window, we need to generate and process the numbers of sub-group candidates shown in Table 3. For example, RCG_2 candidates contain [(A, B), (A, C), (A, D), (A, E), (A, F), (A, G), (B, C), (B, D), (B, E), (B, F), (B, G), (C, D), (C, E), (C, F), (C, G), (D, E), (D, F), (D, G), (E, F), (E, G), (F, G)].

Table 3: Candidate Counts for Each RCG Size

RCG Size	Number of Candidates
RCG_1	7
RCG_2	21
RCG_3	35
RCG_4	35
RCG_5	21
RCG_6	7
RCG_7	1
Total	**127**

B Helper Functions

***Size*(list)**: this function returns the number of elements in a list.

***Process*(agents-trajectories, ϵ, Sen_S, Sen_T, coordinates)**: This function adds noise to locations and timestamps of agents-trajectories at a noise level specified by arguments ϵ, Sen_S, and Sen_T. It also converts agents-trajectories to specified coordinate systems, such as relative coordinates (x, y).

***GetActiveAgents*(agents-trajectories, T_{LP}, T_{CT})**: This function keeps agents who have at least T_{CT} time windows during which there are at least T_{LP} visited locations in each time window. These agents are called active agents. This function returns active agents and their trajectories/location traces.

***MostVisitedPlace*(visited-locations, DBSCAN-eps, DBSCAN-min-samples, T_{LP})**: For a given group of visited-locations, this function uses DBSCAN with specified hyperparameters to find the largest cluster with at least T_{LP} visited locations. This function returns both the MBR of the largest cluster and the area of the MBR. If no such cluster exists, this function returns None.

***TransformRCG*(RCG_k, time windows)**: This function transforms the RCG_k dictionary from one using the recurring co-traveling groups as keys to one using time windows as keys.

***RTree*(agentsMBRs)**: Given a list of agent MBRs, this function constructs an R-tree index of the MBRs and their corresponding agents. The function then returns the R-tree index and the index's boundary, which is the MBR of all given MBRs

***GenerateBufferedGrids*(boundary, $step_y$, $step_x$, T_D)**: This function divides the study area, defined by the boundary input, into grids with specified x and y step sizes. It then adds a buffer to the grid by (2 * T_D). These buffered grids avoid missing patterns on the boundary between grids.

***Query*(buffered-grid, Rtree-index)**: This function uses the Rtree-index to find agents whose MBRs overlap with a given buffered grid. The function then returns a list of agents.

***TwoAgentColocations*(time-window, RCG_1, agents, T_D, T_{AU})**: For a given time window and a given list of agents, this function finds two-agent co-location groups. The candidate two-agent co-location groups are generated by combination. The candidates are then pruned if the distance between the two agents' MVPs is greater than T_D or the GCA is greater than T_{AU}.

***GenerateCandidates*(groupsMBRs)**: This function generates $RCG_{(k+1)}$ candidate groups from RCG_k by matching the first (k-1) agents between any two RCG_k groups.

***GenerateCandidatesFast*(groupsMBRs)**: Given k-agent recurring co-traveling groups and their MBRs, this function generates $RCG_{(k+1)}$ candidates using the anti-monotone property and a divide-and-conquer strategy. Specifically, the function divides k-agent groups into partitions by their first (k-1) agents. For each partition, the function generates $RCG_{(k+1)}$ candidates. Finally, the function collects all candidates and removes duplicates.

***PruneCandidates*(generatedCan, groupsMBRs, T_{AU})**: This function prunes (k+1)-agent candidate groups whose GCAs are greater than T_{AU}. A group's GCA is calculated using groupsMBRs.

***PruneFilterCan*(generatedCan, groupsMBRs, RCG_{k+1}, T_{AU})**: This function first removes generated (k+1)-agent candidate groups (generatedCan) that have not been in RCG_{k+1} yet. Then, it does the pruning as function *PruneCandidates*.

***DetectNextK*(RCG_k)**: Given RCG_k, this function uses Algorithm 4 to output RCG_{k+1}.

***FindContainedRCG*(RCG_k, RCG_{k+1})**: This function finds RCG_k groups that are contained in some RCG_{k+1} groups. An RCG_k group is contained in an RCG_{k+1} group when 1) the RCG_k group members are a subset of the RCG_{k+1} group members, 2) for each recurring co-traveling time window, the GCA of the RCG_k group is a subset of the GCA of the RCG_{k+1} group.

***Cloak*(RCG_K, T_{AL}, T_{AU})**: This function searches RCG_K for GCAs smaller than T_{AL}. The function then generates a random MBR of size in $[T_{AL}, T_{AU}]$ to cloak those small GCAs.

We control two DBSCAN hyperparameters: 1) DBSCAN-eps, which is the maximum distance between two visited locations to be considered neighbors; and 2) DBSCAN-min-samples, which is the minimum number of visited locations needed within a neighborhood to consider a visited location as a core point.

C The RCG Interest Measurement Has the Anti-monotone Property

Lemma 1. *If Distance(GCA_{Gi}, TW_t) <= T_D and Gj ⊆ Gi, then Distance(GCA_{Gj}, TW_t) <= T_D.*

Proof. By definition, Distance(GCA_{Gi}, TW_t) <= T_D is equivalent to max(distance(MVP_a, MVP_b)|a, b ∈ Gi) <= T_D, where distance(MVP_a, MVP_b) is the distance between the MVP of a and the MVP of b. a and b are any agents in Gi. Thus, this condition means the maximum distance between any two agents in Gi is less than or equal to the distance threshold. Since Gj ⊆ Gi, any two agents of Gj belong to Gi. Thus, Distance(GCA_{Gj}, TW_t) <= Distance(GCA_{Gi}, TW_t) <= T_D.

Lemma 2. *If Area(GCA_{Gi}, TW_t)∈ $[T_{AL}, T_{AU}]$ and Gj ⊆ Gi, then Area(GCA_{Gj}, TW_t)∈ $[T_{AL}, T_{AU}]$.*

Proof. By definition, Area(GCA_{Gi}, TW_t)∈ $[T_{AL}, T_{AU}]$ ≡ Area($\bigcup_{a \in Gi}(MVP_a, TW_t)$)∈ $[T_{AL}, T_{AU}]$, where $\bigcup_{a \in Gi}(MVP_a, TW_t)$

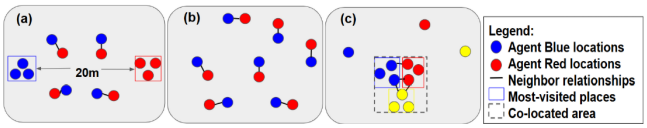


Figure 9: Differences in co-locations between MDCOP and RCPD

is the union of the MBRs of all agents of Gi in TW_t. Since Gj ⊆ Gi, then for any a∈Gj, a∈Gi. Thus, $\bigcup_{a\in Gj}(MVP_a, TW_t) \subseteq \bigcup_{a\in Gi}(MVP_a, TW_t)$. Thus $Area(GCA_{Gj}, TW_t) \equiv Area(\bigcup_{a\in Gj}(MVP_a, TW_t)) \subseteq Area(\bigcup_{a\in Gi}(MVP_a, TW_t)) \equiv Area(GCA_{Gi}, TW_t) \in [T_{AL}, T_{AU}]$.

Theorem 3. *The RCG interest measure has the anti-monotone property. If a group is an RCG, then any sub-group with greater than or equal to K agents of the RCG must also be an RCG. Alternatively, if a group is not an RCG, then none of its super-groups can be an RCG.*

Proof. From Lemma 1 and Lemma 2, we get that for any time window *t*, if $Distance(GCA_{Gi}, TW_t) <= T_D$, $Area(GCA_{Gi}, TW_t) \in [T_{AL}, T_{AU}]$, and Gj ⊆ Gi, then $Distance(GCA_{Gj}, TW_t) <= T_D$ and $Area(GCA_{Gj}, TW_t) \in [T_{AL}, T_{AU}]$. Thus, if there are *n* time windows where the Distance and Area conditions hold for Gi, and $n >= T_{CT}$, then during the same *n* time windows the Distance and Area conditions hold for Gj. In addition, Gj has a cardinality >= K. Thus, $Count(t \in T)(Distance(GCA_{Gj}, TW_t) <= T_D, Area(GCA_{Gj}, TW_t) \in [T_{AL}, T_{AU}]$, Cardinality(Gj) >= K) >= T_{CT}, which meets Definition 3. Thus, Gj is an RCG, and the RCG measure has the anti-monotone property.

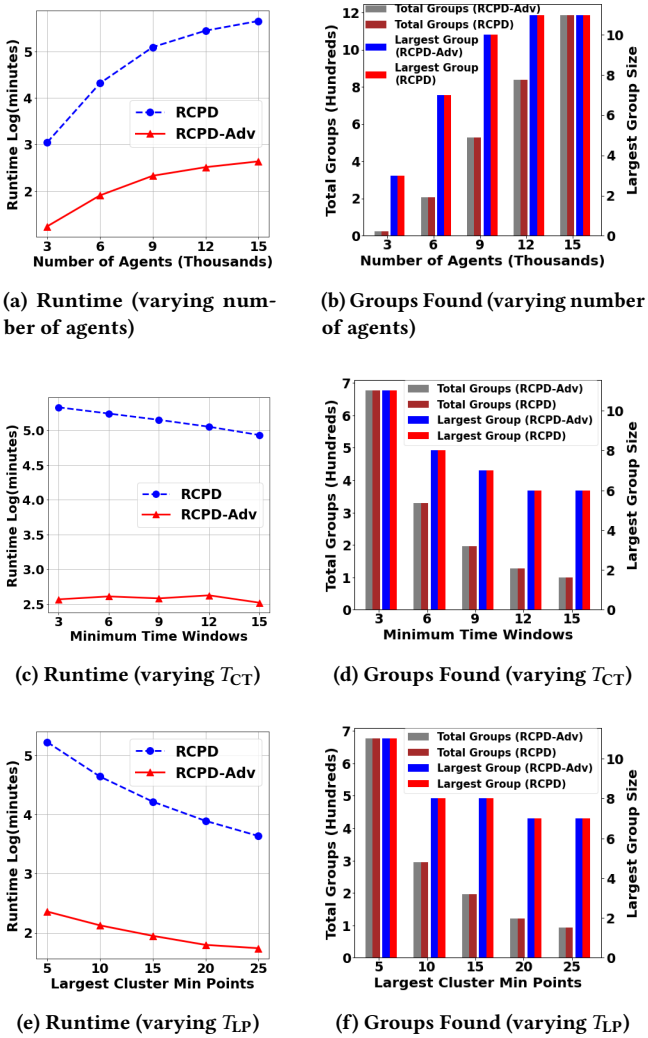
D Example of the Efficiency Strategy of Step 3

Assume there are ten time windows and $T_{CT} = 3$. At the ninth iteration, if a candidate has less than $9 - (10 - 3) = 2$ recurring co-traveling time windows at this step, it will be pruned because even if the last time window qualifies, this candidate would have fewer than three recurring co-traveling time windows, disqualifying regardless the last time window being checked.

E Comparison of MDCOP and RCPD

MDCOP and RCPD differ on four fronts. First, MDCOP detects global co-locations using visited locations, while RCPD detects regional co-locations using concentrations of visited locations. Figure 9 gives examples of differences in co-locations between the two methods. Assume that the participation index (PI) threshold is equal to 0.5 and T_D is equal to 10m, in Figure 9a, Blue and Red co-locate in MDCOP as $PI(Blue, Red) = 4/7 > 0.5$. However, Blue and Red do NOT co-locate in RCPD because the distance between their MVPs is 20m, which exceeds T_D. In Figure 9b, Blue and Red co-locate in MDCOP because $PI(Blue, Red) = 1 > 0.5$. But Blue and Red do NOT co-locate in RCPD because they do not have MVPs. In Figure 9c, Blue, Red, and Yellow do NOT co-locate in MDCOP because $PI(Blue, Red) = 1/3 < 0.5$, while they do co-locate in RCPD as their MVPs are close to each other. Second, the T_{LP} threshold in RCPD requires a minimum number of visited locations participating in co-locations, while the MDCOP approach does not have this requirement. In MDCOP, ten participating visited locations out of twenty total visited locations result in the same participation ratio as one participating visited location out of two total visited locations. Thus, the MDCOP approach does not guarantee recurring patterns. Third, outputs from MDCOP will not satisfy RCPD's societal applications, which require knowing specific GCAs and recurring co-traveling routes. Fourth, MDCOP does not consider privacy protection, while RCPD considers privacy protection through de-identification, differential privacy, and k-anonymity.

F Sensitivity Analysis Results with No Noise



(a) Runtime (varying number of agents)

(b) Groups Found (varying number of agents)

(c) Runtime (varying T_{CT})

(d) Groups Found (varying T_{CT})

(e) Runtime (varying T_{LP})

(f) Groups Found (varying T_{LP})

Figure 10: Sensitivity analysis on runtime and RCG count under varying conditions: number of agents, threshold count time window (T_{CT}), and threshold largest cluster points (T_{LP}). No added noise.