LEARNING OF NETWORK DYNAMICS: MOBILITY, DIFFUSION AND EVOLUTION

by

HAOTIAN WANG

A dissertation submitted to the
School of Graduate Studies
Rutgers, The State University of New Jersey
In partial fulfillment of the requirements
For the degree of
Doctor of Philosophy
Graduate Program in Computer Science

Written under the direction of
Jie Gao
and approved by

New Brunswick, New Jersey
Jan, 2022
Learning of Network Dynamics: Mobility, Diffusion and Evolution

By Haotian Wang
Dissertation Director:
Jie Gao

Within network science and network theory, there has been a lot of success on analysis of complex network in a static setting or at a certain snapshot. The analysis of network dynamics takes interactions of social features and temporal information into account. Different from static networks, dynamic networks consider larger, dynamic, multi-mode, multi-plex networks, and may contain varying levels of uncertainty. Due to the heterogeneity of networks, agent-based modeling and other forms of simulations are often used to explore how networks evolve and adapt as well as the impact of interventions on those networks. In this thesis, we discuss two aspects of this topic.

In the first aspect, we consider human mobility and dynamic networks that are enabled by human trajectories. A human trajectory is a sequence of spatial-temporal data from an individual. Knowledge mined from human trajectories can help us improve a lot of real-world applications. However, the increasing size of the human trajectory dataset brings challenges to our analysis. At the same time, for human trajectory, we still lack for a comprehensive understanding of the relationship among the human trajectories, such as commonality and individuality. Thus, we focused on the statistic tools to measure the similarity between human trajectories. Different from the traditional
geometric similarity measures, such as Hausdorff distance and Fréchet distance, we proposed three novel partial similarity measures, that are more suitable to human trajectories. Using the partial similarity measures can also reduce the storage demand and save computation time. Based on these similarity measures, we designed a advanced unsupervised clustering algorithm, leveraging with the conformal prediction framework. It performs better in varied trajectory datasets compared with the classical clustering algorithms.

In the second aspect, we investigate two problems in the social interactions of the social networks, diffusion and evolution. For the first problem, previous works mainly focused on the information spreading speed in static social networks. Taking mobility into the consideration brings a new dimension to the problem, where interactions between different individuals happen over time. The heterogeneity of mobility plays an important role in the diffusion process. We utilize human trajectory dataset to simulate the physical interactions among individuals which allowed us to predict the diffusion process and dependency on parameters. At the same time, we also investigate the impact of targeted interventions on the social networks. In the second problem, we study the evolution process of opinion dynamics (people’s view on a particular topic). We proposed a co-evolution model, including the opinion dynamics and social tie dynamics, to investigate the community structure and structural balance in opinion dynamics (i.e., consistency of opinion and signs of ties along cycles) in the final state with rigorous theoretical analysis. We also show applications of this model in predicting evolution in a real world data set.
Acknowledgements

My Ph.D. journey is coming to an end, but I will always treasure the precious and monumental moments over the last few years. It is fortunate to have a number of people who made my time enjoyable and helped me in getting this far. I would like to acknowledge of them.

First of all, I would like to give straightforward but truthful appreciation to my advisor, Professor Jie Gao, one of the nicest persons I have ever met. She is an outstanding researcher with broad knowledge, grand vision and great mathematical intuition. As an advisor, she is very patient to guide me in the journey of learning independently, thinking deeply, and experiencing the academic world. In each research step, from selecting topics, to solving problems, to writing papers, to presenting works, she can always provide detailed advice. During the years in working with her, I am impressed, influenced and shaped by her wisdom, kindness and diligence. I am sincerely grateful for all mental, academic, financial supports from her.

I would like to show my gratitude to all my committee members, Professor Feng Luo, Professor Hao Wang, Professor Peng Zhang, and Professor Joseph S.B. Mitchell for their guidances during my Ph.D. study and dissertation writing.

I want to thank all my lab mates and co-authors, including Chien-Chun Ni, Jiaxin Ding, Kin Sum Liu, Hao-Tsung Yang, Aria Rezaei, Shih-Yu Tsai, Chengyuan Deng, Prathamesh Dharangutte, Vikrant Ashvinkumar, Abhirup Ghosh, Rik Sarkar, Niranjini Rajagopal, Anthony Rowe, Bruno Sinopoli, Feng Luo, and Min-ge Xie, for their advice, discussions, assistances, and collaborations.

I have been fortunate to spend my first two and a half years at Stony Brook University and the rest two years at Rutgers University. Both of them have a big family for me to discuss research projects and enjoy life. Thus, I want to thank all of my friends,
Shilei Tian, Shengzhong Liu, Sangtian Wang, Caitao Zhan, Yicheng Lin, Yucheng Xing, Yukun Yuan, Yu Yang, Guang Wang, Shuxin Zhong, Zhiqing Hong, Guang Yang, and Wenjun Lyu. They make my Ph.D. life colorful and fun. I also want to thanks all my friends who shared thier lifes and helped me.

Last but not least, I want to express my deepest thanks to my family. I am deeply indebted to my paraents for their love, encouragement and selfless support. I also sincerely thank my girl friend, Liqun Ding for her accompany, support and encouragement since 10 years ago. This dissertation is dedicated to them.
Dedication

To my parents.

To my advisor Prof. Jie Gao for her guidance and care.
Table of Contents

Abstract .......................................................... ii
Acknowledgements .................................................... iv
Dedication ............................................................ vi
List of Tables ......................................................... xi
List of Figures ......................................................... xii

1. Introduction ...................................................... 1
   1.1. Mobility Properties .......................................... 1
   1.2. Information Spreading ....................................... 3
   1.3. Reference ..................................................... 4
   1.4. Publication .................................................. 5

2. Distributed Human Trajectory Sensing and Partial Similarity Queries 7
   2.1. Introduction .................................................. 7
   2.2. Related Work ................................................. 13
   2.3. Challenges ................................................... 16
   2.4. Trajectory Sensing, Analysis and Query .................... 17
       2.4.1. Partial Similarity ...................................... 18
       2.4.2. Random Collection .................................... 19
       2.4.3. Performance Analysis .................................. 21
   2.5. Evaluation by Simulation ................................... 23
       2.5.1. Experimental Setup. ................................... 24
       2.5.2. Precision and Recall ................................... 26
       2.5.3. Time Efficiency ....................................... 29
3. Unsupervised Clustering Algorithm for Human Trajectories ........................................... 34
   3.1. Introduction ......................................................... 34
       3.1.1. Challenges .................................................. 35
       3.1.2. Our Contribution ........................................... 37
   3.2. Review of DBSCAN and Conformal Prediction ............................................. 41
       3.2.1. DBSCAN ..................................................... 41
       3.2.2. Conformal Prediction ..................................... 42
   3.3. Conformal DBSCAN .................................................. 45
       3.3.1. Algorithm .................................................... 45
       3.3.2. Theoretical Analysis ........................................ 47
       3.3.3. Relation with DBSCAN .................................... 50
   3.4. Evaluation by Simulation .............................................. 51
       3.4.1. Experimental Setup ......................................... 51
       3.4.2. Trajectory Clustering ....................................... 55
       3.4.3. Trajectory Classification .................................. 61
       3.4.4. Discussion .................................................. 63
   3.5. Conclusion .......................................................... 64

4. Information Spreading on Real Mobility Data ................................................. 65
   4.1. Introduction ......................................................... 65
   4.2. Method ................................................................. 66
   4.3. Results ................................................................. 69
       4.3.1. Evaluation of baseline uniform mobility intervention strategy ........ 70
       4.3.2. Heterogeneity in mobility and contagion dynamics ................. 71
       4.3.3. Targeted interventions on the most active individuals and the most popular venues ... 72
4.3.4. Contact graph: Social network abstraction to estimate mobility
based infection spread. ..................................................... 73
4.3.5. Robustness of results. .............................................. 74
4.4. Discussion ............................................................... 82

5. Co-evolution of Opinion and Social Tie Dynamics Towards Structural
Balance ................................................................. 85

5.1. Introduction .......................................................... 85
5.1.1. Our Contribution .................................................. 87
5.2. Related Work ........................................................ 89
5.2.1. Opinion Dynamics and Social Influence ....................... 89
5.2.2. Structural Balance and Signed Networks ....................... 91
5.3. Co-Evolution Model ................................................ 93
5.4. Analysis of the Opinion and Social Tie Evolution ............... 96
5.4.1. Analysis of Opinion Evolution .................................. 96
5.4.2. Analysis of Social Tie Evolution ............................... 104
5.5. Simulation ............................................................ 115
5.5.1. Harmony vs Polarization ....................................... 115
5.5.2. Convergence Rate ................................................. 117
5.5.3. Emergence of Community Structure ............................ 119
5.6. Conclusion and Future Work ....................................... 120

6. Efficient Beacon Placement Algorithms for Time-of-Flight Indoor Lo-
calization ............................................................... 122

6.1. Introduction .......................................................... 122
6.2. Related Work ........................................................ 125
6.3. Problem Definition .................................................. 127
6.4. Algorithm to Verify Unique Localization .......................... 129
6.4.1. Unique Localization of a Point ................................. 129
6.4.2. Unique Localization of a Region ............................... 130
List of Tables

2.1. Dataset Description ......................................... 9
2.2. characteristics of the datasets used in our experiments ........... 25
2.3. Parameters used in Figure 4. .................................. 26
2.4. Running time of different query cases with varied collection probabilities 30
2.5. The number of samples to uniquely identify a trajectory ............... 32
3.1. Dataset Description ........................................... 52
3.2. Clustering result for synthetic trajectories ............................. 56
3.3. The running time of DBSCAN and conformal DBSCAN for human trajectories ........................................... 59
3.4. Classification result of synthetic trajectories .............................. 62
6.1. number of beacons placed by various algorithms ....................... 144
6.2. Performance as random floor plan scales up ............................ 145
List of Figures

2.1. Illustration of the Hausdorff and discrete Fréchet distance of two trajectories. ........................................... 10

2.2. Distribution of distances and partial similarities in Zhengzhou. The blue curve and the orange curve describe the distance of two trajectories of two consecutive days of the same agent. The green line show the distance between the trajectories of two randomly chosen agents in one day. For partial similarity measures, we consider two sample points (without timestamps) match if they are within distance $r$, with $r$ to be the $x$-axis of the plot. We mark the 50 percentile by the vertical dotted line. . . . 15

2.3. Process of sampling and computation: There are two agents who visit 10 checkpoints. When the agent arrive at a checkpoint, with probability $p$, the checkpoint collects the information from all nearby agents. The sample probability $p$ is set as 0.4. Then, for different query cases, we represent the sampled trajectories in different ways. According to the definitions, we compute the similarities between the sampled trajectories and the original trajectories. .................................................. 20

2.4. Performance in terms of precision and recall of our framework on Zhengzhou and Wenzhou. the effect of sample probability $p$, threshold coefficient $k$ and similarity parameter $\alpha$ on the performance. ......................... 27

2.5. Shenzhen: Order-Sensitive similarity compared with result by LSH . . . 30

2.6. Zhengzhou: Distribution of the trajectory pairs based on similarity parameter ......................................................... 31
3.1. Four algorithms: $k$-means, Mixture of Gaussian (GMM), DBSCAN and conformal DBSCAN on three synthetic datasets. The clusters are shown in different colors. 

3.2. Illustration of the conformity score. There are 8 points, $c_1, \ldots, c_8$ in the reference set $C$, with the underlying distribution on a circle (shown in pink). The points $c_9$ and $c_{10}$ are checked against a randomly chosen point (say $c_4$). $\mathcal{A}(x, Y)$ is the sum of distances from $x$ to all points in $Y$. $\alpha^{(4)}_9 > \alpha_4; \alpha^{(4)}_{10} < \alpha_4$ and $\alpha^{(4)}_{11} \approx \alpha_4$. In fact, both $p(c_9)$ and $p(c_{10})$ are close to zero; while $p(c_{11})$ are away from 0.

3.3. Clustering for synthetic trajectories: The first column is the ground truth of three cases. 250 trajectories belong to 5 clusters in different colors, and the outliers are in black. The clustering results by $k$-means, DBSCAN, Conformal DBSCAN are shown in the following columns.

3.4. Clustering results for animal trajectories. (a) and (b) are the trajectories of three species of animals (Red: Elk, Blue: Deer, and Green: Cattle). (c) and (d) show the clustering results by DBSCAN and conformal DBSCAN algorithm.

3.5. Clustering results using Hausdorff distance and Fréchet distance.

3.6. Clustering results of DBSCAN and conformal DBSCAN algorithm using different similarity measures.

3.7. Precision results for human trajectory classification with different similarity measures.

4.1. The model for infection spread (Part I) and dataset properties (Part II to IV).

4.2. Uniform lockdown for limited duration delays and lowers the peak of the infections, but often has a small effect on total infection.

4.3. Heterogeneity of mobility in venues and agents results in varied risks to catch and spread the virus.
4.4. Across all datasets, the most effective strategies in terms of their health and social values consider the heterogeneity of mobility (A to D) and (E to H) present the details of the best strategies. 79

4.5. Contagion simulation in the derived social network model infects a similar number and similar set of agents as the simulation using mobility data in both with and without intervention settings. 80

4.6. Diving agents into cohorts cuts off the contacts between different cohorts, like separating students into different sessions for courses and acts to reduce the spread of the disease. 81

5.1. Structural balance theory: the first two triangles are stable while the last two are not. 86

5.2. The evolution process for a complete graph with $BC = CB$. In the first three plots, the network at the limit achieves structural balance and the network is also polarized. In the last example, the network reaches harmony. 116

5.3. The evolution process for a complete graph with $BC \neq CB$. 117

5.4. The number of iterations till network convergence. 119

5.5. The weight matrix in evolution. 121

5.6. Community detection in two real-world data sets. 121

6.1. Localization with ToF in a floor plan: (a) An infinite number of solutions with a single beacon; (b) Two solutions with two beacons; (c) A unique solution with three beacons; (d-f) A unique solution with two beacons: (d) $p_2$ is outside the floor plan and thus is not feasible; (e) $p_2$ is not visible to $b_2$ and thus is not feasible; (f) $p_2$ is visible to a third beacon $b_3$ and thus is not the correct solution. 123

6.2. An example for which prior work [161] can be suboptimal. 131

6.3. Illustration of Algorithm 2: Each beacon is set the weight and the red beacons are selected beacon locations. The purple region contains all the points with weight greater than $\varepsilon w(B)$. 136
6.4. Beacon placement using *Random Sampling* for various floor plans . . . . 142
6.5. Comparison of expected accuracy with different placement algorithms
   for Figure 6.4(f) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 146
6.6. Comparison between RS-placed beacons (red dot) and user placed bea-
   cons (blue circles) . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 147
Chapter 1
Introduction

The study of network dynamics considers network changes over time. It brings together traditional social network analysis and multi-agent systems within network science and network theory. Compared with static networks, dynamic networks may have time dimension and interactions between nodes. In different time slots, networks might have different shape and structures. Nodes and edges might be crashed in the process. Interactions could take nodes and edges to varied states over time. Thus, dynamic networks bring more challenges into research. In previous work, they mainly use the statistical analysis tool to measure network dynamics data. Owing that each node may have different state over time, multi-agent based simulations are utilized. Both the statistical analysis and simulation analysis are utilized in our research work.

This dissertation has three part. The first part is about the social networks arising from human mobility. We focused on the human trajectories and investigated the commonality and individuality among individuals. We proposed three novel similarity measures for the human trajectories and designed an efficient unsupervised clustering algorithms for trajectories. In the second part, we considered two problems in the information spreading of the social networks. One is the spreading speed in the dynamic social networks and the other is the opinion evolution in the social networks. Both of them take the time information into consideration. The third part includes the indoor localization problem, which is other research topic in my studies.

1.1 Mobility Properties

In this part, we mainly focus on the human trajectory. The ubiquitous availability and mobile devices and wireless coverage has enabled the generation and collection of
large-scale human trajectories. Periodically the mobile agents exchange information with nearby base stations regarding the agent identification (ID), time and location. These records, put together, produces a good representation of the trajectory. Different from previous approaches in which users voluntarily upload their GPS trajectories, the distributed framework of trajectory sensing has allowed for a massive scale of trajectory collection.

Such big-scale human mobility data has motivated human trajectory analysis and mining. There are two fundamental perspectives to study, i.e., commonality and individuality, related to a wide range of applications including anomaly detection [156, 178], crime investigation [106], city planning [129], traffic analysis and optimization [184].

Regarding large-scale big spatial-temporal data, there are several challenges in trajectory collection and analysis. We need to design efficient schemes and algorithms to solve these challenges.

**Effective Sensing Schemes.** In most of previous works, all spatial-temporal data is collected in a centralized manner before any analysis. A lot of redundant data is kept for representations of trajectories. It requires high storage capacity and computation burden for the following analysis. If we only focus on specific properties of trajectories, such as geometric shapes and topology representations, we do not need to store all the data in the central manner. It is desirable to have a distributed sensing system to represent trajectory efficiently to reduce communication, storage, and computation complexity.

**Suitable Similarity Measures.** Existing techniques mainly use the spatial and temporal features of the human trajectories. Sometimes, these low-level position based features are insufficient to distinguish difference among the human trajectories, due to issues such as lack of transparency and interpretability. In general, feature design for trajectories is still a non-trivial problem. It also determines the performance of human trajectory measures. The classical geometric measures for distance of curves using geometric definitions, such as the Hausdorff distance and Fréchet distance, are more suitable to trajectories that are aligned, uniformly and consistently sampled. On a real-world human trajectory dataset, it is nearly impossible for two trajectories to
be close to each other all the time. Thus, we need to develop alternative similarity measures for trajectory analysis.

**Efficient Algorithms.** With the growing demand for analysis of big human trajectory analysis, it is critical to design time-efficiency algorithm with respect to the limited computation capacity. In addition, the trajectories can be regarded as a high-dimensional vector. There is no good understanding about the trajectory distribution and prior information. The traditional algorithms mainly focus on the geometry property of trajectories, like short trajectories taken by cameras. However, in our research, sparsely-sampled human trajectory may not contain obvious geometry property. Thus, these algorithms may not be suitable for trajectory data. Therefore, we also need to design performance-efficiency algorithms for human trajectory data.

In our research, we first provide a few realistic similarity measures for human trajectories. These measures are more suitable to represent the similarity between the human trajectories than the traditional similarity measures, such as Hausdorff distance, Fréchet distance and Dynamic Time Warping distance. At the same time, they can be applied to the distributed collection system, saving storage and computation time. Then, based on the novel similarity measures, we investigated a new unsupervised clustering algorithm. This algorithm can not only group the similar human trajectories together, but also detect the anomalous human trajectories in the dataset. These statistical tools perform well in the real-world human trajectory dataset.

1.2 Information Spreading

In social networks, information spreading is a common phenomena. We live in a continuously changing world in which social interaction dynamically shape who we are, how we view the world, and what decision to make. In this aspect, there are two research topics we mainly focus on. The first one is to investigate the information spreading speed in the social networks, which is called **Diffusion**. The other one is to view the changing of decision making via social influence, i.e., **Evolution**.

**Diffusion.** In previous works, the diffusion phenomena is investigated in a static
setting, such as static networks. In the extreme cases, the social networks can be a complete graph. Information spreads between two connected individuals with a probability. However, in many real-world scenarios, the social networks are dynamics over time. Taking the human mobility as an example, the physical interactions between individuals are different over time. Certain individuals may have more interactions than the average population. This heterogeneity creates fundamental difference in terms of information spreading speed compared to the stable social networks. It is an interesting research topic to simulate the information spreading in dynamic social networks. Thus, we study information spreading in a heterogeneous dynamic network enabled by human mobility.

**Evolution.** Social influence describes how people’s behaviors, habits or opinions are shifted by those of their neighbors. Social influence leads to homophily (similarity of node attributes between friends) and leaves traces in the network structure such as high clustering coefficient and triadic closure. In the literature, there have been a lot of studies of opinion dynamics, where people’s behaviors, habits or opinions are influenced by those of their neighbors. In the process, the social networks evolution can be viewed over time. It is interesting to investigate the final state of the social networks via the evolution. In our research, we consider an extreme case of community structure, dictated by *structural balance*, a common phenomenon observed in many social relationships. A co-evolution model for the dynamic social networks is proposed. We found that node opinions evolve into opposing communities and structural balance natural emerges.

### 1.3 Reference

Parts of the chapters has been or will be published on international conferences or journal. Here is a list of them.

Chapter 2 is based on the work “Distributed Human Trajectory Sensing and Partial Similarity Queries”, co-authored with Jie Gao, appearing in *Proceedings of the 19th International Symposium on Information Processing in Sensor Networks (IPSN 2020)*,
Chapter 3 is based on the work “Clustering of Trajectories using Non-Parametric Conformal DBSCAN Algorithm”, co-authored with Jie Gao and Min-ge Xie.


1.4 Publication


Chapter 2
Distributed Human Trajectory Sensing and Partial Similarity Queries

2.1 Introduction

The ubiquitous availability of mobile devices and wireless coverage has allowed for the collection of a huge volume of human motion trajectories through wireless connections between devices or from devices to base stations. Periodically the mobile agents exchange information with nearby base stations regarding the agent identification (ID), time and location. These records, put together, produces a good representation of the trajectory. Different from previous approaches in which users voluntarily upload their GPS trajectories, the distributed framework of trajectory sensing has allowed for a massive scale of trajectory collection. Examples of trajectories collected in this manner include human mobility traces of over 100,000 mobile phone users [81] and about 95,000 users on a large university campus [216]. Such big human mobility data has motivated human trajectory analysis and mining with a wide range of applications including anomaly detection [156, 178], crime investigation [106], city planning [129], traffic analysis and optimization [184].

Trajectory Sensing and Analysis One of the most fundamental questions is to understand properties of natural human trajectories. Prior work on mining and analysis of human trajectories has revealed two important aspects: individuality and commonality.

On the aspect of individuality, it has been confirmed in multiple datasets [55, 59] that a handful of randomly chosen spatial-temporal data points – the timestamp and location of a user – could uniquely identify the user from a large trajectory database, even with coarse-grained resolution. This suggested that human mobility trajectories, in
the space-time domain, are very sparse. It is unlikely that two different users are at the same location at randomly chosen timestamps. The individuality of human trajectories suggests important security-related applications such as crime investigation (locate the individual who appeared in a few locations at certain times). It also reveals interesting interplay with social structures – individuals who frequently visit the same location (even if not in the same order and not at the same time) are strongly correlated with social ties [190, 24]. The individuality of human trajectories also raises potential privacy concerns for releasing the whole trajectory database to the public.

On the other hand, data mining applications have been trying to identify commonalities among the trajectories, such as convoys (groups of users moving together continuously) [108, 107, 185, 208, 149], frequent sequential patterns (frequently occurring subsequences) [175, 211], or popular paths (paths travelled by a significant fraction of users, possibly at different time, among all users that appears on the paths) [111, 58]. These common patterns and features of trajectories suggest important patterns that could be useful for traffic analysis and civil planning.

Understanding both individuality and commonality of trajectories requires efficient similarity search, possibly with different similarity measures, which is the topic of this paper.

**Dataset and Motivation.** The research problem in this paper is motivated by a trajectory data set from electric motorbikes collected in two different cities over a month. This is a unique dataset – it is large both in the number of users and the time scale, and the motorbikes are individually owned, which makes the dataset to be directly related to individual mobility patterns. In prior work, there have been a lot of studies on mobility datasets of taxis [209, 184, 196], which is suitable for traffic analysis but is less useful for understanding human mobility properties.

Some basic information from one day’s data is shown in Table 2.1. As could be seen in the table, there are two immediate challenges in data collection and analysis. First, the dataset is huge. There are over 600K agents in Wenzhou with over 50 million records for one day only. For Zhengzhou there are fewer agents (still over 40K) but the data is collected with much higher resolution and the number of records for one day
<table>
<thead>
<tr>
<th>City</th>
<th>Zhengzhou</th>
<th>Wenzhou</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensing</td>
<td>GPS location</td>
<td>Appearance</td>
</tr>
<tr>
<td>Geo-range</td>
<td>27km × 33km</td>
<td>110km × 70km</td>
</tr>
<tr>
<td># of Agents (n)</td>
<td>42,380</td>
<td>633,194</td>
</tr>
<tr>
<td># of Records</td>
<td>19,802,231</td>
<td>50,873,192</td>
</tr>
<tr>
<td>Avg # of Records/Agent</td>
<td>467</td>
<td>80</td>
</tr>
</tbody>
</table>

Table 2.1: Dataset Description

is beyond 19 million. Collection, storage, and management of the big data is a great challenge. The immediate questions are: 1) whether one could use sparse sampling to reduce data size; yet 2) whether one could still perform meaningful similarity analysis and query for data mining tasks.

**Trajectory Similarity and Query.** Comparing two trajectories is a heavily studied problem. There has been many commonly used measures to calculate the distance or similarity of two trajectories, including the Hausdorff distance in Figure 2.1(a) (the maximum of the distance from any vertex on one trajectory to the other trajectory), the Fréchet distance in Figure 2.1(b) (the min max distance for a traversal along the two trajectories without backtracking) or Dynamic Time Warping (DTW) distance (min sum instead of min max compared to the Fréchet distance). The Hausdorff distance ignores the order of locations visited by the agent while the Fréchet and DTW distance are sensitive to the orders.

Our datasets are at a city scale and contains user activities in a long period of time. Using these classical measures to analyze the similarity of two users faces two problems.

First, many of these distance measures are extreme measures (of the min-max or max-min type) and capture the worst-case scenario. On a dataset at a city-scale, the distances often come out to be too large to be interesting. It is nearly impossible for any two individuals to be always staying close to one another in a day. In Figure 2.2, we focus on discrete versions of curve similarity, in which the trajectories are given in the form of a sequence of (possibly time-stamped) points.
we randomly selected 1,000 agents in the Zhengzhou dataset and compared the discrete Hausdorff and discrete Fréchet distance (without the timestamp information). About half of the Hausdorff distance and Fréchet distance are more than 20 km, which is the radius of the city. If we want to answer near-neighbor queries (return trajectories similar to a given one), these measures are too sensitive to outlier points and are not very informative.

Second, the computational cost of these distance measures becomes a non-trivial challenge, especially on data at this scale. First, computing the distance of two trajectories could take a quadratic running time (e.g., for Fréchet distance and Dynamic Time Warping distance) on the number of data points of each trajectory. For similarity-based queries, i.e., return the trajectories in the database that are similar to a given one, even a linear running time algorithm (as for the case of Hausdorff distance) becomes infeasible, if we need to scan and compare with the entire database. We need to design super efficient algorithms to answer such queries.

Our Results. In this paper, we propose efficient methods for sensing, processing and answering queries of massive city-scale trajectory datasets.

We develop and use new concepts of partial trajectory similarity measures. The motivation is two-folds. First, we would like to reduce sensitivity to extreme values and develop measures that generate more differentiating results in similarity queries. Second, by using different amount of information in human trajectories, we have measures for different interesting features in terms of individuality and commonality.
• **Time-Sensitive partial similarity**: Each trajectory is a sequence of time-stamped locations. Two trajectories are $\alpha$-Time-Sensitive similar if at least an $\alpha$ fraction of samples on the short trajectory are the same ones (or nearby with a specified distance/time threshold) on the other trajectory, including time stamp and location.

• **Order-Sensitive partial similarity**: Each trajectory is an *ordered sequence* of locations visited by the agent. Two trajectories are $\alpha$-order-sensitive similar, if an $\alpha$ fraction of samples on the shorter trajectory are matched (exactly or with a given distance threshold $r$) with the samples on the other trajectory in the corresponding order. This could be considered as a partial measure for Fréchet distance.

• **Order-Insensitive partial similarity**: Similar to the previous case, but ignore the order of locations an agent visits. This could be considered as a partial measure for Hausdorff distance.

The partial similarity/distances are more robust and differentiating than the classical distances. It is well known that the trajectories of the same agent in different days are similar. The plot using partial similarities clearly demonstrates this while classical measures fail to catch this pattern. See Figure 2.2 and Section 2.3.

The three partial measures, Time-Sensitive, Order-Sensitive, and Order-Insensitive ones, keep a decreasing amount of information (temporal dimension, traversal order, or appearance only, respectively) from the original trajectory data. Therefore, the focus of (partial) similarity in these three measures transitions from individuality to commonality in trajectory data. We empirically evaluated this dimension. By using a random sample from a given query trajectory, we evaluate the minimum number of samples such that there is a unique agent whose trajectory matches the given samples according to each of our defined similarity measures. We find that the temporal property and traversal order are important features to identify an agent with a small number of samples, suggesting these properties to be differentiating and sensitive features for user privacy.
Next, we use random sampling to create sketches for answering partial similarity queries efficiently. We consider a network setting in which the mobile agents communicate with nearby checkpoints/base stations with their appearances and possibly timestamps (exactly the same setting in which the two datasets were collected). Each checkpoint uses a probability $p$ to collect the time-stamped location from the agents nearby. We build hashes using these samples, for the different partial similarity measures. Here the collected records for the same mobile agent can be treated as a sequence, sensitive in the temporal order, or as a set, insensitive to the temporal order. In particular, for the time-sensitive partial similarity, we keep the samples as a set of tuples $(t, c)$, with timestamp $t$ and location $c$; for the order-sensitive/order-insensitive partial similarity, we keep the samples as a sequence/set of locations visited, ignoring the timestamps.

We show that with a small sample probability $p$, this reduces the data rate and communication cost from mobile agents to base stations, yet the sketches allow efficient partial similarity queries. Intuitively, two trajectories of high similarity are still likely to share a lot of samples even with significant down-sampling. Answering queries for partially similar trajectories of a given query trajectory becomes a simple table lookup by using a properly chosen bucket size - simply report the trajectories that are hashed to the same bucket. In all three cases, we analyze rigorously the true positive rate (a trajectory that is $\alpha$-similar is retrieved) of this random hash, and empirically evaluated the precision and recall for different choices of parameters. Specifically, to guarantee a good recall ratio, we just need to maintain very small samples of the original trajectory. For example, for $\alpha$-Time-Sensitive similarity, with $\Omega(\frac{1}{p^2} \log(1/\eta))$ samples, trajectories that are $\beta$-similar to the query trajectory, with $\beta \in (\alpha - \delta, \alpha + \delta)$ can be retrieved with probability $1 - \eta$.

For order-sensitive and order-insensitive cases, we need $\Omega(\frac{1}{\sqrt{l}} \log(1/\eta))$ samples to obtain a good recall ratio, where $l$ is the length of the trajectory in the dataset. In practice, we do not need such many samples.

Last, we implemented our framework and algorithms to evaluate similarity queries for both datasets. Precision and recall are two factors that we mainly focus on. There
are three parameters that have affect on the performance, including similarity parameter $\alpha$, sample probability $p$ and threshold coefficient $k$, which is related to $\delta$, setting the similarity threshold for retrieved instances. In our evaluation, we find that for trajectories that cover a time scale of a week, the sample probability could be less than $0.01 \sim 0.1$, for both recall and precision to be higher than $90\%$, with an appropriate threshold coefficient $k$. This leads to a substantial amount of saving in both communication and data processing costs. The running time can be reduced by $70\%$ approximately for Time-Sensitive similarity and Order-Insensitivity similarity. For Order-Sensitivity similarity, the running time can be reduced by more than $90\%$.

In the following, we start by reviewing the related work in Section 2.2. In Section 2.3, we introduce existing measurements in curve similarity and discuss the reason why they are not suitable for real applications. Then, we proposed our framework and concept of partial similarity in Section 2.4. At the same time, we talk about how to compare two trajectories and analyze the performance guarantee in this section. The simulations are presented in Section 2.5 and Section 2.6 concludes this paper.

### 2.2 Related Work

**Distances between Two curves.** There has been a lot of work on computing distances between two trajectories or curves. For two discrete trajectories of length $l_1, l_2$ respectively, the Hausdorff distance can be computed in time $O(l \log l)$, with $l = l_1 + l_2$. For the Fréchet distance, Alt and Godau presented an $O(l_1 l_2 \log(l_1 l_2))$ algorithm for two polygonal curves with $l_1$ and $l_2$ vertices respectively [15, 80]. Over the decades, studies on the Fréchet distance have developed into a rich field of research, in which many generalizations and variants are studied [14, 41, 45, 62]. However, the quadratic worst-case complexity of Alt and Godau’s algorithm had only a log-factor improvement [40]. Bringmann recently presented strong evidence that the Fréchet distance has no strongly subquadratic algorithms, by proving that any such algorithm would yield a breakthrough for the Satisfiability problem (specifically it would break the Strong Exponential Time Hypothesis) [37]. Similar lower bounds exist for Dynamic Time Warping distance [38]. Thus, in a data set of $n$ trajectories, finding the near neighbors
of a given query, implemented in the naive manner, will run in $O(nl^2)$ time. This is computational quite heavy for the kind of dataset we consider.

**Curves of Special Types.** Due to the high computational cost of the Fréchet distance for general curves, researchers have thus focused on curves of special properties, such as backbone curves [23], $k$-straight curves [16], $\phi$-low density curves [62], and $c$-packed curves [62]. While these notions improved our understanding of characteristics of curves that make Fréchet distance computationally hard, Julian and Karl [28] found that real-world trajectories may not have these properties, for example, in the data set of the ACM SIGSPATIAL GIS Cup 2017, derived from GPS traffic data in San Francisco Bay Area.

**Similarity and Hashing.** A useful algorithm in practice is to efficiently answer similarity-based queries to a trajectory database. For example, given a query curve $q$, the $k$-nearest neighbor query asks for finding the curves that are the top $k$ most similar ones to $q$; the $r$-range query asks for all trajectories within distance $r$ from $q$ [54, 11].

A commonly used idea is Locality Sensitive Hashing (LSH) framework [104]. Indyk used the product metrics [103] to achieve an approximation of $O(\log l + \log \log n)$ for the trajectory with nearest neighbor discrete Fréchet distance search with query time of $O(I^{O(1)} \log n)$, where $l$ is the length of the trajectory and $n$ is the number of trajectories. To improve the storage of data structure and query time, a new LSH scheme was proposed by Drimel and Silvestri [63]. Their idea is to snap trajectories to a grid placed with random shift, or apply a fixed sequence of random perturbations to the vertices of a trajectory before snapping them to nearest grid points, which achieves a constant approximation factor. This is tested on real world data sets in [44]. Similarly, the recent work by Maria et al. [24] proposed to snap trajectories to randomly placed disks. When two trajectories have Hausdorff or Fréchet distance less than $\frac{2R}{a}$, the lower bound of probability that they have the same hash value is $1 - O\left(\frac{R^2 D I}{A}\right)$, where $R$ is the radius of the disk, $D$ is the number of disks, $A$ is the area of the region and $a$ is a constant. This probability bound is good when the trajectories are long.
Fig 2.2: Distribution of distances and partial similarities in Zhengzhou. The blue curve and the orange curve describe the distance of two trajectories of two consecutive days of the same agent. The green line show the distance between the trajectories of two randomly chosen agents in one day. For partial similarity measures, we consider two sample points (without timestamps) match if they are within distance \( r \), with \( r \) to be the \( x \)-axis of the plot. We mark the 50 percentile by the vertical dotted line.

**Addressing outliers.** There was also work in the literature trying to address the issue of outliers in distance calculation. In *partial Fréchet distance* [41], given a threshold \( \delta \), the similarity of two curves is defined as the total length of longest subcurves that are matched with Fréchet distance of at most \( \delta \). It can be computed in \( O(l^3 \log l) \) time under the \( L_1 \) and \( L_\infty \) norm, but computation under the \( L_2 \) norm is still a problem. Alternatively, *Fréchet distance with shortcuts* [61] considers traversals allowing shortcuts [42] or forward jumps [25]. The first one means that the detours are replaced by line segments. They need to decide whether the shortcut Fréchet distance is within the threshold \( \delta \). In the general case, the shortcuts can start and end at any point on the input curve. They provided a 3-approximation algorithm to solve it. For the vertex-restricted case, the shortcuts have to start and end at input vertices. There is an exact algorithm. Both algorithms run in \( O(l^3 \log l) \) time. The forward jump simply skip the detour and and gave an \( O(l^3 \log^3(l)) \)-time deterministic algorithm, but the forward jump is only allowed in one curve. However, all of these algorithms have high computation times which are not suitable to our applications.
**Curve Simplification.** In managing large trajectory datasets curve simplification is often adopted, where the goal is to construct an approximate curve that is close to the original. The Douglas-Peucker algorithm [60] is probably the most popular algorithm of this type. A recent work uses topological persistence to simplify trajectories, and find significant turns at different resolutions [114].

Compared to these work, our work is closely tied to the setup of trajectory sensing scenario. Our random sampling could be considered as curve simplification as well, with guarantee of preserving partial similarity measures to other trajectories. Also our hashing could be considered as locality sensitive hash for partial curve similarity with a better probability bound than the previous solution [24]. If the length of trajectories are almost the same, our work improve the lower probability bound to $1 - 1/e^{O(l)}$, where $l$ is the length of trajectories.

### 2.3 Challenges

Many data mining applications require a similarity measure as a basic routine. We first introduce the classical similarity measurements, Hausdorff and Fréchet distance. We assume a discrete setting in which each trajectory is a sequence of points in the plane.

**Definition 2.3.1. (Hausdorff Distance):** Consider two trajectories $T_1 = \{u_1, u_2, \cdots, u_{l_1}\}$ and $T_2 = \{v_1, v_2, \cdots, v_{l_2}\}$ of length (number of vertices) $l_1$ and $l_2$, respectively. The Hausdorff distance is the maximum of the distance from vertices on one trajectory to the nearest vertex on the other trajectory:

$$H(T_1, T_2) = \max\{\max_{u \in T_1} \min_{v \in T_2} d(u, v), \max_{v \in T_2} \min_{u \in T_1} d(u, v)\}$$

where $d(u, v)$ is the distance between vertex $u$ and $v$.

The Hausdorff distance ignores the traversal order of vertices on a trajectory. The Fréchet distance explicitly considers the order of vertices on a trajectory. We first define the concept of a traversal:

**Definition 2.3.2. (Traversal):** Given two trajectories $T_1$, $T_2$, of size $l_1$, $l_2$ respectively, a traversal $\tau = \{(i_1, j_1), (i_2, j_2), \cdots, (i_l, j_l)\}$ is a sequence of pairs of indices...
referring to a pairing of vertices from two trajectories with the properties:

1. $i_1 = 1, j_1 = 1, i_l = l_1$ and $j_l = l_2$

2. $\forall (i_k, j_k) \in \tau: (i_{k+1} - i_k) \in \{0, 1\}$ and $(j_{k+1} - j_k) \in \{0, 1\}$

3. $\forall (i_k, j_k) \in \tau: (i_{k+1} - i_k) + (j_{k+1} - j_k) \geq 1$

Definition 2.3.3.  
(Discrete Fréchet distance): Let $\mathcal{T}$ be the set of all traversals between two trajectories $T_1$ and $T_2$. The discrete Fréchet distance $DF(T_1, T_2)$ between them is:

$$DF(T_1, T_2) = \min_{\tau \in \mathcal{T}} \max_{(i_k, j_k) \in \tau} d(u_{i_k}, v_{j_k})$$

Both measurements, taking min-max or max-min values, are sensitive to outliers. In Figure 2.2, we show the distribution of Hausdorff distance and Fréchet distance in our trajectory datasets. The green lines describe Hausdorff distance and Fréchet distance between the trajectories of two random chosen agents in one day. About half of the pairs have distance more than $20km$, which is more than the radius of the city. The blue lines consider the trajectories of the same agent in two consecutive days. Indeed the trajectories of the same agent in different days are shown to be more similar to each other. By relaxing to partial similarities (shown by the orange curves), the distance thresholds of achieving $\alpha$-similarity shows much higher level of differentiation of the trajectory pairs.

2.4 Trajectory Sensing, Analysis and Query

We consider the setting in which $n$ mobile agents move within a geographical region with $m$ wireless checkpoints $C = \{c_1, c_2, \cdots, c_m\}$. The checkpoints, being roadside units, WiFi access points or cellular base stations, are assumed to be connected to each other by existing infrastructure. The mobile agents periodically communicate with the checkpoints with their IDs (tagged with timestamps) to nearby checkpoints. The data records may also be appearances only (with or without timestamps) or may carry detailed GPS locations, if the mobile agents have GPS data.
The trajectory set of these mobile agents is represented as \( T = \{T_1, T_2, \cdots, T_n\} \). Each trajectory is a sequence composed of pairs of time stamp and location (either the location of the checkpoint visited by the mobile agents or the GPS location of the mobile agent). The length of trajectory \( T_i \) can be represented as \( l_i \).

In the query process, given a query trajectory \( T_q \), which might be a trajectory of one agent in the data set or a sequence of checkpoints and possibly appearance timestamps, we wish to find trajectories in the dataset \( T \) that are similar to \( T_q \). There are many different ways to define the similarity between two trajectories. We will now define rigorously our similarity measure.

### 2.4.1 Partial Similarity

The motivation for partial similarity is to avoid the measure being overly sensitive to outliers.

First, we talk about Time-Sensitive similarity. A trajectory records the appearance of a mobile agent in the neighborhood of checkpoints and its appearance time. It is a set of tuples \((t, c)\), which means that the agent visited the checkpoint \( c \) at time slot \( t \). Time-Sensitive similarity measures the degree of two agents appearing near the same checkpoint at the same time slot. We denote the number of common tuples in two trajectories by \( CC(T_q, T_i) \).

**Definition 2.4.1. Time-Sensitive Similarity:** Given a query trajectory \( T_q \) and a similarity threshold \( \alpha \), \( 0 \leq \alpha \leq 1 \), the trajectory \( T_i \) is \( \alpha \)-Time-Sensitive similar to the query trajectory \( T_q \) if \( \frac{CC(T_q, T_i)}{\min\{l_q, l_i\}} \geq \alpha \).

Next, we relax the time dimension to consider only the order that an agent visits different checkpoints, termed the Order-Sensitive similarity. A trajectory only records the appearances of a mobile agent in the neighborhood of checkpoints as a sequence of checkpoints visited. The Order-Sensitive similarity is to capture the shared sequential pattern in two trajectories by finding the longest common subsequence between two sequences. It is similar to the definition of Fréchet distance. We denote the length of the longest common subsequence as \( LCS(T_q, T_i) \).
Definition 2.4.2. **Order-Sensitive Similarity**: Given a query trajectory $T_q$ and a similarity threshold $\alpha$, $0 \leq \alpha \leq 1$, the trajectory $T_i$ is $\alpha$-Order-Sensitive similar to the query trajectory $T_q$ if $\frac{\text{LCS}(T_q, T_i)}{\min\{l_q, l_i\}} \geq \alpha$.

Last, we only record the checkpoints visited by agents, ignoring timestamp or traversal order. A trajectory is a set of checkpoints visited. The Order-Insensitive similarity is to consider the common checkpoints visited by two agents, denoted as $CN(T_q, T_i)$. Now we provide the definition of Order-Insensitive similarity:

Definition 2.4.3. **Order-Insensitive Similarity**: Given a query trajectory $T_q$ and a similarity threshold $\alpha$, $0 \leq \alpha \leq 1$, the trajectory $T_i$ is $\alpha$-Order-Insensitive similar to the query trajectory $T_q$ if $\frac{CN(T_q, T_i)}{\min\{l_q, l_i\}} \geq \alpha$.

These partial similarities are more robust to noises or perturbations than the discrete Hausdorff distance and Fréchet distance. In the sensing process, it is common to miss some points or add some noise points in the trajectory, which might increase Hausdorff distance and Fréchet distance, but may not influence the partial similarity much.

**2.4.2 Random Collection**

The current system of trajectory sensing collects location records on the order of every $10 - 15$ seconds. This produces a huge volume of data, resulting in a heavy burden for communication, storage, and analysis. A natural question we ask is, can we reduce data sampling rate/size and develop an efficient query scheme with performance bounds for data mining application?

First, we propose our sampling sketches to collect the trajectories from the checkpoints. The sample probability $p$ is set to reduce the data size. In each collection time slot, each checkpoint decides to collect the information from the agents with probability $p$ and do nothing otherwise. Then, for different settings, we store the trajectories and compute the similarity in different ways:

(1) **Time-Sensitive Similarity**: For any trajectory $T_i$ as a set of tuples, each tuple can be selected with probability $p$ according to the checkpoint and the time slot. If a tuple is selected, it is stored in subtrajectory $H_i$, with the same data structure
- Checkpoint (not activate at that time)
- Checkpoints (activate at that time)

Fig 2.3: Process of sampling and computation: There are two agents who visit 10 checkpoints. When the agent arrive at a checkpoint, with probability \( p \), the checkpoint collects the information from all nearby agents. The sample probability \( p \) is set as 0.4. Then, for different query cases, we represent the sampled trajectories in different ways. According to the definitions, we compute the similarities between the sampled trajectories and the original trajectories.

with \( T_i \). Then we can set a hash table for each tuple, i.e. \( < t, c > \) as the keys of hash tables. To compute Time-Sensitive similarity between two trajectories, we just need to go through the shorter trajectory and check how many tuples are collide with the other trajectory in the hash tables. It is a linear algorithm.

(2) Order-Sensitive Similarity: The trajectory \( T_i \) is stored as a sequence or a vector in the form of \( < c_{i1}, c_{i2}, \ldots, c_{i} > \). To randomly collect this trajectory, we also store the subtrajectory as a sequence or a vector. As computing Order-Insensitive similarity between two trajectories, we have to use dynamic programming to solve the longest common subsequence between them, which is a quadratic-time algorithm.

(3) Order-Insensitive Similarity: The trajectory is stored as a set, containing all the checkpoints visited, as \( \{ c_{i1}, c_{i2}, \ldots \} \). With a sample probability \( p \), we collect the subtrajectory as a set. Using these sets, we maintain a hash table, with the checkpoints as the keys. For each checkpoint, it stores all the agents who visit it. To compute the Order-Insensitive similarity, we take all the trajectories of agents and check whether they visit the checkpoints which are along the query trajectory \( T_q \). We can get the satisfied trajectories quickly. It is a linear algorithm, depended on the number of
samples in the dataset and the number of checkpoints.

The process of our framework is shown in Figure 2.3. The nodes are checkpoints in the city. The red nodes represent when the agent arrives, it records the appearance at that time. The sample probability for the checkpoints is 40%. Through the random collection, according to the query case, we store the trajectories in different ways, as the representations in Figure 2.3. We can see the lengths of trajectories are compressed. Then, we compute the similarity using two sampled trajectories $H_1$ and $H_2$. The result is close to the ground truth.

2.4.3 Performance Analysis

In this section, we analyze the relationship between the similarity of the sampled sub-trajectories and that of the original trajectories. In the following analysis, we mainly use Chernoff Inequality to bound the variance and probability error. Thus, we first introduce Chernoff Inequality.

**Theorem 2.4.1. Chernoff Bound:** Let $X_1, X_2, \ldots, X_t$ be i.i.d. random variables with range $[0, c]$ and expectation $\mu$. Then, if $X = \frac{1}{t} \sum_i X_i$ and $0 < \delta < 1$,

$$P[|X - \mu| \geq \delta \mu] \leq 2 \exp \left(-\frac{\mu \delta^2}{3c}\right)$$

In this part, we mainly focus on Time-Sensitive similarity. Given a query trajectory $T_q$ and another trajectory $T_i$, both of them are sequences of tuples, including the appearance time and checkpoints visited. The parts of trajectories collected by the checkpoints are denoted by $H_q$ and $H_i$ respectively. We use the methods in Section 2.4.2 to compute the Time-Sensitive similarity between $H_q$ and $H_i$.

**Theorem 2.4.2.** Given a query trajectory $T_q$ and another trajectory $T_i$ with time-sensitive similarity at least $\alpha$, with sampling probability $p$, the sampled trajectories are $H_q$ and $H_i$ with length $l'_q$ and $l'_i$ respectively. If $pl \geq \frac{3\log(2/\eta)}{\alpha \delta^2} = \Omega(\frac{1}{\delta^2} \log(1/\eta))$, where $l$ is the shorter length of $T_q$ and $T_i$, by Chernoff inequality, we have

$$\text{Prob} \left[ \frac{CC(H_q, H_i)}{\min\{l'_q, l'_i\}} - \alpha \right] \geq \delta \alpha \leq 2 \exp \left(-\frac{pl\alpha \delta^2}{3}\right) \leq \eta$$
Proof. Suppose the query trajectory is the shorter one and its length is $l$, the expected number of samples in $H_q$ is $pl$. Since $\frac{CC(T_q, T_i)}{\min\{l_q, l_i\}} \geq \alpha$, the number of tuples shared in trajectories $T_q$ and $T_i$ are at least $\alpha l$. Each of these tuples has a probability $p$ to be selected into the sampling sketch – the checkpoint with probability $p$ to take samples of nearby agents and if one tuple is taken by a nearby checkpoint, the one on the other trajectory must be collected within the same time slot as well. Thus, the expected number for $CC(H_q, H_i)$ is $\alpha pl$. Let $X_j = 1$ if the $j$-th pair of $H_q$ are matched with the pair in $H_i$, and 0 otherwise. Let $X = \sum_{j=1}^{pl} X_i$. We have

$$Pr\{|\frac{|(V_q, V_i)|}{\min\{|V_q|, |V_i|\}} - \alpha p| \geq \delta \alpha p\} = Pr\{|\frac{X}{pl} - \alpha p| \geq \delta \alpha p\} \leq 2 \exp\left(-\frac{\mu p^2 l \delta^2}{3}\right) \leq \eta$$

By keeping subtrajectories of length $\Omega\left(\frac{\log(1/\eta)}{\delta^2}\right)$, we can use $\frac{|CC(T_q, T_i)|}{\min\{l_q, l_i\}}$ to estimate $\frac{|CC(H_q, H_i)|}{\min\{l_q, l_i\}}$ with probability at least $1 - \eta$, where $\delta$ is the variance bound and $\eta$ is the probability bound.

We have similar theorems for Order-Sensitive similarity and Order-Insensitive similarity.

**Theorem 2.4.3.** Given a query trajectory $T_q$ and another trajectory $T_i$ with order-sensitive similarity at least $\alpha$, with sampling probability $p$, the sampled trajectories are $H_q$ and $H_i$ with length $l_q'$ and $l_i'$ respectively. If $pl \geq \sqrt{\frac{3l \log 2/\eta}{\alpha \delta^2}} = \Omega\left(\frac{1}{\delta} \sqrt{l \log(1/\eta)}\right)$, where $l$ is the shorter length of $T_q$ and $T_i$, by Chernoff inequality,

$$Prob\left|\frac{LCS(H_q, H_i)}{\min\{l_q', l_i'\}} - \alpha p\right| \geq \delta \alpha p \leq 2 \exp\left(-\frac{p^2 l \alpha \delta^2}{3}\right) \leq \eta$$

**Theorem 2.4.4.** Given a query trajectory $T_q$ and another trajectory $T_i$ with order-insensitive similarity at least $\alpha$, with sampling probability $p$, the sampled trajectories are $H_q$ and $H_i$ with length $l_q'$ and $l_i'$ respectively. If $pl \geq \sqrt{\frac{3l \log 2/\eta}{\alpha \delta^2}} = \Omega\left(\frac{1}{\delta} \sqrt{l \log(1/\eta)}\right)$, where $l$ is the shorter length of $T_q$ and $T_i$, by Chernoff inequality,

$$Prob\left|\frac{LCS(H_q, H_i)}{\min\{l_q', l_i'\}} - \alpha p\right| \geq \delta \alpha p \leq 2 \exp\left(-\frac{p^2 l \alpha \delta^2}{3}\right) \leq \eta$$
For Order-Sensitive similarity and Order-Insensitive similarity, the only difference with Time-Sensitive similarity is that, for matched samples in both trajectories, they may be not within the same time slot. Thus, these matched samples are collected in the subtrajectories $H_q$ and $H_i$ with probability $p^2$. That is the reason that we need more random samples for the same error bound. In our simulations, the number of samples needed is not increased by too much. One reason is that a trajectory may repeatedly visit the same checkpoint and the matched samples in the subtrajectories are not necessarily coming from the samples at the same time slot.

2.5 Evaluation by Simulation

In this section, we present experimental evaluation of our distributed trajectory collection framework. Section 2.5.1 is about the setup of our experiments including the hardware, dataset and baseline algorithms used as references. In Section 2.5.2, we analyze the performance of the sampling based sketches, in particular, we investigate how the sampling probability $p$ and the threshold $k$ affect the performance (recall and precision) in similarity based queries. In Section 2.5.3, the running time of our framework is compared with the framework without sampling. In Section 2.5.4, we include some advice for practitioners, on how to choose the parameters for the best performance and running time. In Section 2.5.5, we use the sketches and similarity measures to analyze the individuality property of human trajectories. We use Time-Sensitive similarity, Order-Sensitive similarity and Order-Insensitive similarity as the agents’ signatures and test how many samples are enough to identify a special agent.

The main take-away messages from our experiments are:

- For a given $\alpha$, by choosing a proper parameter of sampling probability $p$ around $1\% \sim 10\%$, we could achieve high recall and precision ($> 90\%$) by adjusting the threshold coefficient $k$ (Section 2.5.2).

- The sampling probability could be taken as a very small number ($\sim 1\%$) when the trajectories are long while still achieving comparable performance. This means
significant savings in running time for computing partial similarity. For Time-Sensitive similarity and Order-Insensitive similarity, the running time can be reduced by more than 70%. For Order-Sensitive similarity, the running time can be reduced by 99% for long trajectories (Section 2.5.3).

- We randomly select several samples from the trajectory as its signature. Then we check whether these samples are enough to identify a unique agent. We found that for Time-Sensitive similarity and Order-Sensitive similarity, only a constant number of samples are enough to identify the agent. For Order-Insensitive similarity, much more samples (proportional to the length of trajectories) are needed (Section 2.5.5).

### 2.5.1 Experimental Setup.

**Hardware.** We implemented our algorithm in C++ with OpenMP, using GCC 7.4.0. We ran the experiments on a Ubuntu 18.04.03 machine equipped with Intel(R) Core(TM) i7-8559U @2.70GHz (4C8T), and 32GB of RAM.

**Datasets.** The trajectories are collected from electric motorbikes in two cities, Zhengzhou and Wenzhou, China, shown in Table 2.2. The trajectories includes a list of points with vehicle ID, system time, longitude and latitude. For each city, we have more than one-month of data. In the pre-processing step, we remove agents whose trajectories in one day are too sparse (fewer than 20 samples in Zhengzhou and fewer than 50 samples in Wenzhou). Given that each agent may not travel for a long time in one day, we concatenate the trajectories of 7 days to form a trajectory of one-week long for each agent. Then we only keep trajectories that contain more than 10,000 samples.

The dataset from Zhengzhou is appearance-based. The location in a vertex on the trajectory is taken as the location of the checkpoint that has recorded such appearance. There are 5,228 checkpoints in Zhengzhou. For Wenzhou, the dataset is GPS location-based. Thus, it is very unlikely that two agents report exactly the same location. We generate 900 checkpoints in Wenzhou and each checkpoint has a surveillance radius of 0.5km. This way we transform the GPS-based trajectory data to appearance-based.
<table>
<thead>
<tr>
<th>Dataset</th>
<th>Period</th>
<th># of Agents</th>
<th>Length</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>min</td>
<td>median</td>
<td>max</td>
<td></td>
</tr>
<tr>
<td>Zhengzhou</td>
<td>Day</td>
<td>36,951</td>
<td>20</td>
<td>423</td>
<td>5,907</td>
<td></td>
</tr>
<tr>
<td>Zhengzhou</td>
<td>Week</td>
<td>2,085</td>
<td>10,000</td>
<td>11,477</td>
<td>56,592</td>
<td></td>
</tr>
<tr>
<td>Wenzhou</td>
<td>Day</td>
<td>274,307</td>
<td>50</td>
<td>110</td>
<td>6,175</td>
<td></td>
</tr>
<tr>
<td>Wenzhou</td>
<td>Week</td>
<td>241</td>
<td>10,026</td>
<td>15,499</td>
<td>84,225</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.2: characteristics of the datasets used in our experiments

**Baseline Algorithm.** For Time-Sensitive similarity, Order-Sensitive similarity, and Order-Insensitive similarity, we have implemented the classical algorithms to obtain the ground truth similarity. We enumerate all pairs of trajectories in the dataset and calculate the similarity value $\alpha$. For the Time-Sensitive similarity, given that all the checkpoints are stored in a temporal order, we can compute the similarity in linear time. For Order-Sensitive similarity, we compute the longest subsequence between two trajectories. For Order-Insensitive similarity, we just count the number of common checkpoints in these two trajectories.

**Evaluation.** We implemented the proposed algorithms based on C++ and our source code is publicly shared on Github [3]. We provide a tool-chain of our framework, including data collection, similarity computation and performance analysis. Here are the main steps in our tool-chain.

1. **Process Dataset:** The data points are collected with the agents’ ID, system time, monitoring time, latitude and longitude. These entries of data are not necessarily ordered. Thus, we construct the trajectory for each agent in a temporal order and remove outliers (with unrealistic speed or at impossible locations).

2. **Parameter Setting:** There are two datasets, Zhengzhou and Wenzhou, that can be selected. Then, we need to select the measurement method to compare two trajectories, including Time-Sensitive similarity (TS), Order-Sensitive similarity (OS) and Order-Insensitive similarity (OI). According to the corresponding measurement, the sample probability $p$ should be given.
### Table 2.3: Parameters used in Figure 4.

<table>
<thead>
<tr>
<th>Index</th>
<th>City</th>
<th>Measure</th>
<th>$\alpha$</th>
<th>$p$</th>
<th>$k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fig. 2.4(a)</td>
<td>Zhengzhou</td>
<td>TS</td>
<td>0.3</td>
<td>0.05%~5%</td>
<td>1</td>
</tr>
<tr>
<td>Fig. 2.4(b)</td>
<td>Zhengzhou</td>
<td>OS</td>
<td>0.8</td>
<td>0.05%~5%</td>
<td>1</td>
</tr>
<tr>
<td>Fig. 2.4(c)</td>
<td>Zhengzhou</td>
<td>OI</td>
<td>0.8</td>
<td>0.05%~30%</td>
<td>1</td>
</tr>
<tr>
<td>Fig. 2.4(d)</td>
<td>Zhengzhou</td>
<td>TS</td>
<td>0.3</td>
<td>3%</td>
<td>0.5~1.4</td>
</tr>
<tr>
<td>Fig. 2.4(e)</td>
<td>Zhengzhou</td>
<td>OS</td>
<td>0.8</td>
<td>1%</td>
<td>0.5~1.2</td>
</tr>
<tr>
<td>Fig. 2.4(f)</td>
<td>Zhengzhou</td>
<td>OI</td>
<td>0.8</td>
<td>1%</td>
<td>0.4~1.2</td>
</tr>
<tr>
<td>Fig. 2.4(g)</td>
<td>Wenzhou</td>
<td>TS</td>
<td>0.5~0.8</td>
<td>0.5%~20%</td>
<td>1</td>
</tr>
<tr>
<td>Fig. 2.4(h)</td>
<td>Wenzhou</td>
<td>OS</td>
<td>0.5~0.8</td>
<td>0.5%~20%</td>
<td>1</td>
</tr>
<tr>
<td>Fig. 2.4(i)</td>
<td>Wenzhou</td>
<td>OI</td>
<td>0.6~0.9</td>
<td>0.5%~20%</td>
<td>1</td>
</tr>
</tbody>
</table>

(3) **Comparison Algorithm:** We output the similarity measures among all agent pairs in the dataset as the ground truth. We calculate the similarity between the sampled trajectories, and then compare them with the ground truth. Now, the threshold coefficient $k$ can be set and we can retrieve the satisfied instances and analyze the performance (recall and precision).

#### 2.5.2 Precision and Recall

In this part, we analyze how the parameters (sample probability $p$, threshold coefficient $k$, and similarity parameter $\alpha$) affect the performance of similarity based queries. Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall (also known as sensitivity) is the fraction of the total number of relevant instances that are retrieved. The parameters of these experiments are shown in Table 2.3. We only execute the algorithms on the dataset of one-week long trajectories, as the saving is more significant on long trajectories. For different cases, we set an appropriate value for $\alpha$ such that we retrieve less than 5% of all trajectories.

**Sample probability.** First, Figure 2.4(a), 2.4(b) and 2.4(c) show the effect of sample probability $p$ on precision and recall. These simulations are all based on Zhengzhou. In
Fig 2.4: Performance in terms of precision and recall of our framework on Zhengzhou and Wenzhou. The effect of sample probability $p$, threshold coefficient $k$ and similarity parameter $\alpha$ on the performance.
general, recall goes up when sample probability $p$ goes up. As the number of samples increases, by Chernoff Inequality, the variance bound $\delta$ and the probability bound $\eta$ can be reduced. In these cases, the variance bound $\delta$ does not change. Thus, the probability bound $\eta$ drops and recall is higher. On the other hand, precision varies in different ways, because precision mainly depends on the dataset and the parameter setting. Take Figure 2.4(c) as an example: since we may underestimate the similarity parameter between two sampled trajectories, some trajectories with a low similarity (about 0.7) may also be retrieved. The retrieved instances contain all true positive instances but also false positive instances, leading to low precision.

For Time-Sensitive similarity, when the sample probability is about 0.5%, recall is higher than 95%. For Order-Sensitive similarity, the sample probability is close to 1% for the same recall. For Order-Insensitive similarity, when the sample probability is about 10%, recall is close to 1. According to our analysis in Section 2.4.3, the number of samples needed for Time-Sensitive similarity is the fewest compared with the other two similarity measures. The results in experiments confirm this as well. To achieve a good recall performance, we just need fewer than 1% of samples for Time-Sensitive similarity and Order-Sensitive similarity, and fewer than 10% of samples for Order-Insensitive similarity. In either case the data size can be reduced significantly.

**Threshold Coefficient.** According to the analysis in Section 2.4.3, the parameter $\delta$ is to set the variance bound for the retrieved instances. If we want to retrieve all trajectories with high similarity with the query trajectory, we set a similarity threshold as the multiplication of the threshold coefficient $k$ and the expected similarity parameter. Trajectories with similarity to the given query higher than the threshold are retrieved.

The effect of this threshold coefficient $k$ on the performance is shown in Figure 2.4(d), 2.4(e) and 2.4(f). In general, with $k$ increasing, recall is decreased and precision is increased. A higher threshold results in fewer retrieved instances. Thus, when $k$ is small, the retrieved instances contain most of the true positive instances as well as some false positive instances. It makes high recall and low precision. When the value of $k$ is large, most of the retrieved instances are true positive instances, but the number of retrieved instances is fewer than the number of true positive instances.
Thus, it leads to low recall and high precision.

The threshold coefficient can be used to balance the trade-off between recall and precision. The curves for recall and precision intersect when $k$ is around $0.8 \sim 1.2$. At this point, both recall and precision are higher than 90%. The optimal point for $k$ is depended on the dataset and the parameter setting.

**Similarity parameter.** For different similarity parameters $\alpha$, the relationship of precision and recall is shown in Figure 2.4(g), 2.4(h) and 2.4(i). Each color represents a similarity parameter. Nodes of the same color correspond to different sample probabilities (from 1% to 30%). Both precision and recall range from 0 to 1, with 1 being the best. Thus, the closer a node is to the top right corner, the better the performance. In general, most nodes are close to the top right corner, with high recall and high precision. The other nodes not in the top right corner have very small collection probabilities and relatively poor result.

**Comparison with LSH.** We compared with Locality Sensitive Hashing (LSH) [63] to find the $(c, r)$-near neighbor trajectories. The experiment is carried out on Shenzhen taxi dataset as it is publicly available [4]. Each trace is for one taxi for one day sampled at a rate of 30 seconds. We split each trace into trajectories, such that each trajectory is a continuous occupied trip for 1 hours. We apply their algorithm to build hash function families and set the pair of trajectories whose Fréchet distance is within 1km as the ground truth. We compare the performance between their algorithm and our solution. As shown in Figure 2.5, we can see that the performance of LSH is slightly better than our solution. However, LSH requires to store all the appearance of the agents, which might need much larger storage capacity and is not suitable for applications.

### 2.5.3 Time Efficiency

Now, we consider the running time of our algorithms for different query cases, shown in Table 2.4. This process contains the sampling process (collecting samples from trajectories) and computing the similarity between all pairs of agents. Thus, the running time is affected by the sample probability $p$, which determines the data size. We ran
Fig 2.5: Shenzhen: Order-Sensitive similarity compared with result by LSH

<table>
<thead>
<tr>
<th>Similarity</th>
<th>Dataset</th>
<th>sample probability p</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>1%</td>
</tr>
<tr>
<td>Time-Sensitive</td>
<td>Zhengzhou</td>
<td>6ms</td>
</tr>
<tr>
<td></td>
<td>Wenzhou</td>
<td>17ms</td>
</tr>
<tr>
<td>Order-Sensitive</td>
<td>Zhengzhou</td>
<td>4s</td>
</tr>
<tr>
<td></td>
<td>Wenzhou</td>
<td>0.4s</td>
</tr>
<tr>
<td>Order-Insensitive</td>
<td>Zhengzhou</td>
<td>10ms</td>
</tr>
<tr>
<td></td>
<td>Wenzhou</td>
<td>8ms</td>
</tr>
</tbody>
</table>

Table 2.4: Running time of different query cases with varied collection probabilities

simulations on two datasets, Zhengzhou and Wenzhou, with three similarity measures. The sample probability $p$ varies from 1% to 30%.

For Time-Sensitive similarity and Order-Insensitive similarity, the running time grows linearly with the sample probability $p$. Generally, we set the sample probability $p$ less than 10%, achieving recall more than 95%. It means that the running time could be reduced by about 70% approximately, which is a significant improvement. Another interesting observation is that the running time between Zhengzhou and Wenzhou is different in Time-Sensitive similarity and Order-Insensitive similarity. Although the number of agents in Zhengzhou is greater than that in Wenzhou, the average length of trajectories in Wenzhou is longer than that in Zhengzhou. For Time-Sensitive similarity, we enumerated all pairs and compare them in linear time, which depends on
the length of trajectories. For Order-Insensitive similarity, we just need to go through all the samples once. There are more samples in Zhengzhou because there are more agents. Thus, the running time of Zhengzhou is larger than Wenzhou’s.

For Order-Sensitive similarity, the running time grows quadratically with the sample probability $p$. Without subsampling, we compute the Order-Sensitive similarity between all pairs of agents in about 10 hours for Zhengzhou dataset, and about 2 hours for Wenzhou. With sample probability $p$ as 10%, we can compute these similarities within several minutes, which is a big improvement for real applications.

2.5.4 Advice to Practitioners

We have some tips for practitioners on how to adjust parameters to obtain desired precision and recall. The similarity parameter $\alpha$ is typically determined by applications. A high similarity parameter $\alpha$ leads to a low number of retrieved instances and true positive instances. We empirically found the distribution of pairs of trajectories by the similarity parameter $\alpha$ based on Zhengzhou data set, shown in Figure 2.6. The highest similarity parameter appears to be around 0.6 and there are only a few such pairs suggesting 0.6 to be a good threshold to test.

![Figure 2.6](image)

Fig 2.6: Zhengzhou: Distribution of the trajectory pairs based on similarity parameter $\alpha$

With the similarity parameter $\alpha$ fixed, we determine sample probability $p$. From
<table>
<thead>
<tr>
<th>Similarity Type</th>
<th>One-Day</th>
<th>One-Week</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time-Sensitive similarity</td>
<td>4(0.8%)</td>
<td>4(0.03%)</td>
</tr>
<tr>
<td>Order-Sensitive similarity</td>
<td>46(15%)</td>
<td>52(0.5%)</td>
</tr>
<tr>
<td>Order-Insensitive similarity</td>
<td>167(56%)</td>
<td>1032(9.4%)</td>
</tr>
</tbody>
</table>

Table 2.5: The number of samples to uniquely identify a trajectory

our experience, in most cases, the sample probability $p$ calculated by the inequality from the Chernoff Inequality is large enough to achieve recall $\geq 95$%.

Now, with a good sample probability $p$ and a good recall rate, the precision might be poor, as in Figure 2.4(c). In this case, we can adjust the threshold coefficient $k$. By increasing $k$, precision could be increased significantly with a slight reduction to recall. According to our simulations, we can adjust the value of $k$ from 0.8 to 1.2 to obtain a rate of at least 90% for both recall and precision.

2.5.5 Human Mobility Pattern

We ran experiments on using different partial similarity measures in identifying the unique agent. Human movement trajectories over a long period of time are surprisingly unique with strong personal traits. For example, four spatio-temporal points are enough to uniquely identify 95% of the individuals out of a million other [55]. This signature is a special case of our partial similarity, namely, the time-sensitive similarity. We would like to re-examine our data sets using different similarity measures in terms of how much individuality they capture.

Here, we use the dataset of Zhengzhou. In one day, there are 36,951 agents and each one has about 500 samples on average. We use their one-week dataset, with 2,085 agents and each trajectory has about 11,000 vertices on average. In the experiments, we randomly select several samples from the trajectory as the signature of this agent. We want to check whether these samples and other order information could help us to identify this agent uniquely.

For example, we use Time-Sensitive similarity to identify this unique agent. The
similarity parameter $\alpha$ is set to 1. We enumerate all the agents in the dataset and check whether any agent has the same signature. For Order-Sensitive similarity, we just check whether any agent visits all of these checkpoints in the same order as in the signature. For Order-Insensitive similarity, we test whether the agent visits all of these checkpoints in the signature.

For different signatures, we run 1,000 times to get the average successful rate for uniquely identifying an agent. Then we set the threshold as 90% and find the minimum number of samples needed. The result is shown in Table 2.5.

For time-Sensitive similarity, 4 spatio-temporal samples are enough to uniquely identify more than 90% individuals, in both the dataset of daily trajectories and the weekly trajectories. For order-sensitive similarity, 46 and 52 ordered samples are enough to identify most of the agents in the one-day dataset and the one-week dataset. For Order-Insensitive similarity, we need many more samples, 167 and 1,032, for the one-day dataset and the one-week dataset respectively. It is a large fraction, 56% and 9.4%, of the full trajectory length.

These findings show that human trajectories, even with significant down sampling, can still be very sparse and unique with timestamped locations. The visiting order of the locations helps, but not nearly as much. Thus removing the temporal information of a trajectory is crucial for privacy protection.

2.6 Conclusion

In this paper, we proposed a distributed trajectory collection framework to understand the properties of natural human trajectories, individuality and commonality. Due to the large dataset and high computation time of original similarity measurement methods, we design three partial similarities, which are suitable for real-world trajectory analysis. In our framework, we use random sampling to reduce the data size. Through analysis and simulation, we show that with less than 10% of the original trajectories we can get a recall more than 90% and reasonably high precision. Thus, our framework provides an efficient solution for processing massive trajectory data.
Chapter 3
Unsupervised Clustering Algorithm for Human Trajectories

3.1 Introduction

Technology innovation has enabled the possibility to collect a large number of human trajectories. These trajectory traces reveal many interesting characteristics of natural human mobility that are of significance both on an individual level (individual mobility modeling) and on a population level (for gathering and group motion) with numerous applications such as traffic engineering [184, 168], civil planning [129] and public health domains [133]. Trajectory data has been collected by vastly different sensing modalities and has a high variation of spatial and temporal resolution. With dedicated sensors either carried by the agents (e.g., GPS) or carefully instrumented in the environment (e.g., in smart buildings), one can obtain high-resolution spatial-temporal trajectories of high quality and details [68, 144]. On the other hand, a larger category of trajectory traces are inferred from sporadic, discrete occurrences with or without timestamps or orderings – for example, trajectories inferred from proximity with WiFi access points, cellular towers, highway toll stations or public transit ticket booths [213, 191, 134]. This kind of trajectory data is sparse, not uniformly sampled or even disconnected, and lacks details.

Many methods on analysis of trajectory data focus on high-quality trajectory data. There are a lot of rich details and geometric features one can extract, especially with supervised learning. Past work has studied the prediction of the transportation modes [215, 214], turning mode in the crossroad [135], and destination of the trips [30]. These supervised learning methods rely on detailed discriminative statistical features such as velocity [214], acceleration [215, 105], which can only be extracted from fine-grained
trajectories with very dense sampling intervals.

The work in this paper focuses on coarse-grained, sparsely sampled trajectories, or sparse samples of occurrences. We also take an unsupervised setting without any training data or labels. We ask whether one can still infer meaningful clusters in a given set of trajectory traces. A cluster can correspond to trajectories by the same individual or other naturally defined groups (e.g., the same animal species). On the scientific front, we believe that different individuals or species move in a different way and there are hidden features in the traces that can be used to separate them.

3.1.1 Challenges

The immediate challenges in handling sparsely sampled trajectories are to decide on feature selection, distance measurement policies, and clustering algorithms.

**Feature selection.** The first challenge is to find a good representation of an input trajectory. Existing techniques mainly use spatial and temporal features of the trajectory [101, 127]. These low-level position based features are sometimes insufficient to distinguish differences in a small region [203]. Semantic information is taken into consideration, such as the categories of check-in locations [198, 47] and transportation mode parameters [215, 214]. Advanced deep learning techniques can be used to train distinguishing feature vectors [207, 194]. Advanced semantic information requires external labels which are not always available. The neural network approach also falls short in transparency and interpretability. In general, feature design for clustering trajectories is still a non-trivial problem. Although we do expect that mobility trajectories of the same individual tend to be repetitive and regular, there are clearly daily variations and outlier behaviors. It is unclear what are the most discriminating features that separate the mobility patterns of different individuals.

**Trajectory similarity.** To compare two trajectories, classical geometric measures for distances of curves using geometric definitions such as the Hausdorff distance and Fréchet distance [176]. These measures are more suitable to trajectories that are aligned, uniformly and consistently sampled. Examples are trajectories derived from
video footage [203, 21, 132]. For sparsely sampled trajectories with potential missing data and outliers, these measures face challenges. Specifically, these distance measures are extreme measures (of the min-max or max-min type) and capture the worst-case scenario. On a real-world human trajectory dataset, it is nearly impossible for two trajectories to be close to each other all the time. They often come out to be too large (e.g., the radius of the city) to be interesting [76]. Even for two trajectories of the same individual on two consecutive days, the Hausdorff distance and Fréchet distance are on average in the order of several kilometers [191] and thus can not be effectively used to differentiate trajectories from different agents. Motivated by this several recent works [191, 160, 179] define similarity measures for real-world trajectories, by relaxing the dependency on extreme conditions (e.g., by considering partial similarity measures).

**Clustering algorithm.** Existing clustering algorithms are not particularly suitable for trajectory data. Most clustering algorithms in the literature fall into the following two categories. The first one is model-based. One of classical methods is centroid-based methods. The clusters are inherently assumed to be round and the problem is often formulated as an optimization problem to minimize the maximum size of each cluster, such as $k$-center [82], or $k$-means clustering [154]. These algorithms generate less meaningful results when the ground truth has other ‘shapes’ or of different sizes [71]. Another model-based approach assumes the distribution of data points inside each cluster [65], e.g., in the mixed Gaussian model. The clustering algorithm explicitly uses this assumption on input data, which is not applicable for trajectory data.

The other class is density-based methods, e.g., density-based spatial clustering of application with noise (DBSCAN) [66]. The idea is to first identify the core points where the number of points within distance $r$ from each core point is more than a given threshold. Then the cluster grows by including all points that are within distance $r$ from at least one core point in the cluster. The clustering results are influenced by parameter selection and these parameters need to be determined beforehand. A few variations of DBSCAN choose the input parameters automatically, such as Gaussian Means [173], Binary Differential Evolution [112], and Domain Set [98]. These methods mainly help to choose appropriate fixed parameters for different clusters before the
growing process.

We propose our method, which follows the general framework of DBSCAN, but we consider an important improvement in the growing step, that is, adapting the parameter \( r \) automatically in the process. We grow a cluster \( C \) by including the points that are, statistically, believed to be taken from the same distribution that generated \( C \) with high confidence. This allows the clustering algorithm to adapt to different cluster distributions and leads to improved performance (Adjusted Rand Index) on different trajectory datasets. We highlight our contribution below.

3.1.2 Our Contribution

Our goal is to take a non-parametric approach – as we do not have any prior knowledge of the ground truth clusters of trajectory traces (e.g., cluster labels, the number of clusters, the distribution of clusters) – and support a wide range of features and similarity measures on real-world trajectory datasets. Our cluster shall move beyond a centroid-based definition and shall tolerate variation in data density, the existence of outliers, and variations in distributions.

We first take a look at Figure 3.1 to understand where current algorithms fall short. The figure shows three sample datasets and four algorithms: \( k \)-means algorithm, Gaussian mixture models (GMM), DBSCAN and our conformal DBSCAN algorithm. \( k \)-means fails to high-density clusters that are not of a round shape and have varying densities. GMM successfully high-density the three Gaussian clusters but shares similar limitations with \( k \)-means for other non-round clusters. DBSCAN, on the other hand, can successfully recognize ‘gaps’ between clusters. But the algorithm is local – a new data point \( x \) will join an existing cluster \( C \) if \( x \) is considered to be similar (within distance \( r \) for some distance measures) to an existing core data point \( x' \) in \( C \), while \( r \) is a fixed value. Thus, low-density points (in the second and third columns) are either lumped together with the high-density points (if \( r \) is too big) or not recognized by any cluster at all (if \( r \) is too small).

Then, let’s discuss the challenge in trajectory clustering. For trajectory data, there is no standard definition of clusters and we do not have any prior knowledge of the
Fig 3.1: Four algorithms: \textit{k}-means, Mixture of Gaussian (GMM), DBSCAN and conformal DBSCAN on three synthetic datasets. The clusters are shown in different colors.
distribution of trajectories that belong to the same individual. For supervised learning, one can use an empirical distribution of the history data, but this is unavailable for the unsupervised setting. Here we assume that there are $k$ clusters $C_1, C_2, \ldots, C_k$, with $k$ unknown, and the points in $C_i$ are uniformly randomly sampled from a distribution $\pi_i$, which is also unknown. Notice that the distributions $\pi_i$ for different $i$ are sufficiently separated; otherwise, they shall have already been merged to fewer clusters.

We take the DBSCAN algorithm but improve the growing step significantly – starting from an initial set of high-density points, we include new ones with an adaptive radius $r$, which is determined by the current cluster properties. This is achieved by using the conformal prediction framework, a statistical framework to quantify the likelihood of a new data point belonging to a distribution $\pi$ by comparing with random samples independently and identically taken from $\pi$. Thus our algorithm is called conformal DBSCAN algorithm.

The conformal prediction framework [187] is a recent development in statistics and machine learning that quantifies the likelihood of a new data object $x$ belonging to an unknown distribution $\mathcal{D}$, where the only information needed is a set $C$ of data objects randomly sampled from the distribution. The framework does not need any assumption on the knowledge of the distribution $\mathcal{D}$. The conformal prediction framework uses a discrepancy score $A(x, Y)$ which characterizes how different a data object $x$ is relative to a reference set $Y$. If two data objects are similar, they will have similar discrepancy scores with respect to the same reference set. Thus, we compare the new data object $x$ with each object $c \in C$ (with respect to the rest of objects in $C$ as the reference set) – if $x$ is also likely randomly chosen from $\mathcal{D}$, roughly half of the objects in $C$ are expected to have discrepancy scores higher/lower than $x$. The rank of its discrepancy score among all the objects in $C$ provides a rigorous framework to quantify the likelihood of $x$ being an element from $\mathcal{D}$, which also controls the type-I error of labeling. Notice that this allows a wide range of designs for the discrepancy score and covers a wide range of (unknown) distribution $\mathcal{D}$.

To handle the growing phase in DBSCAN we need to modify the conformal prediction framework. Here the initial cluster includes a set of high-density points, instead of
randomly selected from a cluster, which is required in the conformal prediction framework. We provide confidence analysis for this more general setting. We include a new point $x$ if it is within distance $r$ from a point $c \in C$ whose discrepancy score ranks within $[\varepsilon|C|, (1 - \varepsilon)|C|]$ among all points in $C$, where $\varepsilon$ is a small number and is interpreted as the significance level in our work. We reformulate the conformal prediction test to prove that a newly included object belongs to the current cluster with a high significance level $\varepsilon$. In other words, the type-I error of this labeling, false positive rate, is controlled below $\varepsilon$ in the growing process. Our modified conformal prediction framework is applied in an iterative manner to group objects into several clusters.

We also tested our algorithm for different trajectory datasets. For identifying meaningful clusters, we use three different trajectory similarity measures that are robust to missing data and outliers [191]. They can capture different innate features (spatial and temporal features) of mobility trajectories. Three trajectory datasets, including synthetic trajectories and real-world trajectories (wild animal trajectories and individual electric bike trajectories), are tested. Through the experiments, our method significantly outperforms other baseline algorithms ($k$-means and DBSCAN [66]). In the animal species trajectory dataset, our method can separate the trajectories by their species, with Adjusted Rand Index (ARI) value as 0.7449, much higher than the other methods. In the bike trajectory dataset, our goal is to group each individual’s daily trajectories without any prior label. Our method performs well using different similarity measures.

In the rest of this paper, we start by reviewing the previous work, DBSCAN and the conformal prediction framework, in Section 3.2. In Section 3.3, we proposed our conformal DBSCAN algorithm with theoretical analysis. The relationship with DBSCAN is also discussed. The experiments with different trajectory datasets are presented in Section 3.4 and Section 3.5 concludes this paper.
3.2 Review of DBSCAN and Conformal Prediction

In this section, we will review DBSCAN and the conformal prediction framework. We also discuss their limitations for clustering.

3.2.1 DBSCAN

DBSCAN is a heuristic algorithm widely used in data mining and clustering. The main idea of DBSCAN is the following: given a collection of data objects and two predetermined fixed parameters, the radius of neighborhood $r$ (in a distance measure) and the threshold for the number of neighbors $minPts$, the points are gradually placed into clusters. To find one cluster $C$ we take the following steps:

1. **Find the core set:** First, if $x$ has at least $minPts$ points within radius $r$ (including $x$ itself), $x$ and all points within distance $r$ from $x$ are put in $C$. $x$ is called a core point.

2. **Gradually include reachable points:** A point $x$ not yet included in any clusters is added to $C$ if $x$ has at least one core point of $C$ within distance $r$.

The above procedure finds one cluster $C$ in DBSCAN. When $C$ cannot grow anymore, we find another high-density region and repeat. The procedure stops when no more clusters can be identified. The points that are not recognized by any clusters are considered as noise or outlier.

The two parameters in DBSCAN are fixed for all clusters. Later variants to DBSCAN choose different parameters, i.e., the radius $r$ and the threshold $minPts$, for different clusters but the parameters are fixed throughout the growing phase for one cluster. In our algorithm, we use an adaptive parameter $r$ in the growing phase. The choice of $r$ is guided by the points in the current cluster $C$. This is determined by using a revised conformal prediction framework, which is introduced below.
3.2.2 Conformal Prediction

The conformal prediction framework is a statistical test to determine whether a new object belongs to a distribution with \textit{i.i.d.} observation objects. Our discussion focuses on the \textit{Jackknife+ conformal prediction framework} \cite{29}, introduced below.

Let $C = \{c_1, c_2, \cdots, c_m\}$ be a given set of objects from a single cluster with underlying distribution $D$, which is unknown. We would like to check whether or not a new object $c_{m+1} \notin C$ is also a uniform random sample from the same distribution $D$.

First, we define the notion of \textit{discrepancy score} computed by an algorithm or a function $A(x, Y)$, which characterizes the data object $x$ with respect to a reference dataset $Y$, $x \notin Y$. Notice that there is no assumption on this discrepancy score in this framework and our following clustering algorithm. It can be any meaningful measure that characterizes the distance of $x$ with $Y$, the role of $x$ within set $Y$, the influence of $x$ upon $Y$, etc. This discrepancy score can be derived from certain distance measures using extracted features, or an application specific score (e.g., as the output of a preprocessing algorithm such as regression or neural network models). Notice that the discrepancy score can be designed to capture global structural information, such as how close to the other objects. In the following part, we can compare the discrepancy score of all the objects to quantify the probability that this object belongs to the reference set.

Here, the conformal prediction framework works with \textit{any} real-value function $A(x, Y)$ as the discrepancy score. When an appropriate discrepancy score is chosen, the prediction could be more informative. There are some choices proposed previously, such as, the $k$-nearest neighbors algorithm \cite{187}, (kernel) ridge regression \cite{187}, SVM \cite{187}, neural networks \cite{151}, random forest \cite{57} and genetic algorithms \cite{124}.

If the new object $c_{m+1}$ belongs to the cluster $C$, we can compare the discrepancy score $\alpha_{m+1}^{(i)}$ and $\alpha_i$, where $\alpha_{m+1}^{(i)} = A(c_{m+1}, C^{-i})$ and $\alpha_i = A(c_i, C^{-i})$ with respect to the same leave-one-out set $C^{-i} = C \setminus \{c_i\}$, and $c_i \in C$ is a randomly selected object from the cluster $C$. Their order would take a 50-50 chance. In other words, when the new object $c_{m+1}$ is similar to the objects in the cluster $C$, we expect the value of $\alpha_{m+1}^{(i)}$ is comparable to a sizeable fraction of the value of $\alpha_i$, for all $i = 1, 2, \cdots, n$. In the
cases when $\alpha^{(i)}_{m+1}$ is larger or smaller than the majority of $\alpha_i$ values, we would reject the claim that $c_{m+1}$ is the same or similar to the objects in $C$.

Formally, we count the numbers of times that $\alpha^{(i)}_{m+1} < \alpha_i$ and $\alpha^{(i)}_{m+1} > \alpha_i$, respectively. Then we define a \textit{conformity score} in the range of $[0, 1]$ as

$$p(c_{m+1}) = 2 \min\{q^-(c_{m+1}), q^+(c_{m+1})\}$$

(3.1)

where

$$q^-(c_{m+1}) = \sum_{i=1}^{m} \frac{1_{\{\alpha^{(i)}_{m+1} < \alpha_i\}}}{m} + \frac{\sum_{i=1}^{m} 1_{\{\alpha^{(i)}_{m+1} = \alpha_i\}}}{2m}$$

$$q^+(c_{m+1}) = \sum_{i=1}^{m} \frac{1_{\{\alpha^{(i)}_{m+1} > \alpha_i\}}}{m} + \frac{\sum_{i=1}^{m} 1_{\{\alpha^{(i)}_{m+1} = \alpha_i\}}}{2m}$$

(3.2)

and $1(\cdot)$ is the indicator function. When $c_{m+1}$ is similar to $c_i$, we expect that $\alpha^{(i)}_{m+1} \leq \alpha_i$ or $\alpha^{(i)}_{m+1} \geq \alpha_i$ holds by roughly the 50-50 chance. So, if either $q^-(c_{m+1})$ or $q^+(c_{m+1})$ is close 0, it means $c_{m+1}$ is likely different than majority objects $c_i \in C$. In either of the two cases, the conformity score $p(c_{m+1})$ is small and close to 0.

An example is provided in Figure 3.2, where $C$ contains points on a circle. When new points are off the circle (e.g., $c_9$, $c_{10}$), both the conformity scores $p(c_9)$ and $p(c_{10}) \approx 0$. For new points on the circle (e.g., $c_{11}$), the conformity score $p(c_{11}) \approx 0$.

In the case when $c_1, \ldots, c_m$, the points in $C$, as well as the new object $c_{m+1}$ are all independent random draws from the same distribution, it can be shown [29, 201] that

$$\Pr(p(c_{m+1}) \leq \epsilon) \leq \epsilon,$$

(3.3)

where $\epsilon$ is a pre-specified small number in $(0, 1)$. That is, if we use the detection rule to declare “$c_{m+1}$ is a sample point from the same distribution as those in $C$” if and only if “$p(c_{m+1}) > \epsilon$”, then the type-I error to mistakenly reject $c_{m+1}$ is less than $\epsilon$. In fact, some researchers suggested to interpret the conformity score $p(c_{m+1})$ as a p-value for the hypothesis test problem $H_0$: $c_{m+1}$ is \textit{conformal with} $C$ versus $H_a$: $c_{m+1}$ is \textit{not conformal with} $C$, although this test is not formally framed in statistical literatures [200, 201].
Fig 3.2: Illustration of the conformity score. There are 8 points, \( c_1, \ldots, c_8 \) in the reference set \( C \), with the underlying distribution on a circle (shown in pink). The points \( c_9 \) and \( c_{10} \) are checked against a randomly chosen point (say \( c_4 \)). \( A(x, Y) \) is the sum of distances from \( x \) to all points in \( Y \). \( \alpha_9^{(4)} > \alpha_4; \alpha_{10}^{(4)} < \alpha_4 \) and \( \alpha_{11}^{(4)} \approx \alpha_4 \). In fact, both \( p(c_9) \) and \( p(c_{10}) \) are close to zero; while \( p(c_{11}) \) are away from 0.

The above guarantee often requires that the points in \( C \) and new point \( c_{m+1} \) are drawn from \( D \) at random. Intuitively, for a randomly drawn \( x \) from \( D \), the discrepancy score \( A(x, C) \) forms a distribution \( \pi \) in \( \mathbb{R}^1 \). Regardless of what \( \pi \) or \( D \) look like, randomly drawing \( m \) samples from \( D \) will give us a random sample of discrepancy scores from \( \pi \). Thus, we can use the ranking of \( A(c_{m+1}, C) \) with a random sample from \( \pi \) to estimate the likelihood that \( c_{m+1} \) is also taken from the same distribution \( D \). Notice that using the ranking rather than the discrepancy score itself allows for a wide variety of discrepancy functions and embraces inherent robustness. The Jackknife+ framework made a slight adjustment with \( C \) replaced by \( C^{-i} \) when we compare \( c_{m+1} \) against an object \( c_i \) from \( C \), to avoid the influence of \( c_i \in C \) in the discrepancy calculation, which ensures a rigorous mathematical proof of the statistical claims.

However, for the clustering problem, we have no prior knowledge of the distributions and the cluster labels. Thus we do not have a reference set to start with. Generally, we can guess the label of random samples from a truncated distribution – with density function higher than a threshold – and use them as the reference set. This is similar to the first step of DBSCAN of discovering the high-density regions. The issue is that the samples identified from the truncated distribution are not a uniformly random sample from the ground truth cluster. This violates the assumption of conformal prediction. For a new object on the boundary of this truncated distribution, its discrepancy score
is likely to be far away from the majority of the discrepancy score of objects in the reference set. In the following work, we will extend the conformal prediction framework and provide the mathematical explanation for the newly added objects with respect to a truncated distribution.

### 3.3 Conformal DBSCAN

In this section, we propose our algorithm, called *Conformal DBSCAN*, which combines DBSCAN with the conformal prediction framework. We first present the algorithm design and then extend the conformal prediction framework to provide theoretical analysis for our algorithm. Last, we discuss the relation with DBSCAN and try to provide a mathematical explanation for DBSCAN.

#### 3.3.1 Algorithm

Given a collection of data objects $S = \{c_1, \cdots, c_n\}$ in the space $M$, the function $d(c_i, c_j)$ describes the difference between two data objects $c_i$ and $c_j$. Notice that this function does not need to be metric function. We provide the pseudo-code of our conformal DBSCAN algorithm in Algorithm 1. Our algorithm proceeds in three phases.

**Initial Members.** Similar to DBSCAN, we find the high-density region as the initial members of a cluster. Specifically, initialize $\delta = 0$. Given $\delta$ we find the point $x$ with the highest number of points within distance $\delta$. If we cannot find such a point $x$ (i.e., $\delta$ is too small) we increase $\delta$ until we can find one. Then this object and its neighbors are selected as the initial member of a cluster. This step is in Line 3-5 of Algorithm 1.

**Core Objects Determination.** Based on the current cluster members, we include new elements in an iterative manner. In each iteration, we first define the core objects as those whose conformity score is larger than a pre-specified significance level $\epsilon \in (0, 1)$. A smaller value $\epsilon$ results in a larger number of core objects. Generally, we set $\epsilon$ as 5% or 10%. In Algorithm 1 lines 7-8 calculate the discrepancy scores and conformity scores. The first condition in Line 12 is to determine the core objects.

**Adapted Radius and Growth.** Based on the core objects in the current cluster,
ALGORITHM 1: Conformal DBSCAN

Input: The set of data objects $S = \{c_1, \ldots, c_n\}$;

A small number $\epsilon \in (0, 1)$;

Any radius value $\delta$ to select the initial seeds;

Output: The set of clusters $\{C_1, C_2, \ldots\}$

$k = 1$; // Index of the current cluster

while $S$ is not empty set do

    // Initial members
    $N(i) \leftarrow \{c_j|d(c_i, c_j) \leq \delta\}$, for $c_i \in S$;
    $t \leftarrow \arg\max_i |N(i)|$;
    $C_k \leftarrow N(t)$;

    repeat

        // Core objects determination
        $\alpha_i^{(j)} = A(c_i, C_k \setminus \{c_i, c_j\})$, for $c_i, c_j \in C_k$;
        $p(c_i) = \frac{\sum_{c_j \in C_k, i \neq j 1(\alpha_i^{(j)} \leq \alpha_i^{(i)})}}{|C_k| - 1}$;

        // Adapted radius and growth
        $r = \gamma(C_k)$;

        // Radius adaption function
        for $c_i \in C_k$ do
            for $c_j \in S \setminus C_k$ do
                if $p(c_i) \geq \epsilon$ and $d(c_i, c_j) \leq \delta$ then
                    $C_k = C_k \cup \{c_i\}$;
                end
            end
        end

        until There is no $c \in S \setminus C_k$ added into $C_k$;

    $k \rightarrow k + 1$;

    $S = S \setminus C_k$;

end
whose discrepancy scores are ranked in the appropriate range, we include new objects within proximity from the core objects. Our algorithm adapts the radius according to the current cluster, represented by the function $r = \gamma(C)$. Then, all the objects within distance $r$ from the core objects are included in the cluster, as shown in Line 9-13 of Algorithm 1. We will show in the next subsection that these newly included objects have a similar discrepancy score and they are believed to belong to this cluster with a high significance level.

The adapted radius $r$ determines the growth process. In our implementation, we take a data-driven method to adjust it. For each object in the current cluster, we compute the distance to its $k$ nearest neighbor and the largest distance is set as the radius $r$. This radius is changing during the growth and adapts to the property of the current cluster.

### 3.3.2 Theoretical Analysis

In our iterative algorithm, we grow the cluster $C$ one object at a time. Since we start with points in high-density regions (i.e., represented by a truncated distribution), the new point $c_{m+1}$ may not follow the truncated distribution that characterizes the current reference set $C$. Recall that $c_{m+1}$ is within distance $r$ from a point in the core set of $C$, we denote by $\beta = \beta(C, r)$ the upper bound on the difference of the discrepancy scores of $c_{m+1}$ and points of $C$ within distance $r$.

**Definition 3.3.1.** If a new object $c_{m+1}$ and an object $c_i \in C$ are within distance $r$ of each other, define the upper bound on the difference between their discrepancy scores based on the reference set $C^{-\{i,j\}} = C \setminus \{c_i, c_j\}$ by $\beta = \beta(C, r)$:

$$ |A(c_{m+1}, C^{-\{i,j\}}) - A(c_i, C^{-\{i,j\}})| \leq \beta. $$ (3.4)

$\beta = \beta(C, r)$ may depend on the reference set $C$ and the radius $r$.

It suggests that the discrepancy scores of two objects in a $r$-neighborhood, with respect to remainder points in $C$, are controlled with an upper bound $\beta$. For instance, suppose a cluster is formed by random draws from a certain distribution, for the sake
of illustration, say a Gaussian distribution. If the initial cluster and the reference set \( C \) contain points in the high-density region, then the points in \( C \) are from a corresponding truncated Gaussian distribution. The new object \( c_{m+1} \) in the neighborhood of \( C \) and some points at or close to the boundary of \( C \) will be very similar. The discrepancy score of \( c_{m+1} \) and the scores of these close-by boundary points, with respect to the remaining points of \( C \), will not be too far and have an upper bound. It provides a nice feature for the growth process that we do not actually need to calculate the discrepancy score of all the new data objects.

With this definition we modify the conformal prediction framework to handle the truncated distribution. Suppose, for a new object \( c_{m+1} \), there exists a random draw \( \tilde{c} \in C \) such that \( c_{m+1} \) and \( \tilde{c} \) are in a \( r \)-neighborhood. We define a modified conformal score of \( c_{m+1} \):

\[
\tilde{p}(c_{m+1}) = 2 \min\{\tilde{q}^-(c_{m+1}), \tilde{q}^+(c_{m+1})\} \tag{3.5}
\]

where

\[
\tilde{q}^-(c_{m+1}) = \frac{\sum_{x_i \neq \tilde{c}, 1 \leq i \leq m} 1\{\tilde{\alpha}^{(i)}_{m+1} \leq \tilde{\alpha}_i + \beta\}}{m-1}
\]

\[
\tilde{q}^+(c_{m+1}) = \frac{\sum_{x_i \neq \tilde{c}, 1 \leq i \leq m} 1\{\tilde{\alpha}^{(i)}_{m+1} \geq \tilde{\alpha}_i - \beta\}}{m-1} \tag{3.6}
\]

and \( e^{(i)}_{m+1} = A(c_{m+1}, C \setminus \{c_i, \tilde{c}\}) \) and \( \tilde{\alpha}_i = A(c_i, C \setminus \{c_i, \tilde{c}\}) \), for \( i \in \{i : c_i \neq \tilde{c}\} \). Then we have the following theorem

**Theorem 3.3.1.** Under the setting discussed above, we have

\[
\Pr(\tilde{p}(c_{m+1}) \leq \epsilon) \leq \epsilon.
\]

Thus, the type-I error (i.e., the probability of not identifying \( c_{m+1} \) as a sample in the cluster when in fact \( c_{m+1} \) is in the cluster) is less than \( \epsilon \).

**Proof.** Without loss of generality, suppose \( \tilde{c} = c_1 \). We define \( \tilde{C} = C \setminus \{\tilde{c}\} = \{c_2, \ldots, c_m\} \). Since \( \tilde{c} = c_1 \) and \( c_{m+1} \) are in a \( \delta \)-neighborhood, by Definition 3.3.1 we have,

\[
|e^{(i)}_{m+1} - \tilde{\alpha}^{(i)}_1| \leq \beta, \text{ for } 2 \leq i \leq m \tag{3.7}
\]
where \( \hat{\alpha}_{m+1}^{(i)} = \mathcal{A}(c_{m+1}, \tilde{C} \setminus \{c_i\}) \) and \( \hat{\alpha}_i = \mathcal{A}(c_i, \tilde{C} \setminus \{c_i\}) \), for \( i = 2, \ldots, m \).

Furthermore, \( \tilde{c} = c_1 \) is one of the random objects in the set \( C \), so \( c_1 \) is conformal with objects in set \( \tilde{C} \). By treating \( c_1 \) as a ‘new’ object and comparing it to reference set \( \tilde{C} \), the regular conformity score defined in Equation (1) and (2), is

\[
p(c_1) = 2 \min \{q^-(c_1), q^+(c_1)\}
\]

where

\[
q^-(c_1) = \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} < \hat{\alpha}_i\}}{m - 1} + \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} = \hat{\alpha}_i\}}{2(m - 1)}
\]

\[
q^+(c_1) = \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} > \hat{\alpha}_i\}}{m - 1} + \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} = \hat{\alpha}_i\}}{2(m - 1)}
\]

It is followed by Equation (3.3) that \( \Pr(p(c_1) \leq \epsilon) \leq \epsilon \).

Now, we check the modified conformal score \( \tilde{p}(c_{m+1}) \). We first compare the relationship between \( \tilde{q}^-(c_{m+1}) \) and \( q^-(c_1) \).

\[
\tilde{q}^-(c_{m+1}) = \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} \leq \hat{\alpha}_i + \beta\}}{m - 1}
\]

\[
= \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} - \hat{\alpha}_i \leq \hat{\alpha}_1^{(i)} + \beta - \hat{\alpha}_1^{(i)}\}}{m - 1}
\]

\[
\geq \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} - \hat{\alpha}_i \leq 0\}}{m - 1}
\]

\[
= \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} \leq \hat{\alpha}_1^{(i)}\}}{m - 1}
\]

\[
\geq \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} < \hat{\alpha}_i\}}{m - 1} + \frac{\sum_{i=2}^{m} 1\{\hat{\alpha}_1^{(i)} = \hat{\alpha}_i\}}{2(m - 1)}
\]

\[
= q^-(c_1)
\]

The variable \( \hat{\alpha}_1^{(i)} \) is inserted on the both sides on Line 3. According to Equation (3.7), the right side of Line 3 is not less than 0 and Line 4 is obtained. Similarly, we can get \( \tilde{q}^+(c_{m+1}) \geq q^+(c_1) \). It leads to

\[
\tilde{p}(c_{m+1}) = 2 \min \{\tilde{q}^-(c_{m+1}), \tilde{q}^+(c_{m+1})\}
\]

\[
\geq 2 \min \{q^-(c_1), q^+(c_1)\}
\]

\[
= p(c_1)
\]

then, we can conclude this theorem, i.e.,

\[
\Pr(\tilde{p}(c_{m+1}) \leq \epsilon) \leq \Pr(p(c_1) \leq \epsilon) \leq \epsilon
\]
Based on this theorem, when the object \( c_{m+1} \) is close enough to one of the randomly selected conformal objects in the reference set, its discrepancy score is bounded. Then, its corresponding modified conformity score is also controlled. If we reject the neighboring \( c_{m+1} \) to be included in the cluster when its modified conformity score \( \tilde{p}(c_{m+1}) < \epsilon \), then the type-I error of this labeling is controlled to be less than \( \epsilon \).

Theorem 3.3.1 holds for any bound \( \beta \) in Definition 3.3.1. If \( \beta \) is too large the detecting rule based on the corresponding \( \tilde{p}(c_{m+1}) \) can be overly conservative, but the way Algorithm developed allows us to define the modified conformity score \( \tilde{p}(c_{m+1}) \) used in our detection rule as the one computed using the tightest bound that satisfies Definition 3.3.1.

### 3.3.3 Relation with DBSCAN

Conformal DBSCAN can be considered as a generalization and improvement to the vanilla DBSCAN. In both DBSCAN and conformal DBSCAN we define core points and include new points by proximity to these core points. Suppose we define the discrepancy score for \( x \) as the distance to the \((minPts)\)th nearest neighbor in the reference set \( C \).

For DBSCAN, the core points have discrepancy scores of less than \( r \). For conformal DBSCAN, the core objects have conformity score \( p(c) \) larger than \( \epsilon \). The bound in Definition 3.3.1 is \( r \) in a metric space. The directly-reachable objects in DBSCAN are similar to the objects whose modified conformity score \( \tilde{p}(c) \) larger than \( \epsilon \).

The analysis in the previous section for conformal DBSCAN provides theoretical explanations on why points within distance \( r \) of the core objects can be regarded as members of the cluster. These newly included objects belong to the cluster with a high significance level. In conformal DBSCAN, the radius \( r \) is adjusted by using the points in the clusters. Conformal DBSCAN is better empowered to handle clusters with non-uniform distribution.

Suppose the discrepancy score is the sum of the distance to \( k \) nearest neighbors in the reference set. When a new data point is added in the cluster, we need to compute
its discrepancy score and update the other existing objects’ discrepancy scores. We can maintain a min heap for each data point to store the k smallest distance. Each data point needs to compute the discrepancy score once in at most \( O(n \log k) \) time, where \( n \) is the number of objects. Then we can use the hash sort to select the core set, which is done in linear time \( O(n) \).

Regarding the asymptotic behavior on \( n \), the number of trajectories, DBSCAN has a running time of \( O(n^2) \) and conformal DBSCAN has running time \( O(n^2 \log k) \), where the extra factor, \( \log k \), comes from using the min-heap to compute the discrepancy scores. We take \( k \) as a small constant. Thus, this factor can be ignored in the implementation.

### 3.4 Evaluation by Simulation

In this section, we present an experimental evaluation of our conformal DBSCAN algorithm in the trajectory clustering tasks. Section 3.4.1 is about the setup of our experiments including datasets and baseline algorithms as references. The clustering performance is discussed in Section 3.4.2. We also presented the application of conformal prediction framework for supervised classification tasks in Section 3.4.3. In Section 3.4.4, we provide some discussions and observations on trajectory clustering, classification with respect to sampling rate and similarity measures.

#### 3.4.1 Experimental Setup

**Hardware.** We implemented our algorithm in Python with version 3.8. We ran the experiments on the machine equipped with Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, and 32GB of RAM.

**Dataset.** We perform our conformal DBSCAN algorithm on three trajectory datasets, which are collected in different settings, with specific features. The description of them is shown in Table 3.1 and the detail is introduced below:

1. **Synthetic Trajectory:** A public dataset\(^1\) of simulated trajectories was previously created by Piciarelli et al. [156]. It consists of 1,000 randomly generated datasets

\(^1\)http://avires.dimi.uniud.it/papers/trclusl/
Table 3.1: Dataset Description

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Synthetic</th>
<th>Animal</th>
<th>Human</th>
</tr>
</thead>
<tbody>
<tr>
<td># Agents</td>
<td>260</td>
<td>102</td>
<td>633,194</td>
</tr>
<tr>
<td># Clusters</td>
<td>5</td>
<td>3</td>
<td>Varying</td>
</tr>
<tr>
<td># Records</td>
<td>4,160</td>
<td>14,990</td>
<td>50,873,192</td>
</tr>
<tr>
<td>Avg # Records/Agent</td>
<td>16</td>
<td>147</td>
<td>80</td>
</tr>
</tbody>
</table>

for comparison with discords. Each of these datasets contains 260 2-dimensional trajectories of length 16, i.e., exactly 16 points. There are 5 different clusters, and each one contains 250 normal trajectory. The remaining 10 are stray trajectories that do not belong to any cluster. Figure 3.3 shows three cases of this dataset.

(2) Animal Trajectory: The animal movement dataset\(^2\) has been generated by the Starkey project. This dataset contains the radio-telemetry location (with other information) of elk, deer, and cattle from spring through fall for the year 1993 through 1996. It includes more than 287,000 observations acquired from animals in a natural setting. We extract the coordinates, as well as the record time information, from the telemetry data observed in June 1995, shown in Figure 3.4(a). The trajectories can be divided into three classes by species. Elk has 38 trajectories and 7,117 data points; deer has 30 trajectories and 4,333 data points; cattle has 34 trajectories and 3,540 data points. Therefore, there are on average 147 points for each animal over one month.

(3) Human Trajectory: The trajectories are collected from electric motorbikes in Wenzhou, China. They include a list of points with vehicle ID, system time, longitude, and latitude. For each vehicle, there are 30-days trajectories in June 2018. The location on the trajectory is taken as the location of the checkpoints that have recorded such an appearance. There are 5,228 checkpoints in the city with an area \(110\text{km} \times 70\text{km}\). There are 633,194 people’s trajectories are recorded and each person has about 80 points on average in one day. In the experiment, we randomly select a subset of vehicles and do the clustering task for all the trajectories of these vehicles.

\(^2\)http://www.fs.fed.us/pnw/starkey/data/tables/
**Baseline Algorithm.** For the trajectory clustering tasks, we have implemented the classical unsupervised clustering algorithms, $k$-means and DBSCAN, to obtain the baseline results. The distance between trajectories is defined as $1$ minus their similarity. We use three novel similarity measures proposed in the previous work [191], Time-Sensitive Similarity (TSS), Order-Sensitive Similarity (OSS) and Order-Insensitive Similarity (OIS). Note that these three similarity measures are non-metric functions. At the same time, we also use two classical similarity measures, Hausdorff distance and Fréchet distance. A brief introduction of these similarities is shown:

1. **Time-Sensitive Similarity:** Each trajectory is a sequence of time-stamped locations visited by the agent. Two trajectories are $\alpha$-Time-Sensitive similar if at least $\alpha$ fraction of samples on the shorter trajectory are the same ones (or nearby with a specified distance/time threshold) on the other trajectory, including time stamps and locations.

2. **Order-Sensitive Similarity:** Each trajectory is an ordered sequence of locations visited by the agent. Two trajectories are $\alpha$-order-sensitive similar, if an $\alpha$ fraction of samples on the shorter trajectory are matched with the samples on the other trajectory in the corresponding order. This could be considered as a partial measure for Fréchet distance.

3. **Order-Insensitive Similarity:** Each trajectory is a set of visited locations. Two trajectories are $\alpha$-Order-Insensitive similar if at least $\alpha$ fraction of locations on the shorter trajectory are matched with the other trajectory. It ignores the time dimension and visiting order issue. This could be considered as a partial measure for Hausdorff distance.

4. **Hausdorff Distance:** Let $X$ and $Y$ be two non-empty subsets of a metric space $(M, d)$. Their Hausdorff distance $d_H(X, Y)$ is defined as

$$d_H(X, Y) = \max\{\max_{x \in X} \min_{y \in Y} d(x, y), \max_{y \in Y} \min_{x \in X} d(x, y)\}$$

where $d(x, y)$ quantifies the distance between points $x \in X$ and $y \in Y$. 

5. **Fréchet Distance**: Let \( f : [0, m] \to \mathbb{R}^k \) and \( g : [0, n] \to \mathbb{R}^k \) be two polygonal curves or sequences. The Fréchet distance is defined as

\[
d_F(f, g) = \min_{\alpha, \beta} \max_{s \in [1, m+n]} \{d(f(\alpha(s)), g(\beta(s)))\}
\]

where \( \alpha \) and \( \beta \) range over all discrete non-decreasing onto mappings of the form
\[
\alpha : [1 : m + n] \to [0 : m], \beta : [1 : m + n] \to [0 : n].
\]

In the experiments, we only use Hausdorff distance and Fréchet distance as the similarity measures in the human trajectory dataset. It shows that both of them are not suitable for our datasets due to the sparse sampling rates and outliers. Thus, we mainly use the first three partial similarities to evaluate the performance of our conformal DBSCAN.

**Evaluation Metric.** To evaluate the classification results, we use precision and recall to measure the performance. Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while the recall (also known as sensitivity) is the fraction of the total number of relevant instances that are retrieved.

For the trajectory clustering tasks, adjusted for chance measures are widely used to compare partitions/clustering of the same dataset. The Rand Index (RI) computes a similarity measure between two clusterings by considering all pairs of samples and counting pairs that are assigned in the same or different clusters in the predicted and true clusterings. It can be viewed as a measure of the percentage of correct decisions made by the algorithm, computed by

\[
RI = \frac{TP + TN}{TP + FP + TN + FN}
\]

where \( TP \) is the number of true positives, \( TN \) is the number of true negatives, \( FP \) is the number of false positives, and \( FN \) is the number of false negatives. The Adjusted Rand Index (ARI) is the corrected-for-chance version of the Rand Index. It is thus ensured to have a value close to 0 for random labeling independently of the number of clusters and samples, and exactly 1 when the clusterings are identical (up to a permutation). In addition, the Adjusted Mutual Information (AMI) based on Shannon information theory is another popular metric in the clustering community. Simone et
al. [166] proposed that ARI should be used when the reference clustering has large equal size clusters; AMI should be used when the reference clustering is unbalanced and there exist small clusters. Our three trajectory datasets are much suitable for the ARI metric, as our evaluation metric, because all the clusters in our datasets have a similar size.

**Evaluation Step.** We implemented the proposed algorithms based on Python and our source code will be publicly shared after acceptance. We provide a tool-chain of our framework, including data cleaning, similarity computation and clustering algorithms. Here are the main steps in our tool-chain:

1. **Data Preprocessing:** For the human trajectory dataset, the sampling rate is about 10 seconds. There are many outliers in the trajectories, such as unrealistic speed or impossible locations. We need to clean these human trajectory data to make them realistic.

2. **Similarity Computation:** For three partial similarities, the corresponding efficient algorithms are introduced in the previous work [191]. When there are thousands of trajectories in the dataset, we could use the uniform sampling to obtain an approximation similarity, which is bounded by Chernoff Bound. The running time could be reduced by more than 70% with more than 90% accuracy.

3. **Clustering Algorithm:** Based on the similarity matrix, we implemented k-means, DBSCAN and our conformal DBSCAN to output the clustering results with different parameters of the corresponding algorithms. Then compared with the ground truth, we analyze the performance (ARI, recall, and precision).

### 3.4.2 Trajectory Clustering

First, we test our algorithm under the unsupervised settings without any prior knowledge. Thus, for the k-means algorithm, we use the “elbow” method to select the optimal number of clusters. It plots the explained variance as a function of the number of clusters and picks the elbow of the curve as the number of clusters to use. While for the DBSCAN algorithm, all the available parameters combinations (the radius of neighborhoods and the minimum number of points within radius) are enumerated to find the
Fig 3.3: Clustering for synthetic trajectories: The first column is the ground truth of three cases. 250 trajectories belong to 5 clusters in different colors, and the outliers are in black. The clustering results by $k$-means, DBSCAN, Conformal DBSCAN are shown in the following columns.

<table>
<thead>
<tr>
<th>Method</th>
<th>$k$-Means</th>
<th>DBSCAN</th>
<th>Conformal DBSCAN</th>
</tr>
</thead>
<tbody>
<tr>
<td>OSS</td>
<td>0.5792</td>
<td>0.8703</td>
<td><strong>0.8970</strong></td>
</tr>
<tr>
<td>OIS</td>
<td>0.5774</td>
<td>0.8196</td>
<td><strong>0.8507</strong></td>
</tr>
</tbody>
</table>

Table 3.2: Clustering result for synthetic trajectories

best performance. In our algorithm, we use the sum distance to $k$ nearest neighbors as the discrepancy score. For any pair of trajectories within the difference $r$ for these three similarity measures, the bound $\beta$ in Definition 3.3.1 can be written as the function $\beta(C, r) = \frac{kr}{1-r}$. The radius is adjusted with the largest $\lceil \frac{k}{2} \rceil$-nearest neighbor distance to include more trajectories. Then the performance of three datasets are shown below:

**Synthetic Trajectory.** Given that trajectories are generated without specific time information, we only compute the order-sensitive similarity and the order-insensitive similarity between trajectories. There are three cases are shown in Figure 3.3. In Figure 3.3(a), the distribution of raw trajectories is demonstrated, including normal trajectories in 5 clusters with different colors and outliers (black). The ground truth
is that the normal trajectories can be clustered correctly and the rest outliers can be grouped in one cluster.

The clustering results by $k$-means algorithm are shown in Figure 3.3(b). We can see that the clusters are not even. Some trajectories in the same reference clusters are grouped into different clusters, and some in the different reference clusters are mixed. In addition, it cannot distinguish the outlier trajectories. Figure 3.3(c) demonstrates the clustering results by DBSCAN. It is much better than the results of $k$-means, however, we can still find some mis-clustering trajectories, in the second and third cases. The outlier trajectories can be distinguished. The clustering results by conformal DBSCAN algorithm are shown in Figure 3.3(d). In some detailed parts, our results are better than the result of DBSCAN, but part of outlier trajectories are grouped into one cluster. The reason is that we do not know how many outlier trajectories there are. After clustering the normal trajectories, part of outlier trajectories compose of a new cluster, but cannot extend any more. If we have some prior knowledge or we set a small initial radius threshold, our algorithm can perform better in distinguishing these outlier trajectories.

The overall performance of 1,000 trajectory datasets, using two similarity measures, is shown in Table 3.2. It is obvious that the results of our conformal DBSCAN algorithm perform best, with a higher ARI value. In addition, the performance using the order-sensitive similarity is better than using the order-insensitive similarity, because the traversal order is a good factor not only for separating different clusters, but also enlarging the difference between trajectories in one cluster in this dataset. In the following experiment, we will focus on the comparison between DBSCAN and conformal DBSCAN algorithm.

**Animal Trajectory.** Figure 3.4(a) shows all the trajectories of these three animal species in one month. Their activity regions are partially overlapped, which brings difficult to cluster their trajectories. The trajectory distribution of each specie is shown in Figure 3.4(b). They are regarded as the ground truth.

Here, we use time-sensitive similarity to compare two animals' trajectories. As processing dataset, it is found that each animal has about 15 data points, but these data points are not recorded evenly. Thus, we set the time slot as one day, which means
Fig 3.4: Clustering results for animal trajectories. (a) and (b) are the trajectories of three species of animals (Red: Elk, Blue: Deer, and Green: Cattle). (c) and (d) show the clustering results by DBSCAN and conformal DBSCAN algorithm.
that if two animals visit the same region on the same day, it should be counted into the similarity measure. First, we can see the clustering result by DBSCAN, in Figure 3.4(c). The trajectories of deer and elk are not separated. While using our conformal DBSCAN algorithm, the clustering results become much better in Figure 3.4(d), separating deer and elk. The ARI value for DBSCAN is 0.6020, while the ARI for our algorithm is 0.7449, which is much higher.

**Human Trajectory** In this dataset, each person has about 30 daily trajectories, which contain the regular daily routine. For the trajectories of the same person on different days, the three similarities can maintain a high value, close to 1, because they are robust to outliers in the trajectories. Thus, we randomly select some people and retrieve all the trajectories of these people from the dataset. Our goal is to separate all the trajectories into several clusters that belong to each corresponding person, using the similarity
Fig 3.6: Clustering results of DBSCAN and conformal DBSCAN algorithm using different similarity measures.

First, let’s check the performance of using Hausdorff distance and Fréchet distance as the similarity measures, shown in Figure 3.5. In terms of the ARI score, conformal DBSCAN using Hausdorff distance and Fréchet distance performs much worse than that using Time-Sensitive similarity especially when the number of agents in the dataset is increased. The reason is that even for the same agent, her trajectories in two days have a large Hausdorff distance and Fréchet distance. They cannot illustrate the commonality of the trajectories with the sparse sampling rates.

We check the number of people from 10 to 200, and pick some results shown in Figure 3.6. It shows the ARI value of clustering by DBSCAN and conformal DBSCAN algorithm using three similarity measurements. First, we can see that with the incremental number of people, the ARI value is reduced. The reason is that the similar trajectories belonging to different people appear more, which is hard to separate them. Another reason is that the trajectories of the same person are not totally similar, because they might have some unusual trajectories. For these trajectories, it is almost impossible to cluster them correctly. When there are 200 people selected, DBSCAN can only generate about 60 clusters and the conformal DBSCAN algorithm gets about 130 clusters, which is a large improvement.
Second, using the same similarity measure, the conformal DBSCAN algorithm outperforms DBSCAN, especially when the number of people is getting large. The reason is that DBSCAN has a predetermined and fixed radius to include trajectories, while conformal DBSCAN algorithms could adjust this radius according to the local density in the current cluster.

In addition, the same clustering algorithm shows different performance with three similarity measures. Clustering with time-sensitive similarity performs better than that with order-sensitive and order-insensitive similarity. This suggests that the traversal order and time information can enlarge the gap between the similarity of the same people and the similarities of two different people, which makes the clustering accuracy better. The time information might not be important in the human trajectory dataset.

**Running time.** Table 3.3 shows the running time of two methods, DBSCAN and conformal DBSCAN, after the similarity matrix is obtained. As the analysis in the previous section, the running time of conformal DBSCAN is only modestly higher than the classical DBSCAN with essentially the same scaling behavior in the number of trajectories.

### 3.4.3 Trajectory Classification

In this section, we present the application of conformal prediction framework for (supervised) classification problem. For each dataset, we select a part of trajectories as the reference set, and the rest are the test set. The discrepancy score is the sum distance to $k$ nearest neighbors. With different given significance levels $\epsilon$, we compute the conformity score for each test trajectory based on each reference cluster. If the conformity score is larger than $\epsilon$, it will contain the cluster label. In this way, one trajectory might have multiple labels and we check the precision and recall metrics to show its performance.

**Synthetic Trajectory.** For each cluster of 50 trajectories, we randomly choose 20 trajectories as the reference cluster. The rest and the outlier trajectories are the test set. The significance level $\epsilon$ is set as 0.05, 0.1 and 0.15, the classification results with
Similarity | OSS | OIS
---|---|---
\(\epsilon\) | 5% | 10% | 15% | 5% | 10% | 15%
Precision | 0.8067 | 0.9385 | 0.9652 | 0.7329 | 0.9021 | 0.9455
Recall | 0.9617 | 0.9237 | 0.8842 | 0.9617 | 0.9237 | 0.8842

Table 3.4: Classification result of synthetic trajectories

order-sensitive and order-insensitive similarity are shown in Table 3.4.

First, we can see the recall value is higher than \(1 - \epsilon\). It is guaranteed by the framework that the prediction set contains the true cluster label with \((1 - \epsilon)100\%\) significance level. Then, with a higher significance level \(\epsilon\), the precision value is increased. The reason is that the prediction set becomes smaller, so the fraction of relevant trajectories among retrieved trajectories becomes higher. In addition, using order-sensitive similarity has a better precision than the order-insensitive similarity, because considering traversal order, the gap between different clusters is enlarged, reducing the false positive instances.

**Human Trajectory.** For each person, 15 daily trajectories are taken as the reference set, with which we wish to successfully identify the other 15 trajectories of the same agent. Given a significance level \(\epsilon = 0.1\), we check the number of people from 10 to 200 with three similarity measures. The recall value is about 0.82, a little lower than the expected result 0.9. The reason is that 30 daily trajectories of one person in the ground truth are not a perfect cluster in practice. There are some trajectories much different from the regular moving pattern.

The precision results are shown in Figure 3.7. With the number of people increasing, the precision result using time-sensitive similarity is reduced substantially, while the precision with other measures still remains high (above 0.7). The Order-Sensitive Similarity and Order-Insensitive Similarity seem to be good similarity measures for this task because their performance is close when the number of people is above 150. The reason is that the variance of the discrepancy scores of conformal trajectories are large, making it easy for the test trajectory to get a comparable discrepancy score in different
On the other hand, with trajectories of 200 agents, if each trajectory is simply labeled by the label of the reference cluster with highest conformity score, the accuracy is not bad (around 60% – 60% trajectories are assigned the correct label).

3.4.4 Discussion

Our experiments show that the three partial similarity measures have respective merits in different tasks and datasets. The pattern is persistent when we reduce the sampling rate. Similar clustering results are obtained. Thus, it is important to choose appropriate measures to analyze the mobility trajectories.

With technology innovation, trajectory collection efforts often strive for fine-grained mobility data with accurate localization and dense sampling intervals. High sampling rate increases computation time. The experiments carried out here do not observe significant benefit with increases in sampling rate, for the purpose of recognizing trajectories of different individuals.

Similarly, it is tempting to incorporate more features, in order to improve the performance of clustering and classification tasks. Our experimental results suggest that
we should take this with caution. The main factor is to recognize the gap between similarity of two trajectories within the same clusters and across different clusters. With an increasing number of participants, the performance of time-sensitive similarity actually drops, while the performance of order sensitive and order insensitive similarities remain to be high. This suggests that the time-sensitive information could be too detailed to recognize mobility patterns of different individuals.

3.5 Conclusion

In this paper, we proposed an algorithm combining DBSCAN algorithm and the conformal prediction framework. Conformal DBSCAN improves DBSCAN algorithm by automatically adapting the parameters to the data being handled, guided by theoretical analysis of an improved conformal prediction framework. Conformal DBSCAN shows superior performance on a variety of artificial and real-world trajectory datasets. We remark that conformal DBSCAN is a generic unsupervised clustering algorithm that has potential on other unstructured, messy data. In future work we plan to test conformal DBSCAN on other datasets, by incorporating more sophisticated discrepancy scores such as those obtained from deep learning encoders.
Chapter 4
Information Spreading on Real Mobility Data

4.1 Introduction

The global pandemic of a novel coronavirus (SARS-CoV-2) has swept through nearly all countries since December 2019. Due to the high infection rate and therefore high demand for medical resources, most countries have adopted large-scale interventions such as business lockdowns and restricted movements. These intervention methods have proven to be effective, successfully reducing the number of peak daily infected cases and the total number of infected cases so far, as shown by both direct observation from real data [70, 100] as well as indirect metapopulation models and simulations [48, 118].

While these intervention methods have aided in controlling the pandemic, they are not sustainable over long periods. To address this challenge, we must understand the effect of human movement and if a less disruptive intervention strategy can control the epidemic.

The relation between mobility and spread of COVID-19 has been observed in several recent studies [26, 118, 130] that show empirical evidence of a correlation between aggregate mobility and number of cases. These studies do not consider the social costs [142] (for example, the reduction in social and economic activities) and are at large scales that do not provide the understanding necessary for more targeted interventions [100]. Classical epidemiology approaches can model behaviour at scale of agents – these include models such as SIR or SEIR [192] and their variations [123, 157, 48], data-driven models [102], and multi-agent models [69]. By amending the epidemiological models with characteristics of COVID-19 and incorporating the effect of real human movements, we can gain insights into targeted strategies for interventions and the social costs of interventions.
The objective of our study is to understand the effect of human movements in the spread of the virus. We use an agent-based simulation where a mobile agent can probabilistically infect other agents that are in proximity – such as at the same venue. We consider movements of the real people captured in three types of mobility data – check-ins at seven different cities, WiFi connection events on a university campus, and GPS traces of electric bikes – representing different scales, behaviors, and modalities. Across these diverse circumstances, we observe several common features in human movement, the spread of the epidemic, and the effect of interventions.

4.2 Method

Our work takes the approach of multi-agent models but incorporates real mobility traces to study the probable spread of an infectious disease. The study involves three data sets that represent three typical social settings. (Refer to Supplementary material for detailed data description).

- **Foursquare check-ins.** We use check-in data [204, 205] from the Foursquare mobile application, which captures snapshots of human mobility in popular public spaces. The dataset includes seven major cities across the world (New York, Chicago, Los Angeles, London, Tokyo, Istanbul, and Jakarta), and contains a total of 2,293,716 check-ins from 24,068 individuals in 397,610 venues over a period of 140 days.

- **University campus [177]:** WiFi log data on connections of mobile devices with nearby WiFi access points from a large university campus (Tsinghua University) is a representative case of mobility in a large university setting. The dataset contains 106,975 average daily localization points from 47,359 individuals.

- **Personal electric bikes [191]:** Densely sampled GPS mobility traces of 46,078 personally owned electric bikes over the period of a month provides a unique dataset as they capture unstructured personal mobility patterns not tied to specific public spaces or institutions.

All three data sets have fine-grained location resolution of within 10s of meters, which
allow us to investigate inter-personal interactions and interactions of individuals and public venues. These datasets collectively cover three representative slices of human mobility that play important roles in the spread of infectious diseases. Infection can propagate person to person at a meeting event (when two agents are in proximity), or through a venue (when a susceptible agent visits a venue recently visited by an infectious agent). Note that these datasets were collected pre-COVID between 2012 and 2018, representing “normal” human movements. We ran simulations on the three datasets with various intervention strategies and show their impact in slowing down the spread. We study the total number of individuals infected under various intervention strategies, the growth rate of new infections, the maximum number of infected individuals at any time (height of the peak), and the time-varying reproduction number \( R_t \). The social cost of an intervention strategy is measured as the number of check-ins and meetings lost due to the strategy.

The Foursquare datasets contain time-ordered check-ins along with categorical unique identifiers for venues and agents. In the University and Bike datasets, meetings are computed and stored as data pre-processing steps. The meetings in the University dataset are computed first grouping the contacts by venues and then finding the overlapping staying intervals. In the Bike dataset, the spatial proximity is computed by a kd-tree, a spatial indexing structure to find the location within a given distance. The meetings are stored with the participates, location, starting, and ending time. Meetings are ordered by their starting time.

The simulation progresses sequentially over the events ordered by check-ins or meetings and keeps track of the states of the agents and venues. While the state machine for the agents is described in the main article, a venue can be in one of the two states — susceptible and infected. Initially, all venues are susceptible and a venue becomes infected once an infectious agent checks-in. Each infected venue keeps a timer that goes off after 48 hours of being infected and then the venue becomes susceptible again. If an infectious agent visits an infected venue, the timer is reset to the current check-in time. We maintain the timer by operating on it only at check-in event times.

In the person to person infection spread model, the infection probability, \( \beta \), used
in the simulation is derived for a particular dataset based on existing knowledge of the initial reproduction number, $R_0$ for COVID-19. $R_0$ denotes the expected number of people infected from a single person, i.e., $R_0 = c\beta T$, where $c$ is the average number of daily meetings per person, and $T$ is the average number of days an agent remains infectious. While $c$ is estimated from a dataset, model parameters in Fig. 4.1B give the value of $T$ as 9.55 days. For the disease transmission through venues, $\beta$ has a different interpretation and does not correspond to usual parameters such as $R_0$.

There are 10 initial infectious seeds at the beginning of each simulation. The infection probability is set as 0.75 for the Foursquare dataset and takes $R_0$ as 3 to obtain the infection probabilities in the University and Bike datasets.

For both the person to person and via venue transmission models, the probabilistic virus transmission is simulated as follows. A uniform random number is sampled in the range $[0, 1]$ and compared against the infection probability, $\beta$. The probabilistic experiment is successful, i.e., the agent becomes infected, if the sample is less than $\beta$.

We study dynamics of the infection spread in four dimensions – the total number of infected people till date, number of active cases, number of new infections, and epidemiological parameters such as the time-varying reproduction number ($R_t$) or growth rate ($\lambda_t$). The active cases include individuals that have been infected till date but not yet recovered. The time-varying reproduction number, $R_t$ for the day $t$ is the expected number of average individuals infected by a single agent who gets infected on the day $t$. The growth rate, $\lambda_t$, is the ratio of the total number of agents newly infected on the day $t$ to the same number on the day $t-1$. When $\lambda_1$ above 1, the number of infections grows exponentially; when $\lambda_t$ is below 1, the number of new infections converges.

In the contact graph-based study, the contagion simulation uses the same model and parameters as mobility datasets (Fig. 4.1 Part I). It starts with 10 initial randomly selected infectious nodes (seeds) and all other nodes are at the susceptible state. It works in synchronous and discrete-time rounds – at each round, an infectious node infects its susceptible neighbors in $G$ with a probability, $p$. As both the meetings and the check-ins to venues are abstracted to counts instead of timestamped events, the contacts are considered to be distributed uniformly over the duration of the dataset,
and infection transmission probabilities $\beta$ are adjusted per edge as $p = (1 - (1 - \beta)^w)$. In the infection transmission model via venue, an infectious individual infects a venue with probability $\min(1, w)$ at each round, and the infected venue in-turn infects an individual with probability $p$.

Similar to mobility-based simulations, in the social network setting we use $\beta = 0.75$ for transmission via venues. In the person-to-person transmission model, $\beta$ is derived from $R_0$ from the same expression, $R_0 = c\beta T$. Here, $c$ is counted as the average total weight from a node and $T$ remains the same as 9.55.

4.3 Results

We found that mobility statistics are heterogeneous across individuals and venues (Fig. 4.1), with few agents being highly active and some venues attracting many visitors. This heterogeneity and consequent network effects influence the spread of COVID-19. The common strategy of blanket lockdown is seen to delay the peak but comes at a high social cost (Fig. 4.2). High mobility agents and popular venues attract early infections and play an important role in the early spread of the infection (Fig. 4.3). Interventions that protect these entities, such as isolating or immunizing the highly active agents, or closing the popular venues, are seen to have a large effect – both delaying the peak and reducing total infections – while still incurring a relatively small social cost (Fig. 4.4). Basic models (e.g., SEIR [192]) where infection transmission is not related to spatial proximity do not show such heterogeneous effects, which implies that mobility makes a significant difference compared to homogeneous models. Finally, a social network or contact graph – that connects agents to other agents they meet frequently or to venues that they visit frequently – is seen to have the property that a similar number and group of people are infected in this static model as under the dynamic mobility simulation model (Fig. 4.5). This section elaborates on these results, starting with a study of the baseline lockdown strategy.
4.3.1 Evaluation of baseline uniform mobility intervention strategy.

Under a lockdown or stay-at-home order, mobility drops only to a limited extent [12]. We model such a partial lockdown by randomly removing events with a certain probability. This is our baseline intervention strategy. Fig. 4.2 studies the realistic scenario where the intervention lasts for a limited duration, with 80% of the events removed. Thus, during intervention this retains 20% social value. The results agree with the observations across the world – the peak of the infection is lowered and delayed [100]. However, we discover that this strategy alone does not significantly reduce the total number of infections when applied for a limited duration.

Part I of Fig. 4.2 studies the disease spread with a varied starting time of the intervention while keeping its duration fixed at 15 days. In the Foursquare NYC dataset, the intervention starts on the day 9 when 5% of the population gets infected. Compared to the scenario without intervention, this strategy reduces the total number of infected agents by 8%, delays the peak of active infections by 11 days, and lowers it by 6%. The peak of new infections is also delayed by 18 days. The University dataset shows a similar pattern, as the peaks in active and new infection are delayed by 22 and 19 days respectively. Other datasets show trends consistent with the above.

We note that an earlier lockdown is not necessarily the better strategy when implemented for a fixed duration. An imperfect lockdown can allow the infection to survive, which then spreads rapidly on the removal of the intervention and produces a high eventual peak or second wave. On the other hand, a lockdown implemented somewhat later achieves a lower peak later on, and a lower total number of infections.

In Fig. 4.2, Part II, the intervention strategy starts when 10% of the population is infected (day 13 for the Foursquare NYC dataset and day 36 for the University dataset). The longer interventions have a more significant effect in reducing both the peak of active cases and total infections. A 30-days intervention reduces the total infected population by 27% and 5% in the Foursquare NYC dataset, the University dataset respectively. While for shorter interventions (up to 15 days), the active infection curves remain uni-modal, a longer intervention (30 days) produces two peaks and the second
peak is delayed by 32 and 65 days for the Foursquare NYC and the University dataset respectively. The growth rate, $\lambda_t$ for the longer intervention does not go to zero ($\lambda_t$ reaches zero when no new person is infected). Thus the disease remains in circulation for a longer period. Interventions up to two months show similar conclusions: second peaks are observed, and the cumulative number of infected people does not decrease beyond 10% compared to the setting without intervention.

Dividing a population into non-interacting cohorts has been proposed as an intervention strategy [2, 136], with the idea that preventing transmission across group boundaries will reduce the spread. We find that grouping does act to delay and lower the peaks and to reduce the total number of infections in Fig. 4.6. The number of infected people is reduced by 37% in NYC and by 53% in the university with 4 groups. The effect is milder in the Foursquare datasets with larger population size, for example in the Istanbul dataset (Fig. 4.6B). By taking the same sized population as in the NYC dataset (Fig. 4.6C), the effect appears – the total number of infected agents is reduced by 22% with 4 groups.

### 4.3.2 Heterogeneity in mobility and contagion dynamics.

We found that the most active individuals are infected significantly earlier (Fig. 4.3) than the average population. A person’s activity is quantified by the number of check-ins in the Foursquare datasets and the number of meetings in the University and Bike datasets. The number of check-ins measures the popularity of a venue. Of the top 5% most active agents, approximately 60% – 80% get infected at the peak of active infections compared to the peak height about 20% – 40% for the overall population, and this peak is reached about 10 days earlier. Thus, highly active people are at a higher risk, and more likely to be propagators of the disease in the early stages.

The heterogeneity of venues in spreading the contagion is seen in Fig. 4.3E to G. The number of agents infected from a venue has a heavy-tailed distribution for all cities in the Foursquare datasets, and the results for the NYC dataset are shown separately. Most of the venues infect a tiny number of individuals, but a few venues infect a large number of agents - In the Foursquare NYC dataset, 50% people are infected from
0.05% (19 venues) most popular venues. There are 718 and 4,979 individuals who have visited the top 0.01% most popular venues in the Foursquare NYC and Istanbul datasets respectively. While these people are infected in higher proportions (by at least 12% higher), the shape of active infections remains similar to the overall population curve.

4.3.3 Targeted interventions on the most active individuals and the most popular venues.

We study targeted intervention schemes to put a higher level of protection on the most active agents and most popular venues, thus reducing or eliminating their contribution to spreading the virus.

In these simulations, the social cost of an intervention is defined as the fraction of social events (or potential interactions) lost due to the intervention. For example, in the check-ins dataset, this is measured as the fraction of check-ins lost. In other datasets, the social cost is measured as the fraction of co-located pairs of individuals – representing potential pairwise meetings. The social value is correspondingly measured as the complement of the social cost - that is, the fraction of events preserved under the intervention. The health value of an intervention is measured as the fraction of agents who escape infection due to the intervention (but would have been infected otherwise).

Fig. 4.4A to D use the social, and health values to compare the targeted interventions against multiple other strategies including the uniform intervention of staying home, protecting a random subset of agents and closing a random subset of venues. In the Foursquare datasets, closing the most popular venues is the most effective strategy, while protecting the most active agents is the most effective strategy in the University and Bike datasets.

In the NYC dataset, to achieve a similar ∼ 80% social value, the strategy of closing the most popular venues shows 60% to 72% more health value than other strategies. In the Istanbul dataset, to achieve 60% social value closing the most popular venue achieves at least 52% more health value than other strategies. The other cities in the dataset show consistent patterns. In both the University and Bike datasets, at 80%
social value, protecting the most active agents achieves 30% or more health value than other strategies.

The strategy to close the most popular venues is analyzed in detail in Fig. 4.4E and F. In Foursquare NYC and Istanbul datasets, closing a few (∼1%) most popular venues reduces the total number of infected individuals by more than 40%, the peak number of active infections by ∼30% and delays the time of the peak by up to 30 days. In larger sample populations, this strategy has a smaller impact on total infections and the peak number but serves to delay the peak. In the university dataset, closing 5% of the most popular venues can reduce the total number of infected individuals by 50% and the peak number of active infections by more than 20%.

Figure 4.4G and H show the effect of protecting the infection of the most active agents. In the University and Bike datasets, protecting the most active 5% to 10% of the population reduces total infection by 20% to 40% and delays the peak by 90 to 110 days. Protecting the most active individuals in the Foursquare datasets has similar effects – in the NYC dataset, the total number of infected people is reduced by ∼30% and the peak of active cases is reduced by ∼25% when 20% most active people are protected. However, protecting 20% most active people reduces the social value by ∼45%. The rest of the Foursquare datasets show similar patterns.

4.3.4 Contact graph: Social network abstraction to estimate mobility based infection spread.

We designed social network models derived from mobility data such that the spread of infection in a mobility dataset resembles that in the corresponding social network. To model person to person disease transmission as a social network, we define a graph $G$ with agents as nodes and edges connect agents who have met at least once. Each edge has a weight, $w$, as the average number of times they met in a day. For infection spread via venues, the social network is defined using a bipartite graph where the agents and venues constitute the two sets of nodes, and they connect if a person visited a venue. The edges are weighted by the average number of daily visits.
Fig. 4.5 compares the contagion simulation on a social network against the simulation on time ordered mobility data. The total number of infected people matches between the simulations (Fig. 4.5, Part I). Further, nearly the same set of agents get infected starting with independent random seed agents. We measure the similarity of the agents infected in two simulations using Jaccard similarity which is defined for sets $A, B$ as $\frac{|A \cap B|}{|A \cup B|}$. For all datasets, Jaccard similarity between the agents (identified by unique user identifiers in the datasets) infected in the mobility-based simulation and the social network simulation reach above 65%. The active cases’ peak differs by 1% (in University) to 24% (in Istanbul).

For the person to person transmission model, Fig. 4.5, Part II compares the total number of infected people for three intervention strategies: staying home, protecting the most active agents, and dividing people into groups. The strategies are simulated in the social network setting by preprocessing the graph, $G$. The staying home intervention of skipping a meeting with probability $\alpha$ is simulated by removing an edge with weight $w$ with probability $1 - (1 - \alpha)^w$. An agent’s activity is defined as the total weight of the edges incident on the corresponding node. The strategy to protect the most active agents translates to removing the nodes with the highest activities from $G$. The strategy to divide the population into groups is simulated by randomly assigning the nodes to groups and then removing the edges between nodes belonging to different groups. The infection curves for mobility and social network simulations are matched for all strategies for both University and Bike datasets – the difference between the two curves is always below 10%. The percentage of people infected in total goes to zero in two models with similar intervention parameters.

4.3.5 Robustness of results.

The conclusions from the simulations are robust to parameters changes in the simulation, and has been confirmed in a range of experiments using different parameter values - for infection probability ($\beta$), number of seeds, and the probability of an agent being asymptomatic. Decreasing $\beta$ reduces the number of infected people and delays the peak of active infections. In the NYC dataset, 68% of the population gets infected when
\( \beta = 0.75 \) compared to 34\% at \( \beta = 0.25 \). In the Istanbul dataset, the peak is delayed by 10 days when \( \beta \) is reduced to 0.25 from 0.75. Increasing the number of seeds – initial infected people – increase the robustness or predictability of the spread of infection, but otherwise has a relatively small effect: a moderate number of seeds shows a similar behavior to a larger number of seeds – the total number of infected people differs no more than 4\% across datasets for using 20 seeds compared to 1 seed. The fraction of asymptomatic carriers increases the rate of infection (since these carriers remain infectious for a longer period), but does not cause a major growth in infection – in the NYC dataset 14\% more people get infected when increasing the percentage of asymptomatic people from 15\% to 75\%. A similar pattern is found in the other datasets.

The conclusions are also sustained when simulations are carried out on sub-samples of data. Here a fraction of the agents are randomly sampled for simulation. For the same fraction of the most active agents protected within the subsample, a larger population has a larger fraction of the total population infected. However, this growth appears to be sublinear, and the effect tapers off with the size of the population. Sampling has a similar effect under the intervention of closing popular venues.
Fig 4.1: The model for infection spread (Part I) and dataset properties (Part II to IV).
Fig 4.2: Uniform lockdown for limited duration delays and lowers the peak of the infections, but often has a small effect on total infection.
Fig 4.3: Heterogeneity of mobility in venues and agents results in varied risks to catch and spread the virus.
Fig 4.4: Across all datasets, the most effective strategies in terms of their health and social values consider the heterogeneity of mobility (A to D) and (E to H) present the details of the best strategies.
**Fig 4.5:** Contagion simulation in the derived social network model infects a similar number and similar set of agents as the simulation using mobility data in both with and without intervention settings.
**Fig 4.6:** Diving agents into cohorts cuts off the contacts between different cohorts, like separating students into different sessions for courses and acts to reduce the spread of the disease.
4.4 Discussion

We have shown that the diversity of human movement influences the spread of the virus, making the behavior different from homogeneous models. The heterogeneity can be leveraged to devise less disruptive strategies of intervention. Our conclusions are complementary to other non-pharmaceutical interventions such as maintaining person to person distance, wearing masks and avoiding contact. The analysis is based on datasets that are samples of population behavior, and thus, the infection numbers or percentages generated by the simulations should not be regarded as precise representative values but rather as patterns to expect under various interventions. The trends in relative measurements hold across multiple datasets of varying sizes and representing multiple locations, lifestyles, and geographic scales, giving us confidence that they reflect general properties of infection spread due to mobility and corresponding interventions.

Mobility in a region may be affected by policies and interventions applied to other regions such as neighboring districts [86]. This spillover effect can be seen as a network effect on a larger scale. It has been shown [97] that spillover indeed affects aggregate mobility in US counties and thus intervention policies should be coordinated at larger scales. Our datasets do not include mobility across such administrative zones and consequent spillover effects. As more mobility data from the ongoing pandemic becomes available, it may be possible to incorporate and model such larger-scale network effects on agent mobility and infection, as well as incorporate changes in human behavior due to COVID-19 [78, 7]. The datasets used here represent pre-COVID behavior between (2012 and 2018) and representative of the spread of infection under “normal” behavior. The effect of changed mobility under social distancing and COVID awareness will require corresponding datasets and separate study.

In simulations, we found that the existing common strategies come with some caveats. The cohort strategy, i.e. partitioning the population into non-interacting
groups \([2, 136, 10]\) is effective in circumstances such as small populations, a large number of groups, and the specific dataset of University mobility (Fig. 4.6), but not necessarily in other scenarios. The simulations are restricted to certain slices of the society, and in practice, it is infeasible to maintain a consistent partitioning across different communities such as residence, work, and school, which is likely to reduce the impact of this strategy. A larger number of partitions or cohorts will reduce the spread of infection, but a large number of cohorts are difficult to implement and schedule, and they come at a larger social cost of reduced interaction across cohorts. The implementation of a cohort strategy may differ by circumstances, such as separating the groups spatially (different classrooms) versus separating temporally (different time slots), which will influence the person-to-person distancing and infection propagation. For the other common strategy of a blanket lockdown, which many countries have implemented, the results in Fig. 4.2 show that the timing and duration of the action subtly influence the peak and total infections. Since this strategy is a limited resource - in the sense that it cannot be sustained for long - it needs to be deployed with careful planning, and with consideration of available medical and economic resources.

We found targeted interventions on the most mobile agents and most popular venues to have a significant effect on pandemic control at a low social cost (Fig. 4.4). These results are complementary to prioritizing protections for essential workers such as medical professionals \([46, 125]\), as our conclusions are for the mobility component of infection spread, which is not addressed in other works \([193]\). In this respect, those most at risk and likely spreaders of infection are workers whose professions require frequent movement. While isolating them from work is not practical, better protection strategies such as protective equipment, strict regulations, and early vaccinations can help to reduce infections. Closing the most popular venues such as university cafeteria are also unsustainable over long durations, but modified operations and strict social distancing at these venues can help to significantly reduce infections. The influence of these various strategies relative to other non-pharmaceutical interventions remains to be investigated. It has been suggested \([86]\) that with more data, implicit Randomized Controlled Trials may be carried out by comparing similar localities enforcing different
subsets of policies.

The difference in results for data-driven mobility and those from traditional homogeneous models can be attributed to information-structural properties of human mobility, creating a complex network effect. The inclusion of social and economic considerations are likely to add to these complexities, for example, it has been argued that the worldwide food supply chain is being affected by labor shortages, trade restrictions, and factory closures [122]. The relations between mobility, essential industries, and economics will be important to decipher for the development of long term strategies.

In conclusion, from the perspective of actions to control the pandemic, our results suggest interventions on highly mobile people and most popular venues are most likely to be effective. These interventions can take forms of priority in testing, vaccination, protective equipment, or stricter regulations. The infection spread at venues can take the form of limiting large gatherings, reducing the need to visit the venue or different modes of operations (e.g. deliveries and takeaways) – and should be encouraged more at popular venues. Our model and simulation methodology can be used to identify the points of action in other datasets and environments.

We found that large scale general lockdowns may not reduce the total number of infections in call cases, but they can be used to delay infections and lower the peak. Thus, it can be used to temporarily avoid overloading hospitals, and to gain time to build up resources. Partitioning a population into cohorts can be useful in small populations (such as small schools). In large groups where they are harder to implement, they are also less likely to effective if implemented.

We note that our analysis is purely on the mobility dimension. The spread of the infection depends on various other factors, such as the nature and architecture of the venue (indoor vs outdoor), awareness of people, and other complex behavioral factors that should ideally be taken into account. Mobility patterns themselves are dependent on economic and industrial aspects that are beyond the scope of our study. More research is needed for detailed models incorporating complex aspects of infection spread.
Chapter 5
Co-evolution of Opinion and Social Tie Dynamics
Towards Structural Balance

5.1 Introduction

We live in a continuously changing world in which social interactions dynamically shape who we are, how we view the world and what decisions to make. Various social processes, naturally intertwined, operate on both the properties of individuals and social ties among them. Social influence, for example, describes how people’s behaviors, habits or opinions are shifted by those of their neighbors. Social influence leads to homophily (similarity of node attributes between friends) and leaves traces in the network structure such as high clustering coefficient and triadic closure (there are likely social ties among one’s friends).

Social influence and homophily, however, do not fully interpret the global network structure. One of the widely observed structural properties in social networks is the community structure. Nodes within the same community are densely connected and nodes from different communities are sparsely connected. Community detection is an important topic in social network analysis and has been investigated extensively [152, 206, 145, 148, 146, 83, 212, 186, 147, 85, 113]. But why does community structure emerge and become a persistent feature? Are there social processes that encourage or maintain the community structure?

In the literature, there have been a lot of studies of opinion dynamics, where people’s behaviors, habits or opinions are influenced by those of their neighbors. Network models that capture social influence, such as French-DeGroot model [72, 19], Friedkin-Johnsen model [74], Kułakowski et al. model [120, 139] naturally converge to global consensus. Actually it seems to be non-trivial to create non-homogeneous outcomes in these models,
while in reality social groups often fail to reach consensus and exhibit clustering of opinions and other irregular behaviors. Getting a model that may produce community cleavage [73] or diversity [121] often requires specifically engineered or planted elements in a rather explicit manner. For example, bounded confidence models [141, 119, 90, 56] limit social influence only within pairs with opinions sufficiently close. Other models introduce stubborn nodes whose opinions remain unchanged throughout the process.

The extreme case of community structure is dictated by structural balance [91, 92, 43, 53], a common phenomenon observed in many social relationships. The notion was first introduced in a seminal paper by Heider in the 1940s in social psychology. It describes the stability of human relations among three individuals when there are only two types of social ties: positive ties describe friendship (or sharing common opinions) and negative ties describe hostility (or having opposite opinions). Heider’s axioms state that among three individuals, only two kinds of triangles are stable: the triangles where all three ties are positive, indicating all three individuals are mutual friends; and the triangle with two negative edges and one positive edge, describing the folklore that “the enemy of your enemy is your friend.” The other two types of triangles (e.g., a triangle of one negative tie and two positive ties, or three mutually hostile individuals) incur emotional stress or are not strategically optimal. See Figure 5.1. Thus they are not socially stable – over time, they break and change to the stable ones. In fact, the structural balance theory not only describes the local property in a signed complete network, but also predicts the global network behavior – the only type of network in which all triangles are stable must have the nodes partitioned into two camps, inside each of which the edges are all positive and between them the edges are negative (one of the camps can be empty).

![Figure 5.1: Structural balance theory: the first two triangles are stable while the last two are not.](image)

The structural balance theory only describes the equilibrium state and does not
provide any model on evolution or dynamics – what happens when a network has unstable triangles? Follow up work on structural balance dynamic models [19, 139, 120, 50, 181] update the sign/weight of an edge \((i, j)\) towards a more balanced triangle by considering the sign/weights of neighboring edges \((i, k)\) and \((k, j)\) in a triangle \(\triangle ijk\).

These two threads of research, opinion dynamics and structural balance dynamics, are currently orthogonal to each other. In opinion dynamics, opinions on vertices are influenced by each other through edges but researchers struggle to maneuver the model to create community cleavage. In structural balance dynamics, edge weights are dynamically updated to meet structural balance, which is explicitly coded as the projected outcome and optimization objective. Holme and Newman [96] presented a simple model of this combination without any theoretical analysis. In this paper, we propose co-evolution models for both dynamics of opinions (people’s view on a variety of topics) and dynamics of social appraisals (the approval or disapproval towards each other). We show that by using two simple rules, node opinions evolve into opposing communities and structural balance naturally emerges.

5.1.1 Our Contribution

In this paper, we consider the co-evolution of opinions via social influence and tie strength/appraisal/sign updates by discrepancies of node opinions. We assume a set of \(n\) individuals where individual \(i\) has an opinion \(v_i \in \mathbb{R}\), and an appraisal matrix \(W = [w_{ij}]\), where \(w_{ij}\) is interpreted as the influence from individual \(j\) on individual \(i\). Here \(w_{ij}\) does not need to be non-negative and takes values in \(\mathbb{R}\). We consider two update rules:

- **Opinion dynamics:** The opinion of \(i\) is updated as a linear combination of its current opinion and the weighted sum of neighbors’ opinions with coefficients in matrix \(W\).

- **Appraisal dynamics:** The appraisal \(w_{ij}\) is updated as a linear combination of its current value and the agreement of the opinion of \(i\) and \(j\). The agreement of opinion \(v_i\) and \(v_j\) is taken as the dot product \(v_i \cdot v_j\).
The opinion dynamics model is similar to classical social influence models such as the DeGroot model [72] and Friedkin-Johnsen model [74], except that the edge weight matrix $W$ is dynamic as well. The model for appraisal dynamics is motivated by tie dynamics that can be traced back to Schelling’s model of residential segregation [170]. In modern society, tie changes on Facebook [171] and Twitter [202, 117] can be easily triggered by disparities on their opinions [109, 171], especially among the users who are most politically engaged.

Our goal is to analyze the evolution and in particular, the conditions that lead to sign stability and structural balance – for any triangle with individual $i, j, k$, $w_{ij}w_{jk}w_{ki} \geq 0$. We call a network to reach strictly structural balance if $w_{ij}w_{jk}w_{ki} > 0$, ∀$i, j, k$. A structurally balanced network has two possible states: harmony, when all edges are positive; and polarization, when there are two communities with only positive ties within each community and negative ties across the two communities.

We show that with our dynamics model, $W(t)$ evolves by the following matrix Riccati Equation [8] $W' = WW^T + C$, where $C = V(0)V(0)^T - W(0)W(0)^T$ is a symmetric constant $n \times n$ matrix if $W(0)$ is. Further the opinion vector $V(0)$ evolves by the differential equation $V''(t) = 2|V|^2 \cdot V - C \cdot V$. Our main result is to analyze the asymptotic behavior of $V(t)$ and $W(t)$ and prove structural balance at the limit. Our results can be summarized in the following:

1. By analyzing the evolving equation for $V(t)$, we show that either the network reaches strict structural balance or $|V(t)| \to 0$. To prove this limit behavior, one crucial observation is that the length of the opinion vector $|V(t)|^2$ is strictly convex, unless $V(t) \equiv 0$.

2. We show how to solve the general matrix Riccati equation $W' = W^2 + C$, $W(0) = B$, for any parameter $B, C$. In particular, the eigenvector corresponding to the largest eigenvalue of $W(t)$ encodes the two communities formed in the network; those with a positive value in the eigenvector versus those with a negative value in the eigenvector. As a byproduct of the analysis of $W(t)$, we also show that when $V(0)$ is an eigenvector of the initial matrix $W(0)$, $V(t)$ remains to be an
eigenvector of $W(t)$ for and structural balance must occur in finite time. In this case we can write down exact evaluations on the blowup time and the rate convergence.

The evolving of $W(t)$ by $W' = WW^T + C$ is strictly a generalization of the dynamic structural balance model by Marvel et al. [140]. Their model captures the dynamics of edge appraisals by $W' = W^2$ and does not consider user opinions. The behavior of our model becomes much more complex, as the initial user opinions are factored into the system dynamics through the matrix $C$.

We also performed extensive simulations with different initial conditions and graph topology. We examined the network evolution on the final convergence state (harmony v.s., polarization) and convergence rate. We observed that a higher network density or a higher initial opinion magnitude, empirically, speeds up the convergence rate.

We tested our dynamic model on two real world graphs (Karate club graph and a political blog graph [9]). Both networks are known to have two communities with opposing opinions. A small number of seeds, randomly selected, are assigned with ground truth opinions and all other nodes start neutral. The network evolution can successfully detect the final community structure and recover the ground truth with good accuracy. Apart from being a transparent and explainable label propagation algorithm, the model sheds lights on why community structure emerges and becomes a widely observed, sustainable property in complex networks.

5.2 Related Work

5.2.1 Opinion Dynamics and Social Influence

Opinions in a sociological viewpoint capture the cognitive orientation towards issues, events or other subjects, and mathematically represent signed attitudes or certainties of belief. Opinion dynamics is an extensively studied topic about how opinions change in a network setting with social influence from neighbors. One of the first models of opinion dynamics, French-DeGroot model [72, 19], considers a discrete time process of opinion \( \{v_1, v_2, \ldots, v_n\} \) for a group of $n$ individuals. An edge $(i, j)$ carries a non-negative weight
$w_{ij} \geq 0$. The opinion of node $i$ at time $t+1$ is updated by

$$v_i(t+1) = \sum_j w_{ij}v_j(t).$$

The weight matrix $W = [w_{ij}]$ is taken as a stochastic matrix. The dynamics can be written as $V(t+1) = WV(t)$. The continuous-time counterpart is called the Abelson’s model [5] where the dynamics is defined by

$$\frac{dV(t)}{dt} = -LV(t), \quad (5.1)$$

where $V(t) = (v_1(t), v_2(t), \ldots v_n(t))^T$ and $L$ is the Laplacian matrix $L = I - W$. Opinions following the French-DeGroot model or the Abelson’s model typically converge unless the network is disconnected or there are stubborn nodes (with $w_{ii} = 1$).

The most popular opinion dynamics model is probably the Friedkin-Johnsen model [74]. It takes a stochastic matrix $W$ as the influence model, and a diagonal matrix $\Lambda = \text{diag}(\lambda_1, \cdots, \lambda_n)$ where $\lambda_i \in [0, 1]$ is the susceptibility of individual $i$ to social influence. The opinions of the individuals are updated by the following process

$$V(t+1) = \Lambda WV(t) + (I - \Lambda)u,$$

where $u$ is a constant vector of the individuals’ prejudices and is often taken as the initial opinion $V(0)$. When $\Lambda = I$ the model turns to French-DeGroot model.

Most of the literature on these two models assume a fixed weight matrix $W$ and prove asymptotic convergence under favorable assumptions [75]. There have been extensions when $W$ is a time-varying matrix, but $W(t)$ is still independent of $V(t)$ (e.g., [36]).

A significant deviation from the above family considers a time-varying matrix $W$, by incorporating the principle of homophily, that similar individuals interact more than dissimilar ones. This is called the bounded confidence model [141]. A few such models (Hegselmann- Krause (HK) model [119, 90], Deffuant and Weisbuch [56]) introduced a fixed range of confidence $d > 0$: individual $i$ is insensitive to opinions that fall outside its confidence set $I_i = [v_i - d, v_i + d]$, and the opinion $v_i$ is only updated by the average opinion of those opinions within $I_i$. In other words, the matrix $W$ is derived from the set of opinions at time $t$ and thus co-evolves with the opinions. This model generates
situations when the individuals converge to a set of different opinions, and has been extended to the multi-dimensional setting [90].

All models above have only considered the case of positive influence, that the interactions of individuals change their opinions towards each other. It has been argued in both social settings and many physical systems that there is negative or repulsive influence (repulsive interactions in biological systems [52] or collision avoidance in robot swarm formation [165]). Abelson [6] argued that any attempt to persuade a person may sometimes shift his or her opinion away from the persuader’s opinion, called the boomerang effect [13, 99]. Bhawalkar et al. [32] presented game-theoretic models of opinion formation in social networks by maximizing agreement with friends weighted by the strength of the relationships. Thus interactions between individuals with similar opinions move their opinions closer; interactions between individuals with opinions that are very different shift their opinions away from each other. Here the edge weights are fixed. Many models have included negative ties but they are still awaiting rigorous analysis [169, 27, 137, 138, 88]. The most notable work in this direction is by Altafini [18, 17]. The model starts to be similar to Abelson’s model in Equation (5.1) with a fixed weight matrix $W$ except that the weights in $W$ do not need to be non-negative. The system is shown to be Lyapunov stable [159] and studies have focused on the initial conditions of $W$ for the system to converge to harmony or polarization. The matrix $W$ is assumed to be either static or, in very recent studies [158, 159, 93], time-varying (but independent of the opinion changes). The negative influence is closely related to signed networks and structural balance theory, which will be discussed next.

5.2.2 Structural Balance and Signed Networks

Notice that the structural balance theory only describes the equilibrium state and does not provide any model on evolution or dynamics – what happens when a network has unstable triangles? Follow up work proposed a few models, that can be categorized by discrete models or continuous models – depending on whether the appraisal on a social tie takes binary values $\{+1, -1\}$ or a real number. Antal et al. [19] considered the discrete model where the sign of an edge is flipped if this produces more balanced
triangles than unbalanced ones. The balanced graph is clearly a stable state but the
dynamics also has many local optimals called jammed states [19, 140]. Andreida et
al. [181] determine the sign of an edge according to the sign of the other two edges
in the triangles to make more triangles balanced. Samin et al. [22] try to remove the
minimum edges to make the graph balanced which is NP-hard problem.

In the continuous setting, the influence-based model [139, 120] describes an influence
process on a complete graph, in which an individual $i$ updates her appraisal of individual
$j$ based on what others positively or negatively think of $j$. In other words, let us use
$w_{ij}$ to describe the type of the social tie between two individuals $i, j$. $w_{ij} > 0$ if $i, j$
are friends and $< 0$ if they feel negative about each other. The absolute value of $w_{ij}$
describes the magnitude of the appraisal. The update rule says that the update to $w_{ij}$
will take value

$$\frac{dw_{ij}}{dt} = \sum_k w_{ik} \cdot w_{kj}. \quad (5.2)$$

Specifically, when $w_{ik}$ and $w_{kj}$ have the same sign, the value of $w_{ij}$ is guided to the
positive direction; when $w_{ik}$ and $w_{kj}$ have opposite signs, the value of $w_{ij}$ is guided
to the negative direction. Both cases try to enforce a balanced triangle on \{i, j, k\}.

Empirically, it has been observed that for essentially any initial value of $W$, as the
matrix where the (i, j) element is $w_{ij}$, the system reached a balanced pattern in finite
time. In [139], Marvel et al. proved that for a random initial matrix $W$ the system
reaches a balanced matrix in finite time with probability converging to 1 as $n \to \infty$.
They also characterized the converged value and its relationship to the initial value.

In a recent paper [50], Cisneros-Velarde et al. considered a pure-influence model,
where the self-appraisal (such as $w_{ii}$) is taken out of Equation (5.2) to be a more faithful
interpretation of Heider’s structural balance. They proved that when $W$ is symmetric
their continuous-time dynamic model is exactly the gradient flow of an energy function
called dissonance [140], defined as

$$- \sum_{[ij], [jk], [ki] \in E} w_{ij} \cdot w_{jk} \cdot w_{ki}.$$ 

Dissonance characterizes the degree of violation to Heider’s structural balance axioms
in the current network. The global minimum of this energy function corresponds to
signed networks that satisfy structural balance in the case of real-values appraisals. When the initial matrix $W$ is symmetric the authors also provided characterizations of the critical points of the dissonance function (aka the equilibrium states of the dynamic model).

The discussions of opinion dynamics and dynamics with structural balance, so far, have focused on node opinion changes or link appraisal changes, separately. There is little work on combining both dynamics into a co-evolving model, which is the focus of this paper.

### 5.3 Co-Evolution Model

Suppose there are $n$ individuals, each one with its own opinion $v_i \in \mathbb{R}$. Define the opinion vector $V = (v_1, v_2, \ldots, v_n)^T \in \mathbb{R}^n$. The influence model among the $n$ individuals is characterized as an $n \times n$ matrix $W = [w_{ij}]$ with entries taking real values. A positive value of $w_{ij}$ indicates a positive social influence between $i, j$, where the opinions under the influence become similar. A negative value of $w_{ij}$ means a negative influence and their opinions under influence become dissimilar. In our theoretical study, we consider the case of a complete graph. The evolution model that we introduce works for any network. In our simulations we also evaluate networks and opinion co-evolution on a general graph.

Both the opinions of individuals and the influence matrix are dynamically evolving. Assume that the initial opinion vector is $V(0)$ and the initial influence matrix is $W(0)$. In this paper we assume the initial weight matrix $W(0)$ is symmetric, i.e., $w_{ij}(0) = w_{ji}(0)$, $\forall i, j$. Define the opinion vector and influence matrix at time $t$, in a discrete-time model, as $V(t)$ and $W(t)$ respectively. We propose the dynamic system governing the evolution of the relationship over integer time:

\[
\begin{align*}
V(t + 1) &= V(t) + W(t)V(t) \\
W(t + 1) &= W(t) + V(t)V(t)^T.
\end{align*}
\]

In the first equation, the opinion of an individual $i$ is shifted by the weighted sum of its neighbors’ opinions, with coefficients in the influence matrix $W$. In the second equation,
the appraisal value $w_{ij}$ between two individuals $i, j$ is updated by the differences of opinions $v_i, v_j$. If $v_i, v_j$ generally agree (with a positive dot product), $w_{ij}$ moves in the positive direction; otherwise moves in the negative direction.

In a continuous-time model, the dynamics are driven by the following ODE:

$$
\begin{cases}
V' = WV \\
W' = VV^T.
\end{cases}
$$

(5.4)

where $V'$ and $W'$ is the coordinate-wise time derivative of $V$ and $W$.

From this point on, we focus on solving the continuous time model. First we present a couple of basic properties of Equation (5.4). This means we can focus on solving the system defined by Equation (5.4) without losing generality.

**Lemma 5.3.1.**

1. If $[V(t), W(t)]$ solves Equation (5.4) and $U$ is an orthogonal matrix, then $[U^T V, U^T W]$ solves the same Equation (5.4) with initial condition $U^T V(0)$ and $U^T W(0) U$. In particular, $W(0)$ is symmetric if and only if $U^T W(0) U$ is.

2. If $W(0)$ is symmetric, then $W(t)$ remains symmetric for all $t$.

3. If $a, b > 0$ are positive constants, then the equation

$$
\begin{cases}
V'_1 = aW_1 V_1 \\
W'_1 = bV_1 V_1^T
\end{cases}
$$

can be reduced to Equation (5.4) by taking $V = \sqrt{ab} V_1$ and $W = a W_1$.

**Proof.** Part (1) follows from standard computation and that $UU^T = U^T U = I$ for an orthogonal matrix $U$. Also, a matrix $A$ is symmetric if and only if $U^T AU$ is symmetric.

For part (2), note that the equation $W' = VV^T$ implies that $W'(t)$ is always symmetric, i.e., $W'(t) = W'(t)^T$. Now if $W(0)$ is symmetric, then $W(t)$ and $W^T(t)$ are solutions of the same differential equation $W'(t) = W'(t)^T$ with the same initial value. By the uniqueness theorem of the solution of ordinary differential equation, $W(t) = W^T(t)$.

Part (3) follows by rewriting Equation (3) as

$$
\begin{cases}
(\sqrt{ab} V_1)' = (aW_1)(\sqrt{ab} V_1) \\
(aW_1)' = (\sqrt{ab} V_1)(\sqrt{ab} V_1)^T.
\end{cases}
$$
Take $V = \sqrt{ab}V_1$ and $W = aW_1$. This becomes Equation (5.4).

The main objective of this paper is to analyze how this system evolves. In particular, we care about system evolution to reach sign stability for $w_{ij}, \forall i, j,$ and $v_i, \forall i$, as well as structural balance

$$\lim_{t \to T} w_{ij}w_{jk}w_{ki} \geq 0,$$

for all indices $i, j, k$ where $[0, T)$ is the maximum interval on which the solution $W(t)$ exists. Notice that in the classical structural balance theory, the two types of stable triangles – with edge signs as either all positive (+1) or have two negative (−1) and one positive – satisfy this property.

The evolution of $W(t)$ and $V(t)$ is described in the following two lemmas.

**Lemma 5.3.2.** With the co-evolution model as in Equation (5.4), the dynamics of matrix $W$ follows the following Matrix Riccati Type Equation

$$W' = WW^T + C,$$

where $C = V(0)V(0)^T - W(0)W(0)^T$ is a symmetric constant $n \times n$ matrix. If $W(0)$ is symmetric, then $W(t)$ satisfies the Riccati equation

$$W' = W^2 + C. \quad (5.5)$$

**Proof.** We look at $W''$:

$$W'' = (W')' = (VV^T)' = V'V^T + V(V')^T = WVV^T + (WV)^T = WWV^T + VV^T W^T$$

$$= WW' + W'W = W(W^T)' + W'W = (WW^T)'.$$

In the second last step, we use the equation $W' = (W^T)'$. This is because $W'(t) = V(t)V(t)^T$ is always symmetric. Thus, $W' = WW^T + C$, where $C$ is a constant matrix $C = W'(0) - W(0)W(0)^T = V(0)V(0)^T - W(0)W(0)^T$. Notice that $C$ is always symmetric.

If $W(0)$ is symmetric, then $W(t)$ is always symmetric (by Lemma 5.3.1 (2)) and $WW^T = W^2$. \qed
Remark that matrix $C$ in our setting is a special symmetric matrix. Specifically, $C + W(0)W(0)^T = V(0)V(0)^T$ has rank one. This property turns out to be useful for characterization of the system behavior.

**Lemma 5.3.3.** The evolution of $V(t)$ satisfies

$$V''(t) = 2|V|^2 \cdot V - C \cdot V$$

(5.6)

**Proof.** Here, we use $W^2 = W' - C$ (Equation (5.5)), $W' = VV^T$ and $V' = WV$.


time derivative of $V'' = (V')' = (WV)' = W'V + WWV = |V|^2V + W^2V = |V|^2V + (W' - C)V$

$$= |V|^2V + (VV^T - C)V = 2|V|^2V - CV.\]

\[\square\]

### 5.4 Analysis of the Opinion and Social Tie Evolution

Our analysis has two parts. First we focus on the opinion evolution model (Equation (5.6)). Here we provide analysis of the asymptotic behavior for $V(t)$. Then we study the social tie evolution (Equation (5.5)) for $W(t)$. By solving Riccati equation explicitly for $W(t)$ we are able to provide more detailed characterization of the evolving behavior.

#### 5.4.1 Analysis of Opinion Evolution

By analyzing the opinion evolution (Equation (5.6)), our main result is the following.

**Theorem 5.4.1.** Let $[0, T)$ be the maximum interval of existence for the solution $(V(t), W(t))$ of the differential equation in Equation (5.4). For generic initial values $V(0)$ and $W(0)$, either

1. structural balance condition $\lim_{t \to T} w_{ij}w_{jk}w_{ki} > 0, \forall i, j, k$ holds, or

2. $T = +\infty, \lim_{t \to \infty} |V(t)| = 0$ and $\lim_{t \to \infty} V'(t) = 0$.

Furthermore, in the first case, the normalized opinion vector $\frac{V(t)}{|V(t)|}$ converges, i.e., $\lim_{t \to T} \frac{V(t)}{|V(t)|}$ exists.
The theorem says that structural balance is always achieved, unless the opinions converge to a zero vector, in which case the entire network becomes neutral. We can consider the second case as a boundary case of structural balance.

The rest of the subsection will focus on proving this theorem. An important observation is that the norm of $V$, $|V(t)|^2$, is a convex function.

**Lemma 5.4.1.** The length function $\varphi(t) := V^T V = |V(t)|^2$ is strictly convex and $\varphi''(t) > 0$ unless $V(t) \equiv 0$.

*Proof.* Recall that $\varphi(t) = |V(t)|^2$.

\[
\varphi'(t) = (V')^T V + V^T V' = (WV)^T V + V^T W V = 2V^T W V.
\]

Therefore,

\[
\varphi''(t) = 2((V')^T W V + V^T W' V + V^T W V') = 2((WV)^T W V + V^T VV^T V + V^T W W V)
\]

\[
= 2(|WV|^2 + |V|^4 + V^T W^T W V) = 2(2|WV|^2 + |V|^4) = 2(2|V'(t)|^2 + |V(t)|^4) \geq 0
\]

(5.7)

Now if $\varphi''(t_0) = 0$, then $V'(t_0) = V(t_0) = 0$ by Equation (5.7). This shows that $V(t)$ is the solution of the ODE (Equation (5.6)) with the initial condition $V(t_0) = V'(t_0) = 0$. But 0 is also the solution. Therefore, by the uniqueness of solution of ODE with initial value, $V(t) \equiv 0$. This ends the proof. \qed

Now, let us understand Equation (5.6) using coordinates. Since the matrix $C$ is symmetric, by the orthogonal diagonalization theorem, there exists an orthogonal matrix

\[
U = [\beta_1, \cdots, \beta_n] = [u_{ij}]_{n \times n}
\]

such that $C\beta_i = a_i\beta_i$, $i = 1, 2, \cdots, n$, where $a_1, \cdots, a_n$ are eigenvalues of $C$. By our assumption that $C = V(0)V(0)^T - W(0)W(0)^T$, the eigenvalues $a_1, \cdots, a_n$ are non-positive except for one. So we may assume $a_1 \geq 0$ and $a_2, \cdots, a_n \leq 0$. Because $\beta_1, \cdots, \beta_n$ form an orthonormal basis of $\mathbb{R}^n$, we can write

\[
V(t) = \sum_{i=1}^{n} \lambda_i(t) \beta_i = U \lambda,
\]

where $\lambda = [\lambda_1, \cdots, \lambda_n]^T$. \[\]
This implies that $V'(t) = \sum_{i=1}^{n} \lambda_i'(t) \beta_i$, $V''(t) = \sum_{i=1}^{n} \lambda_i''(t) \beta_i$, $|V(t)|^2 = \sum_{j=1}^{n} \lambda_j^2(t)$ and $C \cdot V = \sum_{i=1}^{n} a_i \lambda_i \beta_i$.

Therefore, Equation (5.6) becomes

$$\sum_{i=1}^{n} \lambda_i''(t) \beta_i = \sum_{i=1}^{n} (2 \sum_{k=1}^{n} \lambda_k^2 - a_i) \lambda_i \beta_i.$$  

Since $\beta_1, \ldots, \beta_n$ are independent, we obtain the system of ODE with $\lambda = [\lambda_1, \ldots, \lambda_n]^T$ in the form

$$\lambda_i''(t) = (2 \sum_{k=1}^{n} \lambda_k^2 - a_i) \lambda_i(t), i = 1, 2, \ldots, n. \quad (5.8)$$

Denote $W = [w_{ij}(t)]$. Then $W' = V \cdot V^T$ implies

$$W' = U \cdot \lambda \cdot \lambda^T \cdot U^T.$$ 

Therefore, $w'_{ij}(t) = \sum_{k,l=1}^{n} u_{ik} u_{jl} \lambda_k \lambda_l$ and $w_{ij}(t) = \sum_{k,l=1}^{n} (\int_0^t \lambda_k(s) \cdot \lambda_l(s) ds) u_{ik} u_{jl} + w_{ij}(0)$.

Our next goal is to show the following proposition,

**Proposition 5.4.1.** If $T < +\infty$ or if $T = +\infty$ and $\lim_{t \to T} |V(t)| = L > 0$, then there exists one term $\psi_{hh}$ among $\psi_{kl}(t) = \int_0^t \lambda_k(s) \lambda_l(s) ds$ which has the maximum growth rate as $t \to T$ and $\lim_{t \to T} \psi_{hh}(t) = +\infty$.

Assuming the proposition 5.4.1, Theorem 5.4.1 follows. Indeed, the leading term in $w_{ij}(t)$ is $u_{ih} u_{jh} \psi_{hh}$ as $t \to T$. Therefore the sign of $w_{ij}$ is the same as the sign of $u_{ih} u_{jh} \psi_{hh}$. The leading term of $w_{ij} w_{jk} w_{ki}$ as $t \to T$ is

$$u_{ih}^2 u_{jh}^2 u_{kh}^2 \psi_{hh}^2 \geq 0.$$ 

This shows that structural balance occurs eventually for generic initial values. Here the generic condition is used to ensure that all entries $u_{ij}$ of the orthogonal matrix $[u_{ij}]$ are not zero and $\psi_{hh}$ is the unique term with the maximum growth rate. Finally, if $T = \infty$ and $\lim_{t \to \infty} |V(t)| = 0$, then by the following Corollary, $\lim_{t \to \infty} V'(t) = 0$.

**Corollary 5.4.1.** If $V(t)$ solves Equation (5.6), then

$$|V'(t)|^2 = |V(t)|^4 - V^T C V. \quad (5.9)$$

In particular, if $\lim_{t \to T} V(t) = 0$, then $\lim_{t \to T} V'(t) = 0$.
Indeed, by \(|V(t)|^2)' = 2V^TV'\) and Equation (5.6) that \(V'' = 2|V|^2V - CV\), we have \((|V(t)|^2)'' = 2(V')^TV' + 2V^TV'' = 2|V'(t)|^2 + 2V^T(2|V|^2V - CV) = 2|V'(t)|^2 + 4|V|^4 - 2V^TCV\). Comparing it with Equation (5.7), we see the corollary holds. The last statement of the corollary follows from Equation (5.9).

We now prove Proposition 5.4.1 using several lemmas. For simplicity, if \(f(t)\) is a function defined on an open interval \((a, b)\) (here \(b\) may be \(+\infty\)), we say \(f\) has property \(P\) (e.g., positive, non-negative, monotonic, convex etc) near \(b\) if there exists \(\epsilon > 0\) such that the restriction of \(f\) on the interval \((b - \epsilon, b)\) (if \(b < +\infty\) or \((\frac{1}{2}, \infty)\) (if \(b = +\infty\)) has property \(P\). For example, \(t^4 + 3t^3 - 9t^2 + t - 5\) is positive and convex near \(+\infty\). The notation \(C^k(a, b)\) stands for all functions \(f(t)\) for which \(f, f', \ldots, f^{(k)}\) are continuous on the interval \((a, b)\).

**Lemma 5.4.2.** Suppose \(f \in C^2(a, b)\) and \(p(t) > 0\) on \((a, b)\) such that

\[
f''(t) = p(t)f(t) \text{ on } (a, b)
\]

and \(f(t)\) is not identically zero on any sub-interval. Then

1. \(f\) has at most one root in \((a, b)\).
2. \(f\) has the same sign near \(b\) (i.e., always positive or negative).
3. \(f\) is monotonic near \(b\).
4. The limit \(\lim_{t \to b^-} f(t)\) exists (the limit may be \(\pm\infty\)).

Furthermore, if \(g''(t) \geq q(t)g(t)\) and \(q \geq 0\) on \((a, b)\), then either \(g(t) > 0\) near \(b\) or \(g(t) \leq 0\) near \(b\).

**Proof.** To see part (1), if \(f\) has two roots in \((a, b)\), then since \(f\) is not identically zero on any interval, there exist two adjacent roots \(f(c) = f(d)\) where \(c < d\) and \(f\) has no roots in the open interval \((c, d)\). By replacing \(f\) by \(-f\) if necessary, we may assume that the restriction function \(f|_{(c, d)} > 0\). Then by Equation (5.10), \(f''(t) = p(t)f(t) > 0\) on \((c, d)\). Therefore \(f|_{[c, d]}\) is a convex function which has two minimum values at \(c, d\). This implies that \(f|_{[c, d]} \equiv 0\) which contradicts the assumption.
Part (2) follows from the part (1) easily.

To see part (3), we first show that $f'(t)$ has at most two roots in the open interval $(a, b)$. Suppose otherwise that $f'(t)$ has three roots in $(a, b)$. Then by the Mean Value Theorem, $f''(t)$ has two roots in $(a, b)$, one in each interval bounded by roots of $f'(t)$. But $f''(t) = p(t)f(t)$ with $p(t) > 0$ says $f$ and $f''$ have the same roots. This implies that $f$ has two roots in $(a, b)$ which contradicts the part (1). Since $f'$ has only two roots, it follows that $f'(t) > 0$ near $b$ or $f'(t) < 0$ near $b$. Therefore $f$ is monotonic near $b$.

Part (4) follows from the well-known theorem that if $h(t)$ is monotonic in an open interval $(a, b)$, then the limit $\lim_{t \to b^-} h(t)$ always exists (limit value of the limit may be $\pm\infty$).

Finally, to prove the last statement, we consider two cases. In the first case, there are no sequence $\{r_m\}$ of roots of $g$ such that $\lim_{m \to \infty} r_m = b$. Then clearly $g(t) > 0$ or $g(t) < 0$ near $b$. In the remaining case, we have an increasing sequence of roots, $r_1 < r_2 < \cdots < r_m < \cdots$ of $f$ such that $\lim_{m \to \infty} r_m = b$. We claim that $g|_{(r_1, b]} \leq 0$. Suppose otherwise that $g(t_0) > 0$ for some $t_0 \in (r_1, b)$. Let $c$ (respectively $d$) be the largest (respectively smallest) root of $g$ such that $c < t_0$ (respectively $d > t_0$). By the assumption, both $c$ and $d$ exist. Furthermore $c < t_0 < d$ and $g$ has no root in the interval $(c, d)$. Therefore, due to $g(t_0) > 0$, $g|_{(c, d]} > 0$. By the condition $g'' = q g$, with $q \geq 0$, we see that $q''|_{(c, d]} \geq 0$. Therefore $g(t)$ is convex on $[c, d]$ and has two minimum values $0(= g(c) = g(d))$. But that implies $g|_{[c, d]} = 0$ and contradicts $g(t_0) > 0$. \hfill \qed

**Corollary 5.4.2.** Suppose $\lambda(t) = [\lambda_1(t), \ldots, \lambda_n(t)]^T$ solves the ODE Equation (5.8) and $a_1 \geq 0$, $a_2, \cdots, a_n \leq 0$ on the maximum interval $[0, T)$. Then

1. $\lim_{t \to T} |V(t)|^2 = \lim_{t \to T} \sum_{k=1}^n \lambda_k^2(t)$ exists.

2. For all $i$, $\lim_{t \to T} \lambda_i^2(t)$ exists.

3. Assuming that $\lim_{t \to T} |V(t)|^2 = \infty$, for $i \neq j$, either $\lambda_i^2(t) \geq \lambda_j^2(t)$ near $T$ or $\lambda_i^2(t) \leq \lambda_j^2(t)$ near $T$.

4. For $\lim_{t \to T} |V(t)|^2 < \infty$, then all limits $\lim_{t \to T} \lambda_k^2(t)$ are finite and can be ordered.
Proof. For (1), by Lemma 5.4.1, \( \phi(t) = |V(t)|^2 \) is convex in \([0, T)\). Hence \( \phi(t) \) is monotonic near \( T \) and \( \lim_{t \to T} \phi(t) \) exists.

For (2), let us assume \( V(t) \) is not identically zero. Otherwise the result holds trivially. If \( i \geq 2 \), then by Equation (5.8), \( a_i \leq 0 \) and Lemma 5.4.1, we have \( 2|V(t)|^2 - a_i > 0 \) on \([0, T)\) and \( \lambda''_i(t) = p_i(t) \lambda_i(t) \), where \( p_i(t) = 2|V(t)|^2 - a_i > 0 \). Therefore by Lemma 5.4.2, \( \lim_{t \to T} \lambda^2_i(t) \) exists. If \( i = 1 \), we have \( \lambda^2_1(t) = |V(t)|^2 - \sum_{k=2}^{n} \lambda^2_k(t) \).

Now if \( \lim_{t \to T} |V(t)|^2 < +\infty \), then \( \lim_{t \to T} \lambda^2_i(t) < +\infty \), for \( k = 2, \ldots, n \). Therefore, \( \lim_{t \to T} \lambda^2_i(t) = \lim_{t \to T} (|V(t)|^2 - \sum_{k=2}^{n} \lambda^2_k(t)) \) exists and is finite. If \( \lim_{t \to T} |V(t)|^2 = +\infty \), then \( 2|V(t)|^2 - a_1 > 0 \) for \( t \) near \( T \). The equation \( \lambda''_i(t) = (2|V(t)|^2 - a_1) \lambda_i(t) \) is of the form \( \lambda''_i(t) = p_1(t) \lambda_1(t) \) where \( p_1(t) > 0 \) near \( T \). Therefore by Lemma 5.4.2(4), \( \lim_{t \to T} \lambda^2_i(t) \) exists.

For (3), since \( \lambda^2_i \geq \lambda^2_j \) is the same as \( |\lambda_i| \geq |\lambda_j| \) and if \( \lambda_i \) solve Equation (5.8) so is \(-\lambda_i(t)\). The assumption that \( \lim_{t \to T} |V(t)|^2 = \infty \) implies \( |V(t)|^2 - a_k \geq 0 \) for \( t \) near \( T \).

We may assume, using Lemma 5.4.2(2), that \( \lambda_i(t) \geq 0 \) and \( \lambda_j(t) \geq 0 \) near \( T \). Our goal is to show, under the assumption that \( \lambda_i, \lambda_j \geq 0 \) near \( T \), either \( \lambda_i \geq \lambda_j \) or \( \lambda_i \leq \lambda_j \) near \( T \).

Without loss of generality, we may assume that \( a_j \geq a_i \) and \( a_i \leq 0 \) (Note \( a_1 \geq 0 \) and \( a_2, \ldots, a_n \leq 0 \)). Then using Equation (5.8), we have

\[
(\lambda_i - \lambda_j)'' = \lambda''_i - \lambda''_j = (2|V(t)|^2 - a_i) \lambda_i - (2|V(t)|^2 - a_j) \lambda_j
\]

\[
= (2|V(t)|^2 - a_i)(\lambda_i - \lambda_j) + (a_j - a_i) \lambda_j \geq (2|V(t)|^2 - a_i)(\lambda_i - \lambda_j).
\]

Since \( a_j - a_i \geq 0 \) and \( \lambda_j \geq 0 \). Now due to \( a_i \leq 0 \), \( 2|V(t)|^2 - a_i \geq 0 \). Therefore \( (\lambda_i - \lambda_j)'' \geq g(t)(\lambda_i - \lambda_j) \) when \( g \geq 0 \) near \( T \). By the last proof of Lemma 5.4.2, we see either \( \lambda_i > \lambda_j \) or \( \lambda_i \leq \lambda_j \) near \( T \). This ends the proof.

Part (4) follows from part (2) and the assumption which implies \( \lim_{t \to T} \lambda^2_k \) are finite real numbers. Therefore, we can order them.

Now, let us prove Proposition 5.4.1. *

Proof. There are two cases depending on the maximum interval of existence \([0, T)\) being finite, i.e., \( T < +\infty \) or infinite \([0, +\infty)\), i.e., \( T = +\infty \).

Case 1. \( T < +\infty \). Recall the basic global existence of solution to ODE [188].
Theorem 5.4.2. (Existence) Suppose \( F(t, x) \in C^1(\mathbb{R} \times \mathbb{R}^m) \) and \( [t_0, T) \) is the maximum interval of existence of the solution \( x(t) \) to \( x'(t) = F(t, x(t)) \) with \( x(t_0) = x_0 \). Then the path \( \{(t, x(t))| t \in [t_0, T)\} \) does not lie in any bounded set in \( \mathbb{R} \times \mathbb{R}^n \).

Now, for \( T < +\infty \), Theorem 5.4.2 implies \( \lim_{t \to T} |V(t)| = +\infty \). Indeed, if \( \lim_{t \to T} |V(t)| < +\infty \), then \( V(t) \) is bounded on \( [0, T) \). This implies \( W'(t) = V(t) \cdot V(t)^T \) is bounded, but \( W(t) = W(0) + \int_0^t W'(s)ds \). Therefore \( W(t) \) is bounded. This implies the solution \( (V(t), W(t)) \) for \( t \in [0, T) \) lies in a bounded set in \( \mathbb{R}^n \times \mathbb{R}^{n \times n} \) which contradicts Theorem 5.4.2.

Now \( \lim_{t \to T} |V(t)|^2 = \lim_{t \to T} \sum_{j=1}^n \lambda_j^2(t) = +\infty \) implies, by Corollary 5.4.2(2), there exists \( i \) for which \( \lim_{t \to T} \lambda_i^2(t) = +\infty \). Furthermore, by Corollary 5.4.2(3), there exists an index \( h \) for which \( \lambda_h^2(t) \geq \lambda_j^2(t) \) near \( T \) for all \( j \). Thus \( \lambda_h^2(t) \) has the largest growth rate tending \( +\infty \) as \( t \to T \).

For generic initial value \( V(0) \) and \( W(0) \), \( \lambda_h^2(t) \) is the unique term of maximum growth rate. Therefore, we see that part (1) of Theorem 5.4.1 holds.

**Case 2.** If \( T = +\infty \), let us assume that the limit \( \lim_{t \to T} |V(t)| = L > 0 \) and show that structural balance occurs eventually.

By Corollary 5.4.2, we may assume that \( \lambda_h^2(t) \geq \lambda_j^2(t) \) for \( t \) near \( \infty \) for all \( j \) (if \( \lim_{t \to T} |V(t)|^2 = \infty \) or \( \lim_{t \to T} \lambda_h^2(t) = \max \{\lim_{t \to T} \lambda_i^2(t)|i = 1, \ldots, n\} < \infty \). Let \( L' = \lim_{t \to T} \lambda_h^2(t) \). Then \( L' > 0 \) since \( L > 0 \). In the case of \( \lim_{t \to T} |V(t)|^2 = \infty < \infty \), for generic initial value, we may assume that \( \lim_{t \to T} \lambda_h^2(t) \) is the unique maximum value among all \( \lim_{t \to T} \lambda_j^2(t) \), \( i = 1, \ldots, n \). Then we see that

\[
\int_0^t |\lambda_h^2(s)|ds \geq (t - t_0)L'' + c_0
\]

for some constants \( L'' > 0 \) and \( c_0 \). It tends to \( +\infty \) as \( t \to \infty \).

Furthermore, by the Cauchy inequality,

\[
\left( \int_0^t \lambda_i(s)\lambda_j(s)ds \right)^2 \leq \int_0^t \lambda_i^2(s)ds \cdot \int_0^t \lambda_j^2(s)ds
\]

and

\[
\int_0^t \lambda_i^2(s)ds \leq \int_0^t \lambda_h^2(s)ds + c_1
\]
for $t$ large, we see the growth rate of $\int_0^t \lambda_i(s)\lambda_j(s)ds$ is at most that of $\int_0^t \lambda_k^2(s)ds$ as $t \to \infty$. This shows, by the same argument, that the growth rate of $w_{ij}(t)$ is dominated by $u_{ih}u_{jh}\int_0^t \lambda_k^2(s)ds$. Therefore, structural balance occurs again for generic initial values.

Finally, we prove that in Case 1 or in Case 2 that $T = \infty$ such that $\lim_{t \to \infty} |V(t)| = L > 0$, the limit $\lim_{t \to \infty} V(t)/|V(t)|$ exists.

By corollary 5.4.1 (2), $\lim_{t \to T} \lambda_i$ exists in $[-\infty, \infty]$. Therefore, if $\lim_{t \to \infty} |V(t)| = L$ is a finite positive number, then $\lim_{t \to \infty} V(t)$ exists in $\mathbb{R}^n - \{0\}$. Hence $\lim_{t \to \infty} V(t)/|V(t)|$ exists. In the remaining cases, we have $\lim_{t \to T} |V(t)| = \infty$. In this case, by the argument above, we see that $\lambda_i(t)$ has the same sign near $T$. We claim that the function $\lambda_i/\lambda_j$ is monotonic near $T$. Indeed, by the quotient rule for derivative, the sign of derivative of $\lambda_i/\lambda_j$ is the same as that of $h(t) = \lambda'_i \lambda_j - \lambda'_j \lambda_i$. Now $h'(t) = \lambda''_i \lambda_j - \lambda''_j \lambda_i = (2|V|^2 - a_i)\lambda_i \lambda_j - (2|V|^2 - a_j)\lambda_j \lambda_i = (a_j - a_i)\lambda_i \lambda_j$. Therefore, either $h'(t)$ has the same sign for $t$ near $T$ (when $a_i \neq a_j$) or $h'(t) = 0$ (when $a_i = a_j$). If $h'(t) = 0$, then $\lambda_i/\lambda_j$ is a constant near $T$ and the claim follows. If $h'(t)$ has the same sign near $T$, then $h(t)$ is strictly monotonic near $T$. Therefore, $h(t)$ has the same sign for $t$ near $T$. As a consequence we see that $\lambda_i/\lambda_j$ is a monotonic near $T$.

In particular, the limit $\lim_{t \to T} \lambda_i(t)/\lambda_j(t)$ exists. Since $\lim_{t \to T} \sum_{k=1}^n \lambda_k(t)^2 = \infty$, this implies the limit $\lim_{t \to T} \frac{\lambda_i(t)}{\sqrt{\sum_{j=1}^n \lambda_j(t)^2}} = \lim_{t \to T} \frac{\pm 1}{\sqrt{\sum_{j=1}^n \lambda_j(t)^2/\lambda_i(t)^2}}$ exists for any index $i$.

The last statement is the same as that $\lim_{t \to T} V(t)/|V(t)|$ exists.

This ends the proof of Theorem 5.4.1.

\[\square\]

**Corollary 5.4.3.** Suppose $\lim_{t \to T} |V(t)| = \infty$. Then $A := \lim_{t \to T} W(t)^2/|V(t)|^2$ exists and $\lim_{t \to T} V/|V|$ is an eigenvector of $A$ associated to the eigenvalue one.

To see this, let $v$ be $\lim_{t \to T} V/|V|$. From $W^2 = VV^T - C$ and $\lim_{t \to T} |V(t)| = \infty$, we see that $W^2/|V|^2 = (|V|/|V|)(V/|V|)^T - C/|V|^2$ implies $\lim_{t \to T} W^2/|V|^2 = vv^T$. Since $vv^Tv = v$ due to $|v|^2 = 1$, the result follows.

We end the appendix by making several remarks and a conjecture.

The 1-dimensional case of equation $V''(t) = (2|V(t)|^2 - C) \cdot V(t)$, $V(t) \in \mathbb{R}^n$ is $y''(t) = 2y^3(t) - cy(t)$. The function $f(t) = \frac{a}{\sinh(at+b)}$ solves $f'' = 2f^3 + a^2f$ and
$g(t) = \frac{a}{\sin(at+b)}$ solves $g'' = 2g^3 - a^2g$. We may assume $a > 0$. Therefore, if $b < 0$, then both $f$ and $g$ exist only on a finite maximum interval (it is $[0, -\frac{b}{a})$ for $f(t)$), i.e., $T < +\infty$. If $b > 0$, then the function $f(t)$ exists on $[0, +\infty)$ but the function $g(t)$ exists only on a finite interval $[0, T)$. It indicates that if $C$ has a positive eigenvalue $a_1 > 0$, then the solution $\lambda_1(t)$ may exist only on a finite interval.

This prompts us to conjecture that

**Conjecture 5.4.1.** If the initial value matrix $C = V(0)V(0)^T - W(0)W(0)^T$ has a positive eigenvalue (i.e., $a_1 > 0$), then the maximum interval $[0, T)$ of existence for the solution $(V(t), W(t))$ of the co-evolution equation $V' = WV$ and $W' = VV^T$ is finite, i.e., $T < +\infty$.

If the conjecture holds, by Theorem 5.4.1, we see structural balance must occur eventually for generic initial value $C$ which has a positive eigenvalue. Therefore, it also justifies our experimental observation that structural balance occurs almost all the time.

### 5.4.2 Analysis of Social Tie Evolution

The analysis on the evolution of $V$ in the previous section shows convergence. To further understand the community formed at the limit of convergence, we need to study the evolution of $W$. The following theorem explains the reason behind the appearance of structure balance when at least one eigenvalue of $W(t)$ tends to infinity. This was proved in [140].

**Theorem 5.4.3 ([140]).** Suppose $W(t), t \in [0, T)$, is a continuous family of symmetric matrices such that

1. $W(t)$ has a unique largest eigenvalue, denoted by $\beta_1(t)$, which tends to infinity as $t \to T$,

2. all eigenvectors of $W(t)$ are time independent, and

3. all components of the $\beta_1(t)$ eigenvector are not zero.
Then

\[ w_{ij}w_{jk}w_{ki} > 0, \forall i, j, k, i \neq j, j \neq k, i \neq k, \]

for all time \( t \) close to \( T \), i.e., the structural balance of the whole graph is satisfied.

If (2) does not hold, we have

\[ w_{ij}w_{jk}w_{ki} \geq 0, \forall i, j, k, i \neq j, j \neq k, i \neq k. \]

**Proof.** Let all eigenvalues of \( W(t) \) be \( \beta_1, \ldots, \beta_n \) and \( \beta_1 \) be the unique largest eigenvalue. Then

\[ W(t) = H \cdot \text{diag}(\beta_1(t), \ldots, \beta_n(t)) \cdot H^T, \]

for some time independent orthogonal matrix \( H \) whose first column is the \( \beta_1 \) eigenvector.

This implies that \( w_{ij} = \sum_k H_{ik} \cdot \beta_k(t) \cdot H_{jk} \). Since all \( \beta_i(t) < \beta_1(t) \) for \( i \neq 1 \), the growth rate of \( w_{ij}(t) \) as \( t \to T \) is the same as \( H_{i1} \beta_1(t) H_{j1} \).

Therefore, the sign of \( w_{ij} \) is the same as the sign of \( H_{i1} \cdot H_{j1} \). Thus,

\[ \text{sgn}(w_{ij}w_{jk}w_{ki}) = \text{sgn}(H_{i1}H_{j1}H_{j1}H_{k1}H_{k1}H_{i1}) = \text{sgn}(H_{i1}^2H_{j1}^2H_{k1}^2) > 0. \]

The same argument shows that above sign is non-negative if some component of the eigenvector is zero.

Furthermore, the two antagonistic communities are given by \( U^+ = \{ i \in V | u_i > 0 \} \) and \( U^- = \{ i \in V | u_i < 0 \} \) where \( u = [u_1, \ldots, u_n]^T \) is a \( \beta_1(t) \) eigenvector. All edges connecting vertices within the same community are positive while edges connecting two vertices in different communities are negative. When one of \( U^+, U^- \) is empty, the network has only one community.

The model in [140] is the Riccati equation \( W' = W^2 \), whose solution is \( W(t) = W(0)(1 - W(0)t)^{-1} \). As a consequence, Marvel et. al. [140] showed that structure balance occurs in the Riccati equation \( W' = W^2 \) for generic initial parameter \( W(0) \) with a positive eigenvalue at finite time. Our model strictly generalizes the previous model and consider how the initial opinions may influence the system evolution. In the following we show how to solve the general Riccati equation and also when an eigenvalue goes to infinity.
We carry out rigorous analysis of the general form of the matrix Riccati equation as stated below.

\[
\begin{align*}
W' & = W^2 + C \\
W(0) & = B.
\end{align*}
\] (5.11)

The equation in our co-evolution model, i.e., Equation (5.5), satisfies \( C + B^2 = V(0)V(0)^T \). Notice that the right-hand side \( V(0)V(0)^T \) is an \( n \times n \) matrix with rank one, which is a special condition. The analysis in this subsection applies for general matrices \( B, C \).

By using a result in Reid [163], we can turn the matrix Riccati equation to a linear ODE system.

**Lemma 5.4.3** ([163]). The ODE system in Equation (5.11) is equivalent to the following system

\[
\begin{align*}
Y' & = Z \\
Y(0) & = I \\
Z' & = -CY \\
Z(0) & = -B,
\end{align*}
\] (5.12)

where \( Y' = -WY \) and \( Z = -WY \), and \( Y, Z, W \in \mathbb{R}^{n \times n} \).

First, we show that we can get Equations (5.12) from Equations (5.11), i.e., (5.11) \( \Rightarrow \) (5.12).

We know that \( Y \) exists and \( Z = -WY \). Thus, \( Y' = Z \). Then

\[
Z' = -W'Y - WY' = -(W^2 + C)Y - W(-WY).
\]

\[
= -W^2Y - CY + W^2Y
\]

\[
= -CY.
\]

At the same time, \( Z(0) = -W(0)Y(0) = -W(0) = -B \). Thus, this direction is satisfied.

Second, let us prove that Equation (5.11) can be obtained from Equation (5.12).

Since \( Y(0) = I \), by continuity, \( Y(t)^{-1} \) exists for \( t \in [0, \epsilon] \), for \( \epsilon > 0 \). Since \( Z = -WY \), we have \( W = -ZY^{-1} \). Note \( (Y^{-1})' = -Y^{-1} \cdot Y' \cdot Y^{-1} \), since \( Y \cdot Y^{-1} = I \Rightarrow Y' \cdot Y^{-1} + \).
\[ Y \cdot (Y^{-1})' = 0. \] Therefore:

\[
W' = -Z' \cdot Y^{-1} - Z \cdot (Y^{-1})'
= CY \cdot Y^{-1} - Z(-Y^{-1} \cdot Y' \cdot Y^{-1})
= C + Z \cdot Y^{-1} \cdot (-WY) \cdot Y^{-1}
= C - Z \cdot Y^{-1} \cdot W
= C + W^2.
\]

Clearly \( W(0) = -Z(0) = B \). This completes the proof.

From Lemma 5.4.3, we can focus on solving the linear Ordinary Differential Equation (ODE) in Equation (5.12), which can be written in a matrix form. The analysis below is new.

\[
\begin{bmatrix} Y \\ Z \end{bmatrix}' = \begin{bmatrix} 0 & I \\ -C & 0 \end{bmatrix} \begin{bmatrix} Y \\ Z \end{bmatrix},
\]

where we define \( A = \begin{bmatrix} 0 & I \\ -C & 0 \end{bmatrix} \in \mathbb{R}^{2n \times 2n} \) in block form and \( X = \begin{bmatrix} Y \\ Z \end{bmatrix} \in \mathbb{R}^{2n \times n} \).

Now, let us solve the evolution equation \( X' = AX \), where \( X(0) = \begin{bmatrix} I \\ -B \end{bmatrix} \). It is well known that the solution is,

\[
X(t) = \left( \sum_{n=0}^{\infty} \frac{t^n A^n}{n!} \right) X(0).
\] (5.13)

Let us compute \( A^n \) using the block multiplication of matrices [67, 20].

\[
A^2 = \begin{bmatrix} 0 & I \\ -C & 0 \end{bmatrix} \begin{bmatrix} 0 & I \\ -C & 0 \end{bmatrix} = \begin{bmatrix} -C & 0 \\ 0 & -C \end{bmatrix}.
\]

It implies:

\[
A^{2n} = \begin{bmatrix} (-1)^n C^n & 0 \\ 0 & (-1)^n C^n \end{bmatrix},
\]

\[
A^{2n+1} = A^{2n} \cdot A = \begin{bmatrix} (-1)^n C^n & 0 \\ 0 & (-1)^n C^n \end{bmatrix} \begin{bmatrix} 0 & I \\ -C & 0 \end{bmatrix} = \begin{bmatrix} 0 & (-1)^n C^n \\ (-1)^{n+1} C^{n+1} & 0 \end{bmatrix}.
\]

Recall that \( W = -ZY^{-1} \). Now we are ready to solve for \( W \).

For general matrices \( B, C \) we can solve for \( W(t) \) as shown in the following theorem.
**Theorem 5.4.4.** The solution \( W(t) \) is given by the explicit formula that \( W(t) = -Z(t) \cdot Y(t)^{-1} \), where

\[
Y(t) = \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n} C^n}{(2n)!} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} t^{2n+1} C^n B}{(2n+1)!}
\]

\( Z(t) = \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n} C^n B}{(2n)!} + \sum_{n=0}^{\infty} \frac{(-1)^{n+1} t^{2n+1} C^n B}{(2n+1)!}. \)

*Proof.* According to Equation (5.13),

\[
X(t) = \sum_{n=0}^{\infty} \frac{t^{2n} A^{2n}}{(2n)!} \cdot X(0) + \sum_{n=0}^{\infty} \frac{t^{2n+1} A^{2n+1}}{(2n+1)!} \cdot X(0)
\]

\[
= \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n}}{(2n)!} \begin{bmatrix} C^n & 0 \\ 0 & C^n \end{bmatrix} \begin{bmatrix} 1 \\ -B \end{bmatrix} + \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n+1}}{(2n+1)!} \begin{bmatrix} 0 & C^n \\ -C^{n+1} & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -B \end{bmatrix}
\]

\[
= \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n}}{(2n)!} \begin{bmatrix} C^n \\ -C^n B \end{bmatrix} + \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n+1}}{(2n+1)!} \begin{bmatrix} C^n B \\ C^{n+1} \end{bmatrix}
\]

\[
= \begin{bmatrix}
\sum_{n=0}^{\infty} \frac{(-1)^n t^{2n} C^n}{(2n)!} + \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n+1} C^n B}{(2n+1)!} \\
\sum_{n=0}^{\infty} \frac{(-1)^n t^{2n} C^n B}{(2n)!} + \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n+1} C^n B}{(2n+1)!}
\end{bmatrix}
\]

\[\triangleq \begin{bmatrix}
Y \\
Z
\end{bmatrix}.
\]

\( \blacksquare \)

In our model, we assume that the initial tie matrix \( W(0) \) is symmetric. Thus the social tie evolution follows \( W' = W^2 + C \), where \( W(0) = B \), both \( B \) and \( C \) are symmetric. We are able to derive more detailed closed form solutions for the matrix Riccati equation in this setting.

Since \( C \) is symmetric, there exists an orthogonal matrix \( U \) such that \( U^T C U \) is a diagonal matrix: \( U^T C U = \text{diag}\{a_1^2, \ldots, a_k^2, -d_1^2, \ldots, -d_l^2, 0, \ldots, 0\} \), where \( a_i > 0, d_j > 0 \). Furthermore, if \( BC = CB \), by the simultaneous diagonalization theorem, we may choose \( U \) such that both \( U^T C U \) and \( U^T B U \) are diagonal. By Lemma 5.3.1(1), without loss of generality, we are going to solve the equation with the initial opinion vector \( U^T V(0) \) and initial weight matrix \( U^T W(0)U \). This leads to a system as below

\[
\begin{cases}
(U^T W U)' = (U^T W U)^2 + U^T C U \\
U^T W(0)U = U^T B U.
\end{cases}
\]
The solution of this system can be easily transformed back to the solution to the original system by conjugation. For simplicity, the $(i, j)$-th entry of a matrix $M$ will be denoted by $M_{ij}$. Define $\beta_{ii}$ the $i$th diagonal element of $UTBU$, i.e., $\beta_{ii} = (UTBU)_{ii}$.

Let us now work out explicitly the matrices $Y$ and $Z$ in Theorem 5.4.4.

1. For the positive eigenvalue $a_i$ of $C$,  
   \[
   \sum_{n=0}^{\infty} \frac{(-1)^n t^{2n} a_i^{2n}}{(2n)!} = \cos (a_i t)
   \]
   \[
   \sum_{n=0}^{\infty} \frac{(-1)^{n+1} t^{2n+1} a_i^{2n}}{(2n + 1)!} = -\frac{1}{a_i} \sin (a_i t).
   \]
   
   Thus, $(UTY)_{ii} = \cos (a_i t) - \frac{1}{a_i} \sin (a_i t) \cdot \beta_{ii}$. Similarly, we have
   \[
   \sum_{n=0}^{\infty} \frac{(-1)^{n+1} t^{2n} a_i^{2n}}{(2n)!} = -\cos (a_i t)
   \]
   \[
   \sum_{n=0}^{\infty} \frac{(-1)^{n+1} t^{2n+1} a_i^{2n+2}}{(2n + 1)!} = -\sin (a_i t) \cdot \beta_{ii}.
   \]
   
   So, $(UTZ)_{ii} = -\cos (a_i t) \beta_{ii} - a_i \sin (a_i t)$.

2. For the zero eigenvalues of $C$, i.e., $c_i = 0$, we have $(UTY)_{ii} = 1 - t \cdot \beta_{ii}$ and $(UTZ)_{ii} = -\beta_{ii}$.

3. For the negative eigenvalue $c_i = -d_i^2$, we have
   \[
   (UTY)_{ii} = \sum_{n=0}^{\infty} \frac{t^{2n} d_i^{2n}}{(2n)!} + (-1)^n \sum_{n=0}^{\infty} \frac{t^{2n+1} d_i^{2n}}{(2n + 1)!} \beta_{ii} = \cosh (d_i t) - \frac{1}{d_i} \sinh (d_i t) \beta_{ii}
   \]
   \[
   (UTZ)_{ii} = \sum_{n=0}^{\infty} (-1)^n \frac{t^{2n} d_i^{2n}}{(2n)!} \beta_{ii} + \sum_{n=0}^{\infty} \frac{t^{2n+1} d_i^{2n+2}}{(2n + 1)!} = -\cosh (d_i t) \beta_{ii} + d_i \sinh (d_i t).
   \]
   
   So we can summarize the above formulas as: $UTY = D_1 - D_2 U^T BU$ and $UTZ = D_3 - D_1 U^T BU$, where
   
   \[
   D_1 = \text{diag}\{\cos (a_1 t), \ldots, \cos (a_k t), \cosh (d_1 t), \ldots, \cosh (d_l t), 1, \ldots, 1}\},
   \]
   \[
   D_2 = \text{diag}\{\sin(a_1 t)/a_1, \ldots, \sin(a_k t)/a_k, \sinh (d_1 t)/d_1, \ldots, \sinh (d_l t)/d_l, 1, \ldots, 1}\},
   \]
   \[
   D_3 = \text{diag}\{-a_1 \sin (a_1 t), \ldots, -a_k \sin (a_k t), d_1 \sinh (d_1 t), \ldots, d_l \sinh (d_l t), 0, \ldots, 0\},
   \]
   
   Then we analyze a special case when $BC = CB$. In this case we can characterize the conditions when structural balance is guaranteed to occur.
Now, we consider the special case that $BC = CB$. Using a basic fact that two commuting symmetric matrices can be simultaneously orthogonally diagonalized [94].

Let $U$ be an orthogonal matrix such that

$$U^T C U = \text{diag}(a_1^2, \cdots, a_k^2, -d_1^2, \cdots, -d_l^2, 0, \cdots, 0)$$

$$U^T B U = \text{diag}(\lambda_1, \cdots, \lambda_k, \mu_1, \cdots, \mu_l, \delta_1, \cdots, \delta_h).$$

where $a_i, d_j > 0$.

In this case we can further simplify the solution.

$$W = - Z \cdot Y^{-1}$$

$$= U \text{diag} \left( \frac{a_1 \sin (a_1 t) + \cos (a_1 t) \lambda_1}{\cos (a_1 t) - \frac{1}{a_1} \sin (a_1 t) \lambda_1}, \cdots, \frac{a_k \sin (a_k t) + \cos (a_k t) \lambda_k}{\cos (a_k t) - \frac{1}{a_k} \sin (a_k t) \lambda_k}, \frac{d_1 \sinh (d_1 t) - \cosh (d_1 t) \mu_1}{\cosh (d_1 t) - \frac{d}{\delta_1} \sinh (d_1 t) \mu_1}, \cdots, \frac{d_l \sinh (d_l t) - \cosh (d_l t) \mu_l}{\cosh (d_l t) - \frac{d}{\delta_l} \sinh (d_l t) \mu_l}, \delta_1 \cdots, \delta_h \right) U^T. \quad (5.16)$$

Note that above equation for $W(t)$ implies that $V(t)$ is an eigenvector of $W(t)$ for all time $t$.

Now we are ready to analyze the behavior of $W$ over time in the case of a symmetric initial condition matrix $W(0)$. We start with a technical lemma.

**Lemma 5.4.4.** 1. If $a > 0$, then there exists $T \in (0, \frac{\pi}{a})$, such that for all $\lambda$

$$\lim_{t \to T} \frac{a \sin (at) + \lambda \cos (at)}{\frac{a}{\lambda} \sin (at) + \cos (at)} = +\infty.$$

2. If $d > 0$, then there exists $T \in (0, \infty]$, such that

$$\lim_{t \to T} \frac{\cosh(dt) \mu - d \sinh(dt)}{\cosh(dt) - \frac{d}{\mu} \sinh(dt)} = +\infty,$$

if and only if $\mu > d$. In the case $\mu \leq d$, the limit for $T = \infty$ exists and is finite.

3. If $\delta > 0$, then there exists $T = \frac{1}{\delta}$, such that

$$\lim_{t \to T} \frac{\delta}{1 - t \delta} = +\infty.$$

For all cases mentioned above, the convergence rate is $O\left(\frac{1}{T-\pi}\right)$. 
Proof. For (1), we reorganize
\[
\frac{a \sin (at) + \lambda \cos (at)}{-\frac{\lambda}{a} \sin (at) + \cos (at)} = \frac{\lambda \cot (at) + \frac{a}{\lambda}}{\cot (at) - \frac{\lambda}{a}}.
\]
Since \(\cot (at)\) is strictly decreasing from \(\infty\) to \(-\infty\) in the range \((0, \frac{\pi}{a})\), there exists a \(T \in (0, \frac{\pi}{a})\), such that \(\cot (at) = \frac{\lambda}{a}\). As \(t\) approaches \(T\) from the left, \(\cot (at) - \frac{\lambda}{a} > 0\).

When \(t \to T\) the numerator becomes
\[
\lambda \left(\frac{\lambda}{a} + \frac{a}{\lambda}\right) = \lambda^2 + \frac{a^2}{\lambda} > 0.
\]
This confirms (i).

Then we consider the convergence rate as \(t \to T\).
\[
\frac{\lambda \cot (at) + \frac{a}{\lambda}}{\cot (at) - \frac{\lambda}{a}} = \frac{\lambda \cot (-a(T-t) + aT) + \frac{a}{\lambda}}{\cot (-a(T-t) + aT) - \frac{\lambda}{a}} = \frac{\lambda \left(\frac{\lambda}{a} + \frac{a}{\lambda}\right) \cot (a(T-t))}{1 + \frac{\lambda^2}{a^2} - 1}.
\]
The Taylor series of \(\cot (x)\) is \(\frac{1}{x} - \frac{x}{3} + o(x)\). Given that \((T-t) \to 0\), the convergence rate is \(O\left(\frac{1}{(T-t)^2}\right)\).

For (2), reorganize
\[
\frac{\cosh(dt)\mu - d \sinh(dt)}{\cosh(dt) - \frac{\mu}{d} \sinh(dt)} = \frac{\coth(dt)\mu - d}{\coth(dt) - \frac{\mu}{d}}.
\]
Since the function \(\coth(dt)\) is strictly decreasing from \(\infty\) to 1, if \(\mu > d\), there exists \(T \in (0, \infty)\) such that \(\coth(dT) = \mu/d\). Furthermore, as \(t\) approaches \(T\) from the left, the denominator is positive. But \(\lim_{t \to T} \coth(dt)\mu - d = \frac{\mu^2 - d^2}{d} > 0\). It follows that
\[
\lim_{t \to T} \frac{\cosh(dt)\mu - d \sinh(dt)}{\cosh(dt) - \frac{\mu}{d} \sinh(dt)} = +\infty.
\]
If \(\mu \leq d\), the function is smooth on \([0, \infty)\) and the limit is finite as \(t \to \infty\).

For convergence rate, we rewrite the function:
\[
\frac{\mu}{\cosh (dt) - \frac{d}{\mu}} = \frac{\mu}{\coth (dt) - \frac{d}{\mu}} = \frac{\mu}{\coth (-d(T-t) + dT) - \frac{d}{\mu}} = \frac{\mu}{\left(\frac{d}{\mu} - \frac{d}{\mu}\right) \coth (d(T-t))} = \frac{\mu^2 - d^2}{d}.
\]
The Taylor series of \(\coth (x)\) is \(\frac{1}{x} - \frac{x}{3} - o(x)\). Given that \((T-t) \to 0\), the convergence rate is \(O\left(\frac{1}{(T-t)^2}\right)\).

The limit in case (3) is obvious when \(\delta > 0\). Now we look at its convergence rate.

Here \(T = \frac{1}{\delta}\). Rewrite the function as
\[
\frac{\delta}{1 - \delta t} = \frac{1}{\frac{1}{\delta} - t} = \frac{1}{T - t} = \frac{1}{(T - t)}.
\]
Thus, its convergence is $O\left(\frac{1}{T-t}\right)$.

From the above analysis, we can know the convergence rate is $O\left(\frac{1}{T-t}\right)$, which is an inverse proportional function, under any case.

By Equation (5.17) and Lemma 5.4.4, we know that some diagonal entry of $W$ converges to the infinity at a finite time $T$ under some appropriate conditions.

Specifically, using a basic fact that two commuting symmetric matrices can be simultaneously orthogonally diagonalized [94], we can get the following theorem where the conditions of the eigenvalues of $B, C$ for structural balance are characterized:

**Theorem 5.4.5.** Suppose $W(t)$ solves the Riccati equation $W' = W^2 + C$, $W(0) = B$ where $B, C$ are symmetric with $BC = CB$. Then eigenvalues of $W(t)$ converge to elements in $(-\infty, \infty]$ as $t \to T$; meanwhile there is sign stability, i.e., $\lim_{t \to T} w_{ij}(t) \in [-\infty, \infty]$ exists for all $i, j$. If $U$ is an orthogonal matrix such that

$$U^T CU = \text{diag}(a_1^2, \ldots, a_k^2, -d_1^2, \ldots, -d_l^2, 0, \ldots, 0)$$

$$U^T BU = \text{diag}(\lambda_1, \ldots, \lambda_k, \mu_1, \ldots, \mu_l, \delta_1, \ldots, \delta_h).$$

where $a_i, d_j > 0$, then $W(t)$ is given by the following explicit function,

$$W(t) = U \cdot \text{diag} \left( \frac{a_1 \sin (a_1 t)}{\cos (a_1 t) - \frac{1}{a_1} \sin (a_1 t) \lambda_1}, \ldots, \frac{a_k \sin (a_k t)}{\cos (a_k t) - \frac{1}{a_k} \sin (a_k t) \lambda_k}, \right.$$

$$\left. - \frac{d_1 \sinh (d_1 t)}{\cosh (d_1 t) - \frac{1}{d_1} \sinh (d_1 t) \mu_1}, \ldots, - \frac{d_l \sinh (d_l t)}{\cosh (d_l t) - \frac{1}{d_l} \sinh (d_l t) \mu_l}, \right) \cdot UT. \tag{5.17}$$

Further, structural balance

$$\lim_{t \to T} w_{ij}w_{jk}w_{ki} \geq 0$$

occurs in the finite time for $W(t)$ if $W(t)$ has an unique largest eigenvalue and one of the following conditions holds:

1. There exists some $\delta_i > 0$,

2. There exists some $a_i > 0$,

3. There exists some $\mu_i > d_i$. 

Proof. Since the eigenvalues of $W(t)$ remains the same as $U^T W U$ for any orthogonal matrix $U$, we see the convergence of eigenvalues of $W(t)$ from Equation (5.17). Due to the convergence of the eigenvalues in $(-\infty, \infty]$, we see that $\lim_{t \to T} w_{ij}(t) \in [-\infty, \infty]$ exists for all $i, j$. This means that we have sign stability, that all weights $w_{ij}, \forall i, j$, have fixed signs, as $t \to T$.

To see structure balance, by Theorem 5.4.3, it suffices to check if the largest eigenvalue tends to infinity. We examine the solution $W$ as described in Equation (5.17) and use Lemma 5.4.4.

1. If there exists $\delta_i > 0$, structural balance occurs because $\lim_{t \to 1} \frac{\delta_i}{1 - \delta_i} = +\infty$.
2. If there exists $a_i > 0$, structural balance occurs because of Lemma 5.4.4(1).
3. If $\mu_j > d_j$, structural balance occurs because of Lemma 5.4.4 (2).

In our co-evolution model, $BC = CB$ happens when $V(0)$ is a eigenvector of $W(0)$, an interesting initial condition. To see that, recall $B = W(0)$, $C = V(0)V(0)^T - W(0)W(0)^T$, and $W(0) = W(0)^T$. To check if $BC = CB$, we just need to check if $W(0)V(0)V(0)^T = V(0)V(0)^T W(0)$ and apply the following Lemma.

**Lemma 5.4.5.** Suppose $A$ is a symmetric matrix and $v$ is a non-zero column vector. Then $Avv^T = vv^T A$ is equivalent to $Av = \alpha v$, i.e., $v$ is an eigenvector of $A$.

**Proof.** Clearly if $Av = \alpha v$, then $A$ and $vv^T$ commute. Conversely, if $A$ and $vv^T$ commute, we can find an orthogonal matrix $U$ such that $U^T A U$ and $U^T vv^T U$ are diagonal. We may assume that the $(1, 1)$ entry $\lambda$ of $U^T vv^T U$ is not zero. This shows the first column $c$ of $U$ is an eigenvector for $vv^T$ associated to $\lambda$. But $v$ is also an eigenvector of $vv^T$ associate to $\lambda$. Therefore $c$ is a non-zero scalar multiplication of $v$. But we also know that $c$ is an eigenvector of $A$. Therefore, $v$ is an eigenvector of $A$. 

Further, the equation in our co-evolution model, i.e., Equation (5.5), satisfies $C + B^2 = V(0)V(0)^T$. Notice that the right-hand side $V(0)V(0)^T$ is an $n \times n$ matrix with rank one. This property actually ensures that the conditions characterized in
Theorem 5.4.5 are met and thus structural balance is guaranteed. At the same time, the convergence rate is $O(\frac{1}{|T-t|})$.

**Corollary 5.4.4.** For Equation (5.4), if $V(0) \neq 0$ is an eigenvector of $W(0)$, then $V(t)$ remains to be an eigenvector of $W(t)$ for all $t$ and structural balance must occur in finite time for $W(t)$.

**Proof.** When $V(0)$ is not a zero vector, $V(0)V(0)^T$ is a symmetric matrix with one positive eigenvalue and $(n-1)$ zero eigenvalue. Furthermore, $V(0)$ is an eigenvector associated to the largest positive eigenvalue. Now, since $W(0)$ and $C$ commute, we may simultaneously orthogonally diagonalize both. Since $B^2 + C = V(0)V(0)^T$ is diagonal with only one positive diagonal entry, using the same notation as above, we see that all numbers $a^2_i + \lambda^2_i, ..., a^2_k + \lambda^2_k, \mu^2_i - d^2_i, ..., \mu^2_i - d^2_k, \delta^2_1, ..., \delta^2_n$ are zero except one of them which is positive. If $a^2_i + \lambda^2_i > 0$, then $a_i > 0$ exists and the condition (2) holds. If $\mu^2_i - d^2_i > 0$, then $\mu_i > d_i$ exists and the condition (3) holds. If $\delta^2_i + 0 > 0$, then $\delta_i > 0$ and the condition (1) holds. Thus, by Theorem 5.4.5, the structural balance must occur in the finite time for $W(t)$. \hfill \qed

If $V(0) = 0$, the system stays at the fixed point with $V$ remaining zero and the weight matrix unchanged.

The case when $V(0)$ is an eigenvector of $W(0)$ includes a few interesting cases in practice. When $W(0) = 0$ or $W(0) = I$, this models a group of individuals that start as complete strangers with uniform self-appraisals. Their non-homogeneous initial opinions $V(0)$ may drive the network to be segmented over time.

We conjecture that even in the general case (when $V(0)$ is not necessarily an eigenvector of $W(0)$) the limit vector $\lim_{t \to T} V(t)/|V(t)|$ is an eigenvector of the limit tie relation matrix $\lim_{t \to T} W(t)/|V(t)|$. This is supported by our numerical evidences which says that $\lim_{t \to T} V(t)/|V(t)|$ is an eigenvector of $\lim_{t \to T}(W(t)/|V(t)|)^2$. 


5.5 Simulation

We use the discrete time model as described in Equation (5.3). The convergence of the dynamic process defined by Equation (5.3) is very fast. For visualization purposes, in the simulation, we set our evolution model using Equation (3) with $a = b = 0.01$. When a graph is said to satisfy structural balance, the multiplication of weights along all triangles is non-negative. In addition, either all nodes have the same opinion (i.e., harmony) or the graph is partitioned into two antagonistic groups (i.e., polarization).

5.5.1 Harmony vs Polarization

In this section, we work with a complete graph and examine when structural balance and/or polarization appears.

\textbf{B and C commute.} As mentioned in Theorem 5.4.5, when the matrices $B = W(0)$ and $C = V(0)V(0)^T - W(0)^2$ are symmetric with $BC = CB$, structural balance occurs in finite time for $W(t)$ if $W(t)$ has an unique largest eigenvalue. We take three special cases mentioned in the previous section, namely when $W(0) = I$, $W(0) = 0$, or $W(0) = V(0)V(0)^T$. The evolution process of these three cases are shown in Figure 5.2. The initial opinions are selected uniformly at random in $[-1,1]$. The dash curves represent the evolution of opinions and the solid ones shows the edge weights. The color of the curves represent the final sign of opinions and weights at convergence. In all three cases the network reaches polarization where the opinions of some vertices go to positive infinity and the opinion of the others go to negative infinity. In these cases, the node opinions do not change signs, because $v_i w_{ij} v_j \geq 0$ holds all the time. The gradient direction of opinions and weights are same with their signs. Figure 5.2(d) shows a case when both opinions and edge weights converge to positive values. All the opinions are assigned initially as a positive value. The initial weight matrix is a diagonal matrix with the same diagonal entry. All final opinions and weights are positive.

\textbf{B and C do not commute.} Next, we consider the case when the matrices $B$ and $C$ do not commute. We take different random initial cases and show the evolution process in Figure 5.3. In the first two plots 5.3(a) and 5.3(b), we select a symmetric random
Fig 5.2: The evolution process for a complete graph with $BC = CB$. In the first three plots, the network at the limit achieves structural balance and the network is also polarized. In the last example, the network reaches harmony.

matrix as the initial weight matrix $W(0)$. The opinions are assigned random initial values. In simulation we observed both cases of harmony and polarization as the final state. Some edges and vertices change signs in the process. We also show an example of the evolution process in Figure 5.5 which shows how two communities emerge.

$W(0)$ is not symmetric. In Figure 5.3(c), the initial weight matrix is not symmetric. Both vertex opinions and edge weights go to infinity after several iterations. Figure 5.3(d) shows when all the entries in the matrix and opinions are initially negative. Some opinions and weights change signs in the evolution process. For all cases we have tested when $B$ and $C$ do not commute, structural balance is always satisfied.
Fig 5.3: The evolution process for a complete graph with $BC \neq CB$.

5.5.2 Convergence Rate

In section, we check the convergence rate in different settings. This helps us understand intuitively the factors that influence the convergence rate.

Magnitude of Initial Opinions. Figure 5.4(a) and 5.4(b) show two cases $W(0) = I$ and $W(0) = 0$ respectively. In both cases $BC = CB$. The initial opinions are randomly selected from $(-1, 1)$. We check the number of iterations until all entries in the opinion vector and weight matrix have absolute value larger than $10^{20}$. From the analysis in the previous section, the largest eigenvalue of $W(t)$ determines the convergence rate. Given that the initial weights are determined, we check the positive eigenvalue of $V(0)V(0)^T$. 
We can see that the number of iteration until convergence is inversely proportional to the magnitude of the positive eigenvalue of $V(0)V(0)^T$. The more extreme the initial opinions are, the faster the network reaches convergence. We also tested networks of different sizes, which seems to be generally oblivious to the convergence rate.

**W(0) as a Random Matrix.** In Figure 5.4(c), we compare the case of $W(0) = V(0)V(0)^T$ with $W(0)$ being a random symmetric matrix ($W(0)$ and $C$ are generally not commutative). The orange dots show the number of iterations till convergence when entries in $W(0)$ are selected uniformly in $(-1, 1)$. The green dots show the number of iterations until convergence when entries in $W(0)$ are selected uniformly in $(-5, 5)$.

There are a few observations. 1) When $W(0)$ take random values, it requires more iterations to convergence compared to the case when $W(0) = V(0)V(0)^T$. Specifically, during the evolution when $W(0) = V(0)V(0)^T$ both opinions and weights do not change signs. 2) When the initial weights take greater absolute values in general, the system converges faster. Again the more extreme the opinions/weights are, the faster the system reaches structural balance.

**On a General Graph.** We also test on graphs generated by social network models. In Figure 5.4(d), we show the convergence results on graphs generated by the Erdős-Renyi Model $G(n, p)$, where $n$ is the number of nodes and $p$ is the probability of each pair of nodes connected by an edge. As $p$ increases from 0 to 1, the graph becomes denser and it requires fewer iterations to converge.

In Figure 5.4(e) and 5.4(f), we tested on Watts Strogatz model [197] $G(n, k, p)$, where $n$ is the number of nodes, $k$ is the number of neighbors of each nodes and $p$ is the rewiring probability. The network starts as a regular ring lattice, where each node connects to $k$ nearest neighbors on the ring. Each neighbor has a probability $p$ to be ‘rewired’ to another non-neighbor node. Thus, with a larger value of $k$, there are more edges in the graph. With a larger rewiring probability $p$, there is more randomness in the graph. In these simulations, the initial weight matrices are set as a zero matrix, i.e., $W(0) = 0$. Similarly, with the same number of edges in the graph, the relationship between the positive eigenvalue and the number of iterations follows the similar trend. When there are more edges in the graph, the convergence speed increases. In all cases,
5.5.3 Emergence of Community Structure

In this section, we ran experiments on real-world network data sets. We are particularly interested in the following question. Can a few planted seeds with opposite opinions influence the other nodes and drive the network into structural balance and polarization? Does the final state coincide with the community structure in the network?

Our first experiment is based on a study by Zachary [210] who witnessed the breakup of a karate club into two small clubs. As shown in Figure 5.6(a), the networks captures 34 members, documenting links between pairs of members who interacted outside the club. During the study, a conflict arose between the administrator (label 0) and the instructor (label 33), which led to the split. The red and green nodes represent the choice of each individual in the end. In this experiment, we assign the administrator and the instructor opposite opinions as 1 and −1. The other members start with opinion 0. Given that the links represent interaction and positive friendship between members, each edge is assigned a small positive value in the initial matrix $W(0)$. We
then run our co-evolution dynamics till convergence. The edges with negative weight are removed. The graph is separated into two communities, shown in Figure 5.6(b). It nearly predicted the same division as in the ground truth except for two members (#8 and #19) which are somewhat ambiguous.

The second experiment is based on the political blogs network. It is a directed network of hyperlinks between weblogs on US politics, recorded in 2005 by Adamic and Glance [9]. There are 1,490 nodes and 19,025 directed edges in the graph. Each node has its political preference (−1 as liberal, 1 as conservative) shown in Figure 5.6(c). We randomly select 20% nodes and assign initial opinions according to their ground truth values. All edges are assigned initial weights as a small positive value. When the graph reaches convergence, two big communities appear, as indicated by their final opinions and the sign of edges. Figure 5.6(d) shows the detected communities after negative edges are removed. Compared with the ground truth, the predicted opinions by our dynamical model has an accuracy of 97.21%, averaged by 200 simulation runs. If only 3% nodes are assigned ground truth opinions in the initial state, the prediction accuracy for the final opinions of all nodes, on average, is as high as 82.12%.

Our dynamic model, as shown by these experiments, explains why community structures appear. It can also be understood as an algorithm for label propagation or node classification. Compared with other methods for the same task [199, 110, 180, 31] that generally use data-driven machine learning approaches, our dynamic model has better transparency and interpretability.

5.6 Conclusion and Future Work

In this paper, we have provided a co-evolution model for both opinion dynamics and appraisal dynamics. We provided solutions to the system and rigorously characterized how the stable states depend on the initial parameters.

There are a few follow-up problems that remain open, for example, when the social ties are directional/asymmetric, when the network is not a complete graph, and when each agent has an \(m\)-dimensional opinion vector. We include some discussion and
conjectures on these cases and consider this as interesting future work.

**Fig 5.5:** The weight matrix in evolution.

(a) Initial Karate club graph  
(b) Separated communities results

(c) Initial political blogs network  
(d) Separated communities results

**Fig 5.6:** Community detection in two real-world data sets.
Chapter 6
Efficient Beacon Placement Algorithms for Time-of-Flight Indoor Localization

6.1 Introduction

Several technologies have emerged for indoor localization in the past decade that are able to provide sub-meter ranging using Time-Of-Flight (TOF) or Time-Difference-Of-Arrival (TDOA) ranging. The underlying ranging signals could be based on acoustic, ultrasonic or RF technologies, such as Ultra-Wideband, Bluetooth Low Energy 5 and WiFi 802.11mc. ToF beacon-based systems provide a distance measurement, which is used to estimate a device’s location via trilateration. While ToF-based localization is well understood, the problem of where to place the beacons is not. Current methods used to deploy beacons either require domain experts who leverage intuition and heuristics, or let the system installers over-provision indoor spaces with more beacons than required, in order to ensure full localization coverage. As a consequence deployments can waste resources. Minimizing the number of beacons is particularly important as these systems transition from small deployments, mainly used for demonstration purposes, to commercial ones across large real-world spaces such as airports, museums, malls, and industrial buildings. In these environments an efficient and systematic beacon placement methodology will have a significant impact in terms of cost savings, thus facilitating adoption. Thus in this paper we aim to examine the beacon placement problem systematically, with a focus on both practicality in real world setting and provable guarantees about coverage and beacon count with respect to optimal.

In our formulation of the localization problem, the beacons emit signals that could be picked up by a mobile device within line-of-sight of the beacons. The device at a location $p$ within line of sight of a beacon $b$ can derive a (fairly accurate) distance
estimate from \( p \) to \( b \). If a location \( p \) has line of sight paths to three or more beacons, by trilateration one can uniquely localize \( p \). If a point is visible to only two beacons \( b_1, b_2 \), the two range measurements provide two candidate locations \( p_1 \) and \( p_2 \), which are mirrors of each other with respect to the line joining the two beacons. With the information from the floor plan we can potentially discover the true location among \( p_1 \) and \( p_2 \) if the set of beacons visible to one of them differs from \( \{b_1, b_2\} \). Figure 6.1(d)-6.1(f) illustrates this subtlety. This property that takes into account the geometry of the floor plan makes the beacon placement problem especially interesting.

![Diagram of localization with ToF in a floor plan](image)

**Fig 6.1:** Localization with ToF in a floor plan: (a) An infinite number of solutions with a single beacon; (b) Two solutions with two beacons; (c) A unique solution with three beacons; (d-f) A unique solution with two beacons: (d) \( p_2 \) is outside the floor plan and thus is not feasible; (e) \( p_2 \) is not visible to \( b_2 \) and thus is not feasible; (f) \( p_2 \) is visible to a third beacon \( b_3 \) and thus is not the correct solution.

We present two algorithms with provable guarantees for the problem of beacon selection for unique localization. We also evaluate their performance on multiple floor plans, compared with prior work and user studies. First, we formally define the conditions
for a location to be uniquely localizable without ambiguity and propose efficient algorithms to test whether a given set of beacons provide unique localization for all points in the domain. For this, we employ detailed analysis of the geometric constraints and visibility. On this foundation, we design the beacon placement algorithms.

The first algorithm we propose is a greedy algorithm in which we select the next beacon by optimizing a certain objective function. A greedy algorithm is also favored in practice for its incremental nature. The intuitive approach is to maximize the area that is uniquely localizable (as done in prior work [161]). However, there are a number of problems with this objective function, for example, it is possible that no extra beacon can increase the area that is uniquely localized. There are scenarios where this algorithm can lead to a solution far from the optimal. We overcome this challenge by designing a new objective function that has the submodular property (intuitively, a function with monotonicity and diminishing return) and by optimizing this submodular function, we get provable performance guarantee that the number of beacons selected has an approximation ratio $O(\ln m)$ of the optimal, where $m$ is number of target points to be localized. The key is to consider not only the area to be uniquely localizable but also areas that are covered by at least one beacon and at least two beacons.

We propose a second algorithm that uses random sampling for beacon selection. It uses the concept of $\varepsilon$-net and geometric set cover. For the set cover problem, if the sets have constant VC-dimension (which is a measure of the complexity of the sets), then one can approximate the optimal solution up to $O(\log)$, where $\varepsilon$ is the size of the optimal solution [77]. Again, we cannot directly apply this technique to our problem of unique localization, as checking whether a location is uniquely localizable by two beacons depends on the set of beacons covered by the mirror image of this location. We need to carefully work around the issue and argue that even for our localization problem, the random sampling based algorithm works. In the algorithm, we introduce weights to the beacons. Initially all beacons have the same weights. Now define the weight of a point $p$ as the sum of the weights of the beacons that are visible to $p$. We develop a new proof using the VC-dimension property to show that a randomly chosen set of beacons can uniquely localize all points of high weight. Now considering
the points that are not yet covered (which have low weight), we double the weight of
the beacons they see. Therefore, in the next round of random sampling these beacons
have a higher chance of being selected. It has been shown in [77] that after a number
of iterations, this algorithm stops with all the points uniquely localized, as long as the
number of beacons selected is at least $O(\log)$.

We implement both algorithms in a MATLAB-based toolchain for real-world and
random floor plans and compare with prior work. We also show how our placement
compares to placements performed by users with varying familiarity with indoor lo-
calization and beacon placement geometrical problems. In both cases our algorithms
perform favorably in terms of both coverage and localization quality.

In summary, the main contributions of this paper are:

1. A mathematical formulation of the minimal beacon placement problem for indoor
localization with line-of-sight beacons;

2. An algorithm for checking if a point or a region (sub-domain) is uniquely localiz-
able given a placement;

3. Two approximation algorithms for beacon placement: a greedy algorithm of ap-
proximation factor of $O(\ln m)$ and a random sampling algorithm with approxi-
mation factor of $O(\log)$ guarantee where $m$ is the optimal number of beacons.

4. Implementation and evaluation of the algorithms on a variety of floor plans and
comparison with user selected beacon solutions.

6.2 Related Work

The Art Gallery problem and visibility. Mathematically, the problem of beacon
selection for unique localization is closely related to the classical Art Gallery problem in
computational geometry [150], where a minimum number of guards/beacons are selected
to ensure all points of the domain (region within an indoor geometrical floor plan) have
line of sight paths to (equivalently, are covered by) at least one guard/beacon. The Art
Gallery problem is NP-hard even for simple polygons [126]. For any simple polygon
$P$ with $n$ vertices, it has been proven that $\lfloor n/3 \rfloor$ guards are always sufficient [49] and sometimes necessary. For a polygon $P$ with $h$ holes, it was shown that $P$ can be guarded with $\lceil n+h/3 \rceil$ guards [95, 34]. Eidenbenz et al. proved the problem to be APX-hard [64], implying that it is unlikely that any approximation ratio better than some fixed constant. Ghosh [79] showed that a logarithmic approximation may be achieved by discretizing the input polygon into convex subregions. Valtr [183] showed, the set system derived from an art gallery problem has bounded VC dimension, allowing the algorithm based on $\varepsilon$-net to obtain an approximation ratio as the logarithm of the optimal number of guards [39].

Also related to our work is the $k$-coverage set problem. Here, the goal is to find the minimum set that covers all the points at least $k$ times. Obviously the $k$-coverage problem is NP-hard as well. Several approximation algorithms have been proposed for this class of problems. Cormen [51] applied a greedy approach for $k$-coverage problem with a $O(k \log n)$-approximation solution, where $n$ is the number of points. The $\varepsilon$-net technique was introduced for this problem when the sets have constant VC-dimensions [87, 39] and was shown to achieve approximation factor $O(\log)$ [89, 77], where is the optimal solution.

Our problem is not the Art Gallery problem as each point needs to see at least two or three beacons. It is not the $k$-coverage problem either. The points in the domain have to be uniquely localized, which can be achieved by two or three beacons, depending on geometry.

**Beacon placement for localization.** The optimal beacon placement for a single target location is well understood [33, 167]. Onur and Volkan defined the uncertainty of points and proved that the beacon placement problem, to make the uncertainty of target points below the threshold, is NP-complete [182]. Hence, several heuristic-based optimization algorithms have been proposed for general beacon placement [128, 164, 116]. The state-of-art in beacon placement from commercial beacon vendors [1, 35, 143] suggest guidelines to account for the height of beacons and areas where better accuracy is desired and full coverage. However, this is far from a systematic approach for beacon placement.
A class of prior work has studied optimal beacon placement based on optimizing the placement for certain localization accuracy criteria [174, 153]. Though the accuracy is hard to quantify in the general case, under the assumption that the ranging noise is additive Gaussian noise, independent of the range, one can compute the Cramér-Rao bound (CRB) [115] for a beacon-target geometry and aim to optimize it across the region to be localized.

These prior works rely on at least three beacons for each location. It was first proposed in [161] that two beacons for ToF-based indoor localization system may also work, by considering the floor plan. They present a greedy algorithm for beacon placement for optimizing coverage and accuracy which has no theoretical guarantee. We compare our work with this algorithm in our experiments.

6.3 Problem Definition

In this section, we first provide the background on localizing with line-of-sight (LOS) beacons, state our assumptions, formulate the problem and introduce notation and definitions.

**Localizing with LOS beacons:** We illustrate the localization problem in Figure 6.1. Typical localization approaches use three beacons. When the floor plan information is available, which constrains the beacon’s coverage, we can sometimes localize with just two beacons. This concept is key to our minimal beacon placement problem.

**Assumptions:**

*2D deployment:* We assume that the 2D representation of the floor plan is available, we also perform the beacon placement in 2D. In reality, we deploy the beacons in 3D by varying the heights at which the beacons are deployed. The 2D beacon placement is effective for 3D deployment if we deploy the beacons close to ceiling level and the user holds the device at regular height (around 1m from ground). Most temporary obstructions in the environment such as chairs, tables, etc would not change the beacon coverage. However, our 2D model assumption would not hold for 3D if the beacons are deployed at floor level and blocked by objects or if the beacons are deployed at ceiling
and obstructions such as cubicle partitions are much taller than the user and block the
device held by the user from the beacon. For practical purposes, the beacons can be
deployed at ceiling level in most public spaces such as airports, museums, malls, and
the floor plan 2D coverage and assumption will be applicable.

Ray-tracing coverage: While deploying beacons, we assume pure line-of-sight (LOS)
coverage. However, while the system is in use, we receive non-line-of-sight (NLOS)
signals and cope with it while solving for location. We compute the LOS coverage
based on the beacon range and the ray-tracing coverage area. The ray tracing coverage
is as follows: If a point is visible from a beacon (in line-of-sight), we receive an exact
measurement from the beacon, otherwise, we do not receive a measurement. When
NLOS measurements are received under this deployment (due to reflecting off walls or
signals penetrating through walls), we adopt two approaches to cope with NLOS while
estimating location. First, we apply localization techniques that localize in the presence
of LOS and NLOS measurements with [162] or without the floor plan [195, 131]. The
second approach is to detect NLOS signals based on the signal strength or statistical
properties of the signal [172, 84, 155]. In this way, as we cannot predict the NLOS
signals, we design the beacon placement for LOS coverage and cope with NLOS signals
while estimating location.

Definitions: Mathematically, we formulate the optimal beacon placement problem as
follows. The floor plan is represented by $P$. Any permanent walls and obstructions
inside are modeled as holes in the polygon. Thus, $P$ refers to the region inside the
exterior polygon except the holes. Further, $B$ is a set of candidate beacon locations,$B = \{b_i|1 \leq i \leq n, b_i \in P\}$, that guarantees unique localization. The problem is to find
a minimum set of beacons $D \subseteq B$ of size $k$ such that the entire polygon $P$ or all the
target points of size $m$ can be uniquely localized by beacons located at $D$.

Definition 6.3.1. (Visibility): Two points $p, q \in P$ are visible to each other if and
only if the line segment $pq$ is strictly inside $P$, i.e., does not intersect any point on the
boundary of $P$. We can also say $p$ sees $q$ or vice versa.
Definition 6.3.2. *(Visible Region):* For a beacon $b_i$, the set of points seen by $b_i$, is the visible region of $b_i$, denoted as $V(b_i)$.

Definition 6.3.3. *(Visible Beacon Set):* For a point $p \in P$, the visible beacon set of $p$, $V(p)$, is the set of candidate beacon locations that can see $p$. To represent the subset of beacons in the set $D$ that are visible at $p$, we use $V_D(p) = \{b_i \in D| b_i \text{ can see } p\}$.

Definition 6.3.4. *(Unique Localization of a Point $p$):* Given a point $p$ and a set of visible beacons $V_D(p)$, we say $p$ can be uniquely localized, i.e., $UL_D(p) = 1$, if there is only one location consistent with the range measurements and the visibility information.

If all the points $p \in P$ can be uniquely localized, then we say $P$ can be uniquely localized. We denote by $UL_D(P) = 1$.

The rest of the paper is organized as follows. In sec:algo-unique-loc, we describe in detail the algorithm to verify unique localization of a point and a region. The two algorithms are presented in sec:greedy-algo and sec:enet-algo respectively. Finally in sec:eval, we implement the algorithms and present the placement results.

6.4 Algorithm to Verify Unique Localization

In this section, we describe algorithms for verifying whether a point $p \in P$ or the domain $P$ is uniquely localized, given a set of beacons $D$. We use these subroutines in the beacon placement algorithms.

6.4.1 Unique Localization of a Point

With beacons at $D$, a point $p \in P$ can receive range measurements from beacons in $V_D(p)$. We define $|V_D(p)|$ as the number of beacons visible to $p$. If $p$ can see at most one beacon, i.e. $|V_D(p)| \leq 1$, $p$ cannot be uniquely localized. If $p$ can see three or more beacons, $|V_D(p)| \geq 3$, then $p$ is uniquely localized. The case of $p$ seeing exactly two beacons, say $b_i$ and $b_j$, needs more discussion, as described next. In this case, the range measurements from $b_i$ and $b_j$ generate two circles that intersect (in general) at
two points \( p_1 \) and \( p_2 \). We call these two points *mirrors* of each other; symmetric about the line joining \( b_i, b_j \). Figure 6.1(b) shows the ambiguity produced with two beacons. Both locations \( p_1 \) and \( p_2 \) are consistent with the range measurements and the visibility information. So we do not know which one is the true location. In Figure 6.1(d), \( p_2 \) is outside the domain. In Figure 6.1(e), \( p_2 \) cannot see \( b_2 \). In Figure 6.1(f), \( p_2 \) can see \( b_3 \). In these cases, we can conclude that \( p_1 \) is the unique location. Also, if the range information from two beacons leads to only one intersection (when the two circles are tangential), there is a unique solution.

**Fact 1.** If a location \( p \) is only visible to two beacons \( b_i \) and \( b_j \) among \( D \), the location of \( p \) is uniquely determined by the range information from \( b_i, b_j \) if and only if 1) the range information produces exactly one intersection \( p_1 \); or, 2) the range information produces a pair of mirror points \( p_1, p_2 \) but \( V_D(p_1) \neq V_D(p_2) \).

The test for unique localization of points is summarized below. in Equation 6.1. For each point \( p \in P \), we can enumerate \( k \) beacons and check whether \( p \) can see these beacons. If \( |V_D(p)| = 2 \), we check its mirror point, otherwise, we can get the unique localization status of \( p \) through the cardinality of \( V_D(p) \).

\[
UL_D(p_1) = \begin{cases} 
1, & |V_D(p_1)| \geq 3 \\
1, & |V_D(p_1)| = 2, p_1 = p_2 \\
1, & |V_D(p_1)| = 2, V_D(p_1) \neq V_D(p_2) \\
0, & |V_D(p_1)| = 2, V_D(p_1) = V_D(p_2), p_1 \neq p_2 \\
0, & |V_D(p_1)| \leq 1 
\end{cases} \tag{6.1}
\]

### 6.4.2 Unique Localization of a Region

To check if a domain \( P \) is uniquely localizable by beacons in \( D \), we compute the visible regions of the beacons in \( D \).

**Definition 6.4.1. (Canonical Region)**: Given a domain \( P \) and a set of beacons \( D \), \( P \) can be partitioned into disjoint regions such that all the points in the same region see exactly the same set of beacons. These regions are called the canonical regions, denoted
by $Q_1, Q_2, \ldots, Q_c$.

$$Q = Q_1 \cup Q_2 \cup \cdots \cup Q_c \quad (6.2)$$

$$\forall Q_i, Q_j, i \neq j, Q_i \cap Q_j = \emptyset \quad (6.3)$$

$$Q_i = \{q \in Q| V_D(q) = V_D(p), p \in Q\} \quad (6.4)$$

To test the unique localization of a canonical region $Q_i$, we check the cardinality of $V_D(Q_i)$. When $|V_D(Q_i)| = 2$, we need to further consider the ambiguity due to localization with two beacons. Hence, we define the mirror region of $Q_i$.

**Definition 6.4.2. (Mirror Region):** For a canonical region $Q_i$ with $V_D(Q_i) = \{b_i, b_j\}$, the mirror region of $Q_i$ is the region that is symmetric to $Q_i$ with respect to the line through $b_i, b_j$.

Similarly, if the canonical region $Q_i$ has different visible beacon set from its mirror region’s set, it can be uniquely localized.

Thus, a domain $P$ is uniquely localized when all the canonical regions are uniquely localized.

### 6.5 A Greedy Algorithm

![Fig 6.2: An example for which prior work [161] can be suboptimal.](image)

To find a small number of beacons that uniquely localize the entire domain $P$ or all the target points, an intuitive approach is to use a greedy approach based on some optimization criteria. For example, in [161], the beacon that maximizes the extra area that is uniquely localized is chosen. This algorithm, unfortunately, may place a lot more beacons than the minimum number needed, as in Figure 6.2.
We design a new greedy algorithm by showing a variant that optimizes a different objective function is monotone and submodular. A function on a subset of $\Omega$ is monotone and submodular if

- for every $X \subseteq Y \subseteq \Omega$, $f(X) \leq f(Y)$.
- for every $X \subseteq Y \subseteq \Omega$, and $x \in \Omega \setminus Y$, we have
  \[ f(X \cup \{x\}) - f(X) \geq f(Y \cup \{x\}) - f(Y). \]

We refer to this algorithm as the \textit{Submodular Algorithm}. The submodular property is important because a greedy algorithm that greedily optimizes a submodular function can be shown to have a logarithmic approximation ratio [189]. The area of uniquely localizable region does not have the submodular property. For example, when you add the first beacon, the area of the unique localization region does not increase. When the second beacon is placed, this area is likely to be increased, contrary to the property of the submodular function.

### 6.5.1 Algorithm Design

For simplicity of explanation, we describe this algorithm in a discrete problem setting, uniquely localizing all the $m$ target points.

**Definition 6.5.1. (Visibility Level Region:)** Given a domain $P$ and a beacon location set $D$, $V_1(D)$ ($V_2(D)$) is the set of target points that see at least one (two) beacon in $D$. $U(D)$ is the set of target points that are uniquely localized by $D$.

- $V_1(D) = \{p \in P|\exists b \in D, p \in V(b)\}$
- $V_2(D) = \{p \in P|\exists b_1, b_2 \in D, p \in V(b_1) \text{ and } p \in V(b_2)\}$
- $U(D) = \{p \in P|UL_D(p) = 1\}$

From the definition above, it is clear that $U(D) \subseteq V_2(D) \subseteq V_1(D) \subseteq P$. We define a utility function $F(D)$ below.

\begin{equation}
F(D) = 3|V_1(D)| + 2|V_2(D)| + |U(D)|
\end{equation}
When all the target points are uniquely localized by \( D \), we have \( V_1(D) = V_2(D) = |U(D)| = m \). Thus, the beacon locations need to be selected until \( F(D) = 6m \). Our Submodular algorithm selects the next beacon location that maximizes \( F(D) \) instead of \( U(D) \).

### 6.5.2 Approximation Bounds

We prove that the utility function \( F(D) \) is submodular and our algorithm can achieve an approximation factor of \( O(\ln m) \).

**Theorem 6.5.1.** The utility function \( F(D) = 3|V_1(D)| + 2|V_2(D)| + -U(D) - \) is monotone and submodular.

**Proof.** Any extra beacon added to \( D \) can only help to increase \( |V_1(D)|, |V_2(D)| \) and \( |U(D)| \). Thus For any set \( D \subseteq A \subseteq B \), we have \( F(D) \leq F(A) \). So \( F(D) \) is monotonically increasing.

To prove \( F(D) \) is a submodular function, it suffices to show that \( F(A \cup \{b\}) - F(A) \leq F(D \cup \{b\}) - F(D) \), where \( D \subseteq A \subseteq B \) and \( b \in B \setminus A \). We consider the components of the function \( F \) separately:

\[
F(A \cup \{b\}) - F(A) - (F(D \cup \{b\}) - F(D)) = 3F_1 + 2F_2 + F_3
\]

where

\[
F_1 = (|V_1(A \cup \{b\})| - |V_1(A)|) - (|V_1(D \cup \{b\})| - |V_1(D)|)
\]
\[
F_2 = (|V_2(A \cup \{b\})| - |V_2(A)|) - (|V_2(D \cup \{b\})| - |V_2(D)|)
\]
\[
F_3 = (|U(A \cup \{b\})| - |U(A)|) - (|U(D \cup \{b\})| - |U(D)|)
\]

In the following proof, we use set subtraction. When one set contains the other completely, the result of subtraction is the same as the subtraction of their value. Thus, we have the following equalities.

\[
F_1 = |V_1(b)| - |V_1(b) \cap V_1(A)| - |V_1(b)| + |V_1(b) \cap V_1(D)|
\]
\[
= -|V_1(b) \cap (V_1(A) \setminus V_1(D))|
\]
\[
F_2 = |V_1(b) \cap (V_1(A) \setminus V_2(A))| - |V_1(b) \cap (V_1(D) \setminus V_2(D))|
\]
\[
= |V_1(b) \cap (V_1(A) \setminus V_1(D))| - |V_1(b) \cap (V_2(A) \setminus V_2(D))|
\]
$F_3$ considers only target points that become uniquely localizable with $b$ added. $F_3$ has two parts $F_3 = F_4 + F_5$ where $F_4$ counts the change of the target points that are seen by three or more beacons and newly become uniquely localizable with the addition of $b$; and $F_5$ counts the change of the target points that can be seen by exactly two beacons and newly become uniquely localizable.

$F_4$ can be represented, in the same manner as $F_1$ and $F_2$:

$$F_4 = |V_1(b) \cap (V_2(A) \setminus U(A))| - |V_1(b) \cap (V_2(D) \setminus U(D))|$$

$$= |V_1(b) \cap (V_2(A) \setminus V_2(D))| - |V_1(b) \cap (U(A) \setminus U(D))|$$

The trickiest part is to analyze $F_5$. We consider target points that see two beacons and newly become uniquely localizable.

First, we consider the beacon set $A \setminus D$, the set of beacons in $A$ but not in $D$. If a point $p$ is seen by two beacons, one is $b$, and the other is from $A \setminus D$, $p$ can only possibly become uniquely localizable for beacon set $A$, upon the addition of $b$. $p$ cannot possibly be uniquely localizable for $D \cup \{b\}$.

Second, when a point $p$ is seen by $b$ and exactly one beacon from $D$, and its mirror point $p'$ is already uniquely localizable with $D$, $p$ now becomes uniquely localizable.

Third, when $b$ is added, a point $p$ can become uniquely localizable because its mirror point $p'$ just becomes uniquely localizable (seeing three beacons including $b$).

Thus, we add the three components together.

$$F_5 \leq |V_1(b) \cap (V_1(A) \setminus V_1(D))| + |V_1(b) \cap (U(A) \setminus U(D))|$$

$$+|V_1(b) \cap (V_2(A) \setminus V_2(D))|$$

Now, we can bound these components and obtain the results:

$$F(A \cap \{b\}) - f(A) - (F(D \cup \{b\}) - F(D))$$

$$= 3F_1 + 2F_2 + F_4 + F_5$$

$$\leq -|V_1(b) \cap (V_1(A) \setminus V_1(D))| - |V_1(b) \cap (V_2(A) \setminus V_2(D))|$$

$$-|V_1(b) \cap (U(A) \setminus U(D)) + |V_1(b) \cap (V_1(A) \setminus V_1(D))|$$

$$+|V_1(b) \cap (V_2(A) \setminus V_2(D))| + |V_1(b) \cap (U(A) \setminus U(D))|$$

$$= 0$$

(6.6)

Therefore, $F(D)$ is a submodular function. \qed
Owing to its submodular properties, the greedy algorithm has an approximation factor following the standard argument [189].

**Theorem 6.5.2.** In discrete problem setting, the approximation ratio of the submodular algorithm is $O(\ln m)$. In the continuous problem setting, the target points are replaced by the domain $P$, the approximation ratio is $O(\min\{\ln \frac{|P|}{\Delta}, n\})$, where $|P|$ is the area of the domain, $\Delta$ is the minimum increased area for the objective function $F$ with a new guard and $n$ is the number of candidate beacons.

### 6.6 Random Sampling Algorithm

In this section, we present a different approximation algorithm using random sampling technique with slightly improved approximation factor. This algorithm is motivated by the $\varepsilon$-net based algorithm for geometric set cover which gives an approximation factor of $\log$ where $|P|$ is the size of the optimal solution [77]. But due to the specific requirement of the unique localization problem, our problem is different and we need to carefully get around the technical difficulties.

#### 6.6.1 Algorithm Design

In the algorithm we give weights to each candidate beacon location $b_i$ and each point $p \in P$.

**Definition 6.6.1. (Weight):** Define the weight of a beacon $b_i$ by $w(b_i)$. For a set of beacons $D$, its weight is $w(D) = \sum_{b \in D} w(b)$. The weight of a point $p \in P$ is defined by the weight of the beacons that can see $p$, $w(p) = w(V(p))$.

**Definition 6.6.2. ($\varepsilon$-oracle):** Given a domain $P$, a candidate beacon location set $B$ and the weight function $w$, a subset $D \subseteq B$ is an $\varepsilon$-oracle for $(P, B, w)$ if for any $p \in P$ with $w(p) \geq \varepsilon w(B)$, $p$ is uniquely localized by $D$.

In our algorithm (Alg 2), initially all beacon locations carry the same weight of 1. We randomly select $k$ beacons, in which a beacon is selected with probability proportional to its weight. This random sample has a good probability to be a $\varepsilon$-oracle, by the proof
Fig 6.3: Illustration of Algorithm 2: Each beacon is set the weight and the red beacons are selected beacon locations. The purple region contains all the points with weight greater than $\varepsilon w(B)$. 
in the next subsection. If the beacons provide unique localization for all points of \( P \),
the algorithm terminates. Otherwise, we double the weights of the beacons seen by
the points that are not uniquely localized. We iterate this process and can show that
this terminates when \( k \) is \( O(\log) \), where \( \log \) is the size of the minimum number of beacons
supporting unique localization. Since we do not know what \( \log \) is, we start with \( k = 2 \).
When we execute several iterations and still cannot find a feasible solution, \( k \) is doubled
until a feasible solution is obtained.

**ALGORITHM 2:** Random Sampling Algorithm

**Input:** The floor plan \( P \), Candidate beacon locations set \( B \)

**Output:** A feasible beacon locations set \( D \) that can localize \( P \)

for \( k = 2; k \leq m; k* = 2 \) do

\( \epsilon = \frac{1}{k}; \)

Reset all the weights of beacons in \( B \) to 1;

for \( i = 0; i \leq \frac{2k}{\epsilon} \log_2(\frac{m}{k}); i + + \) do

TotalWeight \( \leftarrow \sum_{b \in B} w(b); \)

Prob\( (b) \leftarrow \frac{w(b)}{w(B)}; \)

Select a beacon set \( D \) of size \( k \) according to Prob\( (b); \)

if \( D \) is \( \epsilon \)-oracle then

if Domain \( P \) is uniquely localized by \( D \) then

\( \text{return } D \)

end

else

Select a point \( p \) is not uniquely localized;

Double the weight of all beacons in \( V(p); \)

end

end

end

Figure 6.3 illustrates each step of the \( \epsilon \)-oracle algorithm. There are 10 candidate
beacon locations with weights. We start with \( k = 2 \) and \( \varepsilon = \frac{1}{k} \). In Figure 6.3(a), all the weights are 1. The total weight is 10. According to Definition 6.6.2, the region in purple contains all the points with weight greater than \( \varepsilon w(B) = 5 \). We select two beacons shown in red. They uniquely localize the purple region but do not uniquely localize the whole region \( P \). Thus, we find a point \( p \) (shown as a green node) that is not uniquely localized. There are two candidate beacons that can see \( p \), and their weights are doubled, from 1 to 2. In Figure 6.3(b), as the weights are changed, the total weight is 12 now and we restart the random sampling. Two red beacons are selected, but the region bounded by the green lines is not uniquely localized. So these beacons are not \( \varepsilon \)-oracle. After a few trials of random sampling, we pause as we believe that two beacons are not enough.

Then, we double the size \( k \) to 4 in Figure 6.3(c). All the weights are reset to 1. Again the region in purple shows the points of high weight. Four beacons are selected, uniquely localizing the purple region. They are \( \varepsilon \)-Oracle, but a point \( p \in P \) can be found that is not uniquely localized. The weights of the beacons covering \( p \) are doubled. Finally, we find the beacon locations shown in Figure 6.3(d). These four beacons can uniquely localize the whole region \( P \). The solution is obtained and the algorithm terminates.

Notice that in the above algorithm \( k \) is always a power of 2. While this is only giving a factor of 2 in the approximation factor theoretically, we would like to optimize \( k \) in practice. Thus, we run binary search to find the smallest \( k \) such that randomly selected beacons provide unique localization. We set lower bound \( LB = 1 \) and higher bound \( HB = |B| \). In each iteration, we set \( k = (LB + HB)/2 \). If a solution is obtained, \( HB = k \), otherwise \( LB = k + 1 \). The iterations are executed until \( LB = HB \), when we find the best \( k \).

### 6.6.2 Algorithm Analysis

To analyze the Random Sampling algorithm, first we introduce the concept of VC-Dimension [87], used in our following analysis.

**Definition 6.6.3. (VC-Dimension):** Given a set system \((X, R)\), let \( A \) be a subset of \( X \). We say \( A \) is shattered by \( R \) if \( \forall Y \subseteq A, \exists R \in R \) such that \( R \cap A = Y \). The
VC-dimension of \((X, \mathcal{R})\) is the cardinality of the largest set that can be shattered by \(\mathcal{R}\).

**Theorem 6.6.1.** [87] For a set system \((X, \mathcal{C})\) with \(|X| = n\) and VC-dimension \(d\), 

\[|\mathcal{C}| \leq n^d.\]

Now, we consider the VC-dimension of our case. We regard each canonical region as an element. Beacon \(b\) with visible region \(V(b)\) is a set that contains all canonical regions in \(V(b)\). Therefore the beacons correspond to sets in a set system, which is exactly the set system of the Art Gallery problem. The VC dimension of the Art Gallery problem is at most 23 by [183].

Notice that our problem of beacon selection for unique localization is not a set cover problem. So we have to develop the bounds from scratch. First, we prove that an \(\varepsilon\)-oracle can be found by random sampling with high probability. Compared with the previous proof of \(\varepsilon\)-net [77], our analysis differs in the following aspects:

1. The definitions and description of some events in the proof are in geometrical ways instead of combinatorial ways.

2. When calculating the probability of some events, owing to the ambiguity of unique localization, some inequalities are used to simplify the proof.

3. Through rigorous analysis, a higher probability than the prior work [77] is obtained, which leads to better running time for the algorithm.

**Theorem 6.6.2.** Given \((P, B, w)\) and

\[k \geq \max \left( \frac{2}{\varepsilon \log_2 \frac{1}{2\delta}}, \frac{1}{\varepsilon \log_2 \frac{4d + 16}{\varepsilon}}, \frac{4d + 16}{\varepsilon} \right), \quad (6.7)\]

let \(D\) be \(k\) beacons picked randomly from \(B\) with probability proportional to their weights. \(D\) is an \(\varepsilon\)-oracle with probability at least \(1 - \delta\), where \(d\) is the VC-dimension of the set system, \(d \leq 23\).

**Proof.** After randomly picking \(D\), we pick another set \(T\) in the same way as \(D\). We
denote \( Z = D \cup T \). Now, we define two events

\[
E_1 = \{ \exists p \in P \text{ s.t. } p \text{ is not uniquely localized}, w(p) \geq \varepsilon w(B) \} \\
E_2 = \{ \exists p \in P \text{ s.t. } p \text{ is not uniquely localized}, w(p) \geq \varepsilon w(B) , |V(p) \cap Z| \geq \varepsilon k \}
\]

First, we prove that \( \Pr[E_1] \leq 2 \Pr[E_2] \), i.e. \( \Pr[E_2|E_1] \geq 1/2 \) according to the conditional probability. When event \( E_1 \) happens, there are some points \( p \) that are not uniquely localizable but \( V(p) \geq \varepsilon w(B) \). We denote \( Y = |V(p) \cap Z| \) and the expected value of \( Y \) is at least \( 2\varepsilon k \), i.e. \( E[Y] \geq 2\varepsilon k \), because the selection probability is proportional to their weights. We can also get that the variance of \( Y \) is at most \( 2\varepsilon k \), i.e. \( \Var[Y] \leq 2\varepsilon k \). Hence, according to Chebyshev’s inequality,

\[
\Pr[Y \leq \varepsilon k] \leq \Pr[|Y - E[Y]| \geq \varepsilon k] \leq \frac{\Var[Y]}{(\varepsilon k)^2} \leq \frac{2\varepsilon k}{(\varepsilon k)^2} \leq 1/2
\]

It could be verified that \( k \geq \frac{4d+16}{\varepsilon} \log_2 \frac{4d+16}{\varepsilon} > \frac{4}{\varepsilon} \) owing that \( k \) is given by Equation 6.7. Thus, we have

\[
\Pr[E_2|E_1] \geq \Pr[Y \geq \varepsilon k] \geq 1/2
\]

Now consider a point \( p \) and fix the set \( Z \), when \( p \) is not uniquely localized by \( D \), \( |V(p) \cap D| < 3 \). Thus, we define the event \( E_p \) as

\[
E_p = \{|V(p) \cap D| < 3, |V(p) \cap Z| \geq \varepsilon k \}
\]

We consider the value of \( \Pr[E_p] \) with a given \( Z \). Suppose \( |V(p) \cap Z| = l \), we have \( l \geq \varepsilon k \). There are at most 2 elements of \( l \) in \( D \). The selected set \( D \) is comprised of \( k - 2 \) elements from \( Z - V(p) \cap Z \) and the other 2 elements elsewhere. So

\[
\Pr[E_p] = \frac{\binom{2k-l}{k-2} \binom{k+2}{2}}{\binom{2k}{k}} \leq 4(k) \frac{(2k-\varepsilon k)}{4 \binom{2k}{k}} \leq (k)^4 2^{-2\varepsilon k+2}
\]

The first inequality can be proved by mathematical method. Based on \( \Pr[E_p] \), the value of \( \Pr[E_2] \) is bounded. For any two points \( p_1, p_2 \in P \) with \( w(p_1), w(p_2) \geq \varepsilon w(B) \) and \( V(p_1) \cap Z = V(p_2) \cap Z \), the events \( E_{p_1} \) and \( E_{p_2} \) are the same in \( E_2 \). Thus, the occurrence of \( E_p \) depends only one the intersection \( V(p) \cap Z \). Recall that our set has a constant VC-dimension, so we can get the following by Theorem 6.6.1:

\[
\Pr[E_2] \leq \bigcup_{p|V(p)\cap Z \text{ is unique}} \Pr[E_p] \leq (2k)^d (k)^4 2^{-\varepsilon k+2} = (2k)^{d+4} 2^{-\varepsilon k-2}
\]
Thus, we can get the probability of $E_1$.

$$\Pr[E_1] \leq 2 \Pr[E_2] \leq (2k)^{d+4} 2^{-\epsilon k - 1}$$  \hspace{1cm} (6.8)

We need to show $\Pr[E_1] \leq \delta$. We can rewrite Equation 6.8 as:

$$\epsilon k \geq \log \frac{1}{2\delta} + (d + 4) \log(2k)$$

Thus, for the value of $k$, when we have $\frac{1}{2} \epsilon k \geq \log \frac{1}{2\delta}$ and $\frac{1}{2} \epsilon k \geq (d + 4) \log(2k)$, the theorem holds. We can verify when $k$ satisfies Equation 6.7, both inequalities hold.

Hence, when $k$ was given by Equation 6.7, the selected set $D$ is an $\epsilon$-oracle with probability at least $1 - \delta$.

Based on the above theorem, we can use random sampling to get an $\epsilon$-oracle with probability $1 - \delta$. However, this $\epsilon$-oracle may not be the solution to our problem because for some points with low weights, they might not be uniquely localized. This will be fixed by the weight doubling process during the iterative procedure. The number of iterations can be bounded using exactly the same analysis as in [87]:

**Theorem 6.6.3.** Suppose $k$ is the size of an feasible solution, we set $\epsilon = \frac{1}{k}$. Then $\frac{2k}{\delta} \log_2 \frac{m}{k}$ iterations of Algorithm 2 are sufficient to find a feasible solution.

Based on the Theorem 6.6.2 and 6.6.3, we get that Algorithm 2 can get a satisfied solution. When we estimate the optimal solution is and set $\epsilon = \frac{1}{k}$, we will get a solution of size $O(\log)$.

### 6.6.3 Considering Localization accuracy

In reality, range measurements are noisy and the localization error depends on the range error and the geometry of the beacons. The contribution due to the beacon geometry on the localization error is captured by the Geometric Dilution of Precision (GDOP). A lower GDOP is associated with lower bounds on the variance of the location estimate and hence lower localization errors. The GDOP metric at a location covered by a set of beacons is defined by the angle subtended between the location and pairs of beacons. When two beacon locations are very close, for most of their commonly visible points,
the angles between two beacons is very small, which causes the GDOP to be large and hence increases the localization error. Thus, we introduce a heuristic to include the incentive to select beacon locations to be far away from each other.

We add this correction to the random sampling algorithm. In Line 7 of Algorithm 2, the beacon set $D$ are selected according to their weight. We can select these beacons one by one and adjust the other beacons’ weights, to avoid selecting beacons that are close to each other. When a beacon location $b$ is selected, from the beacon locations that can see $b$, we select some whose distances with $b$ are less than a predefined threshold. These beacons’ weights are reduced, e.g. taking off 30% of their weight. This way, the probability of selecting a nearby beacon location is decreased. After Line 7, the beacon set $D$ is obtained and their weights are restored to the original weights.

### 6.7 Evaluation by simulation

For evaluation purposes we implemented the proposed algorithms in a MATLAB-based toolchain. We considered different approaches to generate floor plans, select initial
location candidates and beacon placement algorithms.

(1) **Floor plan generation**: We implemented these types of floor plans:

- User drawn floor plan through our GUI on MATLAB
- Randomly generated simple polygon with user-defined number of vertices
- A pre-defined floor plan. This option is for proving real-world floor plans as inputs.

(2) **Candidate beacon locations**: Next, we tested with beacon placements with different ranges. The range can be infinite ($\text{Inf}$), in which case it is only limited by the floor plan boundaries, or can finite, specified in meters. The beacon locations can be vertices or interior points. The interior points can further be randomly generated or user-specifies through the MATLAB GUI.

(3) **Beacon placement algorithms**: We implemented these placement schemes in the toolchain:

- **GDOP**: Prior work in minimizing beacons while maximizing expected accuracy based on GDOP [161].
- **UL**: Prior work in minimizing beacons while maximizing coverage [161].
- **RS**: Our proposed random sampling algorithm described in sec:rs-algo.
- **Sub**: Our proposed submodular-function algorithm described in sec:greedy-algo.
- **mod-RS**: Our proposed modified random sampling algorithm for accuracy, described in sec:GDOP.
- **Opt**: This is the optimal that our algorithm aims to achieve. We obtain this by searching through all possible solutions.
- **3-Place**: Optimal solution where any point in the domain are covered at least three beacons. This represents the best-case scenario with typical placement methodology.
6.7.1 Number of beacons

In this section, we compare the performance of the different algorithms for various floor plans and potential beacon location settings. These floor plans were chosen to be small enough such that we could compute optimal beacon placement. Figure 6.4 shows the floor plans. Figure 6.4(a)-Figure 6.4(c) show a real-world floor plan with different placement settings (vertex and interior placement, infinite and finite beacon range). Figure 6.4(d) shows a user-drawn star-shaped polygon. Figure 6.4(e) shows a floor plan representative of a large area and small rooms. Figure 6.4(f) and Figure 6.4(g) are real-world floor plans. The set of blue and red dots together indicate all possible beacon locations. The set of blue dots indicate the beacon locations selected by the proposed Random Sampling (RS) algorithm for placement. After placement, all regions are uniquely localized. The regions in white are covered by three or more beacons. The regions in dark grey are covered by 2-beacons and uniquely localized. NumberBeacons summarizes the results of the number of beacons placed by the different algorithms. We wish to make a few observations: (1) The number of beacons placed by our RS algorithm is similar to prior work, and close to the optimal number of beacons. Hence, we are able to have good performance in beacon placement, with an algorithm that has guarantees. (2) For Map1, there is not much difference in number of beacons with vertex only or vertex and interior placement when the beacon range is unlimited. We also see that
<table>
<thead>
<tr>
<th># of Vertices in the floor plan</th>
<th># of beacons (% reduction wrt UL)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UL</td>
</tr>
<tr>
<td>20</td>
<td>6.2</td>
</tr>
<tr>
<td>40</td>
<td>11.6</td>
</tr>
<tr>
<td>60</td>
<td>17</td>
</tr>
<tr>
<td>80</td>
<td>22.2</td>
</tr>
<tr>
<td>100</td>
<td>30.6</td>
</tr>
</tbody>
</table>

Table 6.2: Performance as random floor plan scales up even when allowed interior points for placement, half of the beacon locations selected are at vertices due to their high coverage. (3) When the range is limited, interior locations are chosen more often due to them having a higher coverage. (4) Our greedy algorithm does better than the previous greedy algorithm (UL) in some scenarios such as Multi-room map. (5) For real-world floor plans where the beacon range is limited by the floor plans, such as Map 1(b) and Map 3, our placement localizes a large part of the regions with just 2 beacons rather than 3, as seen by the large amount of area shaded in grey. The running time of these algorithms are roughly similar.

6.7.2 Performance at scale

To test the performance of the beacon placement algorithms at scale, we simulated random simple polygons with number of vertices varying from 20 to 100 in steps of 20. In every floor plan, we randomly generated an additional 30 interior points for candidate beacon locations. We generated 5 floor plans for every fixed number of vertices. RandomPoly shows the average number of beacons placed with each algorithm as the number of vertices increases. The algorithm in the prior work (UL) is set as the baseline.

We also show the percentage reduction in number of beacons placed by our algorithms. We observe that the number of beacons placed by RS is lower in general as the complexity of the floor plan grows. The reason is - the algorithms that are greedy optimize for coverage and place beacons sequentially, whereas the random sampling
algorithm updates the weights of points and places all the beacons newly in every iteration based on the weight of points. This benefit is more evident in floor plans that have small geometrical spaces rather than large open spaces (which is the case with random floor plans).

6.7.3 Accuracy improvement with modified random sampling algorithm

In this section, we consider the accuracy that can be obtained from the beacon placement, based on the modified random sampling algorithm described in sec:GDOP. In order to quantify the quality of the beacon placement, we evaluate the CDF of the GDOP for the given placement as in [161]. We have presented the results for Map 2 shown in fig:RealFL. For a fair comparison, we use the first 24 beacons that each algorithm places, and compare the expected accuracy that we can get from each placement. The results are shown in Figure 6.5. We see that the improvement in performance from RS to RS-mod with 60% GDOP improving from 1.7 to 1.5, and comparable with the GDOP from prior work that was optimizing for accuracy (GDOP). Thus we can accommodate practical metrics that relate to accuracy in or algorithm.
6.7.4 Comparison with user-placed beacons

Since our main contribution is a systematic approach for beacon placement that we believe to be difficult for most humans, we evaluated how our approach performs compares to manual placements from a number of users. We created an online form with four floor plans and asked users to place beacons with the challenge of placing as few beacons as required in order to cover as much of the floor plan as possible. Twenty anonymous users completed the challenge and we did not record any identifying information. We asked the users to indicate their experience with "beacon placement, localization systems, geometry algorithms". 40% indicated no experience, 30% indicated some experience and the remaining indicated reasonable or a lot of experience. The results are shown in Figure 6.6 for two floor plans. We see that for Map 3 shown in Figure 6.6(b), the algorithm outperforms the users in terms of the trade-off between number of beacons and area localized. For Map 1 shown in Figure 6.6(a), the algorithm places 8 beacons to localize the entire floor plan but 90% of the floor plan is localized with 6 beacons. This example indicates that when the constraint of unique localization is relaxed, further savings can be achieved in terms of beacon placement. Across the four tested floor plans, perhaps unsurprisingly, we noticed the trend that as the environment becomes larger and more complex, our algorithm begins to more significantly
out perform humans.

6.8 Conclusion

This paper presents a rigorous formulation of the unique localization problem in indoor environments. We observe, via simulation, that our beacon placement algorithm performs better than manual placements conducted by a number of volunteers we experimented with, even well versed with the problem. The proposed algorithm also places 5% fewer in real-world floor plans and 12% fewer on random floor plans compared to prior work based on heuristics. Beyond the improved performance, which, even for small percentage gain, can translate into large savings over sizable deployments, we believe that the value of the proposed methodology lies into its ability to provide a baseline to evaluate past and future deployment and to systematize the process of beacon placement. Future work will include relaxation of some key simplifying assumptions, and extensions to 3D environments.
References


