UNIVERSITÀ DEGLI STUDI DI CATANIA
DOTTORATO IN INFORMATICA (INTERNAZIONALE)
XXXIII CICLO

Claudia Cavallaro

Analysis of large GPS trajectories datasets via multi-agent techniques

TESI DI DOTTORATO DI RICERCA

Relatore:
Chiar.mo Prof O. Muscato
Chiar.mo Prof E. Tramontana

Coordinatore: Chiar.mo Prof. S. Battiato
ANNO ACCADEMICO 2019/2020

Contents

Un passo alla volta,
senza fretta ma,
senza sosta.

Johann Wolfgang Goethe

Abstract

The widespread use of mobile devices and Internet services on users' locations offer the opportunity to acquire information related to users' trips in real time. This availability has given rise to several studies based on geospatial trajectories but, because of the large volume of the collected information, processing them together is usually difficult. In this thesis we illustrate an approach that, using a multi-agent system, provides personalized recommendations of Points Of Interest (POIs), such as libraries, museums, restaurants, etc. to users. In our context, an agent is an application that improves user navigation in a city. It collects opinions, in terms of scores, that quantify the level of satisfaction in visiting a certain place in a certain period of time. In this approach, interesting positions emerge from the analysis of the collected data, hence scores and suggestions may be available for any large city in any place, when a sufficient number of people provide data. In addition, the next places to visit are suggested according to people's behavior and preferences.

Other directions explored are the identification of the flows of multiple users and the intention to predict the paths that will be taken by a user on the basis of the common paths, already known, of other individuals. Given a large dataset of geographic trajectories in an urban metropolitan area, an efficient strategy for detecting corridors is also proposed. These can be defined as geographical paths, of a minimum length, commonly crossed by a minimum number of different users. This approach is important for transportation analytics because it allows to detect missing lines in public transport systems and also to advise private users which route to take to move from one part of the city to another based on the behavior of users who have provided their GPS logs.

Although people like to visit popular places, due to health problems and due to the recent restrictions currently in place around the world for covid-19 influenza pandemic, meetings should be avoided. When planning a trip, one must consider both the attractiveness in terms of general interest for the destinations and the density of people who gather there. In the final part of the thesis, we propose a recommendation system that aims to offer to users some suggestions on useful routes and destinations which balance liveliness and overcrowding. First, we use datasets that store GPS locations as the basis for route and destination statistics. Then, we use an accurate probability algorithm that estimates the number of people moving from one place of the city to another and consequently show a list of destinations to users. Destination points are filtered according to the user's preferences on the density of people. A multi-agent system is used to manage user's requests to find a route for a journey, statistics on possible destinations

and suggestions for users. Thanks to our solution, we can inform users about suitable routes and destinations, as well as alert them when a favorite destination is overcrowded.

Acknowledgements

I would like to thank my Ph.D. supervisors Professors Emiliano Tramontana and Orazio Muscato for their work and their presence. Professor Tramontana has guided and supported me over the years, encouraging my choices. Professor Muscato has given me precious scientific advices, and he has passed on to me the passion for research since my master.

In these years I have had the privilege of carrying out a period of study and research at the "Universitat de Barcelona (UB)". For this I thank Professor Jordi Vitrià who accepted me for visit in his department and guided me in research for 6 months and the "Departament de Matemàtiques i Informàtica" of UB which provided me with a peaceful and stimulating environment to work. This great experience has allowed me to learn many things, grasp new aspects of scientific research and get to know new realities.

A sincere thanks to Professor Sebastiano Battiato for his important guide as Coordinator of the PhD course, for his availability and for his humanity.

I would like to thank Professor Paolo Bellavista for his hospitality at the Department of Computer Science and Engineering (DISI) of University of Bologna in the final period of my PhD.

Thanks to my father, a fundamental presence in my life. Thanks for his patience, for his encouragement and for understanding the importance that the study and the research has to me.

A hug and a final thank goes to all those people I met at the department of Mathematics and Computer Sciences of University of Catania, especially my collegues Francesco S., Gabriella V., Antonio S. and Fabio D. for their advices.

I acknowledge the support provided by my PhD grant PO FSE 2014 – 2020 founded by Regione Siciliana.

Chapter 1

Introduction

The research field explored, during the PhD, is Geo-localized data analysis. This analysis has become an important field of study due to the increase in the volume of geographic information, obtained from cell phones and GPS devices. An example of application for the analysis of geographic data is to help understand the behavior, movement of people and the models in the cities.

Given the widespread use of mobile devices that track their geographical location, it has become increasingly easy to acquire information related to users' trips in real time. This availability has triggered several studies based on user's position, such as the analysis of flows of people in cities, and also new applications, such as route recommendation systems.

The huge amount of data to analyze has led researchers to develop computational tools and data mining techniques combined with machine learning algorithms to allow better management and understanding of geographic information. Collecting the urban trajectories, for example, it might be helpful for people to make better decisions when they do visit an unknown location and there are interesting places in the city or for knowing how users interact with each other. It is possible to collect geographic information (e.g. from travel diaries, GPS data, image tags) from different sources such as smartphones, GPS devices, social networks, etc.

For Microsoft Researchers, the GeoLife project [104], [105] was started to predict the mode of transport between locations in a geographical region and support users to know how such places can be reached.

Geographical locations are widely used in applications such as recommendation systems. The widespread use of location-based Internet services (eg Google Maps) offers the opportunity to collect user locations.

In the first part of this research we propose a multi-agent system for creating recommendations of Points of Interest (POIs). Using collaborative agents and a centralized server, POIs are created from a dataset provided by Geolife. The server dynamically acquires information from agents, which are stored on the user's mobile device, creating new suggestions on the next place to visit. Another theme developed is the search for flows of multiple users. Discovered flows of people are useful, e.g., to find common itineraries, to suggest reachable spots to users, or to submit improvements on the public infrastructure.

In fact, in the second part of this work we propose a new algorithm to detect corridors in the same geographical dataset. Corridors can be defined as geographical paths, with a minimum length, that are commonly traversed by a minimum number of different users. We propose an efficient strategy based on the Apriori algorithm to extract frequent trajectory patterns from the geo-spatial dataset. After that, we refine the results obtained with a subsequent filtering step, by using a Radius Neighbors Graph. This algorithm is relevant for transportation analytics because it is the base to detect lacking lines in public transportation systems and also to recommend to private users which route to take when moving from one part of the city to another on the basis of behavior of the users who provided their logs.

Currently, organising a trip should take into account the number of people that will gather at their chosen destination points, as it is necessary to avoid visiting a place that will become overcrowded to comply with the restrictions due to the covid-19 influenza pandemic. Hence, an estimate of the number of people that will be in some place in a future time can be valuable for people moving and in situations where they could choose visiting some other place. Different techniques have been proposed to mine the knowledge of different users from GPS logs such as a probabilistic model to predict movements of people.

The proposed approaches have been validated by means of several experiments on data concerning locations, for eliciting the meaningful places, starting from data available on taxis, trucks, or people movements and also for predicting gatherings.

1.1 Contributions

Some results described in this thesis were published in journals:

- Cavallaro, C.; Verga, G.; Tramontana, E.; Muscato, O. *Eliciting Cities Points of Interest from People Movements and Suggesting Effective Itineraries.* Intelligenza Artificiale, 14 (1), 49-61, 2020.

- Cavallaro, C.; Vitrià, J. *Corridor Detection from Large GPS Trajectories Datasets.* Applied Sciences. 2020; 10 (14):5003.

and conferences:

- Cavallaro, C.; Verga, G.; Tramontana, E.; Muscato, O. *Suggesting Just Enough (Un)Crowded Routes and Destinations* Proceedings of the 21th Workshop From Objects to Agents, Bologna, Italy. CEUR Workshop Proceedings, 2020. Vol. 2706, pp. 237-251. ISSN=1613-0073.

- Cavallaro, C.; Verga, G.; Tramontana, E.; Muscato, O. *Multi-Agent Architecture for Point of Interest Detection and Recommendation.* Proceedings of the 20th Workshop From Objects to Agents, Parma, Italy. CEUR Workshop Proceedings, 2019, Vol. 2404, pp. 98–104. ISSN=1613-0073.

1.2 Aims and Approach

Nowadays, thousands of people are using their mobile devices to access new information in relation to their geographic location. This innovation has given rise to new services, which for example read GPS coordinates to receive information on nearby Points of Interest (POIs).

The first purpose of my research is to recover POIs and find the similarities between users based on their StayPoints (SPs). Taking advantage of a multi-agent system, we propose an approach providing personalized recommendations of places of interests, such as libraries, museums or restaurants, to users.

Current recommendation systems for tourist attractions or services available for a city are generally managed manually, e.g. a city council publishes a list of public services or a publisher indicates the top places, or POIs, for a place or region [4]. A static list of POIs may be insufficient for the support that people expect, since for a large city or region there may be hundreds of points, each with its own specific offer, where each point can vary over time and sometimes with a high frequency. The manual care of such a list could be cumbersome, and in some cases still a substitute for an updated list. In addition, each POI identified manually is not usually associated with an indication of the most advantageous time period for visitors and with the real-time conditions of the spot [26], [48].

Thanks to the availability of mobile devices, users can provide their comments on the places they visit in real time. Such comments can be collected and be useful for other people looking for advice. The proposed approach uses an agent-based solution to collect location data and user satisfaction in order to offer to users the next place they could visit. This is based on their current location, and on the time to reach the destination. The solution proposed in this thesis may be able to compose a personalized list of POIs for each individual user based on their previous position and trajectory, selecting real-time data, collected on a server, on user experiences in some positions. While users provide their comments and rate a place, their precise location and identity are preserved, and they are only kept on the user smartphone, whereas peer agents and the server do not handle such data.

We identify significant places for groups of users from the behaviors extracted from the GPS trajectories of different people and we want to predict the future behavior of a specific user directed to these locations. We also want to understand the correlation among user behaviors.

The multi-agent system proposed, which exchanges information with the centralized server, has been used for the creation of a recommendation tool for POIs. It provides users with: (i) a list of POIs, and for each point, (ii) additional information based on real-time data collected by other users, which help them choose the next destination with greater awareness. A key objective is to have this information as close as possible to real-time data, and ranks places based on feedback from other users, the most frequent time slots and the time spent visiting a place by other users.

The approach offers a better experience by giving additional dynamic data (such as popularity, number of users in real-time) to the list of POIs and by exploring their temporal relationships. In fact, for POIs, which we determine using the DBSCAN algorithm, we take into consideration the time intervals in which users have visited them, to offer a more advanced service. Finally, the approach was designed to preserve user privacy, i.e. it does not reveal the exact location of users.

The analysis of large human mobility datasets has the potential to provide many useful suggestions to public operators as well as to individual users, but in order to provide this information in a timely manner, efficient algorithms are needed.

In this scenario, corridor detection has emerged as one of the key elements to make informed decisions about public transportation systems as well to recommend optimal routes to individual users. In this case, the use of a brute-force algorithm is not an alternative due to its computational cost and more efficient solutions are necessary. To this end, we are proposing in this thesis a new approach, based on the Apriori algorithm, which is suitable to the use in very large datasets. In order to show its performance, we have analyzed the dataset of the GeoLife project of Microsoft Research Asia, which includes geographical trajectories of 181 users that uploaded GPS recordings corresponding to their routes in Beijing and its surroundings.

The algorithm Apriori, introduced by Agrawal et al. in [1], is a popular data mining algorithm for market basket analysis to detect which items are frequently bought together. This algorithm can be used in our scenario to reduce computational complexity. The key point of its application is to consider different assignments for its *items* and *baskets*. Specifically, we consider two alternating roles for *items* and *baskets*: in the first case, GPS points are represented as a set of *items* and trajectories as a set of *baskets*, but later we reverse this assignment and consider GPS points as a set of *baskets* and trajectories as a set of *items*. This strategy is particularly suitable in our case, where we have to identify corridors for a large volume of data and the naive application of Apriori would exceed any memory setting.

Both versions of the Apriori algorithm can be seen as filters that act on the set of points and trajectories. In the first case, we are able to detect sets of points that are shared by at least a number of trajectories. These points are not necessarily aligned to form trajectories. In the second case, we detect sets of trajectories that contain at least a number of points. By combining both approaches, the set of candidates that must be check to determine the final set of trajectories is reduced to a very low cardinality. The keypoint for this reasoning is to be aware that the complexity of Apriori depends not only on the number of items and bags, but on the number of different itemsets that are present in at least k bags. In the original case of supermarket application, itemsets of 2 elements represent the bottleneck of the methods. In our case, the bottleneck is created by itemsets with higher cardinality. By reversing the roles of items and bags and doing a final check of the remaining candidates, we bypass this bottleneck.

The Apriori algorithm works in a discretized space and its results must be refined. To this end, we use the Radius Neighbors Graph, which uses the mapping graph together with the adjacency structure of the GPS points in the neighborhood of the points of a fixed trajectory.

To communicate to users how many people are in some places, the statistics accumulated over time of recorded rides are usually used to estimate a measure of traffic or gatherings [20], [94], [11]. Moreover, both popular online services and other apps just count the number of people currently present in some place [28], [9], [58]. However, statistics gathered in the past cannot be a reliable indication for the current situation that has to cope with e.g. restrictions on gatherings, lower capacity of public transport means, etc. due to the influenza pandemic.

Additionally, a kind of real time measures of gatherings does not let other people plan their trip, hence understanding whether, e.g., one hour later when arriving at the destination, the place will still be (un)crowded.

A better estimate is therefore needed which takes into account: (i) the current amount of people in some place, and (ii) the statistics on the number of people that being in some origin place typically flow to another place to visit later on. In addition, an app behaving as an assistant agent is needed to timely inform interested people.

Our latest work, presented in the final part of this thesis, proposes an approach to determine the probability that users move along certain paths. Given the recordings of different user locations, we compute the probability for a user that being in place A will move to another place B (i.e. a possible destination), so when he arrives at place B he will contribute to the number of people who gather there. By computing in advance the probability that it will go to place B in a future time, we can guess if a place will become overcrowded. Our proposal for estimating people's destinations is based on the analysis of the co-occurrence of places statistically visited by a number of people above a threshold.

In addition, the app provides means to collect data on the current amount of people in a place and thus on their trajectories. When we collect user data via the app, we ensure that user privacy is preserved by providing only an approximate location to a central server.

Our approach can be useful in many contexts where estimating the number of people in advance can be a crucial factor for a better service, such as when organizing public transport, or for retailers, etc. In addition, it could be enriched with data, coming from competent authorities, which reveal some places where a covid-19-positive was found. Then, using our paths found, we could provide probabilities about other places where the infection may have spread.

The rest of the thesis is organised as follows. In Chapter 2 we survey related works, in Chapter 3 we clarify some definitions and techniques known in the literature. Chapter 4 describes the datasets used and the approach proposed. Chapter 5 shows the test performed and the results obtained. Finally in Chapter 6 we discuss the conclusions and draw future works.

Chapter 2

Related work

This chapter aims to present an overview of the state of the art on geospatial data analysis. In Section 2.1 we present different and general techniques on the study of trajectories, while in Section 2.2 we go into detail on works similar to those covered in this thesis, such as the extraction and recommendation of places of interest. We show below the differences between our proposed work and other papers in the literature, highlighting the innovative points presented in this thesis.

2.1 Trajectory monitoring

There are several approaches in the literature regarding the analysis of geographic trajectories. In fact, this is today an important research topic due to the increase in the volume of geographic information.

Zygouras and Gunopulos propose in [108] the analysis of real GPS trajectories collected from taxis operating in the city of Porto, moving buses in Dublin and Atlantic hurricanes from 1950 to 2004. In order to detect corridors, they discretize the trajectories using a grid and decompose the space in different sets of frequently observed locations. The Latent Dirichlet Allocation (LDA) model is applied to the trajectory dataset to extract frequent traffic patterns. Their LDA formulation is similar to the one commonly used for natural language processing, but replacing documents by grid cells and words by trajectories, respectively. LDA learns the probabilistic distributions of hidden variables that are introduced in order to discover patterns in the dataset. They apply a hierarchical clustering algorithm to the subtrajectories of each frequent set, following a bottom-up approach, using Dynamic Time Warping (DTW) in order to measure the distance between two subtrajectories. Finally, another algorithm selects the set of corridors from the set of candidate corridors minimizing the MDL principle.

Bicocchi et al. [12] propose a system that suggests daily and local transport sharing opportunities for short-term trips, by analyzing the traces of urban mobility in Milan and Turin. Based on "Call Detail Record" (CDR) data, they propose algorithms to recognize similar paths that can be used to recommend shared rides. The data used for the experiments was provided by a telecom mobile operator and support a travel recommendation system for multiple users. General mobility routines are identified through an extension of the probabilistic generative LDA model, performed on the set of available trips, including among others, leisure mobility routines and commuting trips. Both referred works seek paths that are common to multiple users, organizing the movements made in time slots, but with different approaches. In both cases geographical distance is measured through the Haversine formula. Their data are based on a sequence of intermittent positions. In fact, they identify the location of the user with respect to the cell when the user receives or makes calls or messages. Their goal is to extract 10 different positions shared by multiple users; our approach instead is to identify people movements and to detect the foreseeable routes by computing the number of gathering in some place.

Buchin et al. [16] consider the problem of detecting commuting patterns and propose algorithms that cluster subpaths of given trajectories. The idea is to select a reference trajectory and then find all subtrajectories that are close to this one by using the Fréchet distance. They consider the optimisation problem of finding the longest cluster of a fixed size and the largest cluster of a fixed length. The Fréchet distance is a distance measure for continuous shapes, such as curves and surfaces, and it is defined using reparameterisations of the shapes. Since it takes the continuity of the shapes under consideration, it is generally considered to be a more appropriate distance measure for curves than the Hausdorff distance. For polygonal curves, the discrete Fréchet distance (DDF) is a natural variant of the Fréchet distance. In this paper both are used.

Rolim et al. propose in [78] a method to identify movement patterns of sets of trajectories and analyze simulated trajectories in a region of Itaim Quarter in Santa Catarina - Brazil. They consider the frequency distribution of points for identifying a set of frequent regions. The framework consists of two phases: a partitioning phase, in which the trajectories are segmented using the Minimum Description Length principle (MDL) and a clustering phase, that considers the density to group similar segments in a cluster using the Fréchet distance as a measure of similarity between curves.

Devogele et al. [33] describe a new algorithm to compute DDF which aims to lower computation time and improved precision. It includes three different improvements: the first one is the Douglas & Peucker filtering process. Indeed, for GPS trajectories, the number of positions can be dramatically reduced. The second one is a pruning process. Only a small part of the two matrices required to compute the discrete Fréchet distance are computed. The last one is an improvement of the accuracy of the DDF. The proposed method is more complex than the regular discrete Fréchet distance, but CPU time is reduced. In terms of accuracy, a balance between CPU time and precision is required.

Zheng [103] conducts a survey about the major research in trajectory data mining. He evaluates trajectory data preprocessing and management and some other data mining tasks (trajectory pattern mining, outlier detection and trajectory classification). The paper also introduces some methods that transform trajectories into matrices and graphic tensors, to which different machine learning and data mining techniques can be applied.

Bian et al. [10] study the problem of trajectory grouping. Trajectory clustering has been applied in pattern recognition, data analysis and machine learning, and it is prevalent in some application fields such as the prediction of the movement of objects, traffic monitoring, understanding of activity and anomalous detection.

Crociani et al. in [29] offer an unsupervised learning approach for an automatic lane detection in multidirectional pedestrian flows. Unlike our work, they focus on pedestrian dynamics in a video using a distance function that takes into account the angular distance between the vectors. After that they aggregate instantaneous information on both position and speed of pedestrians to form clusters on short time windows. We use a fixed grid to group people positions into cells, and then compute the probabilities using the data arriving from the agents in real time.

Zheng et al. [106] aim to mine interesting locations and classical travel sequences in a given geospatial region. They first model multiple individuals' location histories with TBHG (tree-based hierarchical graph). Next, supported by the TBHG, they propose a HITS (Hypertext Induced Topic Search) inference model, which consider user's location as a directed link from the user to the location.

In [80] Sakamanee et al. describe multiple methods for inferring commuter route choice from cell phone network data and suggest users the best routes to follow. Based on a calendar year of CDR data collected by mobile users in Portugal, they interpolate waypoints of the route, calculate the shortest distance between a choice of possible routes and the positions of mobile use and also exploiting the Voronoi cells, assigning a choice route in coverage areas. Since the real positions of the users are not as frequent as our fine-grained GPS recordings nor available in the time frames in the absence of telephone activity, the tracks are deduced by choosing the most plausible ones among the paths suggested by Google Maps. Their work proposes to draw an inference for travel routes based on the statistics obtained, based on frequency in certain time intervals and origins-destinations of fixed routes. The noise of the processed data is filtered using DBSCAN as the clustering algorithm and the commuting radius as an admissible spatial interval.

Zou et al. [107] present an algorithm for modelling various movement phenomena, such as that of the movement of aircrafts in the airports of Hong Kong and Macau over a period of 30 days. For that purpose, a 4D time density is calculated, representing 3D spatial coordinates as well as speed. Using this representation they can detect hotspots and trajectory convergences.

In [27] Chessa et al. deal with mobility through the PartecipAct campaign of the University of Bologna, part of the projects called Mobile Crowd Sensing (MSC). This research aims to draw some general inferences on the usability and level of realism of these datasets and evaluate research solutions for MCS. The similarity between users, rather than being measured based directly on interests and therefore places of visit in common, is evaluated through social graphs: the more connections you have in common, the greater the probability for the same group of people to frequent the same places on long periods (e.g. university colleagues) and therefore to share the same habits (gyms, recreational activities in the same city). The participants who voluntarily collect data in PartecipAct are all students (170 in total), the datasets we explored are heterogeneous users. Furthermore, the client they use asks users if they want to perform certain activities, and only in case of confirmation does it collect the data (active detection). The back-end takes care of processing and archiving the detected data. Geonotification associates activities with one or more geographical areas and automatically notifies users as soon as they enter them. To verify the popularity of certain areas, the co-location of the users is extracted from the device's position via bluetooth: it is assumed that two devices are co-located and able to communicate if they are positioned within 10 meters of each other for at least 150 seconds. Frequent zones are then aggregated via DBSCAN.

Hosseinpoor et al. [47] introduce an approach for identifying critical points on a large volume of real data. Three trajectory descriptors, namely the curvature areas, the turning points and the self-intersecting points, are determined by detecting the critical point of trajectories using a convex hull (TCP-CH) algorithm. These characteristics allow the detection of the directions of change, represent the changes of the path and the presence of double-level intersections. They are useful in order to show, in addition to anomalous values, the geometric properties of the trajectories: shape, complexity, direction and distance. The approach allows, by selecting some trajectory parameters, to identify the number and positions of the points of interest and identify from them the similarities of the trajectories.

An accurate monitoring of user mobility provides support for efficient resource usage. E.g., it could help avoid traffic congestion [20],[59],[96].

2.2 Extraction and recommendation of places of interest

This work is an intersection of multiple disciplines, including POI recommendation systems, collaborative filtering and privacy preserving systems.

1. Over the years, these analyzes have fueled different studies on trajectories, such as computing the probability of moving from one POI to another, using, for example, the Markov chains [40], [66] and therefore creating methods which predict the subsequent movements of the individual user from the analysis of their POIs.

2. Collaborative Filtering (CF) approach is one of the approaches for creating recommendation systems. It creates suggestions using a similarity metric among users. The assumption is that similar users probably have similar tastes. The concept of CF was introduced in 1992 by Xerox research staff within the Tapestry project, a system that allowed users to track documents based on comments left by other users [44]. Later, several ratings-based automated recommendation systems were developed, e.g. the GroupLens research system [82] provides a pseudonymous CF solution for Usenet news and movies. Other technologies have also been applied to recommending systems such as Bayesian networks [15], [68], [95] and clustering [25], [95].

3. Privacy preserving: collaborative filtering techniques have been very successful in e-commerce and in direct application recommendation. They are widely used and very useful but they often fail to protect user privacy (for example in the case of GPS coordinates transmitted with their timestamp), so they have some disadvantages. In [18], [19] privacy violations are addressed with cryptographic systems, which can reduce the risk to the user. In other research works, e.g. in [75], each user first disguises his private data, and then sends it to the data collector. Therefore, a Randomised Perturbation (RP) technique is used to camouflage private data [2]. Moreover, anonymisation techniques can be used, however these introduce some attack problem, making datasets not very useful [77], [90].

Unlike other approaches, our proposal includes a solution to identify POIs through the use of the DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm. Then, collaboration filtering is used with the dynamic calculation of ratings based on user experiences. For such a rating we use spatio-temporal variables offering a dynamic and realistic outcomes.

This is done by safeguarding the privacy of users because the centralised server only tracks the movements near the POIs. Furthermore, it is important to offer a service that makes the user and her privacy more secure. To do this we have users sharing their position only if they are close enough to a POI and this information is manipulated to ensure user safety. E.g. the position of a user within the radius of a POI will be saved in our central server with an error rate of about 300 meters in order to preserve the user's privacy. This does not corrupt our system data and better protects users.

In the literature, several studies use collaborative filtering to suggest itineraries or POIs. In [48], the authors propose time-sensitive trip routes, consisting of a sequence of locations with associated timestamps. In [101], Yoon et al. present a recommendation for itineraries based on multiple user-generated GPS trajectories. In [49], Hsueh and Huang suggest a user-based collaborative filtering with time preference to explore user preferences on places visited and offer a recommended itinerary. In [28], data on Foursquare were used to find clusters considering both spatial and social proximity, and results can be useful to characterise the amount of people and the their flows on portions of a city. In [76], photos on Flickr were analysed to suggest routes that can be pleasant, beautiful or quiet, according to the geo-location of photos and user comments.

Moreover, in [9], data gathered from both Flickr and Foursquare were used to identify POIs in Milan (Italy).

Compared to the approach we propose, these systems have the following shortcomings:

1. the nearest points for the user are not identified;
2. the data is not updated in real time;
3. POIs are not automatically detected.

This work includes automatic calculation of POIs, starting from a series of trajectories. In addition, while other works offer a complete itinerary, urging the user choose an origin and a destination, in this proposal a user can immediately see the closest recommended points and can dynamically change the next point of the itinerary according to current needs. That allows users to search more effectively through travel information and organize the trip.

In [3], [54] and [65], the identification of the relevant places through various clustering algorithms is based on the data provided by a telecommunications operator who has recorded events such as calls or text messages. This is a statistic on the amount of people close to some place of interest (e.g. a supermarket). For the above approaches, the data is only available to very popular telecommunications operators or Internet service providers. Instead, the analysis proposed in this document can be performed on data collected by users' devices, therefore without the external support provided by large telecommunications operators. Furthermore, in our approach we suggest locations that represent places that are meaningful to people, rather than just popular places.

The algorithm for detecting personal anchor points like home or places of work or regularly visited sites, unlike our approach is based only on frequency, or if these areas are regularly visited in certain time slots, but not on the time spent in the surroundings. We extract the StayPoints from each log route, and from these we determine the POIs, since if a place has meaning and has passed through them, the user will presumably have spent some time nearby.

Furthermore passive mobile positioning is related to the precision of the cell, as each has a certain coverage radius (which has a maximum value of about 35 km), therefore the exact position cannot be determined but with an uncertainty given by the coverage of the cell to which it hooks. Further uncertainties are given by the fact that the size of the cells are not fixed and that a GSM mobile phone can hook to one of the two neighboring cells depending on whether one is crowded or with a disturbed signal.

In such cases, since a visit is recorded through the cell phone only if the user makes a call or receives a text message, it is not possible to determine how long he remains in that area. In our case the entire path associated with time is marked, this analysis based on trajectories with regularly recorded GPS positions allows us to investigate urban mobility in a more accurate way.

In [50], places of interest are collected according to an analysis of geotagged photos and grouping data according to DBSCAN. This work differs from the proposed approach in that it does not consider: time slots for places of interest, comments from people on such places and data updated in real time.

An Area Of Interest (AOI) could be visited only during the night and therefore could not be extracted because it is not popular according to this method. An analysis of these areas dependent on the seasons is missing, as a place could be interesting to visit in summer and not in winter or vice versa.

In this approach it is not easy to find flows of people, because the whole path is not recorded.

Assigning importance to an AOI, through the TF-IDF method (term frequency–inverse document frequency) [81] , based on the tags inserted is influenced by subjectivity: in social networks some users would prefer to use popular tags to attract likes, conversely an AOI that has few tags or little less significant tags would take away importance to the area.

In [58], the places of interest are determined using clustering techniques on the geolocation of people according to the distance of Haversine, also used by us. The time parameter for StayPoints is not changed on the basis of the dataset, because this problem is not analyzed by them. Our work offers further contributions in terms of privacy for users, also providing users with real-time comments and the classification of useful time slots for the selected places.

In recent years, CF recommendation systems have been supported by multi-agents systems. A multi-agent system is a system with a significant number of independent agents interacting with each other [87], [34]. In recent years, multi-agent systems have been widely used as they are considered suitable for systems with modular architecture, thanks to their independence [87]. In general, agents interact in three ways [85]: (i) each agent can communicate directly with any other agent (called “autonomous agents”); (ii) agents communicate with each other indirectly through an intermediary (called “facilitator”); (iii) all agents communicate with each other through an intermediary, however agents may communicate with each other after the communication has been set up by the intermediary (called “Mediator”). In the second case,

Table 2.1: Differences between this approach and other existing works

Feature	[101]	[48, 49]	[7]	[26]	[84]	[50, 58]	[3, 9, 54, 65]	this approach
visiting time		v						v
visiting order	v	v		v				v
visiting duration	v	v		v				v
location based service				v	v	v	v	v
uses data that is always up to date				v	v	v		v
use multi agent system			v	v	v			v
calculates points of interest						v	v	v
preserve privacy								v

the robustness can be low and the overhead is relatively high but the intermediary acts as a protective wall for the privacy of the users because the agents do not communicate directly and process the information received from the users, reducing their work [63] On the other hand, the use of an intermediary has several advantages in terms of synchronization, reusability, scalability and modularity [46].

In [56], the authors proposed a multi-agent system that allows users to optimize the energy consumption of their smart homes. Each electrical device is configured as a virtual agent. These agents work simultaneously and together to reduce consumption while ensuring user comfort, energy costs and maximum energy savings.

Agent-based recommender systems have been proposed in the last years in different scenarios including the tourism. For example, in [7], Batet et. al. present Turist@, i.e. a system based on multi-agent technology to give personalised tour attraction recommendations more effectively, highlighting the usefulness of finding points near the user. Similarly, [26] illustrates an application to better plan travel decisions based on a multi-agent system. The authors of [84] propose a system that produces recommendations for both individuals and groups.

Finally, this approach is the first approach in the tourism sector that deals with preserving user privacy in two ways: by using a centralized server the agents do not directly know each other to exchange information and by extracting only information on POIs, and an approximate position of the user, only when he expresses an opinion on a POI.

Table 2.1 summarises the differences between our approach and other existing works in location recommendation. Note that only the proposed approach makes use of gathered data to compute points of interests, which therefore emerge and change while users interact with the provided system; additionally, several precautions have been taken to ensure anonymity of the user identity and their location.

Chapter 3

Preliminaries and Basic Definitions

This chapter recalls the basic concepts and definitions used in the thesis and describes some algorithms applied in literature in some related works.

3.1 Geographic definitions

Parallels and meridians form, on the Earth's surface, a geographical grid which allows to identify the absolute position of each single point. Latitude and longitude define the geographical coordinates of a place. A basic meridian, which passes for an astronomical observatory located in Greenwich, it has been conventionally set. This meridian is also called zero or origin meridian (has zero longitude).

Longitude is the angular distance of a point from the fundamental meridian. This corresponds to the angle between the meridians plane of the point that we need to detect and the plane of the fundamental meridian. **Latitude** is instead the angular distance of a point from the equator, being measured along the meridian which passes through that point.

According to the rules of spherical trigonometry we can determine the shortest curve that connects two points of the Earth (geodesic). Remember that thinking about the Earth as a sphere is an approximation, since in reality it has the shape of an ellipsoid and the terrestrial ray is longer at the equator ($6,378\ km$) and shorter at the north and south poles ($6,357\ km$).

Definition 3.1.1. *For the Spherical Law of Cosines the distance d between two points, A =(lat_1, lon_1) and B =(lat_2, lon_2), of given latitude and longitude is defined by:*

$$d(A,B) = R * arccos(sin(lat_1) * sin(lat_2) + cos(lat_1) * cos(lat_2) * cos(lon_2 - lon_1)). \quad (3.1)$$

The angles used are expressed in radians. This formula approximates the Geoid to a sphere of medium quadratic radius, $R = 6,372.795477598\ km$, so the distance calculation may have an error of $0.3\ \%$, particularly in the polar extremities, and for long distances crossing several parallels.

Definition 3.1.2. *The Haversine formula is more accurate than the cosine formula for measuring distanced, due to problems associated with small distances. Important in navigation, the* *first table of Haversines was published by Andrew in* 1805*, but the term Haversine was coined in* 1835 *by Inman [53]. The Haversine distance [86]* $\mathrm{d}(A, B)$ *between two points* A *and* B*, which gives the great-circle distance in kilometers between two points on a sphere, is defined as follows:*

$$d(A,B) =$$

$$2R \arcsin \sqrt{\sin^2 \frac{lat_1 - lat_2}{2} + \cos lat_1 \cos lat_2 \sin^2 \frac{long_1 - long_2}{2}}. \tag{3.2}$$

The angles used are expressed in radians, R *is the terrestrial radius (average radius* $6,371.005076123$ *km).*

Since the Earth is divided into equal parts by the meridians, we know that 1 degree of latitude equals 111.121 *km* while 1 degree of longitude varies according to where we are because all parallels have different diameters moving from the equator to the north.

A trajectory log is a set of geographic locations for a user in motion, ordered by time, from the moment he starts recording his journey until the moment he stops.

Definition 3.1.3. *A trajectory* T *is defined as* $T = ((lat_1, long_1, date\_time_1); \ldots; (lat_n, long_n, date\_time_n))$*, where* lat_i *and* $long_i$ *for* $i = 1 \ldots n$ *are the latitude and longitude in decimal degrees of the GPS points of the trajectory,* $date\_time_i$ *is the time and the date in which every point was registered and* n *is the length of the trajectory* T *(the total number of its points).*

Each trajectory in the GeoLife database, that we used for experiments and described in the next chapter, is identified by a trajectory ID and by the user ID who has traveled it.

Definition 3.1.4. *A sub-trajectory is a sub-sequence of a* T *trajectory, that is, a part of points consecutive within the trajectory.*
$SubT = ((lat_{m_1}, long_{m_1}, date\_time_{m_1}); (lat_{m_2}, long_{m_2}, date\_time_{m_2}); ...; (lat_{m_k}, long_{m_k}, date\_time_{m_k}))$*,* *where* $1 \leq m_1 \leq m_2 \leq m_k \leq n$*.*

Since the trajectory data can be collected by different devices, the log routes will be of different sizes and related to different objects in movement and which can be of different sources and categories:

1. **People**
   The trajectories in this case refer to people who have registered their movements in the real world in the form of GPS tracks, and this it can take place both in active and passive mode for an arbitrary period of time.

   - **Active recording**
     The data of this type of source concern people who spontaneously enable the location service on the devices in order to obtain GPS tracks for trajectories, for this reason it is called active recording. This type of data generally affects people in a trip, who record their track for storage purposes travel itself, or cyclists or runners who

record their activities for subsequent sport analysis and comparisons with previous performances. In some social networks it is also possible to recreate a route made from geo-tagged media in different places.

- **Passive recording**
  It is possible to reconstruct a trace relating to a person's movement without the individual activating specifically the GPS detector on their device, in this case we are talking about passive recording. For example by the sequence of telephone cells connected by the mobile devices, with the corresponding time in which you have passed through the associated area, or when using a credit card it is possible to connect the different positions in which you were registered.

2. **Vehicles**
   A multitude of transport vehicles are nowadays equipped with GPS systems that track their journeys, including for example taxis, buses, ships and planes. The related data report the temporal positions recorded by the devices, which can be used for various purposes such as better allocation of resources, traffic analysis or improvements of transport.

3. **Movements of animals**
   Recently the behavior of animals has been widely studied by biologists and zoologists to understand behavior and displacements carried out by groups of animals in different areas, for this purpose different GPS devices have been used for reconstruct trajectories and subsequently analyze their movements.

4. **Mobility of natural phenomena**
   The same principle is applied to natural phenomena, such as hurricanes, tornadoes and ocean currents. The trajectories related to these phenomena capture their movement and allow analysis to be performed deeper on the usual changes and behaviors of these phenomena.

3.2 Other different distances used to compare trajectories

Let S be the symbolic representation of a measurement space and let x, y, z be three points in S. A **dissimilarity** or **semimetric measure** is defined as a function $d(x, y) : S \times S \to R$ which satisfies the following conditions:

1. $d(x, y) = 0 \Leftrightarrow x = y$;
2. $d(x, y) \geq 0 \ \forall x, y \in S$;
3. $d(x, y) = d(y, x) \ \forall x, y \in S$.

The first condition indicates the reflexivity of the relationship, the second requires that the distance, however is not negative, the third finally indicates symmetry.

Under the conditions listed above, if the function also satisfies the following "Triangle inequality":

$d(x, y) \geq d(x, z) + d(y, z) \; \forall x, y, z \in S$
is said to be a **metric**.

Likewise, a **similarity function** is defined to satisfy the conditions:

1. Symmetry: $s(x, y) = s(y, x) \; \forall x, y \in S$;

2. Positivity: $0 \leq s(x, y) \leq 1 \; \forall x, y \in S$.

   If it also satisfies these conditions

3. $s(x, y)s(y, z) \leq [s(x, y) + s(y, z)]s(x, z) \; \forall x, y, z \in S$ and

4. $s(x, y) = 1 \Leftrightarrow x = y$,

it is called a **similarity metric**.

For a data set with input patterns, we can define an symmetric matrix, called proximity matrix, whose $(i, j)th$ element represents the similarity or dissimilarity measure for the ith and jth patterns $(i, j = 1, \ldots, N)$.

A fundamental part of each clustering algorithm is an appropriate choice of a distance or dissimilarity measure that allows to translate numerically the concepts of association between similar elements and distinction between elements belonging to different clusters.

Given two sets of data X and Y of length m:

$$X = x_1,\, x_2,.\ .\ .,\, x_i,.\ .\ .,\, x_m$$
$$Y = y_1,\, y_2,.\ .\ .,\, y_j,.\ .\ .,\, y_m$$

it must be established how close the two series are.

Definition 3.2.1. *The* ***Euclidean distance*** *between X and Y is defined as follows:*

$D(X, Y) = \sum_{i=1}^{m} \sqrt{(x_i - y_i)^2}$.

Definition 3.2.2. ***Minkowski's distance*** *between X and Y is defined as follows:*

$D(X, Y) = (\sum_{i=1}^{m} (x_i - y_i)^r)^{1/r}$.

where r is an input parameter that can take:

- $r = 1$*: Manhattan distance;*
- $r = 2$*: Euclidean distance;*
- $r \to \infty$ *is the maximum distance between the components of a vector.*

Definition 3.2.3. *The **Lagrange-Tchebychev distance** is defined as follows:*

$D(X,Y) = max_{1 \leq i \leq m} \, |X_i, Y_i|.$

Definition 3.2.4. *The **Mahalanobis distance** is defined as follows:*

$Mahal\ (X,Y) = (X-Y)\ \sigma\ (X,Y)^{-1}(X-Y)^T$ *where* σ *(X, Y) is the covariance matrix.*

Many applications, such as those that use financial and marketing trends, require certain temporal similarity criteria on the data. Look for patterns similar to some known behaviors can help predict or test hypotheses.

We use the following conventions. If S and Q are two sequences:

- $len(S)$ is the length of S;
- $S[i:j]$ is the sub-sequence in S identified from position i to j;
- $d(S,Q)$ is the distance between the two sequences.

We can classify, given a sequence, two types of similarity queries.

1. **Whole matching**: given N data sequences $S_1, S_2, ..., S_N$ and a query sequence Q, all of the same length, we want to find all the sequences that they are distant at most ϵ from Q: $d(S_i, Q) \leq \epsilon$.

2. **Subsequence matching**: given N data sequences $S_1, S_2, ..., S_N$ in length arbitrary, a Q query sequence and a tolerance ϵ, we want to identify the S_i sequences $(1 \leq i \leq N)$ which contain subsquences whose distance from Q is smaller than or equal to ϵ.

If sequences X and Y have respectively n and m element, we can align these two sequences using the **Dynamic time warping (DTW) algorithm** [8].

In fact, DTW can lead to a measure of distance between the two aligned sequences. This algorithm is particularly useful for treating the sequences in which individual components have characteristics that require over time, and for which the simple linear expansion or compression of the two sequences does not bring satisfactory results.

For example, similarities in walking could be detected using the DTW, even if one person walked faster than the other, or if there were accelerations and decelerations during an observation. It has been used in various fields of application, from voice recognition to recognition of motor activities.

Example

Consider an $n \times m$ matrix called D, where $d(i,j)$ is the distance between x_i and y_j.

There are many metrics used for computing distances. Typically Euclidean distance is used and therefore, at this point it is possible to build an alignment path between the two sets of data. The optimal alignment between X and Y data series is obtained by minimizing the local distance $d(i,j)$ between the series of points of X and of Y.

A matrix D $n \times m$ is therefore constructed with the following procedure:

the following values are assigned to positions (0,0), (1,1), (i,0), (0,j):

D(0,0)=0,

D(1,1)=d(1,1),

D(i,0)=∞ for $1 \leq i \leq n$,

D(0,j)= ∞ for $1 \leq i \leq n$.

The remaining values of the matrix are assigned using a recursive procedure:

D(i,j)=d(i,j)+ min {D(i-1, j-1), D(i-1,j), D(i, j-1)}.

A further step is required to obtain the optimal alignment between X and Y, that is the optimal warping path W. W is defined as a continuous path of w_k elements in the matrix D, where each w_k corresponds to an element $D(i,j)$ of the matrix D, chosen so as to optimize the alignment between X and Y.

The complete warping path is described as: $W = w_1, w_2, ..., w_k, ..., w_K$

$$max(m,n) \leq K < m + n - 1.$$

The search for the path is bound by the following conditions:

1. **Boundary conditions**. $w_1 = (1,1)$ and $w_k = (n,m)$. This means that warping path must start and end in the elements $D_{1,1}$ and $D_{n,m}$ of the matrix.

2. **Continuity**. Given $w_k = (a,b)$, then $w_{k+1} = (a',b')$, where $a' - a \leq 1$ and $b' - b \leq 1$. This forces warping to move on adjacent cells horizontally diagonally or vertical.

3. **Monotonicity**. Given $w_k = (a,b)$, then $w_{k+1} = (a',b')$, where $a' - a \geq 0$ and $b' - b \geq 0$. This ensures that the path continues at each step towards $D_{n,m}$ and does not go back to $D_{1,1}$.

The optimal path is obtained by minimizing the: $DTW(X,Y) = min(\frac{\sqrt{\sum_{k=1}^{K} w_k}}{K})$.

The constant normalization for the K value is used to compensate for the fact of have warping path with different lengths in order to optimize the alignment local between datasets. To reconstruct the path, you need to memorize another structure in addition to D, used to store the direction of the path. The path is obtained a posteriori using this pointer.

3.2.1 Fréchet Distance e DDF

For an intuitive definition we can imagine a man crossing a finished curved path while walking his dog on a leash, with the dog crossing a separate one. We suppose that the dog varies his speed to keep as loose in his leash as possible and the dog and this owner have to walk without backtracking from an endpoint each other: the Fréchet distance [39] between the two curves is the length of the short leash sufficient both to cross their separate paths.

The Fréchet distance is a measure of similarity between curves that takes into account the location and ordering of the points along the curves. Therefore it is often better than the well-known Hausdorff distance.

A parameterized curve in $\mathbb{R}^d$ can be represented as a continuous function $f : [0, 1] \to \mathbb{R}^d$. A monotone reparametrization α is a continuous non decreasing function : $\alpha : [0, 1] \to [0, 1]$ with $\alpha(0) = 0$ and $\alpha(1) = 1$. Given two curves $f, g : [0, 1] \to R^d$, their Fréchet distance , $\delta F(f, g)$, is defined as:

$$\delta F(f,g) = \inf_{\alpha,\beta} \max_{t\in[0,1]} d(f(\alpha(t)), g(\beta(t)))$$

where $d(x, y)$ denotes the Euclidean distance beetween points x and y, and α and β range over all monotone reparametrizations.

Alt and Godau described the first known algorithm [5] for the continuos Fréchet distance between two polygonal curves in Euclidean space, based on the principle of parametric research. The execution time of their algorithm for two polygonal curves with m and n segments is $O(m * n * log(m * n))$.

The **discrete Fréchet distance** (DFD), also called coupling distance, is an approximation of the Fréchet metric for polygonal curves, defined by Eiter and Mannila [35]. The discrete Fréchet distance considers only leash positions where the endpoints are located at the vertices of the two polygonal curves and never within an edge. This particular structure allows the discrete Fréchet distance to be calculated in polynomial time by an easy dynamic programming algorithm. DFD is computable in $O(m * n)$ time.

Definition 3.2.5. *The discrete Frèchet distance between two polygonal curves $a : [0, m] \to R^k$ and $b : [0, n] \to R^k$ is defined as:*

$$DFD(a,b) = \min_{\substack{\sigma:[1:m+n]\to[0:m],\\ \beta:[1:m+n]\to[0:n]}} \max_{s\in[1:m+n]} \{d(a(\sigma(s)), b(\beta(s)))\}$$

where σ and β range over all discrete non-decreasing onto mappings of the form $\sigma : [1 : m + n] \to [0 : m]$, $\beta : [1 : m + n] \to [0 : n]$.

This similarity measure has been applied to different problems, from morphing and handwriting recognition, to the alignment of the protein structure, to computer graphics, as well as geographic applications, because it captures perceptual and geographical similarity between discrete trajectories.

DFD can tolerate non-uniform sampling rate. Other distance measures require that points along the trajectories are uniformly and densely sampled, but for real datasets it is difficult to have. Another characteristic of Discrete Fréchet distance is the local time shifting, that is the ability of tolerating short term discrepancies (e.g., missing samples, measurement errors) in aligning two trajectories.

Such distances can be used in our case to compare which GPS subtrajectories are closest to each other. We initially calculated the distance between pairs of points for different trajectories using Haversine's formula to find flows (see Sections 4.2.5 and 5.2) and then to determine the frequently visited areas more quickly we used the Apriori algorithm (definitions in Section 3.6) in Sections 4.2.6 and 5.6.

3.3 Recommendation Systems and collaborative filtering

It is difficult to extrapolate relevant information from a large amount of data such as the correlation between individuals. The similarity between users is very important for example in the customers and in the commercial companies. Many searches have been based on the transactions of supermarkets and bookstores or on the behavior of the users in the web communities, to allow an individual to discover potential friends based on interests similar in the areas of books, films and music. In addition, this would allow sellers to improve their sales and marketing strategies by recommending the right products to consumers.

Similarity is also important in geographic information systems, also exploring the correlation between geographic location. We think that if there are users who share the same location history they could share similar interests and preferences. The more places they have in common, the more they could be connected. And so we want to mine the similarity based on GPS trajectories of different users generated in the real world.

Recommendation systems use community opinion to identify more effectively their interest among a range of content. A well known technique used in such systems is called Collaborative Filtering [83] when trying to predict the evaluation of a product for a given user. The general idea behind it is that similar users vote similarly on similar items. Therefore, if the similarity between users and items is established, you can make a potential prediction for a user's rating for some items.

It has also been explored in social networks to facilitate people to identify potential friends and content that is interesting to them on the Web. One of the most commonly used algorithms is the closest neighborhood approach. In our work we extend the direction of similarity from people's online behaviors to the chronology of places visited in the real world.

Recommendation systems are divided into three main categories [13]: collaborative filtering, (CF), content-based filtering and (CB) and hybrid filtering (HF). Content-based filtering [64] makes recommendations based on user choices made in the past (e.g., for a person who likes a carbonated drink such as cola, the system offers recommendations for similar soda drinks). Collaborative filtering [83] allows users to give ratings about a set of elements, so that when enough information is stored on the system, it is possible to make recommendations to each user based on information provided by those users that have the most in common with them (for example, if Bob and Alan have seen the same horror films, one of the films of the same category seen by Bob is suggested to Alan and vice versa). The hybrid technique is a mixture of the first two. In our approach, suggestions are chosen based on the experiences and choices of other users, falling back into the CF systems. These experiences are documented by agents who observe user behaviour in real time.

3.4 Multi-Agent System

Several studies have proposed the use of multi-agent systems (MAS) [30] in a wide range of different domains. In 1998, a study described a supporting system for suggesting possible purchases during shopping based on the GPS position with the use of agents [37]. In general, there are two main approaches to MAS developments: centralised policies (CMAS) and decentralised policies (DMAS) [99].

- A centralised approach consists of taking all of the decisions in one place. In a typical CMAS, a central server collects all the relevant data that come from the different actors (that is, agents) and identifies the decisions for each agent according to the global system state. The centralised view of the system can be described by a multi-agent Markov decision process model, an example is presented in [14].

- A decentralised approach consists of making each entity responsible for its own decision. In a typical DMAS, an agent cannot see other agents local states and local actions, and has to decide the next local action on its own. Thus, each agent has only a partial view of the systems global state, and different agents have different partial views. A good example is in [100] whose authors propose a decentralised multi-agent decision process framework that provides the basis for a decision-theoretic study of decentralised policies.

The decentralized architecture has advantages in synchronisation, reusability, scalability, and modularity [43], [46], [92]. However, the complexity of decentralised systems is greater than that of centralised ones. Although decentralisation shows obvious advantages, decentralisation also has its own drawbacks, including that agents cannot predict the group behavior based only on the available local information, possible instability, and sub-optimal decisions. Due to the importance of total knowledge, the choice for the approach we proposed fell into the first category. In this work, apps on smartphones are agents that indirectly communicate among themselves and take some decisions on behalf of the user. In addition, the centralized server is able to filter information by offering advice to users without sending their sensitive data; this preserves the user's privacy.

3.5 Clustering techniques

Cluster analysis is the search for groups of objects such that objects appear members of a group are "similar" to each other and different from the objects in the others groups. The intercluster distances (between different cluster data) are maximized, intracluster distances (between data belonging to the same cluster) are minimized. Clustering is referred to as unsupervised classification (unsupervised learning): as for classification, the purpose is to segment the data, but without assigning class labels. In fact, we do not have pre-defined classes, but each cluster can be interpreted as a class of similar objects.

Clustering is a collection of clusters. An important distinction is between:

(i) Partitioning clustering: a division of objects into subsets (clusters) not overlapping, where each object belongs exactly to a cluster.

(ii) Hierarchical clustering: a set of nested clusters organized as a tree hierarchical.

Among the types of partitional clusters we find the center-based one, where a cluster is a whole of points such that a point in the cluster is closer (or more like) to the "center" of the clusters rather than in the center of each other. The center of a cluster is called centroid, the average of all points belonging to the cluster, or medioid, the most "representative" point of the cluster.

3.5.1 K-Means

$K-$means [38] is a partitioning clustering technique distance-based, in which each cluster is associated with a centroid. Each point of our data is assigned with the cluster whose centroid is closest. The number of clusters, K, must be specified as input, as well as the matrix of the data to be clustered (in our case the pairs of latitude, longitude of the GPS points).

The choice of the number of K clusters must be well thought out. Generally it is set between a minimum limit of 3-5 (to avoid too large clusters) and a maximum (clusters too small would undermine the simplification process that underlies the grouping).

Given K, the K-means algorithm is implemented in iterative steps, and detects center-based cluster:

1. Choose K points that represent the initial centroids (means) of the clusters. You can randomly choose K observations, for initializing the cluster center.

2. Calculate distances from all observations to each centroid.

3. Assign each object to the center of the nearest cluster.

4. Calculate the average of the observations in each cluster to get new K's centroid positions (distance recalculation).

5. Repeat steps 2 through 4 until the cluster assignments change or the maximum number of iterations is reached.

It is possible to use different metrics for the K-means such as Euclidean distance, Manhattan distance, cosine, correlation, hamming.

There are various evaluation measures to estimate the quality of the clustering obtained for the chosen parameters, such as Calinski-Harabasz Index [17], Davies-Bouldin Index [31], Sum of Squared Error and Silhouette [79]. The Calinski-Harabasz index is computed as a ratio of inter-cluster variance to within-cluster variance, whereby good clustering solutions tend to have large inter-cluster variation and smaller within-cluster variation, so that the optimal cluster number corresponds to the largest ratio value obtained.

3.5.2 DBSCAN

Other partitioning clustering methods are based on the notion of density. Their idea general is to grow a given cluster until the density (number of objects or points of data) in a neighborhood of a certain radius does not exceed a certain threshold. This method can be used to filter noise and discover arbitrarily shaped clusters.

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [36] and its OPTICS [6] extension build clusters based on a density-based connectivity analysis.

Some new definitions are required to understand how density-based clustering algorithms work. DBSCAN requires only two parameters (no prior knowledge of the number of clusters): the maximum radius of the neighbourhood, ϵ, and the minimum number of points in an ϵ neighbourhood of a point, *MinPts*.

The neighborhood of radius ϵ of a given object p of the dataset D is called "eps-neighborhood" object: $N_\epsilon(p) = \{q \in D \text{ s.t. } dist(p,q) \leq \epsilon\ \}$.
The shape of a neighbourhood depends on the distance function.

Given ϵ and *MinPts*, this algorithm categorizes the objects into exclusive groups.

- A point is a **core point** if it has more than a specified number of points (MinPts) within ϵ. These are points that are the interior of a cluster.
- A **border point** has fewer than *MinPts* within ϵ, but is the neighborhood of a core point.
- A **noise point** is any point that is not a core point nor a border point.

The algorithm follows these steps:

1. randomly select a point p;
2. retrieve all points density-reachable from p wrt. ϵ and *MinPts*;
3. if p is a core point, a cluster is formed;
4. if p is a border point, no points are density-reachable from p , then visit the next data point;
5. continue the process until all points have been processed.

The main principle is therefore that a **cluster** is defined as a maximum set of points related to density. It is separated by low density regions (which represent noise).
DBSCAN discovers groups of arbitrary forms (spherical, elongated, linear). Its pseudocode is

in *Algorithm* 1.

Algorithm 1: DBSCAN

Procedure: *DBscan(dataset D, ϵ, MinPts)*
$C = [-]$
foreach *unvisited point P in dataset D* **do**
  mark P as visited
  $N_\epsilon(p)$=getNeighbors(P,ϵ)
  if *size($N_\epsilon(p)$) $<$ MinPts:* **then**
    mark P as NOISE
  else
    C=next cluster
    Expand_cluster(P, $N_\epsilon(p)$, C, ϵ, MinPts)
  end
end

Procedure:*Expand_cluster(P, $N_\epsilon(p)$, C, ϵ, MinPts)*
add P to cluster C **foreach** *point P' in $N_\epsilon(p)$* **do**
  if *P' is not visited* **then**
    mark P' as visited
    $N'_\epsilon(p)$=getNeighbors(P, ϵ)
    if *size_of($N'_\epsilon(p)$) $\geq$ MinPts* **then**
      $N_\epsilon(p)$=$N_\epsilon(p)$ joined with $N'_\epsilon(p)$
    end
  end
  if *P' is not yet member of any cluster* **then**
    add P' to cluster C
  end
end

Procedure: *regionQuery(P, ϵ)*
Return all points within P's ϵ-neighborhood (including P)

The two parameters can be determined by a heuristic and must be chosen according to the application environment.

Among the advantages we find that it handles noise well and is particularly suitable for spatial data sets: geomarketing, tomography, satellite images.

In the context of GPS trajectory analysis we have used clustering techniques in order to group points of different trajectories spatially close together quickly.

3.6 Apriori algorithm

The algorithm Apriori, introduced by Agrawal et al. in [1], is a popular Data Mining algorithm for Market Basket Analysis to detect which items are frequently bought together.

Market Basket Analysis is indeed a technique which aims to use information on what, when and how much you buy for building models of purchasing behavior. These are then used to establish the arrangement of the shelves, the composition of promotions, the issue of discount coupons.

The data mining technique that is correlated with the market basket analysis is the automatic generation of **association rules**. This is based on the search for recurring configurations within the data relating to *sales transactions* or *basket*. By **sales transaction** is meant the set of products (the contents of a shopping cart for a customer), accompanied by the information relating to the time and place of purchase. An **association rule** says that there is a strong correlation between the purchase of two or more products.

Association rules can also be applied outside of this context, when it is possible to identify a transaction or "basket": it represents the choice of different elements by a "user"; in our case a user saves his trajectories, which contain his positions in the geographic network, recorded as time changes.

Regarding the Apriori algorithm, we have to define the concept of *itemset* and its *support*. A set of unique items is usually referred as **itemset**, and an itemset containing K items is called **K-itemset** or *K-tupla*. An itemset satisfies **minimum support** (min_sup), if the occurrence frequency of the itemset is greater than or equal to min_sup. If an itemset has minimum support, then it's called **frequent itemset**.

Apriori consists of a "Join" function, which generates a new combination of elements (the candidates) and a "Prune" function in which combinations of elements that do not satisfy the minimum support are eliminated. Its pseudocode is presented in Algorithm 2.

In this iterative search algorithm the set of elements of cardinality K is used to analyze the set of elements of cardinality at $K + 1$. The algorithm is based on a simple principle: if a set of items is frequent, all subsets of this set are also frequent.

The advantage is that at each Apriori level, the algorithm checks whether a tupla is frequent only if it is made up of other frequent tuples with smaller support. This search space reduction strategy depends on whether the support of an itemset never exceeds the support of its subsets. This property is known as *antimonotonicity* of the support measure.

In fact, a "small" itemset occurs in all transactions where there is a larger "itemset", plus possibly other transactions. For example if $\{A, B, C\}$ is a frequent itemset, a transaction containing $\{A, B, C\}$ must clearly also contain its subsets $\{A, B\}$, $\{A, C\}$, $\{B, C\}$, $\{A\}$, $\{B\}$ and $\{C\}$. Each subset occurs in the transaction database at least as much as the item $\{A, B, C\}$ and consequently it is itself a frequent itemset. Conversely, if an itemset like $\{A, B\}$ is not frequent, then none of its own subsets is frequent.

The "Join function", procedure for generating candidates (*Apriori_gen* in Algorithm 2), takes as input the set of $(K-1)$-itemset frequent and returns a subset of all frequent K-itemsets. The purpose of the procedure is to return the fewest candidates, so you have to do the calculation of support on as few candidates as possible.

By discretizing the data through a grid, adapting the roles of itemsets and baskets of this algorithm to our context, we find with Apriori algorithm the longest and closest sub-trajectories contained in the cells of the grid and shared by a minimum number of users.

Algorithm 2: Apriori

Input: D' : transaction database; min_sup: the minimum support threshold for every itemset.
Output: frequent itemsets of D'
Procedure: *Apriori*(D')
L_1=find_frequent_1-itemsets(D')
for $(k = 2;\ L_{k-1} \neq \emptyset;\ k++)$ **do**
    C_k=Apriori_gen(L_{k-1})
    foreach *transaction* $t \in D'$ **do**
        $C_t = \text{subset}(C_k, t)$
        // get the subsets of t that are candidates
        foreach *candidate* $c \in C_t$ **do**
            $c.count++$
        end
    end
    $L_k = \{c \in C_k |\ \text{c.count} \geq min\_sup\}$
end
return $E = \bigcup_k L_k$

Procedure: *Apriori_gen*(L_{k-1}: frequent($k-1$)-itemsets)
foreach *itemset* $\mathcal{L} \in L_{k-1}$ **do**
    foreach *itemset* $\mathcal{N} \in L_{k-1}$ **do**
        if $(\mathcal{L}[1] = \mathcal{N}[1]) \cap (\mathcal{L}[2] = \mathcal{N}[2]) \cdots \cap (\mathcal{L}[k-2] = \mathcal{N}[k-2])$ $\cap(\mathcal{L}[k-1] < \mathcal{N}[k-1])$ **then**
            $c = \mathcal{L} \times \mathcal{N}$
            // join step: generate candidates
            if *has_infrequent_subset*(c, l_{k-1}) **then**
                delete c
                // prune step: remove unfruitful candidate
            else
                add c to C_k
            end
        end
    end
end
return C_k

Procedure: *has_infrequent_subset*(c : candidate $k-$itemset;
L_{k-1} : frequent($k-1$)-itemsets)
foreach *($k-1$)-subset s of c* **do**
    if $s \notin L_{k-1}$ **then**
        return TRUE
    else
        return FALSE
    end
end

3.7 Sentiment Analysis

Sentiment Analysis (SA) also called Opinion Mining is a field of Natural Language Processing (also studied in Data Mining) that analyzes people's opinion, extracting assessments and emotions from their written language referring to entities such as services, products, events, attractions.

The term *Sentiment Analysis* first appeared in Nasukawa and Yi, 2003 [71]. With the growth of social networks and the amount of texts such as reviews, forums, discussions, the expansion of this research area and the great application of the SA in many social, economic, political and business areas has led. For example, consumers want to know other users' reviews about a product before buying it and voters want to know about a political figure before making a voting decision for the election.

This volume of text opinions is not easily decipherable, especially for long texts, and a human reader has difficulty identifying the most relevant information. Furthermore, humans do not consider clear criteria for interpreting and classifying the sentiment of a text: it is estimated that different individuals agree only about 60-65% of the time when evaluating sentiment for a particular text: therefore an automated SA system is necessary.

In addition, sentiment analysis can be used to identify critical information during specific scenarios in real time. Therefore, the advantages of SA can be summarized in Scalability, Coherent Criteria and Real-time Analysis. Moreover, to identify the opinion, the SA systems classify the attributes of the expression as *Polarity*, that is if an opinion is positive or negative and the *Object*, that is what we are talking about, the intensity of that opinion and the relevance of the object of analysis with respect to the context.

There are different types of sentiment analysis and the SA tools are mainly distinguished in systems that focus on polarity (positive, negative, neutral); in systems that detect feelings and emotions (angry, happy, sad, etc.) and those that identify intentions (e.g. not interested) [51],[60]. If it is important to accurately distinguish the level of polarity of the opinion, we consider the analysis of the fine-grained sentiment, which considers the following categories:

- Very positive
- Positive
- Neutral
- Negative
- Very negative.

It could be mapped to a 5-star rating, for example: Very positive = 5 stars and Very negative = 1 star.

Sentiment analysis is studied at three levels:

- Document level: in which the positive or negative opinion of the entire document concerning a single entity is classified, but it is not applicable to documents that express comparisons and assessments on multiple entities.

- Sentence level: in which the positive, negative or neutral opinion of each sentence is assessed.

- Entity and aspect: o level of functionality, which is made up of a feeling (negative or positive) and a target (of opinion). For example, in the sentence “The call quality of the mobile X is good, but the battery life is short”, two different aspects of the same mobile phone with different feelings (positive and negative) are emphasized.

Pre-processing of the text is necessary as the texts often contain grammatical errors or the same word can be written in different ways or abbreviated by different individuals. If you want to compare opinions relating to the same entity through multiple comments written in different languages, the procedure usually used is to automatically detect the language in the texts, then form a customized model for a chosen language (e.g. English) and finally execute analysis.

Sentiment words, also called opinion words, make up the Sentiment Lexicon or Opinion Lexicon. Over the years, some researchers have designed algorithms to compile these lexicons, even if problems are found that are difficult to manage in some cases, such as for sentences that contain sentiment words but do not express any opinion, sentences contain sarcasm, sentences that do not contain sentimental words but may imply opinions; words that in one domain can have a positive meaning but in another negative and finally the presence of opinion spamming used for example to promote or discredit products on behalf of a company by posting false opinions.

Short documents are easier to analyze, such as Twitter posts that contain internet jargons and emoticons and that have a limit length of 140 characters, so they are directed to the point.

An opinion can therefore be defined as a pair (g, s, t) where g is a target, that is, any entity or aspect of it on which an opinion can be expressed; s is a sentiment on the target, which can be positive, negative or neutral (sentiment orientations or polarity) or a numerical score that expresses the intensity of the sentiment and t is the time in which the opinion is expressed.

Since even a single sentence can express more than one sentiment, the algorithms, belonging to the so-called knowledge-based techniques, analyze an opinion of a text add up the sentiment scores of its various terms considering negatives and intensifiers. If the final score is less than 0, the review is negative, otherwise positive. During the classification of sentiment, the opposite words present and the words of negation or adverbs are taken into account as almost and only, which change the orientation of the sentiment.

Monitoring the opinion to vary the time of a given entity can help make improvements on it and understand which aspects need to be changed, and also check through the users’feelings if these changes have given effect.

In the multi-agent architecture proposed by us in this work, we will use sentiment analysis to analyze the comment that a user enters on each place visited (through his agent) in order to assign an interest score and direct subsequent visit options based on this.

Chapter 4

Problem Analysis

In this chapter we describe the large-scale datasets used in our experiments, which show the feasibility of our method. They have various characteristics, because they are detected by different types of devices and users (tourists, students, commuter workers). The recorded tracks are real GPS trajectories, which we preferred over CDR raw data because of finer grain (compared to the latter). Then, using this data as a basis for our analysis, we explain the method used. The problem we faced was to find SPs and POIs from big geo-spatial data and, starting from these, to propose a multi-agent system that could satisfy the needs of users to reach places of interest. The real-time data, collected by the architecture in action, is added to those previously saved for a more realistic and accurate result.

In the same way we start from the tracks recorded by the taxis, relating to Cabspotting data, to show how the forecast of overcrowded places and the alert to users takes place, applying some tools used by market basket analysis.

4.1 Proposed Datasets

The GeoLife dataset presented in Section 4.1.1 is used for the extraction of Points Of Interest and the identification of flows of people, also through a more efficient strategy for finding corridors.

Taxis and trucks of the next two sections confirm the usefulness of the approach presented for the SPs and flows and the veracity of the results found, as the POIs are verified by the nature of the dataset.

The flow detection algorithm is performed on the Cabspostting data (dataset described in Section 4.1.4) and we used it mainly as a basis for the statistics for the co-occurrence of highly frequented areas. With this latest test, we show how to avoid crowding by predicting the massive movement of people to stationery areas.

4.1.1 GeoLife

The GeoLife dataset was collected for a Microsoft Research Asia project for more than four years (from April 2007 to October 2011) and is freely downloadable from their website [42]. It consists of a set of GPS trajectories, ordered sequences of timestamped points with latitude and longitude, recorded by 182 users during their daily movements.

The 17,621 total trajectories were recorded by cell phones or other portables GPS devices, for a total distance of 1,292,951 km with a total duration of 50,176 hours. Most of the trajectories were logged in a dense representation, e.g. every 3 seconds or every $5 \sim 10$ meters per point.

73 of the users participating in the GeoLife project had labeled their trajectories with the mode of transport: bus, taxi, car, foot, bike, subway, train, plane, boat, race and motorbike. The registered routes cover more than 30 cities in China, but also some cities of Europe and USA, but most are distributed over Beijing.

This dataset collects users'outdoor movements, such as home-to-work life routines or trips for entertainment and sports activities (shopping, sightseeing, restaurants, hiking and cycling).

4.1.2 Taxi

Taxi trajectory data is the GPS trajectories collected by taxis equipped with GPS sensors in Beijing during a period from October 30, 2010 to November 30, 2010. The sampling rate was one point per minute. Currently, a small part of Taxi (101 trajectories) are available [91]. The sampling rate was one point per 60 s, and a trajectory has 39,100 data points on average. Each sample records detailed information of taxis including taxi ID, location, time stamp, occupancy indicator, orientation and speed. The time at which each GPS point was recorded appears as a timestamp in Unix time format. Unix time (also known as Epoch time) is a system for describing a point in time. This is the number of seconds that have passed since the Unix era, which is 00:00:00 UTC on January 1, 1970, minus leap seconds.

The dataset has good temporal and spatial coverage, including weekdays, weekends and public holidays of all the main urban areas in Beijing (longitude and latitude in [115.421387; 39.437614] x [117.321785; 40.609333].

4.1.3 Truck

Truck trajectory data [91], referred to as Truck, is the GPS trajectories collected by trucks equipped with GPS sensors in China during a period from August 2015 to October 2015. The sampling rate varied from 1 s to 60 s. Currently, a small part of Truck (101 trajectories) are available, and they mostly have around 50 to 90 thousand data points. Every GPS point is represented by latitude, longitude and timestamp in Unix time format.

The space covered by the registered paths is the large area [86.882817; 0.230753] x [172.467424; 43.405276].

4.1.4 Cabspottingdata

This dataset [74] includes the trajectories collected in May 2008 by 536 taxis, for a total of 11,219,424 GPS points. Mobility traces are provided by the Exploratorium - the museum of science, art and human perception through the cabspotting project (http://cabspotting.org)

To gather data each vehicle was outfitted with a GPS tracking device that was used to efficiently reach customers. The data were transmitted from each taxi's cab to a central receiving station, and then delivered in real-time to dispatch computers via a central server.

For each mobility trace file, associated to a taxi ID, contains in each line: (latitude, longitude, occupation, timestamp). Where latitude and longitude are in decimal degrees, the occupation indicates whether a taxi has a passenger (1 = busy, 0 =free) and the time is in the UNIX era format.

The area covered by these routes corresponds to the county of San Francisco of USA and its surroundings in California, with minimum and maximum longitude and latitude of the range $[-127.08143; 32.8697] \times [-115.56218; 50.30546]$. The total size of the trajectories registered with a customer on the taxi consists of $5,017,659$ points.

4.2 Proposed Approach

When an individual travels to an unknown city, he looks for information about the most interesting sites, and the ways to follow to connect them. A simple map could confuse him, therefore a user through his mobile device searches very often for updated information.

This is the motivation of our work regarding the extraction of GPS points and user customizable routes.

In this section, attention is given to finding significant user locations, this analysis is possible thanks to the growing availability of a large amount of data regarding individual trajectories.

To achieve this, the data to be analyzed mainly comes from cellular or mobile GPS devices, which allow you to keep a track of the history of positions by GPS paths. The first tests were carried out on the GeoLife dataset provided by Microsoft Research Asia.

GPS points provides fine-grained information on the spatio-temporal trajectory with respect to, for example, CDR (Call Detail Record) data, but also they are more difficult to manage. In fact, given the huge amount of these data to be analyzed, it is not easy to process them through algorithms with acceptable times.

For this research we have chosen the range of longitude and latitude of $[116.1; 39.7]$ x $[116.7; 40.13]$ (Beijing metropolitan area). In this area, of 51 per 48 km, there are $18,401,631$ GPS points.

By plotting the trajectories we can see a second problem: some paths appear broken / not continuous probably due to the presence of buildings or tunnels that disturb the GPS signal. Therefore in some areas the GPS device did not register, but we have decided not to tackle this problem.

Before analyzing the GeoLife paths, we performed a cleaning in order to remove some of its inconsistent data because these may influence the study results. For each trajectory, after having computed the instantaneous speed of each point with the consecutive one, we considered the GPS points with a speed between 0 to 100 m/s.

After having computed this, there were some undefined speeds, because the time was zero probably due to the GPS device stops working properly not recording the elapsed time: with the speed filter we eliminated the points with these anomalies. The total number of recordings

Table 4.1: Information about different time slots of GeoLife dataset.

Time Slot	**Total number of**		
	GPS Points	***Trajectories***	***Users***
1	$3,978,234$	$5,878$	156
2	$3,729,429$	$4,302$	150
3	$4,976,744$	$6,613$	166
4	$3,107,232$	$4,702$	168
5	$889,076$	$1,505$	114
6	$1,341,196$	$2,537$	129

we have worked on was $17,976,308$ points, therefore from the area considered we have deleted $425,323$ points.

GeoLife dataset was published in .plt, a vector graphic format, converted by us in .csv files to be processed.

The first step of this work was to build a rectangular grid of dimensions compatible with the maximum and minimum latitudes and longitudes previously chosen: in order to have an accurate statistic we decided to divide our rectangular grid into intervals of 100, 200, 400 and 800 meters.

Since, approximately, for the whole area under examination 100 meters of distance between two points at the same longitude correspond to a difference of latitude of 0.0009 degrees (direction 90 degrees), while for points on the grid with same latitude 100 meters are about 0.0012 degrees of longitude difference (direction 180 degrees), we can give the appropriate divisions of the grids.

Then, we analyzed how many points of different paths fall into our boxes transforming our grid into a matrix M, where every element of M contains the total occurrences of GPS points that fall in that square ($100 * 100$ meters for example) and the position of the element $m_{i,j}$ corresponds to a certain square of the created grid.

Through an histogram we can evaluate which subsets of area the greatest traffic of users. The contour plot (graph of the isolines) of the matrix M allows you to immediately go back to the main "hotspots", that are the areas more visited.

From Figures 4.1 and 4.2 we can see some hotspots, around the Tsinghua University, at Yuanmingyuan Park, the Hepingli Residential District and Peking University.

We divided the trajectories into 6 time slots from 4 hours each Slot1 = [00:00:00, 03:59:59], Slot2 = [04:00:00, 07:59:59] and so on, in order to analyze the traffic in Beijing during different time slots (Figures 4.3, 4.4, 4.5, 4.6, 4.7, 4.8). More details are visible in Table 4.1.

An extensive series of experiments in the other two datasets, Taxi and Truck, has been performed in order to study the movements of users in different situations. We mined shared routes in real-word and not only drop-off or pick up taxi. The heatmap of Taxi trajectories is visible in Figure 4.9.

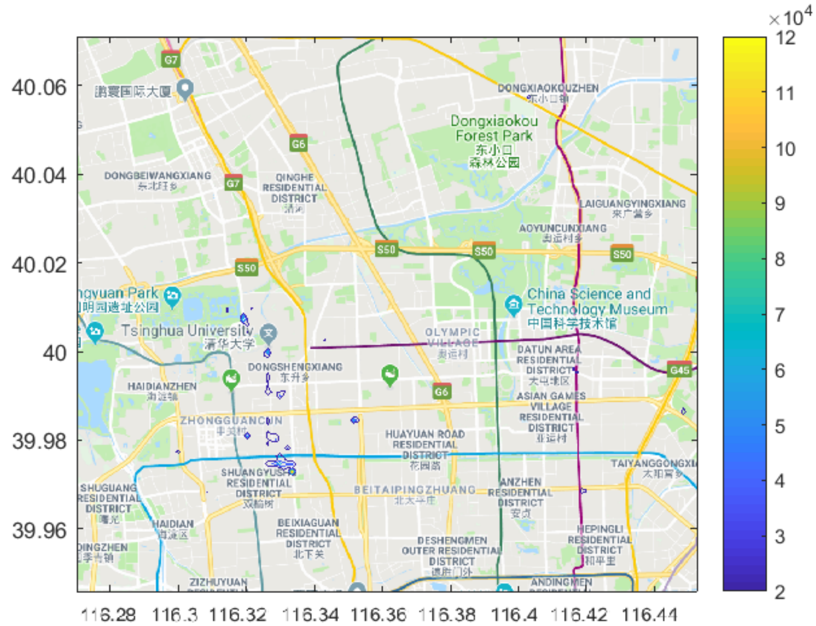

Figure 4.1: Contour plot created with grid with intervals of 100 meters, zoom on map.

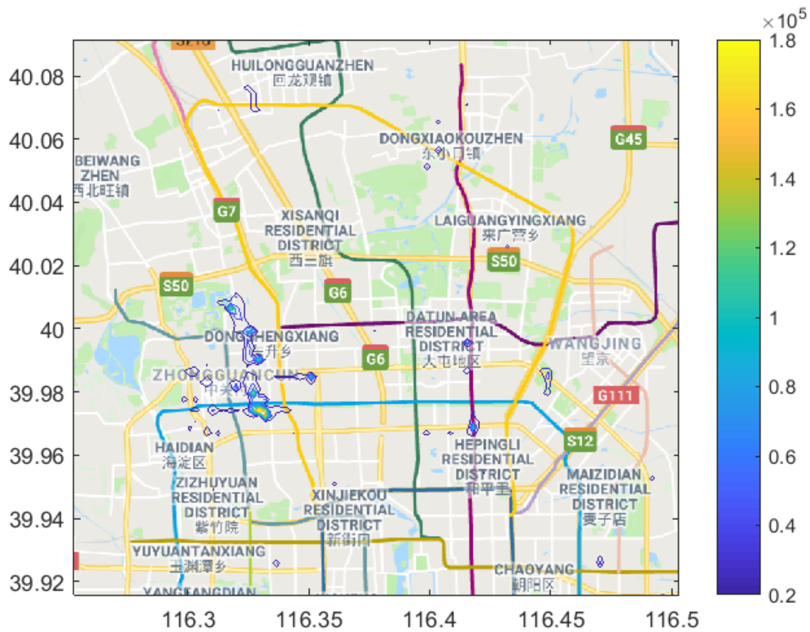

Figure 4.2: Contour plot created with grid with intervals of 200 meters, zoom on map.

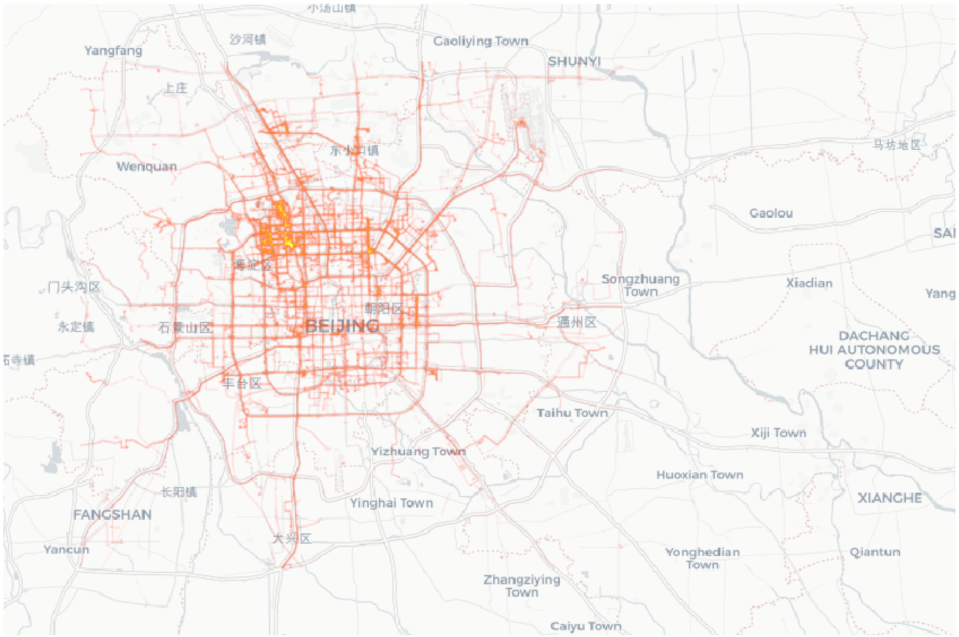

Figure 4.3: Trajectories in Beijing for time slot 1: [00:00:00, 03:59:59].

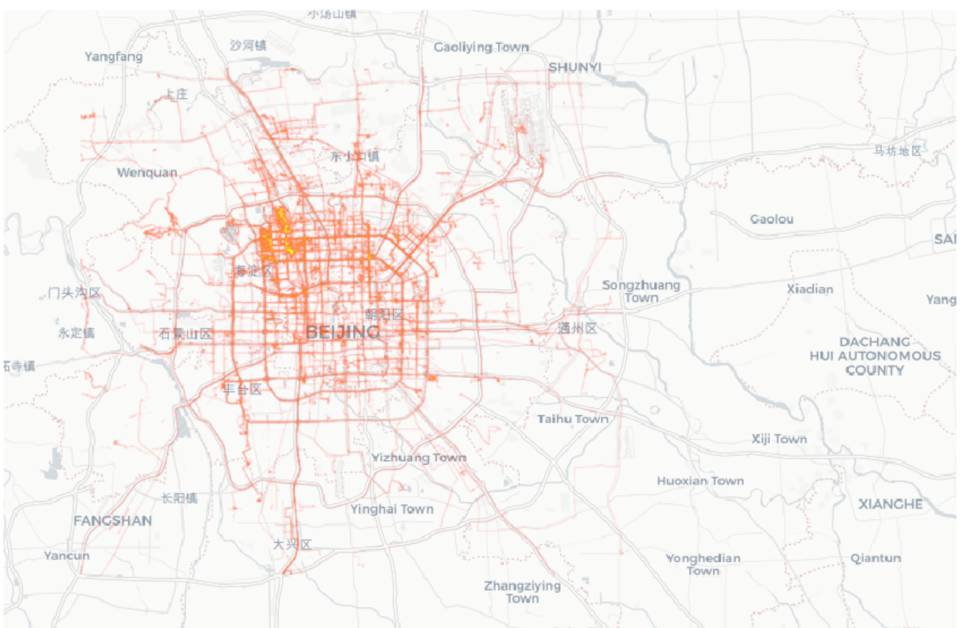

Figure 4.4: Trajectories in Beijing for time slot 2: [04:00:00, 07:59:59].

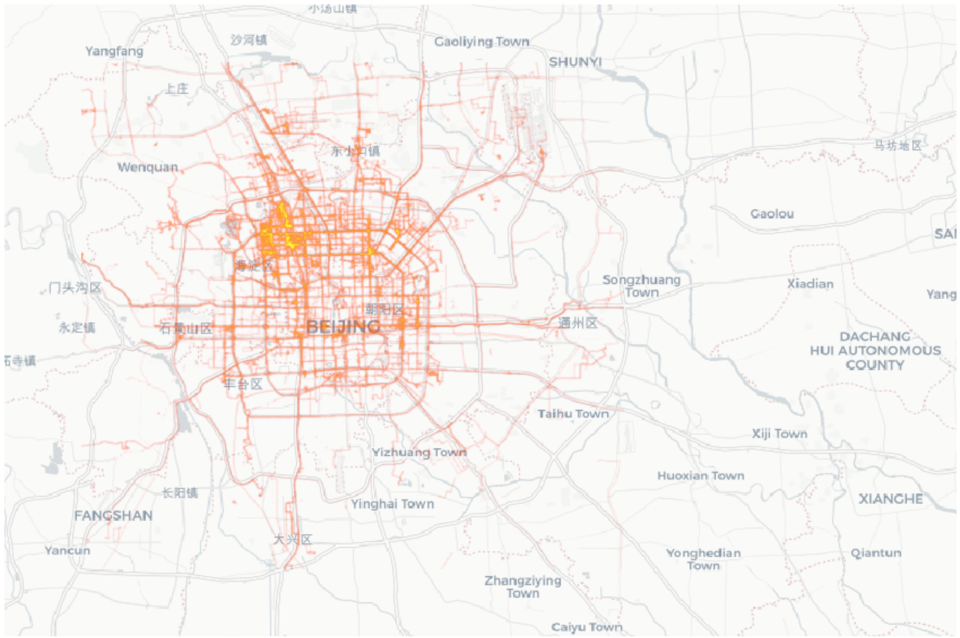

Figure 4.5: Trajectories in Beijing for time slot 3: [08:00:00, 11:59:59].

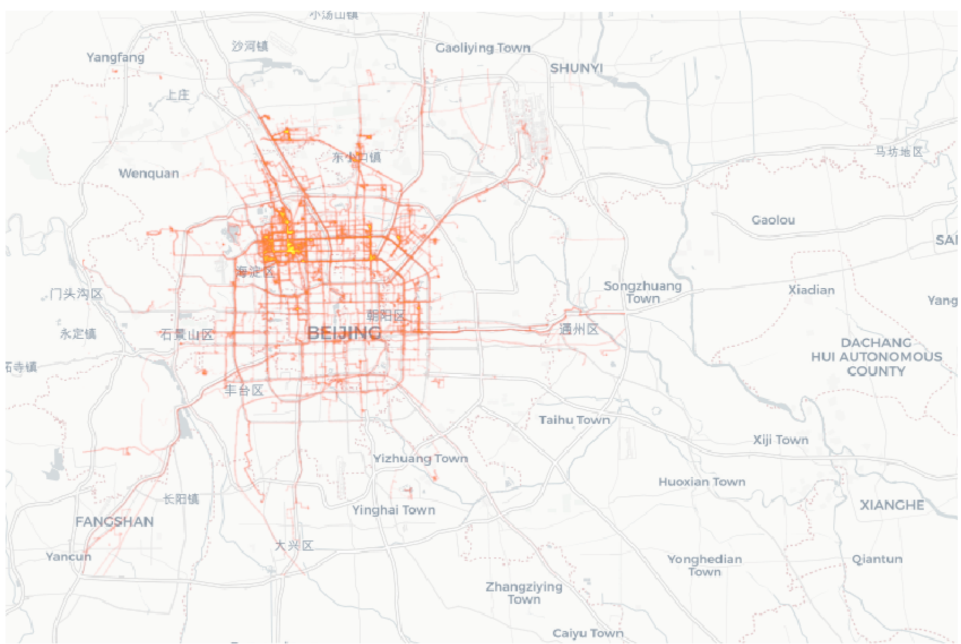

Figure 4.6: Trajectories in Beijing for time slot 4: [12:00:00, 15:59:59].

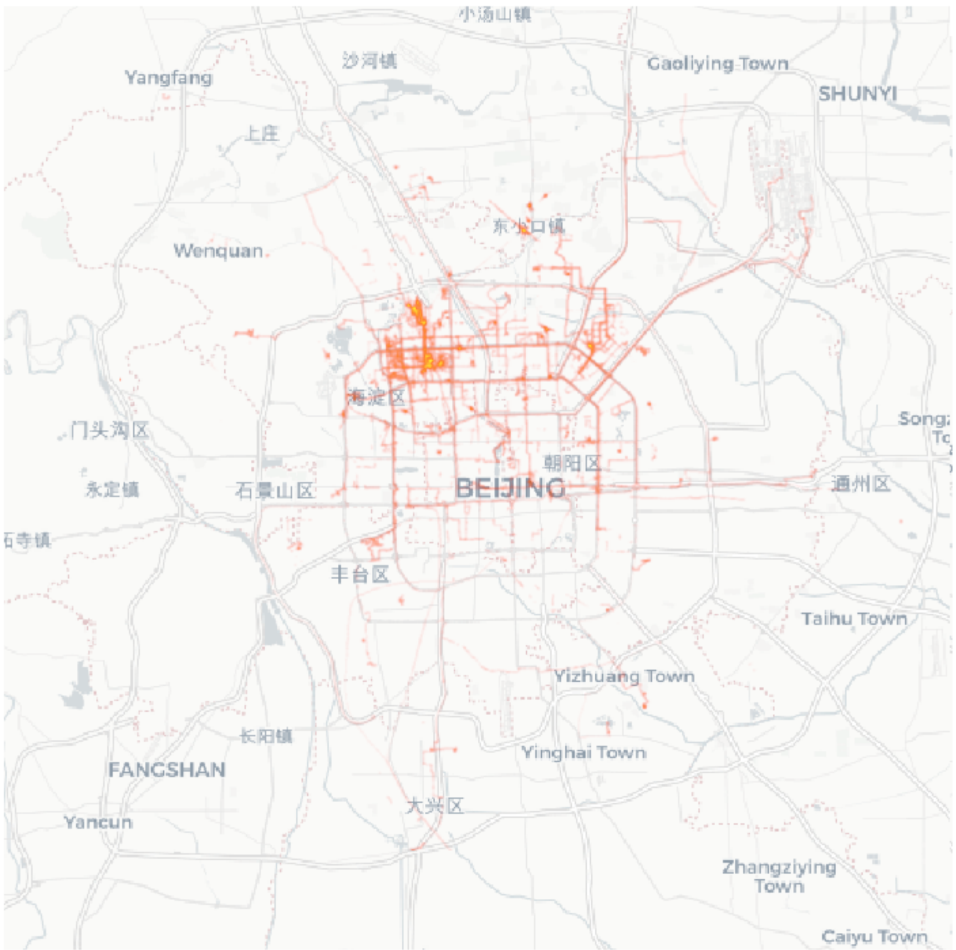

Figure 4.7: Trajectories in Beijing for time slot 5: [16:00:00, 19:59:59].

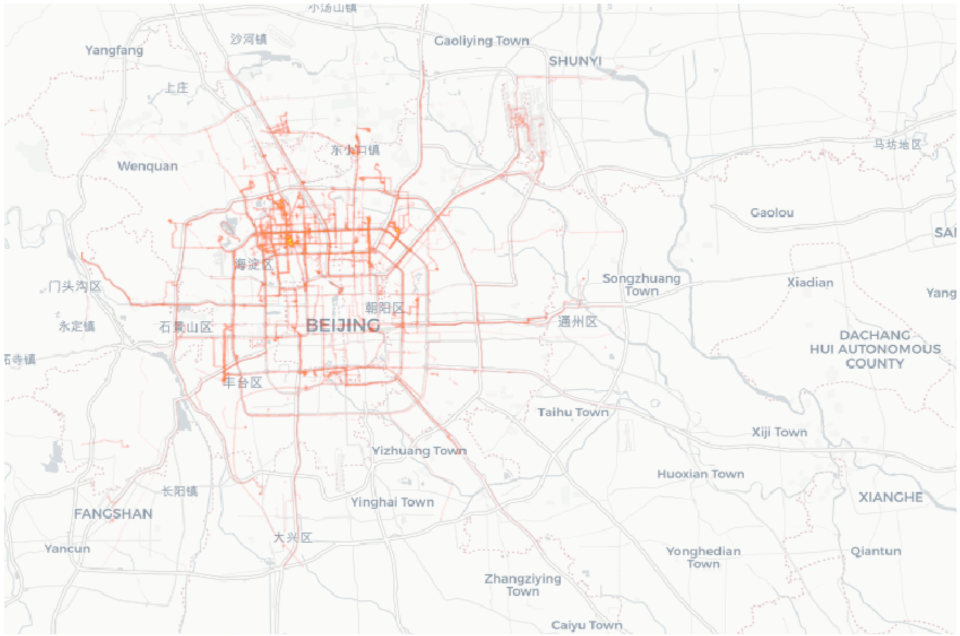

Figure 4.8: Trajectories in Beijing for time slot 6: [20:00:00, 23:59:59].

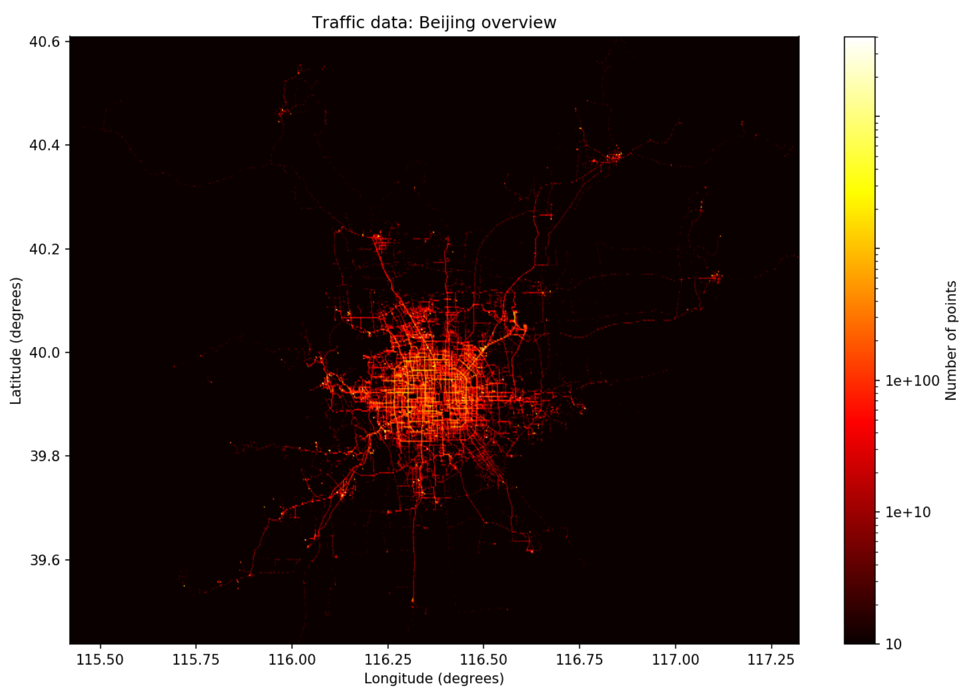

Figure 4.9: Heatmap of trajectories in Taxi dataset.

4.2.1 StayPoints extraction

For every slot of time in GeoLife dataset, after grouping trajectories by users, the second step of our work was the StayPoints (SPs) detection [73]. When we find a region in which a user has spent a considerable time on its surroundings, the centroid (the mean of coordinates of the points belong to it) of this cluster represents an SP. The algorithm that we implemented for the SP detection (pseudocode in Algorithm 3) needs as input a time threshold ($TimeThr$) and a distance threshold ($DistThr$). For example, if an individual stays over 20 minutes ($TimeThr$) within a distance of 200 meters ($DistThr$), a SP is detected.

The Algorithm 3 analyses each trajectory individually, scanning its points in the order of registration. For each, considering the pairs between it and its successors, calculate the spatial distance between the two points p_i; and p_j ; and the time span between them. All points are located in an area enclosed by the $DistThr$ distance threshold and for which the time span is greater than $TimeThr$ form a SP region. The average of their coordinates determines the centre of the area, and represents a StayPoint (SP) of the analysed trajectory. This captures the behaviour of the user, who is probably going around some building for a while. We want to study mainly the parts of the path in which he dwells for some reason in a spatial environment. We also extract the *arrival time* information for it, i.e. the time relative to point p_i; and the *leaving time*, i.e. the time relative to point p_j.

With reference to the log routes, each SP has a particular semantic meaning such as the restaurant where we go and the places we visit, etc. Figure 4.10 shows two categories of StayPoints. The first situation, such as SP 1, occurs when an individual remains stationary in a single point for a period of time greater than a threshold. In most cases, this state occurs when people enter a building and the satellite signal is lost for a certain amount of time until arrival of the outdoor signal again.

In the other situation, like SP 2, a user wanders within a given spatial region for a period of time greater than the threshold considered. Consequently, we have to calculate the average coordinates of the GPS points of that spatial region and the final SP point is represented by its arrival time and its left time. In most cases cases, this is a situation arises when people travel and are attracted by the surrounding outdoors.

If we group through clustering some significant places (SPs) by setting a minimum number of points necessary to form the cluster sufficiently, the few GPS points will be excluded like point 1. Therefore, if they are not surrounded in the vicinity by other significant places, the density of the points recorded there cannot satisfy the condition for the formulation of a cluster and instead the SPs "type 2" will be extracted.

4.2.2 Solution for Points Of Interest detection

Points of Interest, commonly abbreviated POI, are a well-known concept in literature [4], [52], [102]. A POI is defined as an object associated with a latitude and a longitude which at least one person would reasonably be expected to have an interest or an utility. POI recommendation suggests places to visit, which are taken from an automatic analysis of the three real dataset:

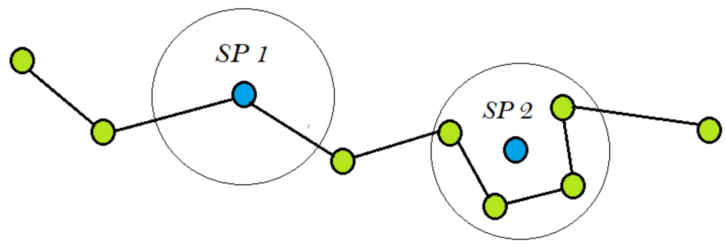

Figure 4.10: Example of the two types of StayPoint.

Algorithm 3: Pseudocode for SPs detection

```
Input A trajectory T={p_i,p_{i+1},...}, a distance threshold (DistThr) and time span
  threshold (TimeThr)
Output A set of StayPoints SP ={S}
i=0, cardinality_traj= |T|
while i < cardinality_traj do
    j = i + 1;
    while j < cardinality_traj do
        dist=distance ( p_i, p_j );
        if dist > DistThr then
            δtime=time_p_j−time_p_i;
            if δtime >TimeThr then
                S.coords=MeanCoords({p_k|i < k < j});
                S.arrive_time=time_p_i;
                S.left_time=time_p_j;
                SP.append(S);
                break;
        j = j + 1;
    i = j;
return SP
```

Taxi trajectories data, GeoLife and Truck trajectories data.

After the SP detection, we focused on POIs that cluster together SPs of different users (at least 10), and checked if, in different time slots, the users that previously had a common POI move together to another one. We applied DBSCAN, to the SPs obtained, as it works well with large geographical dataset and likewise it can be adapted for any distance function.

For our data, the clustering algorithm DBSCAN has determined clusters for all SPs. For example in Geolife dataset we set MinPts equal to 10 or 15 and ϵ from a minimum of 200 meters to a maximum of 400 meters.

Geospatial clustering must depends on geographic information domain knowledge and the context of the users. For example 200 meters was chosen as a spatial parameter because it is the average space between an intersection or a square around an attraction.

Both parameters control the local neighbourhood of the points: making better use of geospatial and clustering knowledge to select suitable constraints and parameters is likely to yield better and more meaningful clusters.

For example, applying another partition clustering method such as K-Means to the whole GPS Geolife dataset, we obtain a result like Figure 4.12. The K spatial partitions obtained will depend on the number K chosen and we can rely on the Calinski-Harabasz index for this. As shown in Figure 4.11 on the left, this criterion suggests us to use the highest value of the plot, i.e. $K = 9$. This value was confirmed as optimal for the division of different zones also by the Elbow method, as indicated on the right of Figure 4.11; the execution time for this clustering algorithm on our data was 1 hour and 27 minutes.

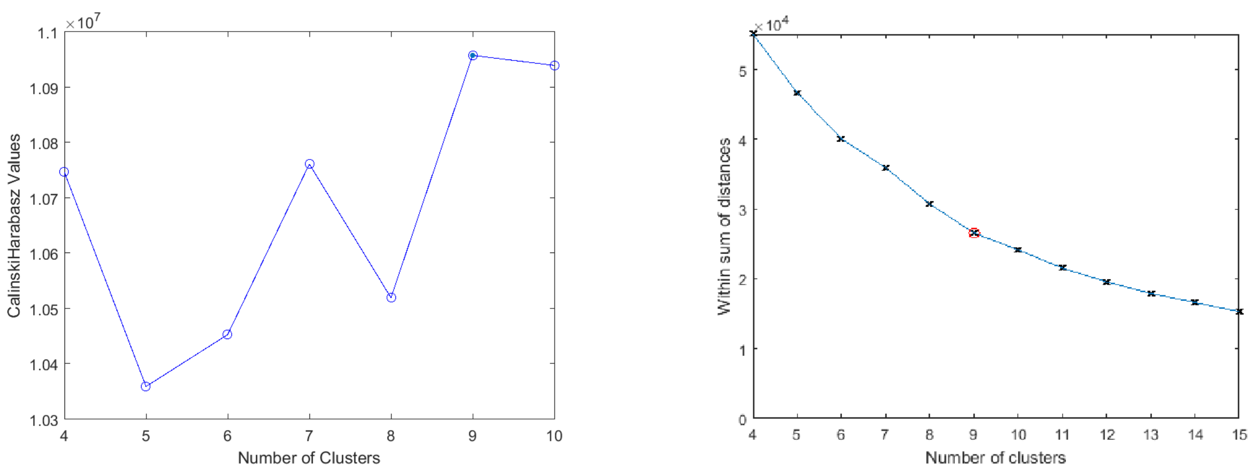

Figure 4.11: For Caliski-Harabasz criterion (shown with the plot on the left) the optimal number of clusters for the set of GeoLife GPS points is 9.
The Elbow method is a heuristic used in determining the number of clusters in a dataset. The method consists of plotting the explained variation as a function of the number of clusters, and picking the elbow of the curve (on the right) as the number of clusters to use, in this case $K = 9$.

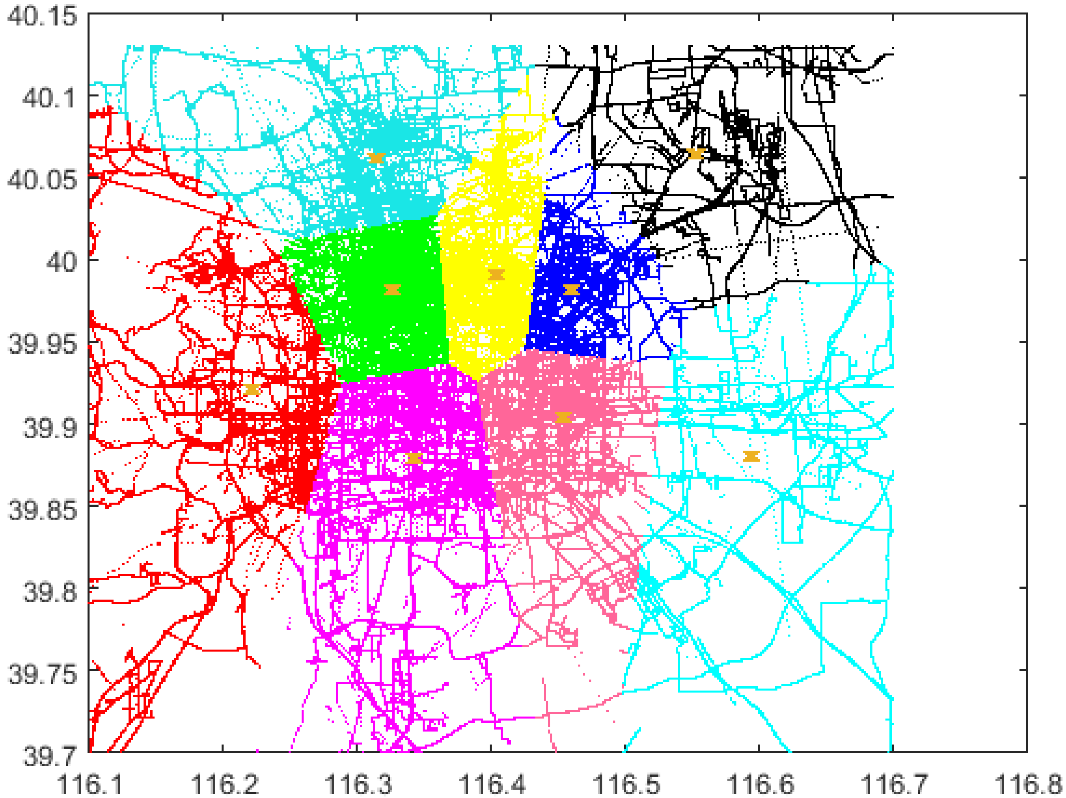

Figure 4.12: Cluster assignments and centroids, K-means with $K = 9$.

So, we decided to cluster not directly on the total GPS points because it is computationally expensive and we also want to consider only those of greater significance in the trajectories, therefore we apply the DBSCAN algorithm based on density only on the StayPoints.

The points found by our analysis were verified by matching the results with Google Maps data. It was confirmed that they correspond to real POIs, i.e. parks, restaurants, etc., hence validating our approach.

4.2.3 POIs Recommendation (proposed Multi-Agent system)

In our proposed software architecture a centralized server collects and analyzes data from different agents, with the aim of offering suggestions to users and some real-time data on visiting POIs. In addition, each user (with reference to an application) was modeled as an independent agent who communicates with the centralized server. In this approach, an agent is an application installed on an Android device that tracks the user movements around Points Of Interest, in order to suggest new places to visit. Therefore, our recommendation on POIs is based on a multi-agent system that performs the following steps:

STEP 1: the server sends to agents the list of known POIs for a city. Each POI was previously determined by the DBSCAN algorithm discussed in Section 3.5.2. Each POI has the following information: the most visited time slots; the number of agents present on the site (POI) in real time; a series of feedback on the site, created by a Sentiment Analysis algorithm that analyzes the comments released by users; a rating estimated from previous information that recommends (or dismiss) the POI to the user. The union of this information is used to drive the recommendation system.

STEP 2: thanks to the evaluation, the agent chooses a place of interest that the user can visit. As soon as the GPS coordinates are within a radius of less than one kilometer from the coordinates of a POI, then the location will be sent to the centralized server, which can determine the number of users present. GPS coordinates are sent only in controlled areas to preserve user privacy.

STEP 3: once the visit is over, the user can use the agent to comment on the place visited. This information will be sent to the server, and analyzed there using Sentiment Analysis algorithms [61], which in turn allows the server to determine a score that identifies whether the POI was satisfactory for the user.

For each POI visited, the data forwarded to the server are: the inaccurate GPS position of the device, visiting hours (through a timestamp), the name of a place, an evaluation score of the place (i.e. a number from 0 to 5). To forward such data, the user needs to authorise the application to share his GPS position.

STEP 4: the agent will receive the list of POIs again adding information for a specific POI related to the site already visited based on previous users experiences. One of the aspects considered is the sequentially: we take into account not only the places visited, but also the sequence of them. Therefore if a group of users who are considered similar to Claire have visited POIs A-B-C-D and she is on C, then the system will advise her to visit D or B.

Finally, these points are sorted according to the user current distances. E.g. if there is a POI near her in which the number of people is adequate (i.e. according to the preferences of the user), the time slot matches the current time, and users have appreciated the visit (having assigned a positive score), then such a POI is suggested to the user by alerting her agent. Otherwise, if the place has not been appreciated by other users or the current time is not within the time slot referred by available data, then the place is labelled as not recommendable.

Agents accept suggestions from the server and filter them, since the exact location of the user is known only to the local agent. The suggested points, also given to the experiences acquired by other agents, are sorted based on the user's current location and filtered for the time slot corresponding to the current time.

Moreover, points not yet visited by the user are marked. As for the time slots, the hours in a day were organised into six 4−hour slots; Slot1 = [00:00:00, 03:59:59], Slot2 = [04:00:00, 07:59:59], Slot3 = [08:00:00, 11:59:59], etc. Therefore, data gathered for a POI are associated with a time slot based on the start time of the visit. E.g., if the visit starts at 10 a.m. this point will be labeled in the 4th time slot.

Such cooperation can generate benefits for groups of people who share the same interests (tourists, students, etc.). Thanks to the exchange of information between agents and servers, it is possible to define the rules for our POI-based system recommendation.

The most important properties of the agents in this proposed recommendation system are the following:

- Agents are independent: if an agent stops working the others continue their work without consequences.
- Agents are mobile: they are easily transportable thanks to their integration in the mobile application.
- Agents are reliable: given that the GPS coordinates are taken directly from the device, avoiding to obtain false data.
- Agents work by preserving user privacy: there is no continuous sending of the users positions, instead the server only receives data when agents are near POIs, and the coordinates sent to the server are displaced by a bounded random amount, thus preserving the users privacy. More specifically, two important steps are taken to offer greater security to users: (i) users share the location only if they are close to a well-known POI, i.e. their position falls within 300 m from the POI); (ii) the position is processed through data masking techniques, i.e. a variable and unknown displacement will be added before being sent to a central server storing it. Having altered the position does not affect the recommendation system and guarantees better protection for the user.

In this context, security and robustness are features embedded in all the agents, working together by means of a centralised server. Thus, thanks to a system based on multi-agents, and a centralised server, it is possible to create a recommendation system based on the experience and appreciation of users, able to suggest points of interests to users, while ensuring user privacy.

The agent takes some decisions on behalf of the user, i.e. the agent determines:

- when geolocation data are sent,
- which displacement is added to geolocation data to preserve user privacy,
- how to rank incoming data.

For the latter, since the agent shields the user preferences to the outside world, hence the agent is entrusted with the task of muting some alerts when the user has low interest for it, or they are not related to the actual flow of the user, and with the ranking of incoming comments and selected places of interests according to match user preferences. Each agent sends data to a centralised server, and is identified by an app id that the server has provided. User identity is preserved since: the server does not disclose such ids to the agents; agents can not directly communicate with each other; the user is not providing his personal data.

4.2.4 Privacy Preserving

In the literature, user privacy is a topic much discussed. The unconditional use of smartphones and the multiple downloaded applications put a strain on the protection of user rights. Existing systems have mainly taken three approaches to improving user privacy: (a) introducing uncertainty or error into location data [41], [69], (b) relying on trusted servers or intermediaries to apply anonymisation [55],[57], and (c) using cryptography techniques [70], [72]. However, each system has weaknesses: the first approach loses accuracy because the uncertainty rate degrades the data, the second approach can be risky, since private data is exposed to proxies that could be violated and the third approach is often computationally expensive. Hence, hybrid approaches are often a proper compromise. This work offers a hybrid system while preserving privacy in two ways:

1. by using a centralized server the agents do not directly know each other and can not exchange information (as for systems (b) above);

2. by only extracting information on POIs, and an approximate position of the user, only when he expresses an opinion on a POI (as for systems (a) above).

In addition, this approach is the first approach in the tourism sector that deals with preserving user privacy. There is no user ID, no precise position is provided to the other participants. Users interact with the system a centralized and not directly with other users. Allowing the direct exchange of information between the various devices, which are in a certain area, would make known the user position that we do not want. To ensure privacy, therefore, the details -as the precise position- are obscured, We considered more disadvantages to leaking private information than the advantages.

Data collected are processed and cleaned up in the user device, in order to respect anonymity, and then processed data are sent the server. In detail, each agent: analyses only user data located in the vicinity (about one kilometre) of known points of interest and adds a displacement to the data before making it available to the server. In this way, it preserves the user's privacy in two ways: it does not always share the user's position to the server, and when sharing the position, it is affected by a variable uncertain amount.

4.2.5 People movements identification

It is difficult to find significant user behaviors from the available data. In this section and in the following one, great attention is paid to identifying significant paths for users. The Section 4.2.7 aims to predict user behavior.

Many people flows are discovered on the datasets analyzed before and these are useful, e.g. to find common itineraries, to suggest places that can be reached by users or to present improvements on public infrastructure [22].

The proposed approach was validated by three experiments on location data, to arouse significant places, starting from the data available on: taxi, truck and people movements.

We use an alternative technique, by means of significant flows of people within the city, to identify POIs. It can detect people or taxis flow to extract POIs with a high flow of interest, starting from a series of GPS trajectories. The detailed results are presented in Sections 5.2 and 5.3.

We define two *spatially similar* trajectories if they cross the same n points (unless of a certain tolerance) in the same temporal order. The distance between two points is defined in the Haversine formula. A *flow* is a continuous and uniform movement of entities (such as people, cars or truck) in one direction and a *flow density* is the number of entities traveling through the same flow. Finally trajectories (spatially similar) are called temporal similar if entity movements occur in the same time slot. The first necessary step is the preprocessing of data, that is, each trajectory is cleaned up by removing the noise and outliers. We also cleaned the trajectory from points too close to each other, so you can work with others homogeneous trajectories. Outliers removal compares each point with the previous and next points, looking sharp speed changes. In case of abrupt changes, this point it is an abnormal label and is discarded. Once the data preprocessing phase is complete, a the algorithm that calculates all flows is executed their density, finding spatially similar trajectories. The search for comparison was done in parallel in order to speed up the procedure.

Using this approach, relevant flows are extracted in the city. Starting from these shared paths it is possible to extrapolate POIs by making a match between the POIs of the analysed city (see Section 5.3) and the points close to each detected flow (with a maximum distance of 100 m).

4.2.6 Mining corridors from GPS trajectories, through Apriori

In this part of work we performed different tests in order to detect shared routes, so we applied a grid of uniformly sized cells to discretize the trajectories. The collection of trajectories D of the original dataset is transformed into a new collection of trajectories D', where each trajectory is represented as a sequence of grid cells. If two or more consecutive coordinates are mapped into the same grid cell, we report only the first instance, not allowing a trajectory to have consecutive points of the same grid cell.

In this case, we have considered that trajectories are not characterized by traveling direction: going forward or backwards on a sequence of cells results in two instances of the same trajectory. Hence, trajectories can be seen as a path in an undirected graph.

Given this trajectory representation, we can define what is a *corridor*:

Definition 4.2.1. *A (M,S)-corridor is a sequence of M cells in D' such that all cells are shared by at least S trajectories in a given time slot.*

A first naive approach to detect corridors is given by this pseudocode:

Algorithm 4: Brute force approach to find corridors

Procedure: Find_Corridors$((T_1, T_2, \ldots, T_L), M, S)$
Input: A set T_1, T_2, $\ldots$,T_L of L trajectories formed by N cells each, M: the minimum number of cells for a corridor, S: the minimum number of trajectories for a corridor.
Output: C the set of (M, S)-corridors.

```
C = {}
for T from 1 to L − S + 1 do
    for i from 1 to N − M + 1 do
        A = (T[i], T[i + 1], ..., T[i + M − 1])
        s = 1
        for T' > T do
            for j from 1 to N − M + 1 do
                if A = (T[j], T[j + 1], ..., T[j + M − 1]) or
                A(T[j + M − 1], ..., T[j]) then
                    s = s + 1
                end
            end
        end
        if  s ≥ S then
            add s to C
        end
    end
end
```

For every trajectory, this algorithm scroll down its elements to decide whether a pair of consecutive cells are shared by at least other $S - 1$ trajectories. If this happens, it must be checked whether the subsequent cell, together with the frequent pair of cells found, constitutes a frequent triple for at least L trajectories. And so on, until you find the largest set of frequent cells.

The complexity of this algorithm is a function of L and N. The number of performed operations is $\sum_{i=1}^{L-S+1}(N - M + 1) * (L - i) * (N - M + 1) * 2$, which results in a complexity $O(N^2L^2)$.

The proposed method comprises two phases: first the generation of *candidate corridors*, which are coarse traces of several distinct trajectories, and then filtering candidates for the selection of *fine-grained corridors*.

First, we analyze the database once to generate a mapping graph on real roads and get information on trajectories. To this end, we define a grid of uniformly sized cells on the map of interest, mapping the coordinates (in decimal degrees) of the points for every trajectory to discrete grid cells. For example with square cells 1 km wide, we get a $51 * 48$ cell grid to apply to the selected dataset (see Figure 4.13).

In Figure 4.13, it is possible to notice more intense cells which indicate a greater concentration of trajectories that cross them.

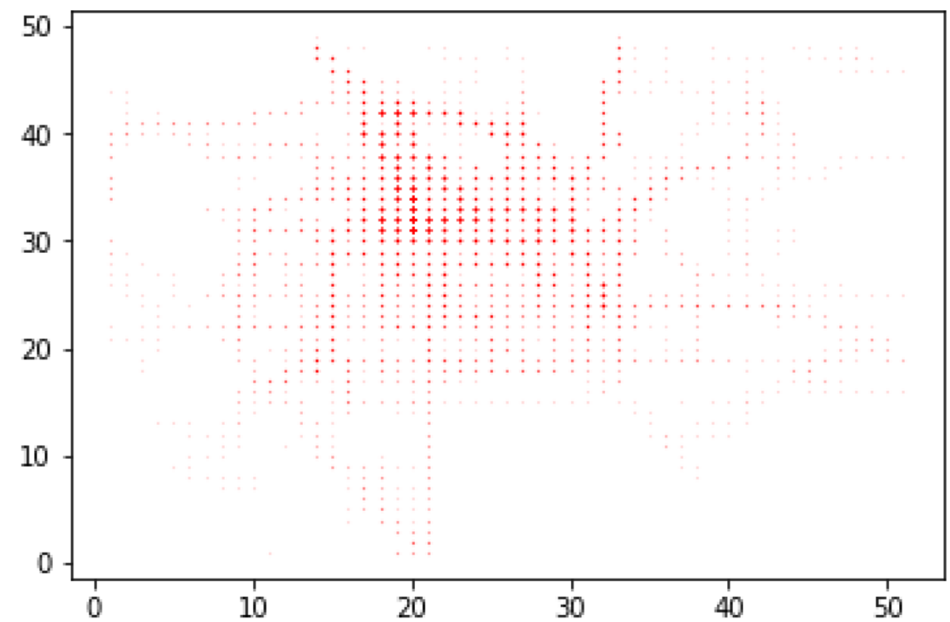

Figure 4.13: Trajectories mapping into a grid with cells of 1 km for side.

Based on this trajectory representation, we can generate the candidate corridors to obtain information on the areas that approximate the sub-routes common to several users with the Apriori algorithm.

In the second phase, this information is filtered using Radius Neighbors Graph and used to find fine-grained frequent corridors.

Apriori algorithm is currently used for example for marketing strategies in supermarket chains, so you want to find out which products are often bought together by the customers. This frequency-based approach that allows us to find sub-paths of the road network which are shared by at least a fixed number of trajectories.

In our scenario this algorithm can be used to reduce computational complexity. The key point of its application is to consider GPS points as a set of items and trajectories as a set of baskets.

Then the minimum support to be set as input in the algorithm corresponds to the minimum number of trajectories that must have in common tuple of cells. The application of the Apriori algorithm in the case under consideration can be improved by setting constraints: for example, by considering that K-tuples can only be formed by groups of adjacent cells. Extending the search for other frequent cells to be aggregated only to the neighborhood located at the edge of these cells, since a cell can have at most eight adjacent cells, would lead to the reduction of the execution time in the generation of the candidate cells.

We propose three different strategies for the application of Apriori (Step 1, Step 1.2, Step 2

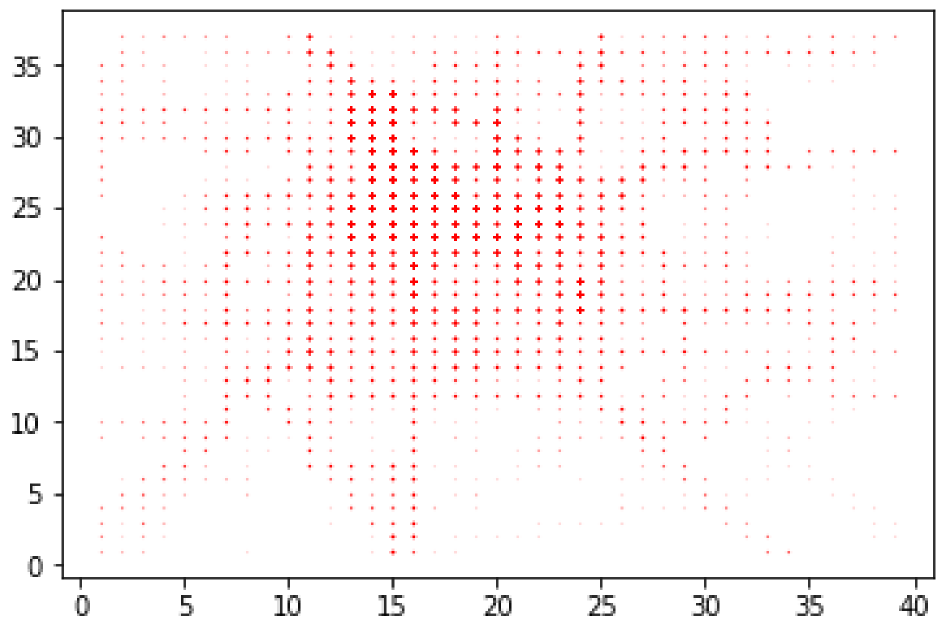

Figure 4.14: Trajectories corresponding to a grid with cells of 1300 meters.

below) to improve execution time significantly. The results are then compared, the experiments performed in GeoLife dataset show that for different approaches they lead to overlapping paths. We worked on a random sample of 1000 trajectories on the first slot by setting min_support = 50 trajectories to 1000 or considering all of them, the lowest value was chosen at 15 out of 5878.

In our approach Apriori algorithm allows the progressive identification of cells that are common to several trajectories and the identification of subsets of cells with low support of trajectories.

Step 1

The first step of the algorithm is to discretize the trajectories. In our case, we used a grid consisting of $1,443$ square cells (39 horizontally per 37 vertically) which result in cells whose side is $1,300$ meters (see grid in Figure 4.14) and about half of these are crossed by at least one trajectory.

Each trajectory, that was first considered as a sequence of GPS points, now becomes a list of consecutive and distinct cells.

The application of the Apriori algorithm performed in this step is as follows:

- First the 1-frequent cells L_1 set is found, scanning the dataset to count number of occurrences of each item. All the individual cells that satisfy the minimum support are counted.

That is, they are crossed by at least S trajectories (one of the parameter of the corridor detection algorithm).

- From the elements of L_1, the set C_2 is formed by considering all possible pairs of cells (candidate frequent pairs of cells).
- To find the L_2 set, the algorithm applies a pruning step, considering among the pairs of C_2 only those that satisfy the *min_sup* of trajectories (2-frequent cells).
- By increasing the size of the cells at each K iteration of Apriori the sets C_K and L_K are formed and C_k is generated from L_{k-1}.
- The algorithm stops at job K when it is no longer possible to find a set of frequent cells of size $K + 1$.

As a result of this step, we have identified the most frequent cell sets in trajectories. It is important to stress the fact that those cell sets may not be necessarily close in geographical terms, and for this reason they could represent disconnected paths.

Step 1.2

Once we have identified in the previous step the most frequent cell sets in trajectories, we can proceed to process the data by reversing the roles of items and bags in the Apriori algorithm. In this case, we will consider that trajectories correspond to "items" and geographical cells correspond to "bags". In this way, at the end we will get the most frequent trajectory sets, that is, the sets of trajectories that share at least a given number of cells.

In this step we apply the Apriori algorithm on a grid of 500 meters ($102 * 96$ cells), visible in Figure 4.15.

By setting the minimum support parameter equal to 51 cells (which make a total length of $51 * 500\ m = 25.5\ km$) and taking into account that the "populated" cells in the first slot, for example, are $4,603$ (cells crossed by trajectories), we have obtained a percentage of support greater than that of Step 1.

The description of the Apriori algorithm performed in Step 1.2 is as follows:

- The algorithm starts by first identifying all the individual trajectories which satisfy the minimum support, that is, those which are at least M cells long (one of the parameter of the corridor): this is the set L_1.
- From the elements of L_1 with the Join step the set C_2 is formed. That is, all the possible pairs of trajectories (candidate pairs of trajectories). Candidates are the itemsets containing all potentially frequent itemsets.
- To form L_2 the algorithm does the pruning, considering among the pairs of C_2 only those that satisfy the *min_sup* of cells (2-frequent trajectories).

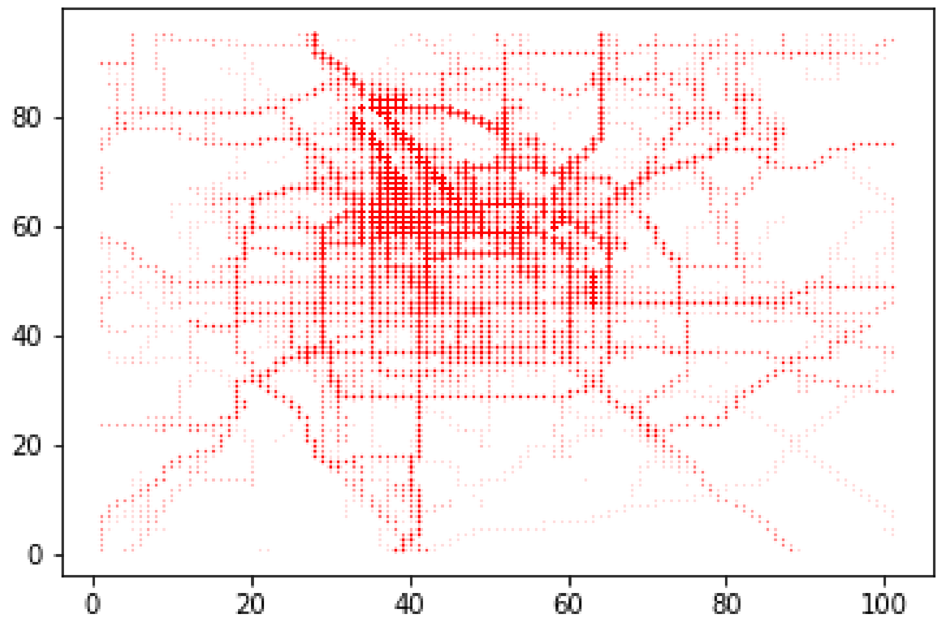

Figure 4.15: Trajectories mapping into a grid with cells square of 500 m wide.

- By increasing the number of trajectories at each K^{th} iteration ($K \geq 2$) of Apriori, it generates candidate K-trajectories C_K from the frequent $(K-1)$-itemsets L_{K-1} of the last iteration. Now only the frequent sets L_{K-1} and candidates C_K reside in memory, whereas other itemsets of previous iterations are discarded.

- The algorithm stops at job K when it is no longer possible to find a set of frequent trajectories of size $K+1$, (that is, when there are no $K+1$ trajectories which cross at least M cells together).

Compared to the previous strategy, in this step, fixed the desired minimum length of the corridors in terms of cells, Apriori finds those with size greater than or equal to this and shared by an increasing number of trajectories at each iteration of the algorithm.

Step 2

Since we only considered the last level of the output of the Apriori algorithm, now we want to analyze the subsets of cells that not appear in the last result of Apriori, the trajectories which do not participate in frequent itemsets of the longest cells. In addition, to optimize the execution time and vary the results, instead of using all the trajectories of every time each time, we considered a subset of $1,000$ random trajectories. We chose to build a grid of cells 1 km wide, so the total number of cells are 51 (horizontal) $*$ 48 (vertical).

We applied Apriori by following this convention: the role of "baskets" is played by trajectories and the role of "itemsets" is played by cells. As for the input of Apriori we chose a minimum support of 50 trajectories on the total number of the random set: the support of a cell is the percentage of trajectories in which that cell occurs.

This support being greater than that chosen for Step 1 and Step 1.2 leads to an improvement in the execution time of the algorithm. In fact, the output in this case is the tuples of cells shared by at least 50 trajectories and so there are less candidate corridors that contain 50 subtrajectories compared to Step 1 where the minimum support was 15. Moreover, the reduction in the number of the sample of trajectories also contributes to bringing this advantage. By varying the sample several times, different results will be obtained.

4.2.7 Probability of movement for GPS points

We aim to determine the probability that users are moving from one point somewhere in a city to another point. Such a probability is then used to provide recommendations accordingly. Recommendations, alerts, or user requests, are communicated by means of a smartphone app. Therefore, our proposed solution comprises two parts: (i) an algorithm for the movement prediction and (ii) an app on the user's device to track movements and suggest destinations.

A. Determining the movement prediction

To determine the probability of movement, we perform two steps. The first step is to determine the flows of people shared by the points recorded during each user's previous movements. Therefore, the second step consists in obtaining statistics on the amount of people who, having stopped at point A, move subsequently on to point B.

By analyzing each GPS trajectory, that is the set of recorded GPS points ordered in time, we have extracted its StayPoints (SP). They are the centers of the areas within which a user stays for more than a certain time: for some reason that area is of interest.

Then, the geographical area where the SPs of all the datasets are located has been discretized by means of a grid, composed of equal Square Cells. Each given SP was associated with a single square cell if contained in that space. Of course, a cell could contain multiple SPs if these are close enough together, depending on the width of the cell. Cells that do not contain SPs were not considered.

We then determined the subset of **frequently visited cells** consisting of all cells that have at least one SP within them and that were visited by at least 10% of people. For the sake of reliability, we only calculate the statistics between frequently visited cells and consider *Confidence* as used by the Market Basket Analysis. **Confidence** indicates the percentage of trajectories that frequently visit a cell B that also frequently visit cell A. For a Confidence value above a threshold (set as 60% in our experiments), we can say that a large group of people who have visited cell A moves together with cell B. Confidence is an estimate of the conditional probability. Two or more cells for which there is greater than 60% Confidence that have been visited by a large group of people are called **co-visited cells**.

Then, we check the reliability of the association rules obtained ($A \Rightarrow B$) via *Lift*, which will confirm that the transition of a user from the SP in A to the SP in B has a positive correlation.

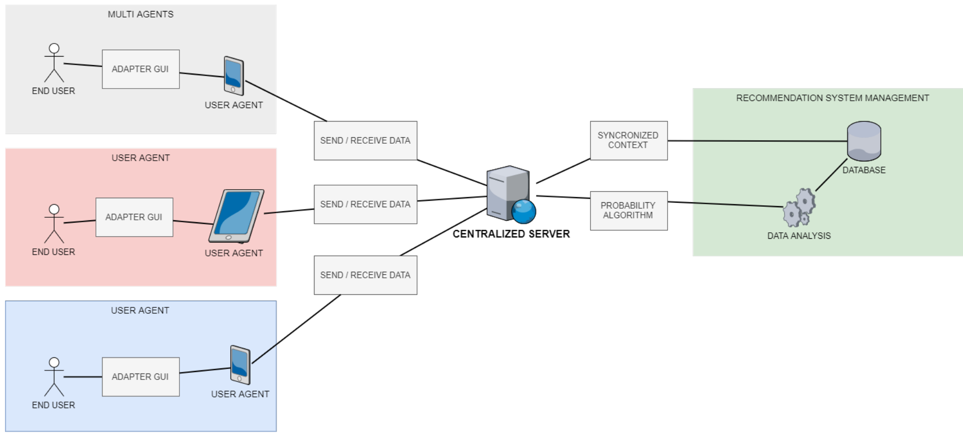

Figure 4.16: System architecture showing the interaction between agents and server: each agent sends his preference for crowded places and where he is, the server gathers data and creates the recommendation system.

B. A Multi-Agent Recommendation System

This section describes the multi-agent system that provides users with recommendations. In general, an agent, according to Wooldridge [97], is simply "a software (or hardware) entity that is in an environment and is able to react autonomously to changes in that environment". Each agent has the basics to learn and communicate, and in our case learning takes place by acquiring the user's GPS positions and, communication is achieved by connecting to a centralized server, which alerts all agents when necessary and stores the geographic coordinates of the points visited by users. Figure 4.16 shows the main components of the proposed multi-agent system. An agent runs on a smartphone like an app to get suggestions about the possible destination. The agent offers advice by highlighting any "hot", which is very busy or A "cold" place, that is, not very crowded, using the collected statistics as described in the previous section. For this, the agent periodically reads the user's location and checks if a known StayPoint (SP) is nearby. Then, the agent communicates to the server if it is close to an SP. This allows the server to determine the number of people close to an SP, rather than providing their actual location, thus preserving user privacy. In this context, the protection of privacy aims to prevent the disclosure of information relating to the exact location of the user. Figure 4.17 shows the app providing information to the user.

The server, having acquired the closest SP position from the user, returns the list of other SPs that could be visited according to the estimate of the probability of passing throught that point (0 equals low probability, 1 equals high probability). In this way, we create a recommendation based on the Collaborative Filtering system [67], as it is based on the choices of other users.

Finally, the user via an administration panel, can set with a flag, if he prefers "hot" or "cold" places. Therefore, the agent, based on the choice of users and the list received from the server,

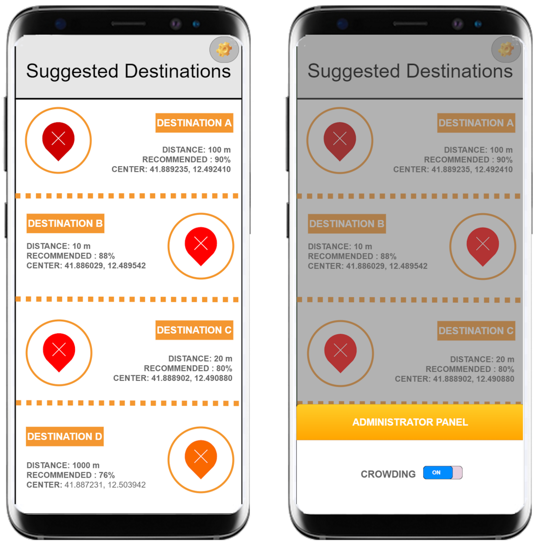

Figure 4.17: User communicates with agents via application GUI. The left panel shows the list of destinations suggested by the multi-agent system and the right panel is the administration view where the user gives his preferences on (un)crowded places. The colour of the icons represents the intensity of crowding, that is, more (less) red equals more (less) crowded.

decides what information to show and then suggests to the user. Destinations are displayed via a map or a list of suggested destinations.

All agents are independent of each other and since they extrapolate data directly from the device they are reliable, making the architecture stable and trustworthy.

Figure 4.18 summarizes the methods and the contributions of this thesis.

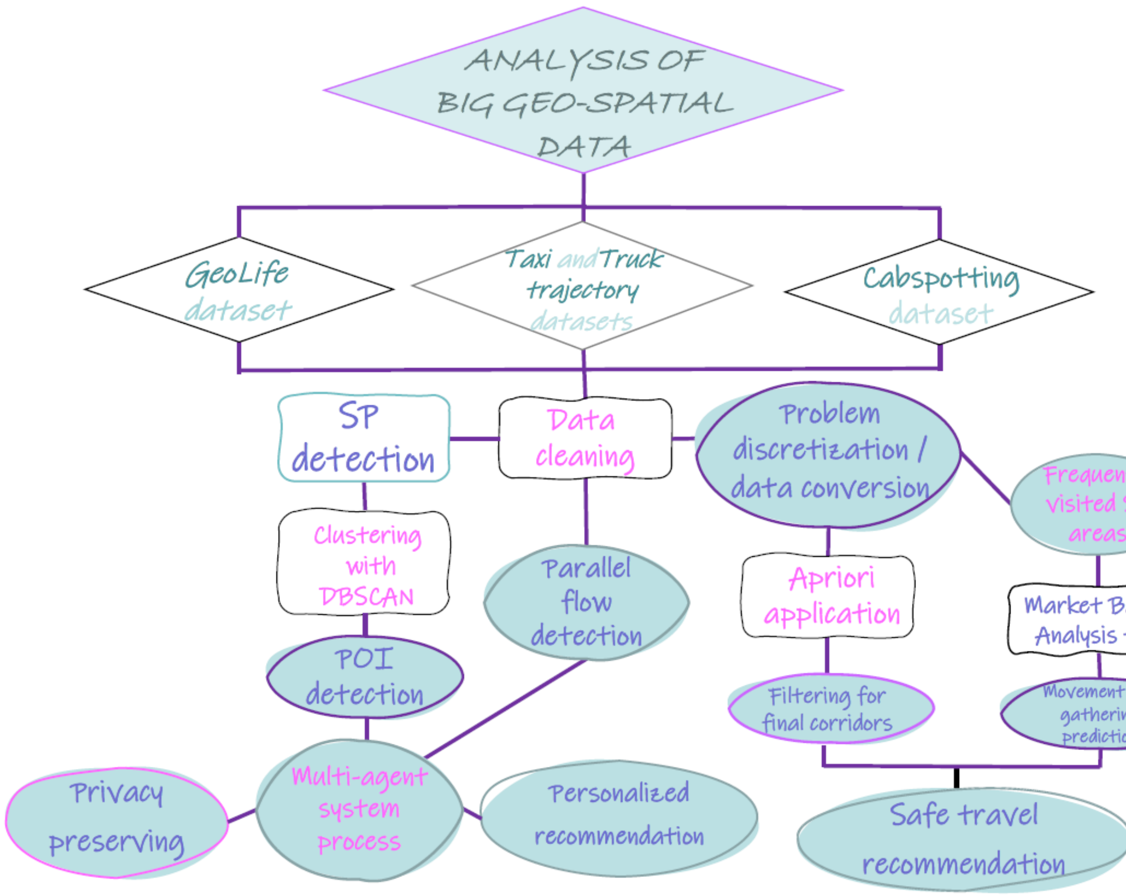

Figure 4.18: The analysis of the datasets is shown through the used methods (rectangular shapes) and the contributions (ovals).

Chapter 5

Evaluation

This chapter describes the experiments carried out using a dataset that collects real movements from one part of the city to another by taxis and/or people. Positions have been gathered by periodically reading geo-coordinates from tablets or smartphones. The experiments in Section 5.5 focus on data analysis for determining the probabilities of moving from one StayPoint to another. This approach is used in our centralized server in order to select the list of suggestions to send to the agents. The used dataset allows us to simulate the behaviour of a reasonable number of users, showing the useful of the app. Over time, data are updated as provided by agents. Below we describe the tests carried out and the results that have been found.

5.1 SPs and POIs obtained

The approach presented in this work finds the POIs (starting from a series of user trajectories) and uses the exchange of information with a centralized server to improve city services and user knowledge by creating software of content filtering: this creates personalized recommendations specific to the user to help him in his choices.

The execution time of the SPs detection algorithm (see Algorithm 3 for its pseudo-code) in GeoLife dataset for 100 trajectories is about 16 minutes. We obtained many SPs for every time slot, as shown in Table 5.1.

Applying DBSCAN to SPs detected in trajectory of Geolife, we obtained on average 20 clusters for every time slot (120 POIs in total, for a minimum of 9 POIs to a maximum to 29) that represent significant places for users, i.e. the centroids of these clusters are POIs. E.g. when we considered time Slot 3 and we set that the minimum number of SPs necessary to make a cluster as 15 and the ϵ equal to 200 meters, we obtained 29 clusters, hence 29 POIs (Figure 5.1).

The next step of our work was to filter the POIs detected according to popularity. We considered only POIs with a number of users greater than 10 (called *Popular POIs*), in order to understand users interaction and similarity. E.g., for time Slot 3 we obtained 9 POIs shared by a minimum of 11 individuals to a maximum of 80 individuals (see Figure 5.2).

Our experiments in GeoLife dataset have shown that in different time slots a set of different

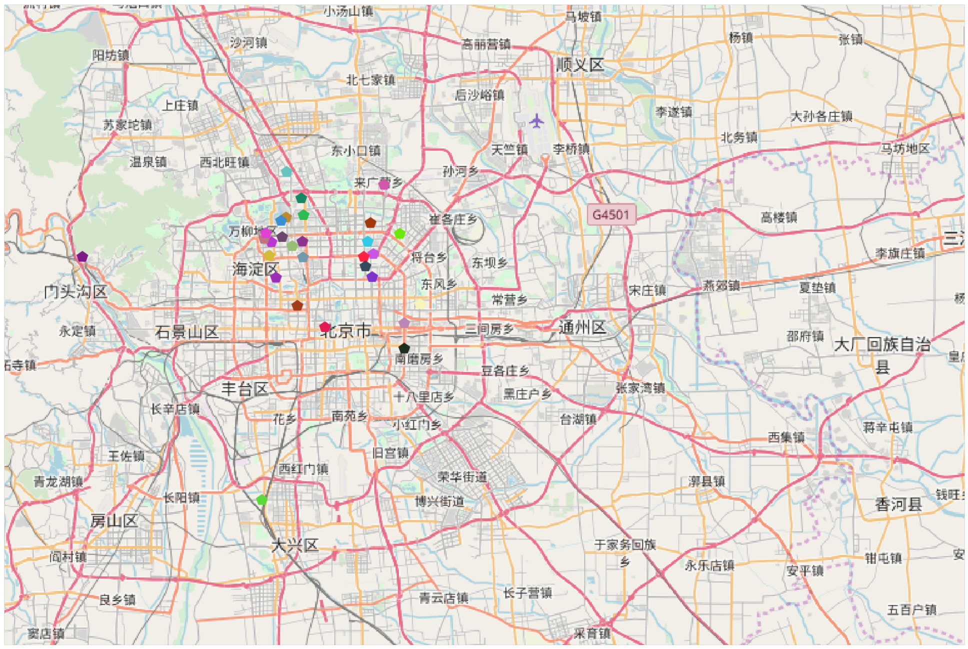

Figure 5.1: POIs obtained for the trajectories on time Slot 3.

Table 5.1: Results about SPs and POIs obtained for GeoLife's trajectories.

Time Slot	**Total number of**				**DBSCAN**	
	SPs	***Users***	***POIs***	**Popular *POIs***	***Eps*** *(km)*	***MinPts***
1	2966	122	18	8	0.3	15
2	3772	124	27	8	0.25	15
3	4146	145	29	9	0.2	15
4	1899	130	24	6	0.2	15
5	751	84	13	4	0.4	10
6	545	84	9	1	0.4	10

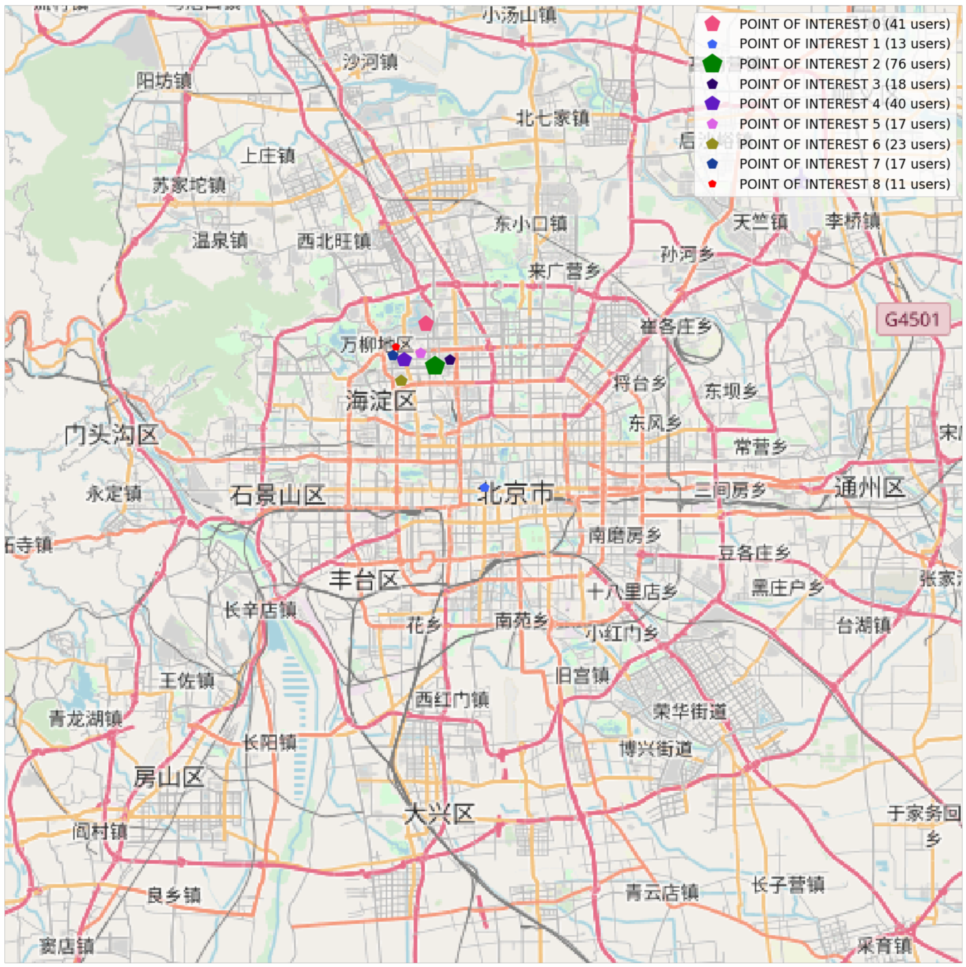

Figure 5.2: Popular POIs in time slot 3 on the map, Geolife dataset.

individuals move together to the same POIs, like parks, departments of Universities, shopping centres, hostels, parking spaces, libraries, stadiums, banks, Metro and bus stops. This suggests us a similarity between users.

The total of Popular POIs on all 6 time slots was 36; looking for the most distant pairs of points they have 8 km of longitude difference, 25 *km* of latitude difference on this area $[39.908309, 116.262296]$ x $[116.368098, 40.128495]$. These places of interest in question started from the north in Yangyang Paradise (amusement park), up to the Cultural Palace of Nationalities in Fuxingmen Inner Street in south, crossing Changping District, Haidian District with Tsinghua Park, Beijing Shi and Zhongguancun.

For detected POIs we can further say that our experiments show a correlation of people moving from one POI to another: users remain in these areas in certain common time slots. In our experiments the execution time of DBSCAN on the 6 time slots for GeoLife ranges from a minimum of 240 ms to a maximum of 1.44 s. Our implementation uses Python 3, and the experiments were run in a host having an Intel Xeon CPU E5-2620 v3 2.40GHz, with RAM 32 GB.

Figure 5.3 shows a list of POIs close to the user. Thus, for each POI the user can access the scores collected from other people's comments, as well as his comments. As mentioned in STEP 1, the POIs are detected by the implemented algorithm discussed in the next section. We can see four nearby points labeled as POI:

- Chaofan Weiye Kejiao Library with coordinates: 39.98405510061326, 116.3204636235443;
- Haidian Stadium with coordinates: 39.987213527969644, 116.30248430595732;
- Beijing Zhongguancun Branch Commercial Rural Bank with coordinates: 39.980016801082485, 116.30856309688643;
- Beihang University with coordinates: 39.98011363182701, 116.34218061609567.

Another nearby POI has been associated with a parking area (however, it is not listed in Figure 5.3:

- Parking of the satellite building with coordinates: 39.97673497237701, 116.33137904408086.

To validate the results of our algorithm in finding POIs, each discovered site has been verified on Google Maps. So, the list above consists of real sites, which are POIs according to Google Maps, which are located within a radius of 100 meters from the POIs detected by our algorithm.

In *Taxi trajectory data*, for the SP detection algorithm we chose a distance threshold of 200 meters and a time threshold of 5 minutes.

The obtained SPs were $31,621$, with an execution time of 51 minutes and 9 seconds. An example of SPs related to trajectories of a taxi is shown in Figure 5.4.

Relatively to the POI detection in this dataset, the parameters set in the DBSCAN were ϵ equal to 200 meters and minimum points (SPs) equal to 8. This algorithm produced 560 clusters whose centroids are the POIs, with an execution time of 10.2 seconds. Thirdly, resulting POIs were filtered to find the popular POIs, i.e. the POIs shared by at least 8 taxis. The total

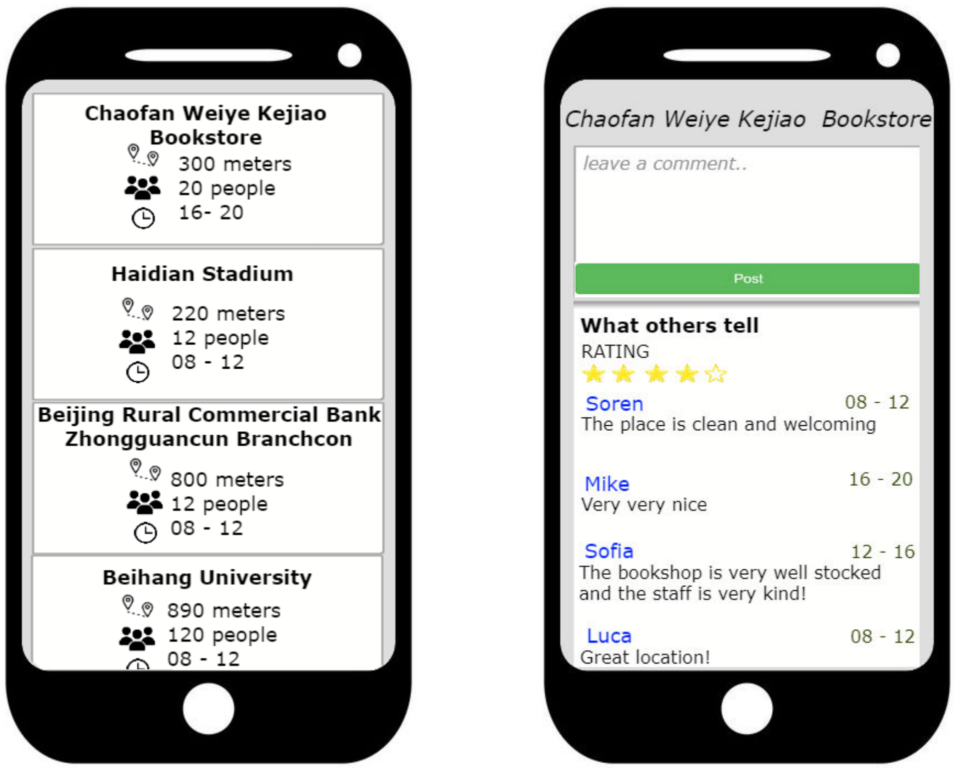

Figure 5.3: A list of POIs and associated dynamic data presented by an agent.

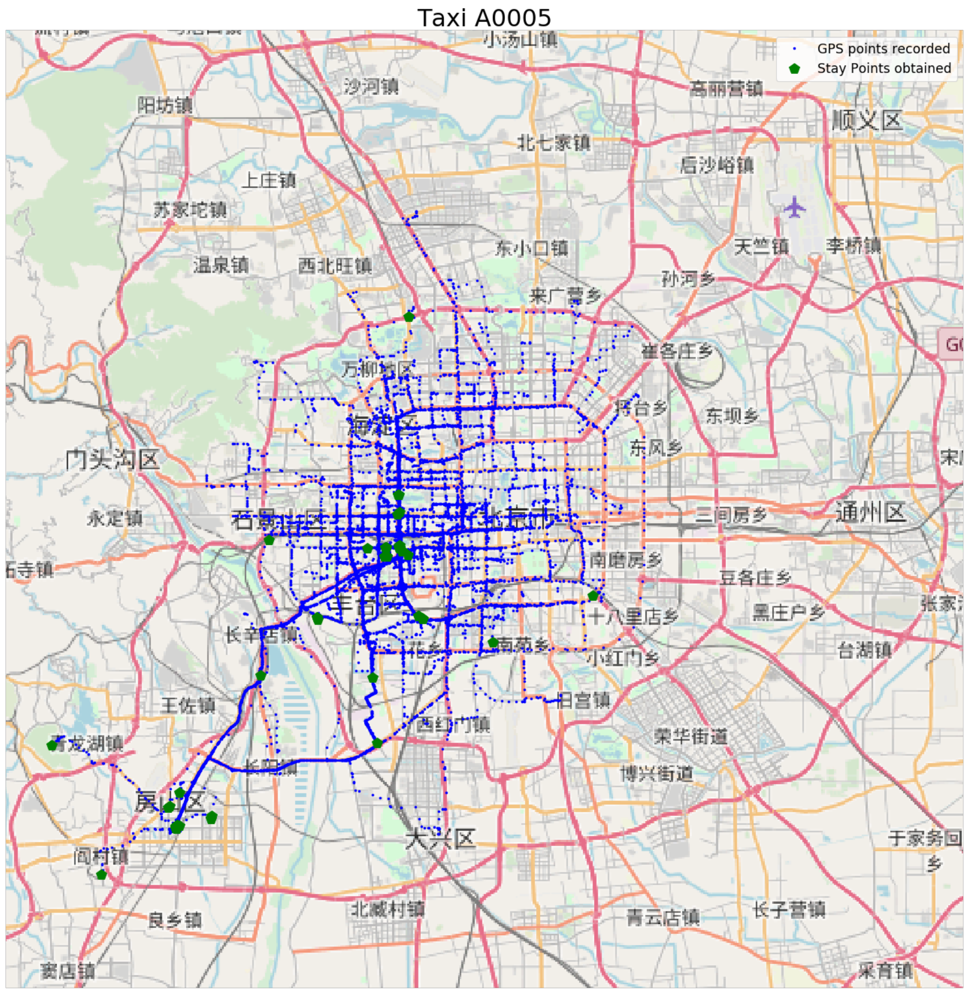

Figure 5.4: StayPoints of a taxi on its trajectories.

Figure 5.5: StayPoints of trucks on their trajectories.

number of such popular POIs was 257. Out of a total of 101 taxis, the popular POIs obtained were visited by 8 to 95 taxis.

The identified popular POIs can be found in the area $[116.093765, 39.79077]$ x $[116.608337, 40.092962]$. They are in Beijing and the most far apart pairs of points have distances 45 km for the longitude and on 35 km for the latitude.

Looking for SPs in *Truck* dataset the parameters DistThr $= 200$ meters and TimeThr $= 10$ minutes were chosen. A total of $54,962$ SPs were identified, in a time of 5 hours, 1 minute and 5 seconds.

In Figure 5.5 it is possible to see the trajectories recorded by trucks and their SPs, they pass through these provinces: Shānxī Shěng, Shǎnxī Shěng, Gansu, Henan, Hubei, Hebei, Beijing Municipality, Hunan, Sichuan, Guizhou, Yunnan, Guangxi Zhuang Autonomous Region, Guangdong, Jiangxi, Anhui, Fujian, Zhejiang and Shanghai Municipality.

Figure 5.6: Popular POIs in Truck dataset.

The choice of different values of time threshold set in the SP detection is due to the different nature of the three datasets. For taxis, a reasonable time of stay is 5 minutes, for trucks 10 minutes if it has to consider the bays, for Geolife dataset, which includes routes of users on foot, DistThr was chosen equal to 20 minutes. These parameters were validated by the average speed value relative to the flows found, in the vicinity of the Popular POIs.

For POIs detection the spatial threshold in DBSCAN remained unchanged ($\epsilon = 200\ m$) and the MinPts $= 8$ as in the previous case. The clustering execution time was 5.31 seconds with $1,065$ POIs detected.

According to the minimum number of taxis (8 out of 101), the popular POIs obtained were 127: they are shared by a minimum number of 8 trucks and a maximum number of 24 trucks. They cross the counties: Shangsi, Longzhou, Tiandong, Long'an, Mashan, Pingnan, Teng, Yunan, the prefecture cities: Chongzuo, Wúzh–ou, Yunfu, the districts Jinchengjiang and Yun'an, the Luoding city and the Kunming Subdistrict (see Figure 5.6).

Popular POIs obtained covered the area $[106.880591, 21.609655]$ x $[113.670971, 24.685619]$, with 700 km of difference in longitude and 350 km of latitude difference between the two most distant pairs of points. From west to east touched the cities: Chongzuo, Nanning, Guigang, Qinzhou, Fangchenggang, Beihai, Yulin, Wuzhou, Zhaoqing, Foshan, Canton and Dongguan.

5.2 Movement of people detected

The analysis of Geolife dataset has found 25 flows, shared by a minimum of 20 users to a maximum of 53 users and with a minimum distance equals to 150 meters between the trajectories. The length of paths that covered these flows is between 2 and 5 kilometres. More details are visible in Table 5.2. One flow for 21 walking users is shown in Figure 5.7.

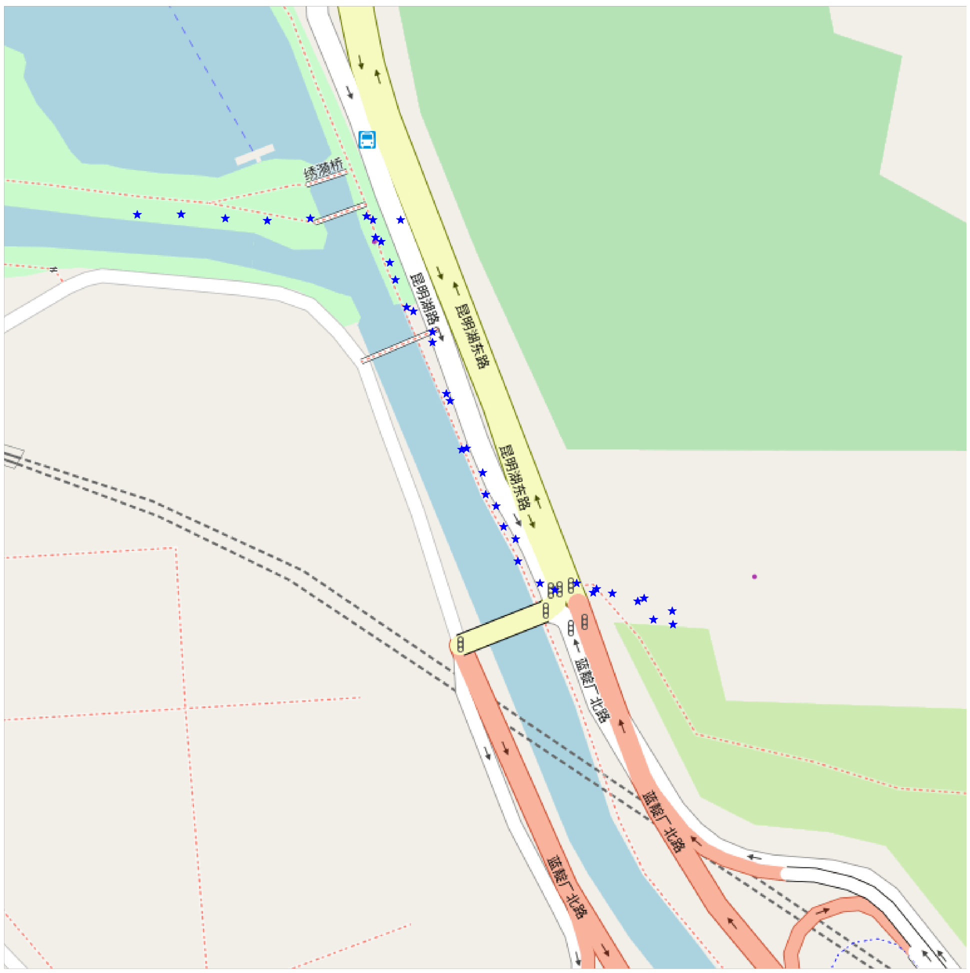

Figure 5.7: A flow for walking users, detected in GeoLife dataset.

Table 5.2: Details about flows obtained in GeoLife dataset.

Starting point	***End point***	***Distance (km)***	***Total number of users***
(39.932367, 116.385852)	(39.931153, 116.36233)	2.010	20
(39.881835, 116.269233)	(39.910339, 116.268027)	3.171	20
(39.988628, 116.483763)	(39.972852, 116.466223)	2.304	29
(39.986877, 116.399344)	(39.987707, 116.429464)	2.567	44
(39.90712, 116.350367)	(39.930045, 116.349922)	2.549	29
(39.979295, 116.284629)	(39.93731, 116.267806)	4.883	21
(39.971537, 116.274198)	(39.950695, 116.269099)	2.357	27
(39.93856, 116.268389)	(39.962849, 116.268952)	2.701	24
(39.931233, 116.321381)	(39.96773, 116.314436)	4.101	23
(39.972308, 116.327603)	(39.951754, 116.329522)	2.291	29
(39.984809, 116.333326)	(39.986664, 116.392776)	5.069	41
(39.986462, 116.375304)	(39.985521, 116.340627)	2.956	53
(39.976682, 116.453254)	(39.987662, 116.419631)	3.114	33
(39.985884, 116.361805)	(39.986612, 116.388662)	2.289	50
(39.984303, 116.326575)	(39.985593, 116.356151)	2.523	53
(39.986183, 116.433872)	(39.968712, 116.464021)	3.220	25
(39.948103, 116.345853)	(39.974371, 116.333989)	3.090	28
(39.943232, 116.34601)	(39.961544, 116.339601)	2.108	37
(39.98915, 116.331707)	(39.96691, 116.338599)	2.541	20
(40.023843, 116.347193)	(40.040263, 116.333505)	2.166	38
(40.032671, 116.313707)	(40.00121, 116.331161)	3.800	35
(39.931233, 116.321381)	(39.949994, 116.317978)	2.106	22
(39.944018, 116.318824)	(39.96041, 116.316824)	1.830	41
(39.969003, 116.313939)	(39.988966, 116.3099)	2.246	35
(39.991503, 116.333198)	(39.990766, 116.310343)	1.948	35

In Taxi dataset, thanks data preprocessing, about 400 outliers were removed. A minimum distance equals to 150 meters has been chosen for the algorithm and 77 flows were found with a minimum length equals to 1 km and a minimum number of taxis equal to 9. More details in Table 5.3.

Then, an estimate was made of the useful time slot to suggest to potentially interested agents. According to available data, it has been found that the preferred time slot was between 00:00:00 and 07:59:59.

The algorithm takes a trajectory as a reference to find flows. In this case, according to data, checks to find flows were performed taking about 10% of the trajectories (randomly chosen) and all the results were assembled together. Table 5.5 shows the details for all the datasets considered, and the first row of data is about taxis. In the table, each row contains: the name of the dataset used (Dataset); the time slot most chosen by users (Time Slot); the number of flows found (Number of flows); the maximum density detected among flows (Maximum Density); the maximum distance in meters of the longest flow found (Maximum Distance).

Subsequently, the algorithm was started in Truck dataset by taking 10 trajectories at random and using them as a reference for a comparison between all other trajectories. With data preprocessing, data removed were approximately between 0.001% and 0.007%, by eliminating outliers. In this case, a minimum distance between two points was set equal to 250 meters for elimination of noise. This choice was made due to the high number of points present in each trajectory and almost allows us to halve the response times of the algorithm.

Therefore, it has obtained about 114 flows with a minimum distance of 1 km and with a minimum number of users equal to 8%. For such experiments, it has been checked that the preferred time slot is between 16:00 and 19:59, though each POI has been associated with its own time slot. Table 5.4 gives the details of the experiments.

5.3 Corrispondence between flows and POIs

With reference to the performed analyses using the Taxi dataset, a 68% correspondence was found between the popular POIs and the shared flows, as can be seen in Figure 5.8 that illustrates both flows and POIs (black diamonds). Such places correspond to real useful sites and attractions, like: World park, Grand View Garden, Muzhi Gongdian amusement park, Ancestral Temple, Dongcheng Chongwen Science & Technology Museum, financial buildings, airport area, Long-distance Passenger Transport Terminal, Chaoyangmen SOHO shopping center, UK Embassy and Silver Bridge on Shichaha river.

The density of flows detected in this test was between a minimum of 9 taxis to a maximum of 70 taxis (among a total of 101 taxis). The intersections between the identified popular flows and POIs are found in the Wanliu, Haidian, Fengtai Districts, in the Tianzhu, Cuige, Lai Guang Yin, Shibalidianxiang Villages, in the Dongfeng area and in Nanmofang residential district.

Experiments on the Truck dataset have shown that there are 56 occurrences of Popular POIs near the 114 identified flows. It indicates a meaningful 50% correspondence, which gives a proximity within 100 m on the latitude and longitude of the points with respect to the identified flows. Points and flows were calculated on a minimum number of 8 trucks sharing them.

Table 5.3: Details about flows obtained in Taxi dataset.

Starting point	***End point***	***Distance*** (km)	***Total number of users***
$(40.01391983, 116.4700165)$	$(40.01517487, 116.4484177)$	1.302	12
$(40.05770493, 116.5829163)$	$(40.07411575, 116.5801697)$	1.654	48
$(39.94765091, 116.3908005)$	$(39.94740677, 116.3716965)$	1.510	10
$(39.96813583, 116.4283142)$	$(39.96772766, 116.4118347)$	1.199	10
$(39.88315964, 116.4419022)$	$(39.88715744, 116.4552155)$	1.028	15
$(39.94997406, 116.365387)$	$(39.93468475, 116.3665466)$	1.143	14
$(39.93462753, 116.4555817)$	$(39.94881439, 116.455368)$	1.262	23
$(39.87185669, 116.3431015)$	$(39.8875885, 116.3426819)$	1.604	11
$(40.02097321, 116.4004517)$	$(40.02082825, 116.4183197)$	1.505	9
$(39.94997406, 116.365387)$	$(39.9397049, 116.3663635)$	1.081	21
$(39.90634155, 116.330368)$	$(39.89759445, 116.3229828)$	1.051	10
$(39.98168945, 116.287735)$	$(39.98122025, 116.3092651)$	1.642	10
$(39.91970825, 116.3504028)$	$(39.90489197, 116.3504639)$	1.350	28
$(39.95053864, 116.4556808)$	$(39.96003342, 116.4426346)$	1.283	9
$(39.85622406, 116.3926163)$	$(39.85575867, 116.4096146)$	1.404	9
$(39.94730377, 116.3659515)$	$(39.94137955, 116.3498306)$	1.520	23
$(39.89347839, 116.4391632)$	$(39.90650558, 116.4296646)$	1.618	18
$(39.91085434, 116.4288025)$	$(39.92374802, 116.4281006)$	1.127	28
$(39.97738647, 116.28302)$	$(39.98391342, 116.2976837)$	1.343	14
$(39.86820984, 116.2659988)$	$(39.87461853, 116.2800522)$	1.020	18
$(39.89577484, 116.2631836)$	$(39.89577484, 116.2804489)$	1.209	11
$(39.90711975, 116.462883)$	$(39.90679169, 116.4413147)$	1.559	12
$(39.91335297, 116.3504181)$	$(39.89775085, 116.3500671)$	1.500	21
$(39.97241974, 116.295433)$	$(39.96090317, 116.301918)$	1.213	15
$(39.9569931, 116.3484344)$	$(39.9441452, 116.3498993)$	1.348	14
$(39.88085175, 116.4550858)$	$(39.89391327, 116.4554977)$	1.398	29
$(39.8991394, 116.455452)$	$(39.91464996, 116.4552536)$	1.624	17
$(39.90274048, 116.4475021)$	$(39.89246368, 116.4366989)$	1.468	16
$(39.87311172, 116.3432999)$	$(39.88673782, 116.3427353)$	1.436	20
$(39.89627075, 116.3370667)$	$(39.90134811, 116.3506317)$	1.233	12
$(39.9216423, 116.3496017)$	$(39.90307236, 116.350502)$	1.781	15
$(39.94994736, 116.3995514)$	$(39.93841171, 116.402298)$	1.069	10
$(39.98218918, 116.2887192)$	$(39.9839325, 116.3079834)$	1.452	17
$(39.91148758, 116.4289856)$	$(39.92631912, 116.4281158)$	1.379	30
$(40.02723312, 116.3072968)$	$(40.0164032, 116.3104858)$	1.024	12
$(40.0593605, 116.5826035)$	$(40.07794189, 116.5806198)$	1.992	23
$(39.93490982, 116.3380661)$	$(39.94364929, 116.353653)$	1.424	12
$(39.91411209, 116.3503036)$	$(39.89731216, 116.3494492)$	1.589	24

Starting point	***End point***	***Distance (km)***	***Total number of users***
(39.9209938, 116.3500671)	(39.90653992, 116.3503342)	1.323	70
(39.92617798, 116.4555359)	(39.9408226, 116.455513)	1.502	15
(39.90365601, 116.4556198)	(39.91807556, 116.4553299)	1.286	57
(40.06372833, 116.5824814)	(40.07794189, 116.5806198)	1.580	51
(39.94777298, 116.3972855)	(39.94748688, 116.3790817)	1.269	21
(39.89669418, 116.344902)	(39.91262817, 116.3507309)	1.553	10
(39.85474014, 116.3630981)	(39.85559464, 116.3847809)	1.848	12
(39.91291809, 116.4304504)	(39.92630386, 116.4309845)	1.214	25
(39.90378571, 116.455513)	(39.91992569, 116.4580688)	1.548	17
(39.94758224, 116.3987656)	(39.94828033, 116.4191971)	1.569	22
(39.92295456, 116.426033)	(39.92560577, 116.4419327)	1.049	11
(39.85756302, 116.4300995)	(39.85603333, 116.4152679)	1.194	12
(39.93951797, 116.4056015)	(39.93938828, 116.3879166)	1.379	11
(40.05770493, 116.5829163)	(40.07120132, 116.5812683)	1.461	53
(39.9569397, 116.4121323)	(39.94761276, 116.4052505)	1.002	9
(39.96665955, 116.3703537)	(39.96731186, 116.3871002)	1.040	14
(39.9621048, 116.2922821)	(39.97661209, 116.2945175)	1.099	9
(39.94763565, 116.3919983)	(39.94746399, 116.372818)	1.219	22
(40.00532532, 116.2823639)	(40.01837158, 116.2777863)	1.495	11
(39.92563629, 116.3669815)	(39.90939331, 116.3677673)	1.398	12
(39.91980743, 116.3500671)	(39.90393448, 116.3505173)	1.403	50
(39.96784592, 116.4162979)	(39.96609116, 116.4334335)	1.240	22
(39.98101807, 116.3206863)	(39.96736145, 116.319397)	1.211	13
(39.90114594, 116.4554825)	(39.91568375, 116.4553986)	1.207	42
(39.89970398, 116.4366837)	(39.90280914, 116.4532013)	1.418	12
(39.88513565, 116.4499969)	(39.89596939, 116.4547653)	1.083	27
(39.85876846, 116.4561691)	(39.85668564, 116.454567)	1.003	14
(39.97903824, 116.2846527)	(39.98388672, 116.3008804)	1.266	28
(40.01387787, 116.4676666)	(40.00965118, 116.4824829)	1.218	14
(39.96792221, 116.4197693)	(39.96332169, 116.4377518)	1.365	15
(39.96615219, 116.3343353)	(39.96644974, 116.3530502)	1.232	19
(39.94676208, 116.363266)	(39.93883514, 116.3483658)	1.227	43
(39.88864517, 116.4552002)	(39.90485001, 116.4554672)	1.473	15
(39.94742584, 116.383316)	(39.94783401, 116.4070129)	1.928	9
(39.8993988, 116.3507004)	(39.91819, 116.3501816)	1.804	15
(39.86477661, 116.4526367)	(39.87872314, 116.4550018)	1.425	23
(39.88868332, 116.4384689)	(39.90473557, 116.4297638)	1.598	20
(39.95137024, 116.3627014)	(39.9397049, 116.3663635)	1.125	22
(39.91291809, 116.4304504)	(39.92887497, 116.4280472)	1.497	31

Table 5.4: Details about flows obtained in Truck dataset.

Starting point	***End point***	***Distance*** (m)	***Total number of users***
$(23.46398, 110.1776)$	$(23.45764, 110.1636)$	1329.87	9
$(23.19097, 109.7451)$	$(23.18311, 109.7294)$	1490.506	13
$(23.38019, 110.0936)$	$(23.39321, 110.1007)$	1230.484	11
$(23.32668, 110.0111)$	$(23.3253, 109.9933)$	1492.372	9
$(23.36797, 110.0626)$	$(23.37064, 110.0579)$	1213.874	11
$(23.35973, 110.0617)$	$(23.37147, 110.0706)$	1570.259	9
$(23.53082, 110.399)$	$(23.51622, 110.3932)$	1293.346	9
$(22.16008, 108.1142)$	$(22.15369, 108.1282)$	1282.383	12
$(22.15512, 107.9694)$	$(22.16267, 107.9869)$	1696.31	9
$(23.1298, 109.3957)$	$(23.13622, 109.4109)$	1419.325	12
$(22.73798, 109.3333)$	$(22.72276, 109.3332)$	1476.157	10
$(22.8669, 108.3745)$	$(22.87254, 108.3894)$	1503.326	28
$(23.11833, 109.5236)$	$(23.11325, 109.5407)$	1451.289	17
$(23.15197, 109.4091)$	$(23.13818, 109.4114)$	1469.86	9
$(23.1204, 109.5085)$	$(23.11804, 109.5263)$	1693.694	11
$(22.87388, 108.3946)$	$(22.86811, 108.3807)$	1226.46	20
$(23.11759, 109.5282)$	$(23.11086, 109.5443)$	1722.323	18
$(23.13085, 109.3981)$	$(23.13505, 109.4145)$	1402.94	12
$(23.11982, 109.5211)$	$(23.1212, 109.5031)$	1465.044	17
$(23.12318, 109.483)$	$(23.12212, 109.4974)$	1204.83	13
$(22.83292, 108.3841)$	$(22.82915, 108.3879)$	1118.652	9
$(22.8275, 108.3957)$	$(22.82808, 108.4114)$	1094.745	9
$(22.90489, 108.2954)$	$(22.91195, 108.2957)$	1640.532	11
$(22.66003, 108.3862)$	$(22.67287, 108.3863)$	1163.431	18
$(22.88428, 108.2993)$	$(22.87448, 108.3139)$	1459.984	16
$(22.88168, 108.3084)$	$(22.88609, 108.2946)$	1456.035	16
$(22.73972, 108.3798)$	$(22.75565, 108.3777)$	1390.175	11
$(22.76152, 108.3703)$	$(22.75287, 108.3796)$	1205.48	15
$(22.8559, 108.3662)$	$(22.86585, 108.3544)$	1216.196	23
$(22.83669, 108.3734)$	$(22.85397, 108.3667)$	1785.581	10
$(22.84362, 108.3709)$	$(22.85662, 108.3656)$	1466.063	12
$(23.43506, 110.3099)$	$(23.44214, 110.2959)$	1625.982	9
$(23.39799, 111.1887)$	$(23.39769, 111.1798)$	1165.062	11

Starting point	***End point***	***Distance*** (m)	***Total number of users***
$(23.53223, 110.3998)$	$(23.51722, 110.3931)$	1293.346	9
$(23.37135, 110.0665)$	$(23.37696, 110.0852)$	1915.182	11
$(22.83352, 108.3868)$	$(22.8329, 108.3744)$	1200.724	9
$(22.93619, 108.2913)$	$(22.92397, 108.2971)$	1161.448	10
$(22.96408, 108.2891)$	$(22.97635, 108.2969)$	1308.752	10
$(22.71571, 108.3417)$	$(22.72903, 108.3304)$	1392.192	20
$(22.7945, 108.2684)$	$(22.80962, 108.2667)$	1364.938	12
$(22.86854, 108.3314)$	$(22.88037, 108.3378)$	1162.064	15
$(22.88345, 108.3021)$	$(22.88009, 108.3198)$	1693.3	15
$(22.88591, 108.2952)$	$(22.88088, 108.3118)$	1525.455	19
$(22.87914, 108.3368)$	$(22.86574, 108.332)$	1553.411	12
$(23.14492, 108.2352)$	$(23.15158, 108.247)$	1412.635	9
$(22.85945, 108.3627)$	$(22.86626, 108.3734)$	1057.851	29
$(22.86559, 108.3569)$	$(22.8696, 108.3838)$	2492.366	10
$(22.92994, 108.5176)$	$(22.93651, 108.5323)$	1501.728	9
$(22.92137, 108.4967)$	$(22.91548, 108.4802)$	1500.757	10
$(22.87822, 108.2934)$	$(22.89584, 108.2946)$	1624.175	29
$(22.88225, 108.3057)$	$(22.89031, 108.2942)$	1340.602	13
$(21.75837, 108.6056)$	$(21.75968, 108.5937)$	1261.628	9
$(22.14857, 108.5792)$	$(22.14441, 108.5931)$	1107.592	9
$(22.11363, 108.6227)$	$(22.10348, 108.6327)$	1152.363	14
$(22.3357, 108.399)$	$(22.32056, 108.4011)$	1450.596	18
$(22.08649, 108.6327)$	$(22.1009, 108.6329)$	1411.097	14
$(22.2349, 108.4161)$	$(22.25158, 108.4173)$	1664.687	13
$(22.09361, 108.6313)$	$(22.10851, 108.6261)$	1319.371	11
$(22.59992, 108.385)$	$(22.61542, 108.3855)$	1432.261	13
$(22.24552, 108.417)$	$(22.26123, 108.4164)$	1646.367	12
$(22.61952, 108.3845)$	$(22.63658, 108.3863)$	1495.875	18
$(22.28093, 108.4189)$	$(22.29208, 108.4063)$	1541.909	19
$(22.26574, 108.4183)$	$(22.28073, 108.4192)$	1603.248	15
$(22.27401, 108.4218)$	$(22.28527, 108.4135)$	1399.102	17
$(22.36413, 108.3838)$	$(22.37866, 108.3773)$	1450.092	17
$(22.09086, 108.6327)$	$(22.10606, 108.6308)$	1659.565	10
$(22.39749, 108.3841)$	$(22.38282, 108.3774)$	1430.2	19
$(22.74021, 108.3797)$	$(22.75572, 108.3777)$	1390.175	12
$(22.78894, 108.2468)$	$(22.7846, 108.261)$	1135.41	9
$(22.0058, 108.6383)$	$(22.00215, 108.6533)$	1454.266	15
$(21.96763, 108.5858)$	$(21.97193, 108.5985)$	1213.778	9
$(21.74269, 108.6015)$	$(21.73132, 108.5916)$	1119.86	12
$(22.07253, 108.6324)$	$(22.05612, 108.6317)$	1414.589	14
$(22.00949, 108.6315)$	$(22.00352, 108.6457)$	1364.895	24
$(22.06896, 108.6328)$	$(22.05241, 108.6305)$	1513.593	15
$(21.96581, 108.6823)$	$(21.95049, 108.6838)$	1439.659	17

Starting point	***End point***	***Distance (m)***	***Total number of users***
(21.95528, 108.6852)	(21.97146, 108.6771)	1539.229	15
(22.02115, 108.6338)	(22.00407, 108.6414)	1850.013	13
(22.0795, 108.6326)	(22.06653, 108.633)	1198.458	9
(22.047, 108.6287)	(22.0311, 108.631)	1426.291	19
(23.0508, 112.6255)	(23.04628, 112.6094)	1494.967	9
(23.05515, 112.6359)	(23.04925, 112.6203)	1348.628	14
(22.7636, 108.2572)	(22.76369, 108.2725)	1553.032	26
(22.85393, 108.2529)	(22.84287, 108.248)	1278.944	16
(22.88176, 108.3012)	(22.87944, 108.3169)	1312.716	17
(22.88609, 108.3395)	(22.87303, 108.331)	1235.096	10
(22.86431, 108.3297)	(22.87877, 108.335)	1442.315	15
(22.89781, 108.2826)	(22.88789, 108.2942)	1539.527	10
(22.91355, 108.4698)	(22.90682, 108.4566)	1323.153	10
(21.60608, 108.3474)	(21.61972, 108.3512)	1212.667	14
(22.86503, 108.369)	(22.87001, 108.3843)	1459.871	31
(22.8648, 108.3592)	(22.86687, 108.376)	1540.272	22
(21.60355, 108.3447)	(21.6161, 108.3486)	1368.788	14
(22.88035, 108.416)	(22.87374, 108.4025)	1150.239	12
(22.8604, 108.3175)	(22.86467, 108.3284)	1170.169	17
(21.76312, 108.3744)	(21.74969, 108.3731)	1387.631	10
(21.6694, 108.3656)	(21.67165, 108.362)	1244.098	12
(21.61372, 108.3506)	(21.63112, 108.3576)	1966.214	10
(22.87388, 108.3946)	(22.8696, 108.3839)	1378.008	12
(22.96362, 108.2673)	(22.95115, 108.2609)	1199.331	9
(23.62062, 107.0523)	(23.6268, 107.0366)	1291.845	9
(23.61226, 107.0659)	(23.61055, 107.0808)	1192.61	10
(22.92252, 107.9986)	(22.92764, 107.9814)	1493.632	12
(22.8472, 108.1149)	(22.84776, 108.1292)	1321.734	16
(22.73798, 109.3333)	(22.72276, 109.3332)	1218.709	10
(22.80511, 108.2687)	(22.81823, 108.2577)	1419.84	9
(22.82901, 108.2062)	(22.83938, 108.2102)	1100.438	11
(22.84139, 108.1561)	(22.83088, 108.1669)	1258.154	18
(22.76363, 108.2608)	(22.7749, 108.2531)	1307.189	12
(22.76152, 108.3703)	(22.7492, 108.3799)	1385.364	15
(22.83818, 108.161)	(22.82777, 108.1718)	1600.854	14
(22.76432, 108.2508)	(22.76374, 108.2681)	1548.94	27
(22.7491, 108.2928)	(22.73492, 108.29)	1183.701	12
(22.82743, 108.1799)	(22.82482, 108.1947)	1435.011	16

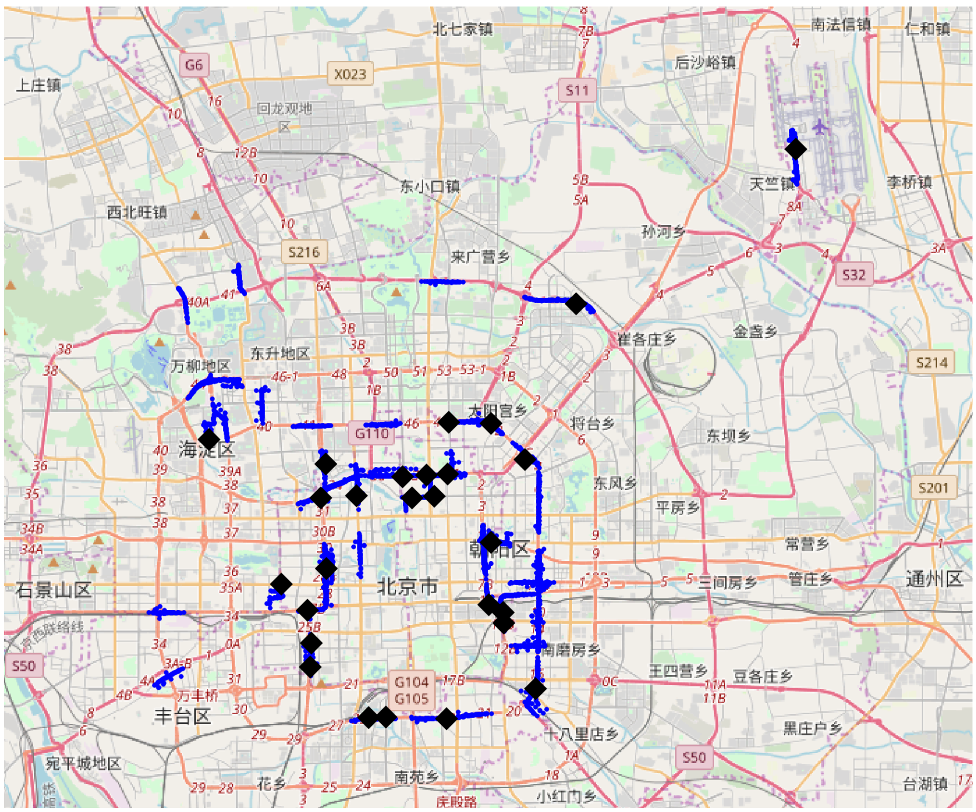

Figure 5.8: POPULAR POIs in correspondence of Taxi dataset's flows.

Table 5.5: An overview of the tests performed about flow found.

Dataset	Time Slot	Number of flows	Maximum Density	Maximum Distance
Taxi	00:00-07:59	77	70	1716
Geolife	08:00-11:59	25	53	3052
Truck	16:00 - 19:59	114	31	1718

Popular places of interest found near the flows correspond to: service areas, parking lots, grocery stores, tollgates, ATMs, auto parts stores, lottery ticket dealer, Driver Examination Center, hotel, restaurants, repair service area, gas station, lubricating oil sale service and Port of Fangcheng. They belong to Nanning, the capital of the Guangxi Zhuang Region, to the prefecture cities: Fangchenggang, Qinzhou, Guigang, Zhaoqing, to the Guiping county-level city and to the district Wuming.

As for the Geolife dataset, on 25 flows found, there were 13 matches with popular POIs (hence 52% of correspondence).

Figure 5.9 shows the longest route found, recorded by 35 users and also the Popular POIs discovered, which are in the same area. Such places are found on 5 of 6 different time slots, from 00:00:00 to 19:59:59 (near flow number 21), and they correspond to the sites: the Tsinghua Park for time Slot 1, Tsinghua University West Stepping Classroom for time Slot 2, the Tsinghua University Human Resources Service & Employment Center for time Slot 3, the Tsinghua University Biomedical Library for time Slot 4 and the High School Attached to Tsinghua University for time Slot 5.

Other popular POIs near the flows detected in Geolife dataset are: China Academy of Space Technology, China Aerospace Zhongguancun Astronautics Community, Satellite Building Parking Lot, Jade Palace Hotel Office Building and Kangtuo Science e Technology Mansion.

The above analysis shows that the correspondence among flows and popular destinations is 50% to 68%. Such values are high when we consider that we matched the flows of common people, rather than simply tourists. For the considered flows, many people would go to their homes and working places, many of which are not popular destinations.

We have used three datasets in order to show the generality and robustness of our approach in finding POIs, and flows of people. Finally, the analysis let us reveal how close the main flows of people are to the real points of interests, which were taken from a curated list. Hence, we can show that the suggestion of Points Of Interest can actually be convenient for users who need not travel long distances if they accept the suggestion.

5.4 Relations with the proposed multi-agents system

POIs provide the initial knowledge base to the agents. As, at the beginning, when agents have not yet collected and shared the user opinions, the multi-agent system is based on data obtained from the detection of POIs and flows, as explained above. Therefore, at the beginning all POIs have been computed by means of the approach detailed in Section 4.2.2 and used as a knowledge base for the multi-agent system. The POIs are defined by the name and GPS coordinates. Such

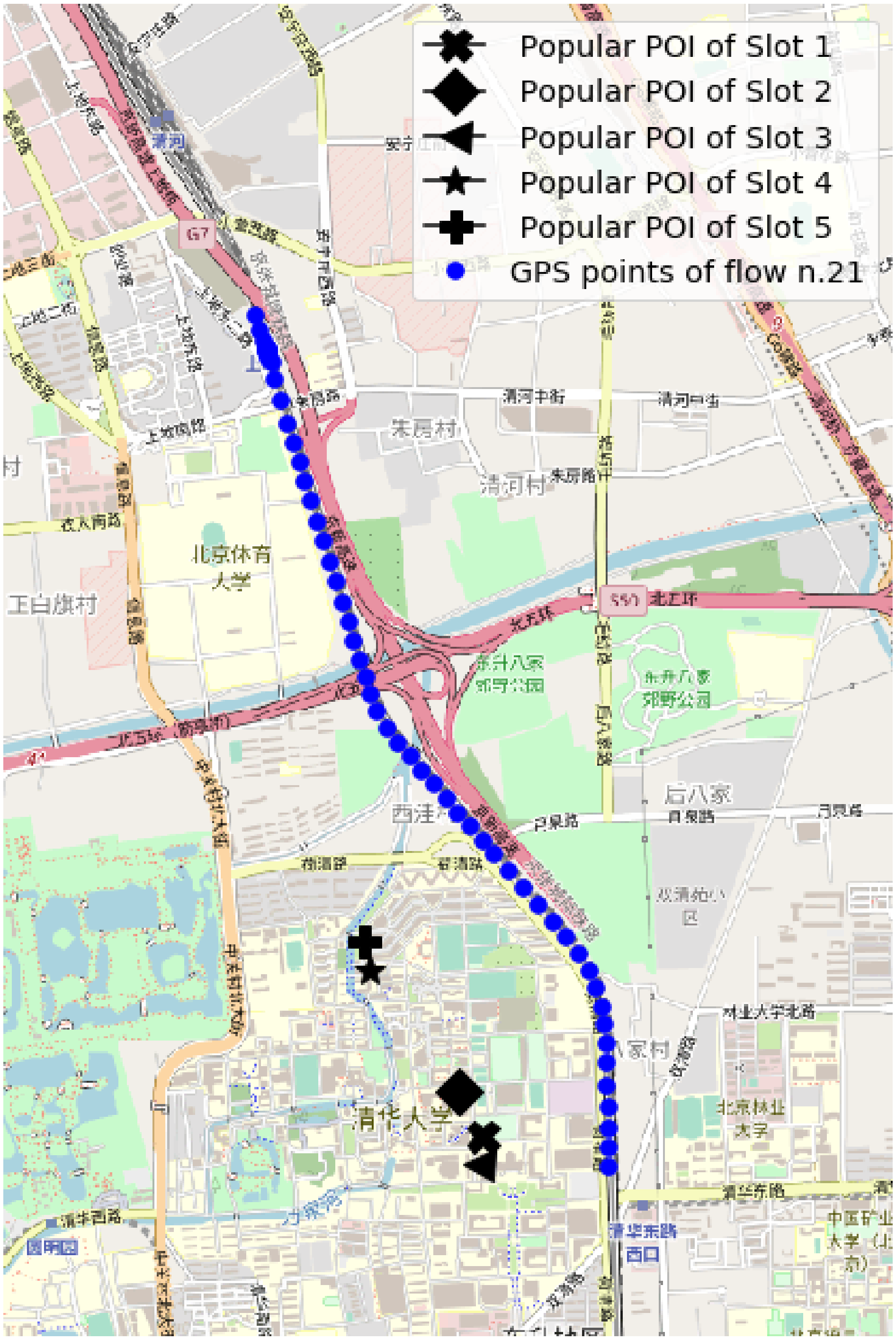

Figure 5.9: POIs of 5 time slots near Geolife's flow 21.

points have been identified by using the SPs, and other details (i.e. the name of the place) can be obtained, via APIs, since available on the web using services such as e.g. OpenStreetMap or Google Maps.

Experiments have shown how POIs vary according to the needs of the users. For available data, by analysing the flows of people for the different datasets, we could determine that users moving on foot mainly want to visit banks, bars, restaurants; whereas users moving by taxi (generally tourists) mainly want to visit popular places like churches and museums.

This allows us to offer a more efficient recommendation system, since the POI emerges from the gathered user positions and their trajectories, hence are related to the users.

In addition, thanks to the GPS position taken by the agent, the POIs are recommended based on the users distance from it and near the POI, the user's position is monitored, taking care to protect his privacy, as explained in the multi-agents section 4.2.3 and Section 4.2.4.

Finally, data and suggestions offered by means of the proposed multi-agents system are enriched by using the flow detection.

Such an approach has allowed us to validate the POIs previously found, and then to highlight the time slot that is generally preferred by users. The analysis has been performed for six time slots, in this way the recommendation system improves its efficiency, offering information on time slots. Suggestions to users are always updated and improved by the exchange of data between the independent agents, guaranteeing a reliable and updated service to the end user.

About the multi-agent architecture proposed by us, summarizing:

1. The assistants in the literature have been used as a personal assistant (via app) and we have proposed an assistant who proposes the place to visit (first characteristic of the agents: *ASSISTANCE*);

2. each agent has an *AUTONOMY* (second characteristic of the agents) and we have a (mobile) device in which the individual application does not depend on the applications of the other devices (*INDEPENDENCE*);

3. there is a *COLLABORATION* (third characteristic of the agents) through the central server with the mediator and they communicate with a client-server approach.

5.5 Test performed to investigate probability of users'movement

We considered the trajectories of Cabspottingdata and the data cleaning was done in order to eliminate noise, due for example to GPS errors. It was performed by computing the instantaneous speed of each point of the rides recorded on the taxi. The maximum acceptable speed threshold has been set for 150 $\frac{km}{h}$. We considered 6 time slots of 4 hours each, to visualize the movement of the vehicles at different times and the trajectories were therefore split according to the 6 time slots.

For each trajectory we apply the StayPoint detection algorithm (see its pseudocode in Algoritmh 3) with time threshold, TimeThr, equal to 10 minutes, and distance threshold, DistThr, 100 meters. Such thresholds should suffice to select the positions in which a user dwells (in several SPs) as he finds the place interesting, and removes the locations where a user is stopping because e.g. he is blocked at the traffic light.

The execution time for the SP detection algorithm on the whole Cabspotting dataset (536 taxis and more than $11M$ points) was 36 minutes and 54 seconds. We obtained a total of 4261 SPs, which is an average of 8 SPs per vehicle journey. The results show that 98% of users have at least one StayPoint associated with their trip (523 users out of 536). The implementation of StayPoints detection algorithm used Python and the experiments were executed in a host having an Intel Xeon CPU E5-2620 v3 2.40GHz, with RAM 32 GB. Figure 5.10 shows the recorded trace for each trip in blue, and the detected SPs in yellow.

To predict people's movements, firstly a grid was built, covering the map of all SPs: it consists of square cells with a side of 1 km. This grid allows us to discretize the data and estimate the probability of movement from a cell that has one SP inside it in another cell also having at least one SP. There are two distinct geographical areas that include some SPs represented as two square cells without intersection, therefore a spatial partition is formed. Figure 5.11 shows such a grid, having dimensions 80×46 cells (latitude by longitude) and the SPs obtained are mapped into the grid and displayed as a red diamonds.

Some cells have much more SPs than others, so some red diamonds are denser in some areas than others, such as shown in said figure. To determine if cell A is a frequent destination, the support for each SP cell was calculated. The **Support** of a cell is the ratio between the number of trajectories that contain the cell and the total number of trajectories present in the grid. If this ratio exceeds a certain threshold, i.e. if cell A is crossed by a certain number of different trajectories (a value of 10% has been chosen for **Minimum Support**, which is 0.1), then the cell (containing one or more SP) will be a *frequently visited cell*.

Our experiments on the taxi dataset above showed that there are 43 cells visited by a number of users greater than or equal to 52. That is, we can say that in the dataset there are 43 frequently visited SPs cells. This means that there has been a probable meeting in that cell, as users have remained stationary in the same time slot in the same cell. The data are updated in real time through the agents running on smartphones as an app, therefore the *Minimum Support* is fixed, however the number of frequent cells in output and the position of these frequently visited cells will vary over time. By lowering the Minimum Support, i.e. the threshold of the minimum amount of people sharing the same cell, the number of cells considered as having a sufficient amount of people will increase and then the number of cells considered overcrowded will increase.

In order to compute *Association Rules* only between the frequently visited cells in the dataset, we considered the **Confidence** of the *Market Basket Analysis* for our approach. Given two cells called A and B we have that:

$$Support(A \Rightarrow B) = \frac{Frequency(A, B)}{N}$$

the Support of the association rule $(A \Rightarrow B)$ denotes the percentage of trajectories containing A which contain also B, where N is the total number of trajectories.

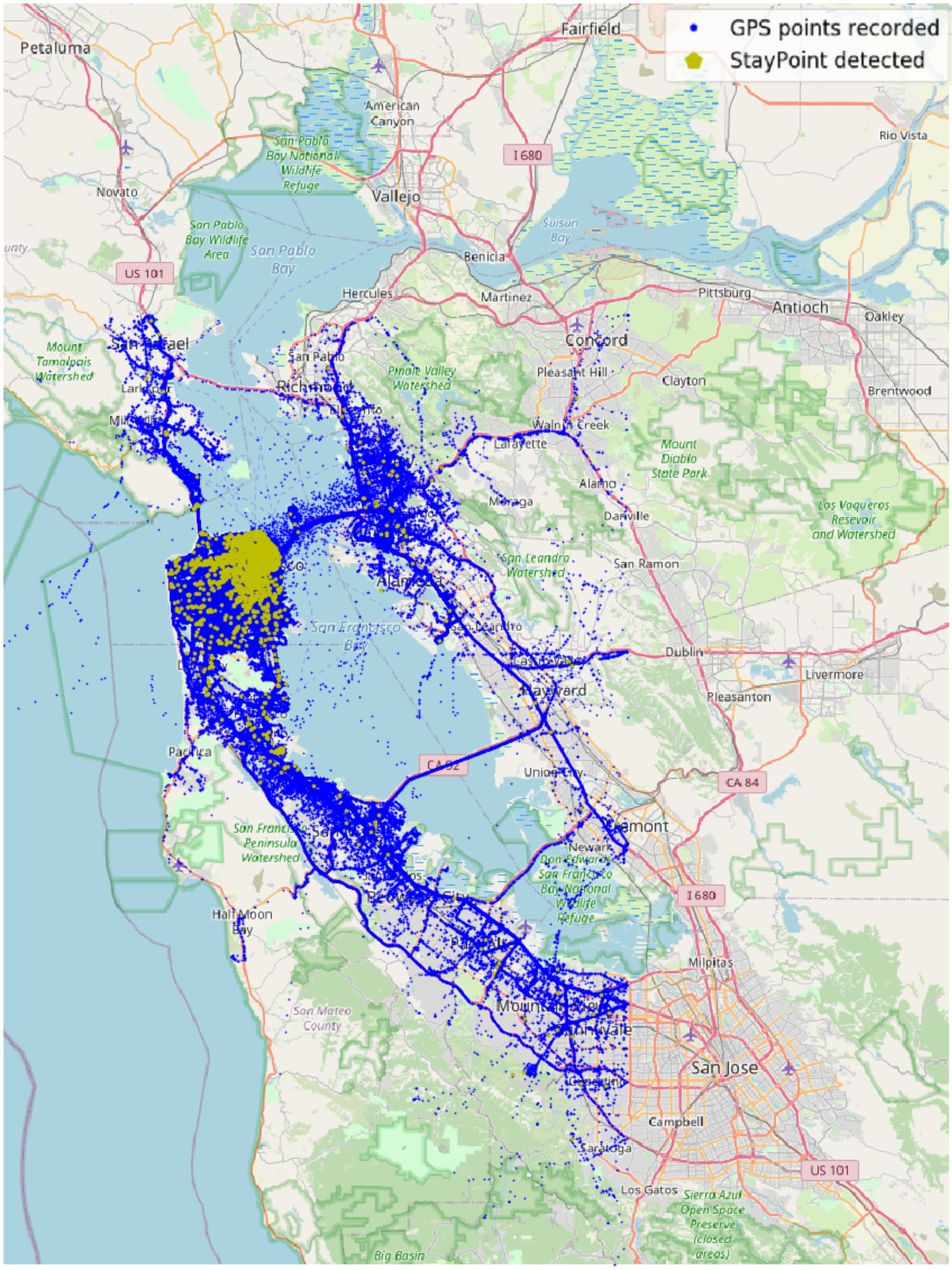

Figure 5.10: Blue points for trajectories and StayPoints obtained in yellow.

$$Confidence(A \Rightarrow B) = \frac{Support(A \Rightarrow B)}{Support(A)}$$

Hence, confidence is an estimation of conditioned probability, which can be expressed as follows:

$$Confidence(A \Rightarrow B) = \frac{p(A \cap B)}{p(A)} = P(B|A).$$

In Market Basket Analysis, *Confidence* is the probability of purchasing item B, said *consequent*, given the purchase of object A, said *antecedent*, within the same transaction. The higher the *Confidence*, the greater the reliability of the $(A \Rightarrow B)$ rule (more details can be found in [45]). In our context, the value computed as the *Confidence*$(A \Rightarrow B)$ gives the probability that a user is in a *SP* in cell B moving there together with at least 10% of the total number of users, if he has already been in cell A and dwelling in one of its *SPs*.

Going forward along this procedure, we compute *Confidence*$(A, B \Rightarrow C)$ and after that *Confidence*$(A, B, C \Rightarrow D)$, in order to determine a common path that crosses several cells having highly visited *SPs*. We compute:

$$Confidence(A, B \Rightarrow C) = \frac{Frequency(A, B, C)}{Frequency(A, B)}$$

and so on.

Therefore, the results obtained are useful to predict the number of gatherings on some place. Moreover, given that there is knowledge about an infected person on some area, our results can be used to predict whether a user can be potentially infected (as his trajectory is estimated), and predict who else he will infect (i.e. people whose trajectories are expected to pass through the same areas).

The Confidence limit is due to the fact that it does not consider the Support of the item on the right side of the Association Rule and therefore does not provide a correct evaluation in case the groups of items are not stochastically independent.

A measure that takes this eventuality into account is *Lift*$(A \Rightarrow B)$, defined as:

$$Lift(A \Rightarrow B) = \frac{Confidence(A \Rightarrow B)}{Support(B)} = \frac{p(A \cap B)}{p(A) * p(B)}$$

Lift$(A \Rightarrow B)$ takes into account the importance (the Frequency) of B. Using such an amount, then we can say

- if $Lift > 1$ the events are positively correlated;
- if $Lift \leq 1$ the events are negatively correlated or independent.

Therefore, **Lift** indicates how the occurrence of one event raises the occurrences of the other. At this point, the setting would be a Minimum Confidence (0.6 i.e. 60%) to skim the results and obtain only the Association Rules that had a higher Confidence and also a Support higher

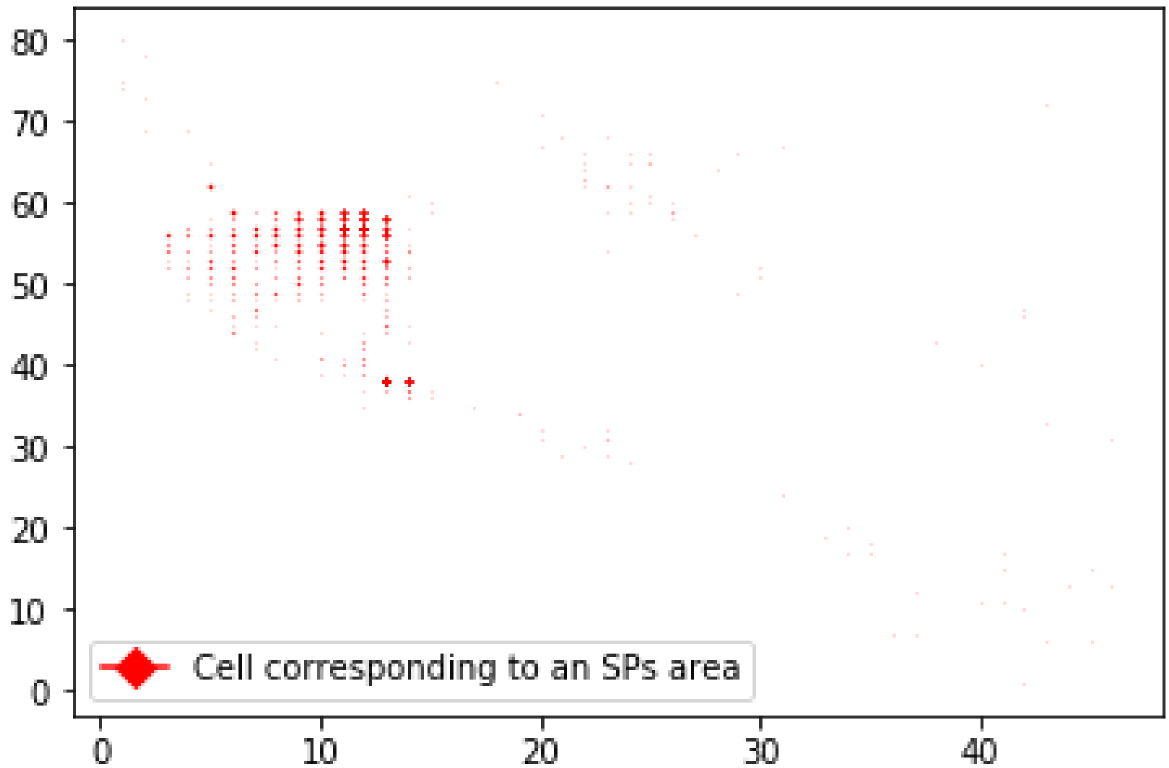

Figure 5.11: Grid formed by squared cells of 1 km per side, each red dot represent a cell having at least one SP.

than the Minimum Support (0.1 chosen), such a setting is named *Strong Rules*. Finally, these rules were checked with the Lift, the last column of Table 5.6. Then, for the Association Rule ([2587] $\Rightarrow$ [2588]) in row 6 the events of movement from cell A to cell B are negatively correlated.

Figure 5.12 shows the plot of every Strong Rule obtained as a point, as a value for its Support and Confidence (the latter according to the Support).

For Association Rules with higher support the Confidence, that is the probability of moving to the frequently visited cell B, decreases.

The Lift and the Confidence of the Strong Rules obtained are directly proportional, as we can see in Figure 5.13. The Pearson correlation coefficient between them is 0.9999999999999999 and this implies an exact linear relationship.

At the end, in order to find common subtrajectories between different users, the trajectories of Cabspotting have been summarized for the comparison of the distance: we considered for each path two successive points in temporal order only if they were at a minimum distance of 140 m. This was done to decrease the size of the dataset and therefore will allow a reduction in the execution time of the algorithm. To carry out a statistical analysis, 90% of trajectories

Table 5.6: Strong Association Rules: Minimum Confidence 60%, Minimum Support 10%.

$(A \Rightarrow B)$	A	B	$Support(A)$	$Support(B)$	$Support(A \Rightarrow B)$	$Confidence(A \Rightarrow B)$	$Lift(A \Rightarrow B)$
1	[1715]	[2588]	0.395793	0.671128	0.284895	0.719807	1.072533
2	[1716]	[2588]	0.313576	0.671128	0.223709	0.713415	1.063008
3	[2405]	[2588]	0.151052	0.671128	0.108987	0.721519	1.075084
4	[2541]	[2588]	0.137667	0.671128	0.099426	0.722222	1.076132
5	[2543]	[2588]	0.242830	0.671128	0.175908	0.724409	1.079391
6	[2587]	[2588]	0.435946	0.671128	0.292543	0.671053	0.999888
7	[2633]	[2588]	0.145315	0.671128	0.103250	0.710526	1.058704
8	[2634]	[2588]	0.281071	0.671128	0.206501	0.734694	1.094715
9	[2635]	[2588]	0.235182	0.671128	0.177820	0.756098	1.126607
10	[2679]	[2588]	0.202677	0.671128	0.147228	0.726415	1.082379
11	[2680]	[2588]	0.242830	0.671128	0.175908	0.724409	1.079391
12	[1715, 1716]	[2588]	0.149140	0.671128	0.112811	0.756410	1.127073
13	[2587, 1715]	[2588]	0.156788	0.671128	0.112811	0.719512	1.072094
14	[2634, 2587]	[2588]	0.141491	0.671128	0.101338	0.716216	1.067183

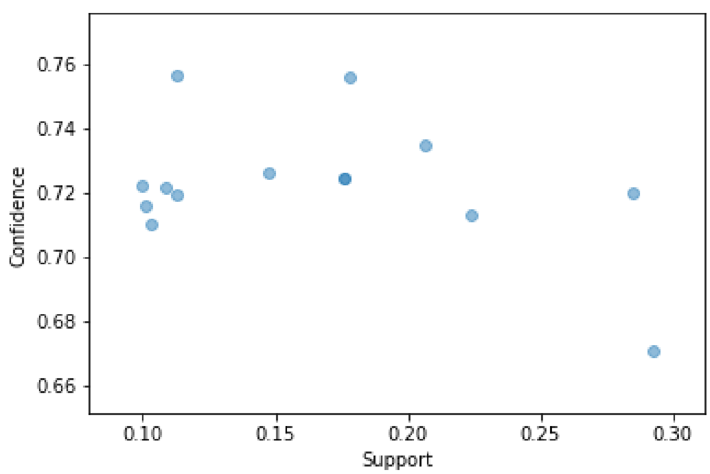

Figure 5.12: Support vs Confidence for Strong Rules obtained by our analysis and given as rows in Table 4.1. The points show a pair of cells (A and B), or a triple of cells (for the last three rows of Table 4.1).

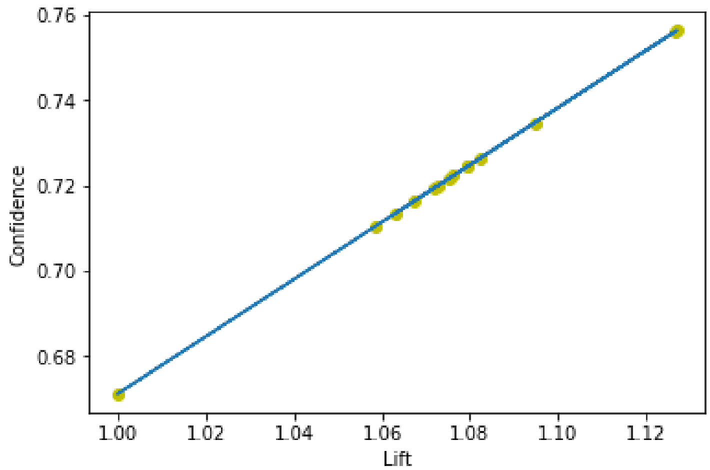

Figure 5.13: Lift vs Confidence in this test, as for the points shown in Figure 5.12.

were randomly selected, and this set was the *Train* set for the the flow detection algorithm. The complementary set, that is the remaining 10% of the trajectories, consists in the *Test* set, that is the verification set.

To identify the sub-trajectories common to different users in the same time slot, a maximum tolerance distance was set between two different points of different users as 280 m. The distance between two points was computed by using the Haversine distance.

We define **flows** as close sub-trajectories, belonging to different users, spatially similar and recorded in the same time slot. The density of a flow is the number of users that pass through it. To detect flows in this dataset, the minimum density threshold was set to 25. According to these parameters, 12 flows were identified, ranging from 1 to 2 km in length. The minimum density of the flows found is 26 taxis, while the maximum density found is 192 taxis. Then, by taking the complement of the trajectory sample (10% of the taxis, as the Test set) we checked where their GPS points were compared to the previous train set. We found that the points of the Test set intersect with the 12 paths identified on the Train set. Another check was carried out by confirming the correspondence of the points of the flows on a map. It consists of the process of matching the coordinates of the obtained flows and the road segments, and assessing that there are no external points with respect to road segments (see Figure 5.15).

Figure 5.14 shows the detected flows in magenta and the SPs in that area in green.

Moreover, the results tell us that the probability of transitioning from one cell with SPs to another is high even in correspondence with the indicated flows and that different highly visited cells having SPs belong to different flows. For each cell we have checked which taxis passed there and which passed at a later time on other flows passing through other frequently visited cells.

Our proposed work uses a multi-agent system communicating with a server that acts as

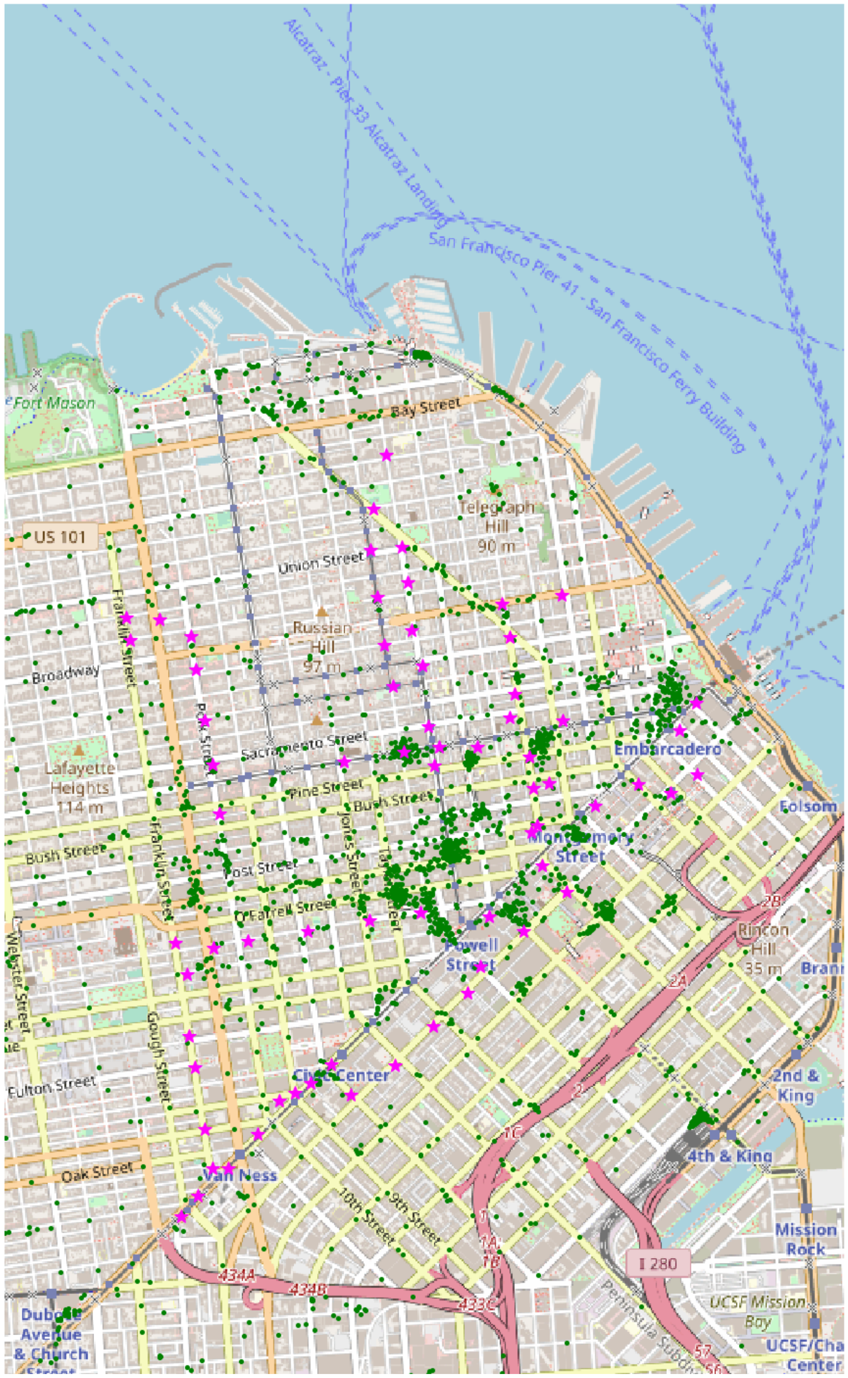

Figure 5.14: A zoomed in map of an area in Figure 5.10, showing flows in magenta and the nearby StayPoints in green.

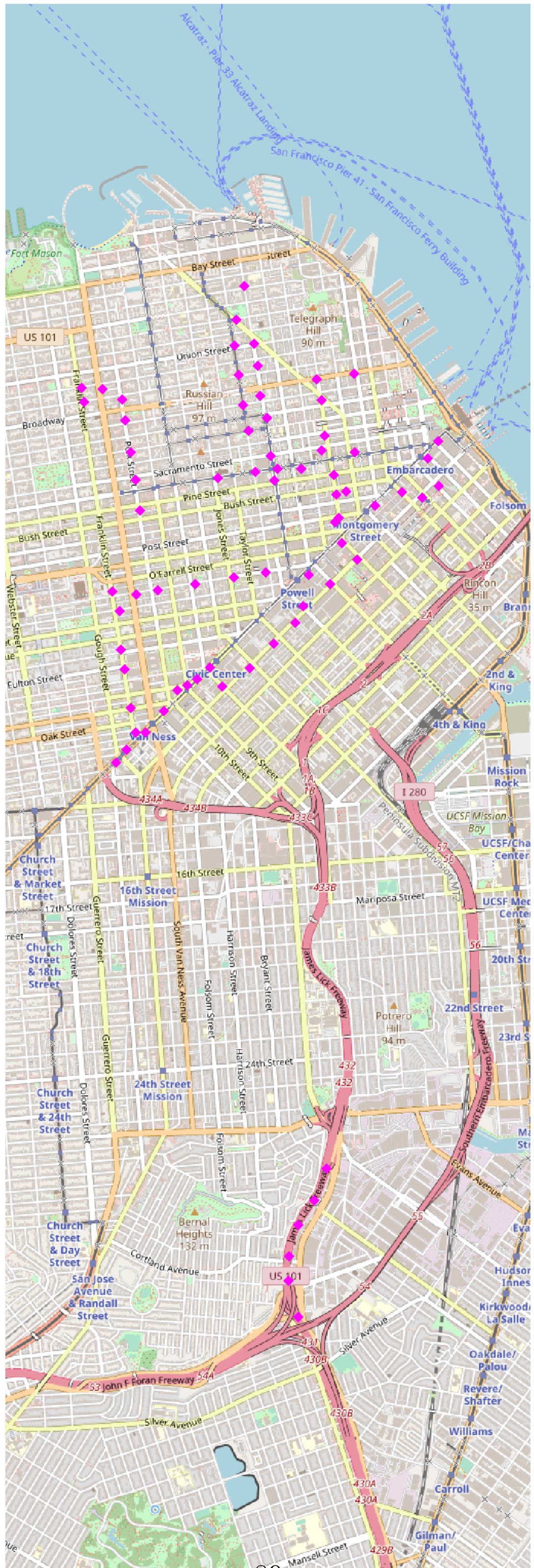

Figure 5.15: Flows detected for the Cabspotting dataset.

an intermediary and predicts movements by means of an innovative and reliable mathematical solution. Unlike the work presented in [11], our goal is to detect the foreseeable routes by computing their probability, whereas their method determines the probability that a group of users is moving together.

The paper [32] presents personalised recommendations for guiding tourists through the city of Melbourne by observing their actions. This system is modelled as a Markov decision process that recommends the user in sequence the next place to visit. However, unlike the StayPoint analysis presented here, it does not consider the stationary nature of visitors over a period of time and this is a key element in avoiding overcrowding.

5.6 Different methods by varying the corridor parameters

The output obtained is frequent K-tuples of cells, that are shared by at least $n \geq$ min_support (number of trajectories) Apriori uses a bottom-up approach, starts from the smallest subsets (the single cells) and checks whether they are *Frequently*.

For $K \geq 2$ since if an element (set of cells) is frequent, so too are its subsets, it generates the cardinality candidates $K + 1$ only those formed from the frequent K-tuples. This allows you to greatly reduce counting calculations.

The execution time is 1 hour and 22 minutes for 15 trajectories, considering instead the inverted problem: cells = baskets that contain/are crossed by the trajectories the algorithm ends in about 22 minutes, with a min_support of 55 cells of 500 meters each. So the advantage is the reduction of execution times significantly. The algorithm is very fast on the sample of $1,000$ trajectories and with a minimum support of 50 trajectories it ends in 297 ms.

We compared our proposed method with the computation of Discrete Fréchet distances between curves, which can be used to find those close to each other within a certain fixed tolerance. Considering the same random set of trajectories in the first time slot (a total of 1000 curves), the DDF between pairs of trajectories (one at a time) is determined in a total time of 6 hours and 24 minutes. Given that the total average number of trajectories for each time slot is about 4500 this strategy would be overly time-consuming (about 4 times and half more) than the three strategies we implemented (in the worst case 1 hour and 22 minutes have passed). Furthermore, determining the matrix of the distances for the trajectories to be compared would be impractical. In the machine available for our experiments we get a memory error, it is not possible for us to memorize a matrix so large.

Experimental results of Step 1

The application of Apriori results in the list of the K-frequent itemsets (i.e. subsets of frequent cells of cardinality K, the second parameter of the corridors), where frequent means that they are shared by a number of trajectories greater than or equal to the minimum support of trajectories chosen as Apriori parameter in input.

In Step 1 we chose 15 trajectories for each time slot as the minimum support threshold. For example, the total number of trajectories in the first time slot is $5,878$, so the set percentage of

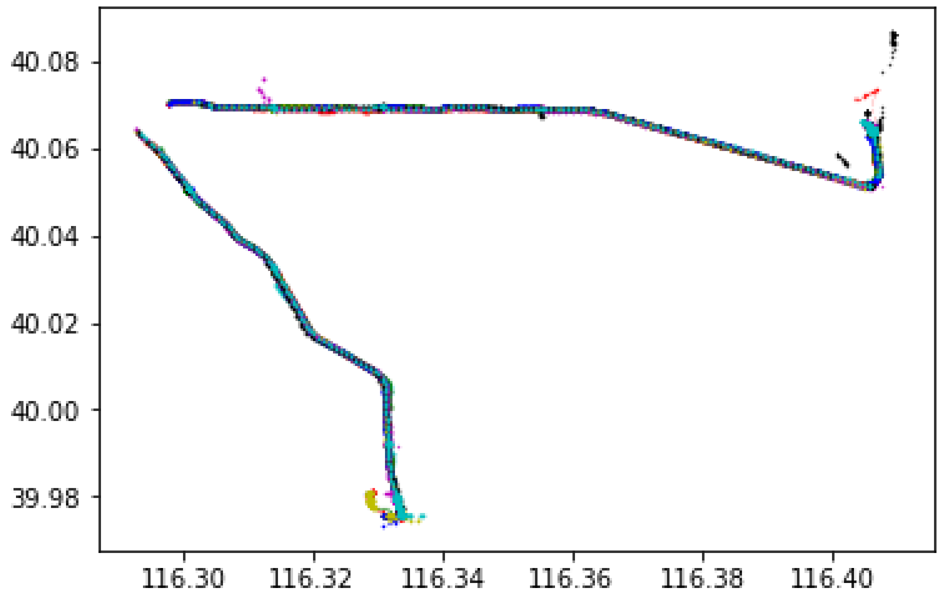

Figure 5.16: A candidate corridor detected with Step 1, with parameters $M = 19$ cells and $S = 15$ trajectories.

min_sup on the total was low but the advantage is to be able to find corridors even for a limited number of trajectories. Setting the minimum number of trajectories, the length of the corridor candidates obtained at each iteration of the Apriori algorithm increases.

In our case, Apriori returns a list of over 4 million K-frequent itemsets of cells, for K from 1 to 21. The largest subset of cells, which is the last iteration of the algorithm, corresponds to a path of about 25 km. At this point we considered all the trajectories passing from the first cell of the 21-tupla of cells. We deleted all the trajectories that did not pass for the second cell, then we deleted the trajectories that did not pass for the third cell and so on, until the 21st cell. At the end we found 15 trajectories, that are exactly the trajectories passing for all the 21 cells. After verifying that these trajectories cross these cells in the same or in the reverse order and consecutively, we considered the 21-tuple as a candidate corridor. Figure 5.16 shows the longest candidate corridor obtained in Step 1: the interruption in the upper left part of the graph is probably due to the loss of the GPS signal during the recording.

Experimental results of Step 1.2

The output produced by the algorithm consists of $55,109$ sets of K elements of frequent trajectories (in the time slot Δt_1), for K from 1 to 14: these trajectories share candidate corridors of over 25 km.

The advantage is that, compared to the previous step, it improves the execution time. Indeed the scanning of the trajectories to check if they are frequent or not is faster because they have

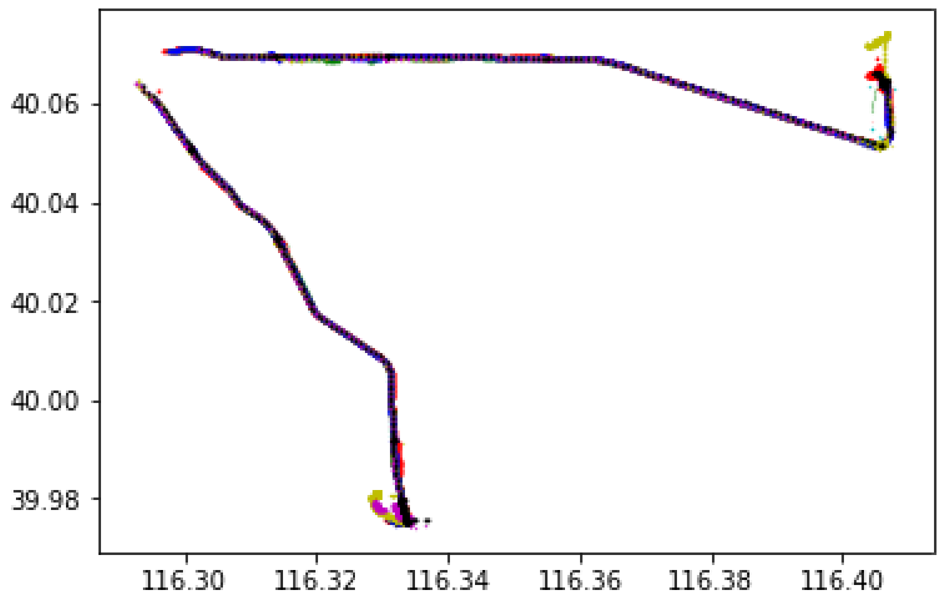

Figure 5.17: Candidate corridor detected with Step 1.2.

few elements (on average they have a length of about 16 cells of 500 m each) and also the support threshold has been raised. In fact, if the min_sup is larger, then the cardinality of the frequent set L_K and therefore also those of C_{K+1} are much smaller than in Step 1. Moreover, the number of iterations of the algorithm is also lower (it made 14 iterations instead of 21). Another improvement was the use of a thinner grid which gives a greater precision of the road results.

The longest corridors of these 14 trajectories are similar to the ones found in the previous case (Step 1). Among these 14 trajectories, 7 are in common with the 15 obtained in the Step 1. The plot of trajectories crossing 51 cells consists of about 25 km visible in Figure 5.17, and like the previous case we verify that this sequence is a candidate corridor of parameters $M = 51$ and $S = 14$.

Step 1.2 is a verification of Step 1, also allowed to refine the grid and therefore to have a greater precision (for instance we used a grid with side cells 500 meters wide instead of a grid with cells $1,300$ meters per side as in Step 1 - see Figure 4.15 -).

A further advantage is the reduction of execution times significantly. We found two candidate corridors, in the same street shared by a union of 22 trajectories, found in Steps 1 and 1.2.

The grid is out of phase, because 500 (meters) is not a multiple of $1,300$ (meters), but this gives a greater number of trajectories that are recorded on the same road, with the union of the results obtained.

Experimental results of Step 2

The output obtained for data in Δt_1 was a list of frequent itemsets of cells, a table of $1,398K$-frequent itemsets, for K from 1 to 9. The longest sets of cells were three 9-frequent cells:

$F = \{1956, 1703, 1804, 1550, 1905, 1906, 1652, 1754, 1855\}$,

$G = \{1956, 1703, 1550, 1905, 1906, 1652, 2007, 1754, 1855\}$ and

$H = \{1956, 1703, 1804, 1905, 1906, 1652, 2007, 1754, 1855\}$.

We set $A = F$. Among the 8-frequent itemsets we looked for those that were not a subset of A. If there was a 8-frequent itemset that shared at most one cell with F, then we saved it as a potential corridor. At the end, we added to A all the elements of the 8-frequent itemsets that were not in A and we repeated the procedure with the 7-frequent itemsets. Continuing in this way until the 2-frequent itemsets, we found 7 potential corridors with at least 2 cells.

Doing the same starting from G or H we found the same potential corridors of parameter M from 2 to 5:

C_1=[1601, 1602, 1603, 1604, 1656], shared by 53 trajectories;

C_2=[1753, 1702, 1703], shared by 58 trajectories;

C_3=[1548, 1549], shared by 58 trajectories;

C_4=[1548, 1599], shared by 64 trajectories;

C_5=[1550, 1551], shared by 91 trajectories;

C_6=[1600, 1601], shared by 74 trajectories;

C_7=[1804, 1805], shared by 53 trajectories.

The performance of this procedure depends on the number of K- and $(K-1)$-cells and size K for every iteration and the number of new cells found in until level K.

Starting for example from the candidate corridor C_1 obtained, we carried out the filtering which is the final phase of our work. The plot of this candidate corridor is shown in Figure 5.18.

5.6.1 Filtering process

Given the problem of approximation of the grid, the previous results does not assure that all sub-trajectories have a similar initial and final part. Because of this, we must restrict the search for the corridor to a sub-area, starting from the determined corridor candidate.

Given a radius r set by the user, the Radius Neighbors Graph algorithm constructs a graph in which all the points of a trajectory, seen as nodes of the graph, are connected to all the other points of the other trajectories that are close to it within distance r. The adjacency matrix allows to quickly distinguish which sub-trajectories are in the neighborhood and which sub-trajectories are to be discarded.

Every corridor selected with this filtering phase (applying Radius Neighbors Graph to the cells containing the candidate corridors) is a road segment that can be considered as a “hot” route. The result of the filtering process on candidate corridor detected above is shown in Figure 5.19. Finding the frequently repeated patterns of sub-routes can help to analyze and predict the movements of objects; starting from the results obtained we could propose an estimate of the similarity between users (who share corridors) using the Jaccard Similarity index.

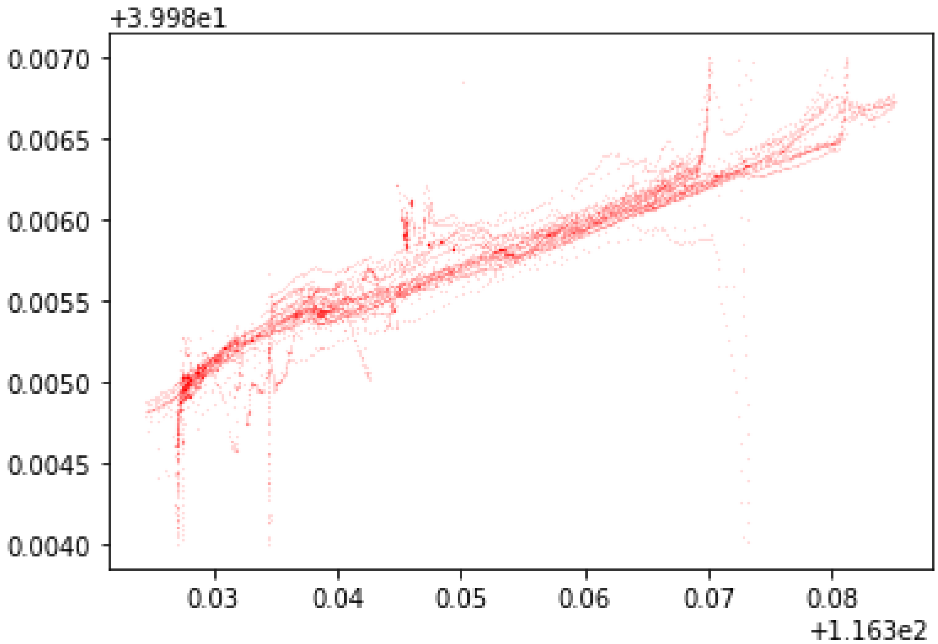

Figure 5.18: C_1 candidate corridor detected in Step 2, with parameters $M = 5$ cells and $S = 52$ trajectories.

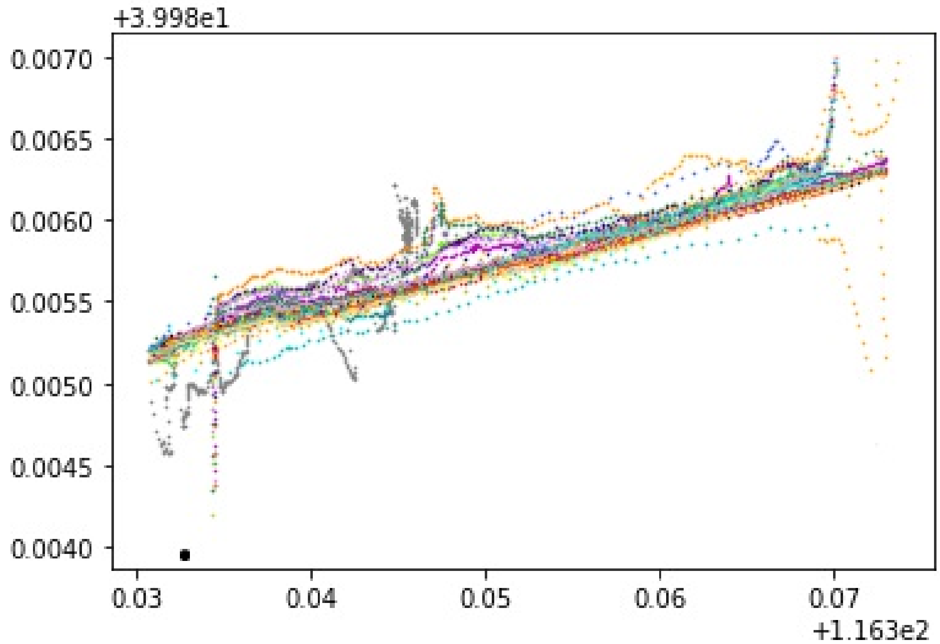

Figure 5.19: A final corridor obtained by the filtering process, shared by $S = 50$ trajectories and $3,7$ km long.

Chapter 6

Final discussion, remarks and future work

Given the widespread use of mobile devices and various technologies that track their geographical position, it is nowadays easier to acquire information about users' GPS in real time. This availability has triggered numerous studies based on user positioning, such as analyzing flows of people [93] in cities or predicting people's movements. This has also led to the improvement of the services that identify the points of interest of a city to offer advantages to users who want to reach a place but do not have sufficient knowledge for an immediate choice.

The problem of "hot routes" detection is important for transport services, which are interesting for finding corridors for public transport of multiple routes (e.g. by bicycle, subway, car sharing) taking advantage of GPS sub-trajectory datasets.

6.1 Epilogue

The first proposed approach provided a solution for sharing collective knowledge about popular and useful points about a city. Users can share their comments on a place they visit thanks to an app available on them smartphones and can receive suggestions on the next place to visit.

For the proposed multi-agent system, a setting in our agent application on the smartphone allows to collect GPS coordinates. This is useful for a geographical position where there are no previously collected trajectories. Thus, we can continuously extract user trajectories and update both user tips and POI recommendations. Using this setting, each agent periodically releases both the GPS coordinates and his identity to the server, however, to take into account privacy issues, the user's identity is masked and the GPS position is randomly moved up to 300 meters. So, we get a trajectory that is not very precise, but still useful. As trajectories are dynamically collected, POIs are also determined dynamically as data arrive to the server.

The experiments in GeoLife dataset confirm that users stop in the same areas for some common reasons, such as visiting a tourist attraction or taking advantage of the same service and remain in certain areas in common time slots. Additionally, these tests reveal that people move together from one POI to another.

The main contribution of the first part of our work is:

- automating the extraction of the list of places of interests;
- updating such a list in real time;
- characterizing places of interests with time slots;
- providing to the user comments on places of interests;
- selecting comments and places of interests according to user preference and position;
- preserving privacy for the user.

In the second study we carried out we analyzed two other geo-datasets real thunderstorms, Taxi and Truck Trajectories with different characteristics, applying the algorithms explored but with parameters adapted to the cases in question.

By analysing the available people positions, it has been shown that it is possible to find: Points Of Interests and people flows.

To validate the new Points Of Interest found, flows shared by different users (within a fixed radius) were identified by similar sub-trajectories. The density of the corridors that we were looking for (it is equal to the number of users that passed through the same corridor) has been fixed and a great match between these and the points of interest identified was found.

Experiments confirm the possible hypothesis that the vehicles have been booked to reach a site of interest or after visiting one of these to leave another destination, also being a popular place.

The POIs found were verified by means of Google Maps and compared with a curated list brought to us by a native of the analysed places. For each dataset used, consisting of data referring different categories of people and movements, it was shown that the points of interests are compatible with the nature of the dataset and the behaviour of the users who collected them. In fact, Points Of Interests have been found in service areas, supplies and spare parts sales outlets with regard to users with trucks; service buildings and companies for businessmen travelling by taxi; and places of study and work for participants of the Microsoft Research Asia Geolife project.

A third path followed during my PhD, that in particular I started during a period of study and research at the Universitat de Barcelona in Spain under the supervision of Professor J. Vitrià, concerns the identification of the corridors, which are shared by a considerable number of users.

The dataset has been discretized in order to solve the problem by applying the Apriori algorithm. Having an iterative layer-by-layer approach, it allowed us to check more quickly which common routes were frequent, starting only from frequent sub-trajectories smaller than the ones we wanted to identify and excluding the others. The new approach has reduced execution times compared to a comparison between pairs of trajectories, in which the discrete Fréchet distance or other similarity measures between curves are used.

The total run time for Apriori algorithm in every step is defined as (see [89] Suneetha and Krishnamoorti, 2011):

$$\sum_{i=1}^{n}(t_s * m_k + t_c + p_{k+1} * k + 1/2 * t_s * n_k/B)$$

where, t_s is the time cost of a single scan of the database, t_c is the time cost of generating C_{k+1} from L_k, m_k is set to be the amount of itemsets in C_k, the variable $p_k + 1$ is set to be the amount of itemsets in C_{k+1} and the variable n_k is set to be the amount of itemsets in L_k. B is the number of records in the database and n represents the dimension of the data. The Apriori execution time, with minimum support of 15 trajectories in Step 1 is 1 h and 22 min; for min_sup of 51 cells in Step 1.2 the execution time is 21 min and 44 s; for $1,015$ random trajectories, with min_sup of 50 trajectories in Step 2: 297 ms.

A subsequent filtering of the results, according to a desired tolerance, returned us outputs cleaned from initial discretization errors.

The application of the Apriori algorithm and the subsequent filtering of the results have this interpretation: If different users start at the same point P in the city with $\pm$ a certain radius r, with r smaller than or equal to the side of the cell of the grid, and they share a second consecutive cell, or more than one, then there is a high probability that such individuals move together on the same road to reach another point in the same time slot. The corridor will end when any of these N trajectories will not share the next cell (when the number of trajectories "traveling" close together becomes less than the min_sup threshold).

Finally, we have proposed an approach for predicting how many people move to certain destinations when it is known that a certain amount of people is in some other place. We use an app that detects the location of people and sends such data to a server. So, being able to calculate how likely a group of people will move, along with the previous stats, is useful for estimating the amount of people in another specific place to track later. Having an educated guess on the amount of people that will gather in some place before planning a trip can be very useful to avoid overcrowded places and to keep with the current regulations.

6.2 Conclusions

It is possible to calculate the similarity between users simply by checking the number of times each pair of individuals has visited the same region (POI). For this purpose, we can use the Jaccard similarity between users, by calculating the set of intersections and the union set of the regions visited for each couple of users. Computing the similarity of people based on their position history is useful to understand the interaction of the individuals in a particular region. Since GeoLife is a dataset composed primarily by GPS logs of academic individuals, here POIs can be a university building, a residence, a restaurant or a research center. In fact, for the city of Beijing, most of the POIs are being built around Peking University, Tsinghua University, Microsoft Research Asia and some leisure places like Yuanmingyaun Park.

For individuals with a high similarity, they are likely to know and have each other similar visit / travel preferences. For the GeoLife dataset, these people can be, for example, university colleagues. This approach can be useful for suggestions from friends in social networks. Facebook, for example, suggests a friend based on the number of friends that two users have in common. Likewise, this approach can be applied in the same way in a social network based on geographic location about. In addition, this strategy can be applied for different GPS logs or other sources of geographic information than by users, in order to find the similarity of people based on their geographic location record, StayPoints or Points Of Interest.

The use of spatio-temporal information aims to understand the movement patterns (corridors) of voluntary users in GeoLife and it can help to make an appropriate recommendation (to know, for example, in which time slot you mostly visit a certain urban area through certain roads).

Therefore, the discovery of the flows in cities is very important as it has a wide range of applications: for example, it can facilitate the planning and optimization of transport services even in mass events to avoid the traffic.

6.3 Open Issues

Another strategy is to use QuickBundles (QB) for clustering the trajectories for each user. With QB we could find for each individual few prototypes of trajectories (at each trajectory it assigns its closest trajectories) and finally compare pairs of clustered trajectories with the Discrete Fréchet Distance.

QuickBundles, which is used in neuroimaging, finds dense road segments (i.e., with high number of trajectories) and merges them into dense routes (with similarity and minimizing the distance. At the end we will have to filter the trajectories that have low Fréchet distance (or also with Radius Neighbors Graph, with the help of the adjacency matrix). These methods will be the topics of future research.

This analysis could have many interesting applications: for instance it would be possible to create a social network to connect people in the same travel location, give travel recommendation or point out the lacking of public transport lines in certain areas to the city council.

The experiments that we have performed on previously gathered geographical locations have shown the viability and reliability of our approach. The more people use the app (our agent) the more the approach would give a correct estimate. To make the approach more robust, it could be extended in order to include data available online from other services that give indications on queues, road traffic and gatherings.

Future work can take into account the geometry of some stations, museums, etc. of some popular destinations to compute the average distance of people given the estimated amount of people gathering. Moreover, alerts about overcrowding could be sent to both the people present in some place for a peak of incoming people, occurring later on, and the people that are moving towards there.

Bibliography

[1] Agrawal, R.; Srikant, R. *Fast Algorithms for Mining Association Rules in Large Databases.* 20th International Conference on Very Large Data Bases. Morgan Kaufmann, Los Altos, CA, 1994, pp. 478–499.

[2] Agrawal, R.; Srikant, R. *Privacy-preserving data mining.* ACM Sigmod Record, 29(2):439–450, 2000.

[3] Ahas, R.; Silm, S.; Järv, O.; Saluveer, E.; Tiru, M. *Using mobile positioning data to model locations meaningful to users of mobile phones.* Journal of urban technology, 17(1):3–27, 2010.

[4] Aliannejadi, M.; Crestani F. *Personalized context-aware point of interest recommendation.* ACM Transactions on Information Systems (TOIS), 36(4):45, 2018.

[5] Alt, H.; Godau, M.*Measuring the resemblance of polygonal curves.* In Proc. 8th Annu. ACM Sympos. Comput. Geom., pages 102–109, 1992.

[6] Ankerst, M.; Breunig, Markus M.; Kriegel, Hans-Peter; Sander, J. *OPTICS: Ordering Points To Identify the Clustering Structure.* ACM SIGMOD international conference on Management of data. ACM Press. pp. 49–60, 1999.

[7] Batet, M.; Moreno, A; Sánchez, D.; Isern, D.; Valls, A. *Turist@: Agent-based personalised recommendation of tourist activities.* Expert Systems with Applications, 39(8):7319–7329, 2012.

[8] Berndt, D. J.; Clifford, J. *Using dynamic time warping to find patterns in time series.* In KDD Workshop, pp. 359–370. AAAI Press, 1994.

[9] Berzi, C.; Gorrini, A.; Vizzari, G. *Mining the social media data for a bottom-up evaluation of walkability.* In Proceedings of International Conference on Traffic and Granular Flow, pages 167–175. Springer, 2017.

[10] Bian, J.; Tianb, D.; Tangc, Y.; Taod, D. *A survey on trajectory clustering analysis.* arXiv preprint, [arXiv:quant-ph/1802.06971].

[11] Bicocchi, N.; Mamei, M. *Investigating ride sharing opportunities through mobility data analysis.* Pervasive Mobile Computing 14 (2014) 83–94.

[12] Bicocchi, N.; Mamei, M.; Sassi, A.; Zambonelli, F. *On Recommending Opportunistic Rides.* IEEE Transactions on Intelligent Transportation Systems 2017, 18, 3328–3338.

[13] Bobadilla, J.; Ortega, F.; Hernando, A.; Gutiérrez, A. *Recommender systems survey.* Knowledge-based systems, 46:109–132, 2013.

[14] Boutilier, C. *Sequential optimality and coordination in multiagent systems.* In Proceedings of International Joint Conference on Artificial Intelligence (IJCAI), volume 99, pages 478–485, 1999.

[15] Breese, J. S.; Heckerman, D; Kadie, C. *Empirical analysis of predictive algorithms for collaborative filtering.* In Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence, pages 43–52. Morgan Kaufmann Publishers Inc., 1998.

[16] Buchin, K.; Buchin, M.; Gudmundsson, J.; Löffler, M.; Luo, J. *Detecting Commuting Patterns by Clustering Subtrajectories.* International Journal of Computational Geometry & Applications 2011, 21, 253–282. doi:10.1142/S0218195911003652.

[17] Harabasz, Calinski T.; Karoński, M. *A dendrite method for cluster analysis.* Communications in Statistics. Vol. 3. No. 1. 1974. 1-27.

[18] Canny, J. *Collaborative filtering with privacy.* In Proceedings IEEE Symposium on Security and Privacy, pages 45–57, 2002.

[19] Canny, J. *Collaborative filtering with privacy via factor analysis.* In Proceedings of international ACM SIGIR conference on Research and development in information retrieval, pages 238–245, 2002.

[20] Castro, P.; Zhang, D.; Li, S. *Urban traffic modelling and prediction using large scale taxi gps traces*, in Pervasive Computing, pp. 57–72, 06 2012.

[21] Cavallaro, C.; Verga, G.; Tramontana, E.; Muscato, O. *Multi-Agent Architecture for Point of Interest Detection and Recommendation.* In Proceedings of the 20th Workshop From Objects to Agents, Parma, Italy. CEUR Workshop Proceedings, 2019, Vol. 2404, pp. 98–104.

[22] Cavallaro, C.; Verga, G.; Tramontana, E.; Muscato, O. *Eliciting Cities Points of Interest from People Movements and Suggesting Effective Itineraries.* Intelligenza Artificiale, 14 (1), 49-61, 2020.

[23] Cavallaro, C.; Verga, G.; Tramontana, E.; Muscato, O. *Suggesting Just Enough (Un)Crowded Routes and Destinations.* In Proceedings of the 21th Workshop From Objects to Agents, Bologna, Italy. CEUR Workshop Proceedings, 2020. Vol. 2706, pp. 237-251.

[24] Cavallaro, C.; Vitrià, J. *Corridor Detection from Large GPS Trajectories Datasets.* Applied Sciences. 2020; 10(14):5003.

[25] Chee, S. H. S. ; Han, J. ; Wang, K. *Rectree: An efficient collaborative filtering method.* In Proceedings of International Conference on Data Warehousing and Knowledge Discovery, pages 141–151. Springer, 2001.

[26] Chen, C.-C.; Tsai, J.-L. *Determinants of behavioral intention to use the personalized location-based mobile tourism application: An empirical study by integrating tam with issm.* Future Generation Computer Systems, 2017.

[27] Chessa, S.; Girolami,; M.; Foschini,L.; Ianniello, R.; Corradi, A.; Bellavista, P. *Mobile Crowd Sensing Management with the ParticipAct Living Lab.* Pervasive and Mobile Computing (2016), http://dx.doi.org/10.1016/j.pmcj.2016.09.005

[28] Cranshaw, J.; Schwartz, R.; Hong, J. Sadeh, N. *The livehoods project: Utilizing social media to understand the dynamics of a city.* In Proceedings of International AAAI Conference on Weblogs and Social Media, 2012.

[29] Crociani, L.; Vizzari, G.; Gorrini, A.; Bandini, S., *Identification and Characterization of Lanes in Pedestrian Flows Through a Clustering Approach.* In AI*IA 2018 – Advances in Artificial Intelligence. AI*IA 2018. Lecture Notes in Computer Science; Springer Verlag, 2018; Vol. 11298, pp. 71–82. doi:10.1007/978-3-030-03840-3_6.

[30] Cruz, J.; Silva, E.; Rossetti, R. J.; Silva, D. C.; Oliveira, E. C. ; Neto, J. *Application of multi-agent systems to shared transport services: A review.* In Proceedings of Iberian Conference on Information Systems and Technologies (CISTI), pages 1–6. IEEE, 2018.

[31] Davies, D. L.; Bouldin, D. W. (1979). *A Cluster Separation Measure.* IEEE Transactions on Pattern Analysis and Machine Intelligence. PAMI-1 (2): 224–227.

[32] De Nijs, F.; Theocharous, G.; Vlassis, N.; De Weerdt, M. M.; Spaan, M. T. J. *Capacity-aware sequential recommendations.* In Proceedings of International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), 2018, p. 416–424.

[33] Devogele, T.; Etienne, L.; Esnault, M.; Lardy, F. *Optimized Discrete Fréchet Distance between trajectories.* In Proceedings of the 6th ACM SIGSPATIAL Workshop on Analytics for Big Geospatial Data; Association for Computing Machinery: New York, NY, USA, 2017; BigSpatial'17, p. 11–19. doi:10.1145/3150919.3150924.

[34] Earl, R. W. *Book reviews : Models of man: Social and rational* by Herbert A. Simon. New York: John Wiley & sons, 1957. pp. xiv 287. Educational and Psychological Measurement, vol. 21, no. 1, pp. 244–246, 1961.

[35] Eiter, T.; Mannila, H. *Computing discrete Fréchet distance.* Technical report, Technische Universitat Wien, 1994.

[36] Ester, M.; Kriegel, H.-P.; Sander, J.; Xu, X. et al. *A density-based algorithm for discovering clusters in large spatial databases with noise.* In Proceedings of International Conference on Knowledge Discovery and Data Mining, pages 226–231, 1996.

[37] Fano, A. E. *Shopper's eye: using location-based filtering for a shopping agent in the physical world.* In Agents, volume 98, pages 416–421, 1998.

[38] Forgy, Edward W. (1965). *Cluster analysis of multivariate data: efficiency versus interpretability of classifications.* Biometrics. 21 (3): 768–769.

[39] Fréchet, M.*Sur quelques points du calcul fonctionnel.* Rendiconti del Circolo Mathematico di Palermo, 22:1–74, 1906.

[40] Gambs, S.; Killijian, M.-O.; del Prado Cortez, M. N. *Next place prediction using mobility markov chains.* In Proceedings of the First Workshop on Measurement, Privacy, and Mobility, page 3. ACM, 2012.

[41] Gedik, B.; Liu, L. *Location privacy in mobile systems: A personalized anonymization model.* In Proceedings of IEEE International Conference on Distributed Computing Systems (ICDCS), pages 620–629, 2005.

[42] GeoLife, Microsoft Research Asia, http://www.microsoft.com/en-us/download/details.aspx?id=52367

[43] Giunta, R.; Pappalardo, G.; Tramontana, E. *Aspects and annotations for controlling the roles application classes play for design patterns.* In Proc. of IEEE Asia Pacific Software Engineering Conference (APSEC), pages 306–314, Ho Chi Minh, Vietnam, December 2011.

[44] Goldberg, D.; Nichols, D.; Oki, B. M. ; Terry, D. *Using collaborative filtering to weave an information tapestry.* Communications of the ACM, 35(12):61–71, 1992.

[45] Han, J.; Kamber, M.; Pei, J. *Data Mining: Concepts and Techniques*, 3rd ed.; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 2011.

[46] Hernández, J. Z.; Ossowski, S.; Garcıa-Serrano, A. *Multiagent architectures for intelligent traffic management systems.* Transportation Research Part C: Emerging Technologies, 10(5-6):473–506, 2002.

[47] Hosseinpoor, A.H.; Ali Abbaspour, R.; Claramunt, C. *A Geometric Framework for Detection of Critical Points in a Trajectory Using Convex Hulls.* ISPRS Int. J. Geo-Inf 2018, 7, 14. doi:10.3390/ijgi7010014.

[48] Hsieh, H.-P.; Li, C.-T.; Lin, S.-D. *Exploiting large-scale check-in data to recommend time-sensitive routes.* In Proceedings of the ACM SIGKDD International Workshop on Urban Computing, pages 55–62. ACM, 2012.

[49] Hsueh, Y.-L.; Huang, H.-M. *Personalized itinerary recommendation with time constraints using gps datasets.* Knowledge and Information Systems, 60(1):523–544, 2019.

[50] Hu, Y.; Gao, S.; Janowicz, K.; Yu, B.; Li, W.; Prasad, S. *Extracting and understanding urban areas of interest using geotagged photos.* Computers, Environment and Urban Systems, 54:240–254, 2015.

[51] Hu, M.; Liu, B. *Mining and summarizing customer reviews.* In Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2004). 2004.

[52] Hwang, J.-R.; Kang, H.-Y.; Li, K.-J. *Spatio-temporal similarity analysis between trajectories on road networks.* In Proceedings of International Conference on Conceptual Modeling, pages 280–289. Springer, 2005.

[53] Inman, J. *Navigation and Nautical Astronomy, for the Use of British Seamen.* F. & J. Rivington: London, UK, 1849.

[54] Isaacman, S.; Becker, R.; Cáceres, R.; Kobourov, S.; Martonosi, M.; Rowland, J.; Varshavsky, A. *Identifying important places in people's lives from cellular network data.* In Proceedings of International Conference on Pervasive Computing, pages 133–151. Springer, 2011.

[55] Jiang, T.; Wang, H. J.; Hu, Y.-C. *Preserving location privacy in wireless lans.* In Proceedings of International conference on Mobile systems, applications and services, pages 246–257, 2007.

[56] Jiménez-Bravo, D. M.; Pérez-Marcos, J.; De la Iglesia, D. H.; Villarrubia González, G.; De Paz, J. F. *Multi-agent recommendation system for electrical energy optimization and cost saving in smart homes.* Energies, 12(7):1317, 2019.

[57] Kalnis, P.; Ghinita, G.; Mouratidis, K.; Papadias, D. *Preventing location-based identity inference in anonymous spatial queries.* IEEE transactions on knowledge and data engineer ing, 19(12):1719–1733, 2007.

[58] Lau, B. P. L.; Hasala, M. S.; Kadaba, V. S.; Thirunavukarasu, B.; Yuen, C.; Yuen, B.; Nayak, R. *Extracting point of interest and classifying environment for low sampling crowd sensing smartphone sensor data.* In Proceedings of IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops), pages 201–206, 2017.

[59] Lee, J.; Han, J.; Li, X. *A unifying framework of mining trajectory patterns of various temporal tightness*, IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 6, pp. 1478–1490, 2015.

[60] Liu, B. *Sentiment Analysis and Subjectivity.* Handbook of Natural Language Processing, Second Edition, N. Indurkhya and F.J. Damerau, Eds. 2010.

[61] Liu, B. *Sentiment analysis and opinion mining.* Synthesis lectures on human language technologies, 5(1):1–167, 2012.

[62] Liu, B.; Fu, Y.; Yao, Z.; Xiong, H. *Learning geographical preferences for point-of-interest recommendation.* In Proceedings of ACM SIGKDD international conference on Knowledge discovery and data mining, pages 1043–1051, 2013.

[63] Liu, Q.; Gao, L.; Lou, P. *Resource management based on multi-agent technology for cloud manufacturing*, in 2011 International Conference on Electronics, Communications and Control (ICECC), pp. 2821–2824, IEEE, 2011.

[64] Lops, P.; De Gemmis, M.; Semeraro, G. *Content-based recommender systems: State of the art and trends.* In Recommender systems handbook, pages 73–105. Springer, 2011.

[65] Mamei, M.; Colonna, M.; Galassi, M. *Automatic identification of relevant places from cellular network data.* Pervasive and Mobile Computing, 2016, https://doi.org/10.1016/j.pmcj.2016.01.009

[66] Mathew, W.; Raposo, R.; Martins, B. *Predicting future locations with hidden markov models.* In Proceedings of ACM conference on ubiquitous computing, pages 911–918, 2012.

[67] Melville, P.; Sindhwani, V. *Recommender systems.*, Encyclopedia of machine learning, vol. 1, pp. 829–838, 2010.

[68] Miyahara, K. ; Pazzani, M. J. *Collaborative filtering with the simple bayesian classifier.* In Proceedings of Pacific Rim International conference on artificial intelligence, pages 679–689. Springer, 2000.

[69] Mohamed, F.; Chi-Yin, C.; Walid, G. *The new casper: a privacy-aware location-based database server.* In Proceedings of International Conference on Data Engineering, pages 1499–1500, 2007.

[70] Narayanan, A.; Thiagarajan, N.; Lakhani, M.; Hamburg, M.; Boneh, D. et al. *Location privacy via private proximity testing.* In NDSS, volume 11, 2011.

[71] Nasukawa, T.; Jeonghee Y. *Sentiment analysis: Capturing favorability using natural language processing.* In Proceedings of the KCAP-03, 2nd Intl. Conf. on Knowledge Capture. 2003.

[72] Papadopoulos, S.; Bakiras, S.; Papadias, D. *Nearest neighbor search with strong location privacy.* volume 3, pages 619– 629. VLDB Endowment, 2010.

[73] Pérez-Torres, R. ; Torres-Huitzil,C. ; Galeana-Zapién, H. *Full on device stay points detection in smartphones for location-based mobile applications.* Sensors, 16(10):1693, 2016.

[74] Piorkowski, M.; Sarafijanovic-Djukic, N.; Grossglauser, M. *CRAWDAD dataset epfl/mobility(v. 2009-02-24)*, 2009. doi:10.15783/C7J010.

[75] Polat H.; Du, W. *Svd-based collaborative filtering with privacy.* In Proceedings of ACM Symposium on Applied Computing (SAC), pages 791–795, 2005.

[76] Quercia, D.; Schifanella, R.; Aiello, L. M. *The shortest path to happiness: Recommending beautiful, quiet, and happy routes in the city.* In Proceedings of ACM conference on Hypertext and social media, pages 116–125, 2014.

[77] Reiter, M. K.; Rubin, A. D. *Crowds: Anonymity for web transactions.* ACM transactions on information and system security (TISSEC), 1(1):66–92, 1998.

[78] Rolim, V.B.; Silva, M.R.; Flamarion, F.C.; Filho, C.C.G.F.; Braz, F.J. *A Method for Identifying Patterns of Movement of Trajectory Sets by Using the Frequency Distribution of Points.* GEOProcessing 2017 : The Ninth International Conference on Advanced Geographic Information Systems, Applications, and Services, 2017. ISBN: 978-1-61208-539-5.

[79] Rousseeuw, P. J. *Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis.* Computational and Applied Mathematics. 20: 53–65, (1987).

[80] Sakamanee, P.; Phithakkitnukoon, S.; Smoreda, Z.; Ratti, C. *Methods for Inferring Route Choice of Commuting Trip From Mobile Phone Network Data.* ISPRS Int. J. Geo-Inf. 2020, 9, 306.

[81] Salton, G.; Buckley, C. *Term-weighting approaches in automatic text retrieval.* In Information Processing & Management, 24(5), 513–523, 1988.

[82] Sarwar, B. M.; Konstan, J. A.; Borchers, A.; Herlocker, J.; Miller, B.; Riedl, J. *Using filtering agents to improve prediction quality in the grouplens research collaborative filtering system.* In Proceedings of the ACM Conference on Computer Supported Cooperative Work (CSCW), GroupLens Research Collaborative Filtering System, 1998.

[83] Schafer, J. B.; Frankowski, D.; Herlocker, J.; Sen, S. *Collaborative filtering recommender systems.* In The adaptive web, pages 291–324. Springer, 2007.

[84] Sebastia, L.; Giret, A.; Garcia, I. *A multi agent architecture for tourism recommendation.* In Trends in Practical Applications of Agents and Multiagent Systems, pages 547–554. Springer, 2010.

[85] Shen, W.; Norrie, D. *Facilitators, mediators or autonomous agents.* In Proceedings of the Second International Workshop on CSCW in Design, Bangkok, Thailand, pp. 119–124, 1997.

[86] Sinnott, R.W. *Virtues of the Haversine.* Sky Telesc. 68, 159, 1984.

[87] Sycara, K. P. *Multiagent systems*, AI Magazine, vol. 19, p. 79, Jun. 1998.

[88] Su, X.; Khoshgoftaar, T. M. *Collaborative filtering for multi-class data using bayesian networks.* International Journal on Artificial Intelligence Tools, 17(01):71–85, 2008.

[89] Suneetha, K.; Krishnamoorti, R. *Web Log Mining using Improved Version of Apriori Algorithm.* International Journal of Computer Application 2011, 29, 23–27. doi:10.5120/3569-4923.

[90] Syverson, P. F.; Goldschlag, D. M.; Reed, M. G. *Anonymous connections and onion routing.* In Proceedings of IEEE Symposium on Security and Privacy, pages 44–54, 1997.

[91] Taxi and Truck datasets, http://mashuai.buaa.edu.cn/traj.html

[92] Tramontana, E. *Minimising changes when refactoring applications to run multiple threads.* In Proceedings of IEEE Asia-Pacific Software Engineering Conference (APSEC), Nara, Japan, 2018, pp. 713–714.

[93] Tramontana, E.; Verga, G. *Get spatio-temporal flows from gps data.* In Proceedings of IEEE International Conference on Smart Computing (SMARTCOMP), pages 282–284, 2018.

[94] Trasarti, R.; Rinzivillo, S.; Pinelli, F.; Nanni, M.; Monreale, A.; Renso, C.; Pedreschi, D.; Giannotti, F. *Exploring real mobility data with m-atlas*, in Machine Learning and Knowledge Discovery in Databases (J. L. Balc ´azar, F. Bonchi, A. Gionis, and M. Sebag, eds.), pp. 624– 627, Springer, 2010.

[95] Ungar, L. H.; Foster, D. P. *Clustering methods for collaborative filtering.* In Proceedings of AAAI workshop on recommendation systems, volume 1, pages 114–129, 1998.

[96] Wang, Z.; Lu, M.; Yuan, X.; Zhang, J.; Wetering, H. v. d. *Visual traffic jam analysis based on trajectory data*, IEEE Transactions on Visualization and Computer Graphics, vol. 19, no. 12, pp. 2159–2168, 2013

[97] Wooldridge, M.; Jennings, N. R. *Intelligent agents and multi-agent systems. i*, Applied artificial intelligence, vol. 9, no. 4, 1995.

[98] Xu R.; Wunsch D.*Survey of clustering algorithms.* Trans. Neur. Netw. 16, 3 (May 2005), 645–678.

[99] Xuan, P.; Lesser, V. *Multi-agent policies: from centralized ones to decentralized ones.* In Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3, pages 1098–1105. ACM, 2002.

[100] Xuan, P.; Lesser, V.; Zilberstein, S. *Communication decisions in multi-agent cooperation: Model and experiments.* In Proceedings of the fifth international conference on Autonomous agents, pages 616–623. ACM, 2001.

[101] Yoon, H.; Zheng, Y.; Xie, X.; Woo W. *Smart itinerary recommendation based on user-generated gps trajectories.* In Proceedings of International Conference on Ubiquitous Intelligence and Computing, pages 19–34. Springer, 2010.

[102] Yuan, Q.; Cong, G; Ma, Z.; Sun, A.; Thalmann, N. M. *Time-aware point-of-interest recommendation.* In Proceedings of the international ACM SIGIR conference on Research.

[103] Zheng, Y. *Trajectory data mining: An overview.* ACM Trans. Intell. Syst. Technol. 2015, 6, Article 29. doi:10.1145/2743025.

[104] Zheng, Y.; Li, Q.; Chen, Y.; Xie, X.; Ma, W.Y. *Understanding Mobility Based on GPS Data.* In Proceedings of ACM conference on Ubiquitous Computing (UbiComp 2008), Seoul, Korea. Association for Computing Machinery, 2008, pp. 312–321. doi:10.1145/1409635.1409677.

[105] Zheng, Y.; Xie, X.; Ma, W.Y. *GeoLife: A Collaborative Social Networking Service among user, location and trajectory.* IEEE Data(base) Engineering Bulletin (2010).

[106] Zheng, Y.; Zhang, L.; Xie, X.; Ma, W.Y. *Mining interesting locations and travel sequences from GPS trajectories.* In Proceedings of International conference on World Wild Web (WWW 2009), Madrid, Spain. Association for Computing Machinery, 2009, pp. 791–800. doi:10.1145/1526709.1526816.

[107] Zou, Y.; Chen, Y.; He, J.; Pang, G.; Zhang, K. *4D Time Density of Trajectories: Discovering Spatiotemporal Patterns in Movement Data.* ISPRS Int. J. Geo-Inf 2018, 7, 212. doi:10.3390/ijgi7060212.

[108] Zygouras, N.; Gunopulos, D. *Corridor Learning Using Individual Trajectories.* 19th IEEE International Conference on Mobile Data Management (MDM). IEEE, 2018, pp. 155–160. doi:10.1109/MDM.2018.00032.