International Workshop on Finding Patterns of Human Behaviors in Networks and Mobility Data

NEMO’11

September 9, 2011

Athens, Greece
Editors:
Albert-László Barabási
CCNR, Northeastern University, USA
Michele Berlingerio
KDD Lab, ISTI-CNR Pisa, Italy
Dino Pedreschi
KDD Lab, Department of Computer Science, University of Pisa, Italy
Dashun Wang
CCNR, Northeastern University, USA
Preface

The International Workshop on Finding Patterns of Human Behaviors in NEtwork and MObility Data (NEMO 2011) was held in Athens, Greede, on September 9th 2011 in conjunction with the The European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2011).

In the past decade, network theory has revolutionized our understanding of systems of interacting objects, impacting a wide array of disciplines. In the area of complex systems we are witnessing yet another seismic shift. Indeed, thanks to the increasing availability of large-scale data in many diverse settings, researchers now have access to patterns of human behavior at an unprecedented level of details. These large-scale datasets, offering objective description on human activity patterns, are expected to revolutionize our understanding of human behavior.

Researchers have attempted to study, measure, model and predict human behaviors from many different perspectives. For example, much effort has been devoted to understanding how people connect, interact, and exchange information with others. At the same time, there has been wide and multidisciplinary research on understanding human mobility patterns: where do people go? How fast do they move? How regular are their movements? These advances in social networks and human mobility have also turned the interplay between these two aspects into an emergent focus in our understanding of human behavior.

The aim of the workshop was to bring together pioneering researchers in the fields of data mining and machine learning who are focusing on above topics, and thus intensify the exchange of ideas among different research communities to foster devising tools and models for creation, analysis, and visualization of network and mobility data.

Eleven contributions were originally submitted, eight of which were accepted for presentation. Each submission was evaluated by at least two independent referees. Besides paper presentations, the scientific programme also featured an interactive panel session involving all the authors, the audience, and the workshop chairs.

We would like to thank all the authors who submitted papers and all the workshop participants. We are also grateful to members of the program committee members and external referees for their thorough work in reviewing submitted contributions with expertise and patience. A special thank is due to both the ECML PKDD Workshop Chairs and the members of ECML PKDD Organizing Committee who made this event possible.

Athens, September 2011

Albert-László Barabási
Michele Berlingerio
Dino Pedreschi
Dashun Wang
Workshop Organization

Workshop Chairs
Albert-László Barabási  CCNR, Northeastern University, USA
Michele Berlingerio  KDD Lab, ISTI-CNR Pisa, Italy
Dino Pedreschi  KDD Lab, Department of Computer Science, University of Pisa, Italy
Dashun Wang  CCNR, Northeastern University, USA

Sponsor and Award Chair
Mirco Nanni  KDD Lab, ISTI-CNR Pisa, Italy

Program Committee
Leman Akoglu  Carnegie Mellon University
James P. Bagrow  CCNR, Northeastern University
Bjørn Bringmann  Deloitte & Touch GmbH, Munich, Germany
Tanya Berger-Wolf  University of Illinois at Chicago
Vincent D. Blondel  Université catholique de Louvain
Nick Blumm  CCNR, Northeastern University
Ilaria Bordino  Yahoo! Research
Dirk Brockmann  Northwestern University
Michele Coscia  University of Pisa
Marta C. Gonzalez  CEE, Massachusetts Institute of Technology
César A. Hidalgo  Media Lab, Massachusetts Institute of Technology
U Kang  Carnegie Mellon University
David Lazer  Dept of Political Science and Computer Science, Northeastern University
Sune Lehmann  Technical University of Denmark
Ching-Yung Lin  IBM TJ Watson Research Center
Yu-Ru Lin  IQSS, Harvard University
Andreas Nürnberger  Otto-von-Guericke-Universität Magdeburg
Tina Eliassi-Rad  Rutgers University
Fabio Pinelli  KDDLab ISTI-CNR Pisa
Salvatore Rinzivillo  KDDLab ISTI-CNR Pisa
Xiaolin Shi  Stanford University
Chaoming Song  CCNR, Northeastern University
Pedro Olmo Stancioli Vaz  Federal University of Minas Gerais
de Melo
Shirish Tatikonda  IBM Almaden
Hanghang Tong  IBM TJ Watson Research Center
Roberto Trasarti  KDDLab ISTI-CNR Pisa
Alessandro Vespignani  School of Informatics and Computing, Indiana University
Pu Wang  Massachusetts Institute of Technology
Jiang Yang  University of Michigan
Philip Yu  University of Illinois at Chicago
Wen Zhen  IBM TJ Watson Research Center
Augmented Betweenness Centrality for Mobility Prediction in Transportation Networks

Yaniv Altshuler¹, Rami Puzis², Yuval Elovici², Shlomo Bekhor³, and Alex (Sandy) Pentland¹

¹ MIT Media Lab
{yanival, sandy}@mit.edu

² Deutsche Telekom Lab, Department of Information Systems Eng., Ben-Gurion University
{puzis, elovici}@bgu.ac.il

³ Transportation Research Institute, Technion
sbekhor@technion.ac.il

Abstract. Measuring and predicting the human mobility along the links of a transportation network has always been of a great importance to researchers in the field. Hitherto, producing such data relied heavily on expensive and time consuming surveying and on-field observational methods. In this work we propose an efficient estimation method for the assessment of the flow through links in transportation networks that is based on the Betweenness Centrality measure of the network’s nodes. Furthermore, we show that the correlation between those two features can be significantly increased when additional (pre-defined and known) properties of the network are taken into account, generating an augmented Mobility Oriented Betweenness Centrality measure. We validate the results using a transportation dataset, constructed using cellular phones data, that contains a high resolution network of the Israeli transportation system. We show that the flow that was measured using this expensive and complicated method can be accurately estimated using our proposed Augmented Betweenness technique.

1 Introduction

The analysis of mobility trends and demands forecasting in transportation networks relies heavily on household survey data that provides the required input for calibrating the mathematical models that represent decisions people make related to travel [1]. However, a well known problem common to all interview-type surveys is non-response. Complex methods to correct for non-response have been developed, however, these alleviate the problem only partially [2].

As mentioned in [3], another limitation of household surveys is the need for active cooperation from the respondents, relying on their memory and patience. The need for active participation reduces the ability to capture complex travel and activity patterns, and the ability to collect data over a long period of time. The problems mentioned above, coupled with budget constraints, explain the fact that typical household surveys collect data regarding a period of merely one or two days for each household.
As a result, there exists a strong need for finding an alternative mechanism of assessing mobility and traffic demand in transportation networks, one that could be used without the necessary, tedious and inaccurate process of surveying.

Betweenness Centrality (BC) stands for the ability of an individual node to control the communication flow in the networks \([4, 5]\). Formally, for a node \(v\) it denoted the total portion of shortest-paths between every pair of nodes in the network that pass through \(v\) (see more details in Section 3). In recent years Betweenness was extensively applied for the analysis of various complex networks \([6, 7]\) including among others social networks \([8, 9]\), computer communication networks \([10, 11]\), and protein interaction networks \([12]\). Holme \([13]\) have shown that Betweenness is highly correlated with congestion in particle hopping systems. Extensions of the original definition of BC are applicable for directed and weighted networks \([14, 15]\) as well as for multilayer networks where the underlying infrastructure and the origin-destination overlay are explicitly defined \([16]\).

In this paper we discuss the applicability of BC and certain augmented types of it for the prediction of mobility patterns in transportation networks. Specifically, we show that there is a strong positive correlation between a traffic that flows through a node in a transportation network and its BC measures. In this study we use a comprehensive transportation network of the Israeli roads and highways system, containing over 15,000 directed links.

The rest of the paper is organised as follows: Section 2 describes the transportation data that was used in this study. Section 3 discusses the correlation between betweenness centrality and traffic flow, whereas concluding remarks appear in Section 4.

2 Transportation Network Dataset

The widespread use of cellular phones in Israel enables the collection of accurate transportation data. Given the small size of the country, all cellular companies provide national wide coverage. As shown in \([3]\), the penetration of cellular phones to the Israeli market is very high, even to lower income households, and specially among individuals in the ages of 10 to 70 (the main focus of travel behavior studies). Such penetration enables a comprehensive study of travel behavior that is based on the mobility patterns of randomly selected mobile phones in the Israeli transportation system. This data was shown in \([3]\) and \([17]\) to provide a high quality coverage of the network, tracking 94% of the trips (defined as at least 2km in urban areas, and at least 10km in rural areas). The resulting data contained a wealth of traffic properties for a network of over 6,000 nodes, and 15,000 directed links. In addition, the network was accompanied with an Origin Destination (OD) matrix, specifying start and end points of trips.

The network was created for the National Israeli Transportation Planning Model. In urban areas the network contains arterial streets that connect the interurban roads. For each link of the network, there is information about the length (km), hierarchical type, free-flow travel time (min), capacity (vehicles per hour), toll (min), hourly flow (vehicles per hour), and congested travel time (min). The hourly flows and congested travel times were obtained from a traffic assignment model that loads the OD matrix on the network links.
2.1 Network Structure

Based on the dataset described above we have created a network structure, assigning running indices from 1 to 6716 to the nodes (junctions). We have examined the directed variant of the network where each road segment between two junctions was represented as either one or two directed links between the respective nodes.

In order to get a basic understanding of the network we first extracted and studied several of its structural properties (see Table 1). We have partitioned the network into structural equivalence classes of the nodes and bi-connected components and computed the betweenness centrality indices of the nodes [18, 19, 4]. Structurally equivalent vertices have exactly the same neighbors and the set of these vertices is called a structural equivalence class. As can be seen from Table 1 the number of structural equivalence classes is roughly the number of vertices in the network and the size of the largest class is three. This means that there are no “star-like” structures in the network and alternative paths between any two vertices are either longer than two hops or have other links emanating from the intermediate vertices. On the other hand the number of biconnected components in the network is low compared to the number of nodes, meaning that there are significant regions of the network that can be cut out by merely disconnecting a single node.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Nodes</td>
<td>6716</td>
</tr>
<tr>
<td>Edges (undirected representation)</td>
<td>8374</td>
</tr>
<tr>
<td>Edges (directed representation)</td>
<td>15823</td>
</tr>
<tr>
<td>Number of structural equivalence classes</td>
<td>6655</td>
</tr>
<tr>
<td>Largest equivalence class</td>
<td>3</td>
</tr>
<tr>
<td>Number of bi-connected components (BCC)</td>
<td>931</td>
</tr>
<tr>
<td>Avg BCC size</td>
<td>8.2</td>
</tr>
<tr>
<td>Largest BCC</td>
<td>5778</td>
</tr>
</tbody>
</table>

2.2 Congestions

In this paper we define the impact of congestion as the difference between the time to travel through a congested link and the free-flow time to travel. Congestion of a junction can be either inbound or outbound. Inbound congestion is the sum of all congestions on inbound links of some junction. Figure 1 presents the distribution of congestion on network nodes (junctions). Power law nature of this distribution means that vast majority of nodes are not congested but there are a few nodes whose congestion can be arbitrarily large. Based on the Wardrop’s User Equilibrium [20] this also implies a low number of yet significant deviations between the routes chosen by travelers during free-flow and during congestions. In Section 3.3 we use this fact to merge between two routing strategies.
2.3 Flow

The analyzed dataset contains traffic flow through links provided as the number of vehicles per hour. In the next section we will compare the flow through nodes estimated using Betweenness Centrality to the measured flow. We compute the total inbound flow through a node by summing flows on all of its inbound links, where outbound flow is computed symmetrically. Unless a specific junction is a source or a destination of traffic we expect the inbound flow to be equal to the outbound flow. Figure 2 demonstrates the correlation between inbound and outbound flow. We see that vast majority of the nodes are located on the main diagonal, however, there are some deviations, caused by the fact that the data represents average measurements that were carried out along a substantial period of time.

Figure 3 presents the distribution of inbound flow on network nodes. This distribution is exponential, meaning that a vast majority of nodes have little flow through them. However, in contrast to network congestion, there are no “unbounded fluctuations”, i.e. the flow through the most “busy” junctions is not as high as can be expected from the power law distribution of betweenness and congestions (Figures 1 and 4). In fact, congestions significantly limit the flow through the busiest junctions, which subsequently is the reason we do not see the long tail in flow distribution.

3 Betweenness Centrality vs. Traffic Flow

Betweenness centrality is defined as the total fraction of shortest paths between each pair of vertices that pass through a given vertex [4]. Let $G = (V, E)$ be a directed transportation network where $V$ is the set of junctions and $E$ is the set of directed links.
as described in Section 2. Let $\sigma_{s,t}$ be the number of shortest paths between the origin vertex $s \in V$ and the destination vertex $t \in V$ (in some applications the shortest path constraint can be relieved to allow some deviations from the minimal distance between the two vertices). In the rest of this paper we will refer to the shortest or “almost” shortest paths between two vertices as routes. Let $\sigma_{s,t}(v)$ be the number of routes from $s$ to $t$ that pass through the vertex $v$. The Betweenness centrality can hence be expressed by the following equation:

$$BC(v) = \sum_{s,t \in V} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}.$$  \hspace{1cm} (1)

Note that in this definition we include the end vertices ($s$ and $t$) in the computation of Betweenness since we assume that vehicles can be inspected also at their origin and at the point of their destination.

After computing the Betweenness centrality for the given transportation network, we can easily see that the distribution of Betweenness centrality follows a power law (Figure 4). Long tail distributions such as the power law suggest that there is a non negligible probability for existence of vertices whose Betweenness centrality can be arbitrarily high. This is in contrast to the exponential flow distribution depicted in Figure 3. The different nature of these two distributions suggests that BC as defined above will overestimate the actual traffic flow through nodes especially for the most central vertices.

Next we would like to check the correlation between BC and traffic flow. Although the correlation is significant the square error is very low ($R^2 = 0.2021$) as shown in Figure 5 (a). Every point in this Figure represents a vertex with the x-axis corresponding to the measured traffic flow and y-axis corresponding to the computed BC.
We now discuss augmented variants of the Betweenness centrality measure that significantly improve the correlation with the traffic flow.

### 3.1 Origin-Destination based Betweenness Centrality

BC definition according to Equation 1 BC assumes equal weights of routes between every pair of vertices in the network. In other words every vertex acts as an origin and as a destination of traffic. We would like to utilise the measured origin-destination (OD) flow matrix in order to prioritize network regions by their actual use. For this, we shall use the following altered definition for betweenness, as suggested in [16]:

\[
BC(v) = \sum_{s,t \in V} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \cdot OD_{s,t}
\]  

(2)

where \(OD\) is the actual measured origin-destination matrix. This method produces a better correlation \((R^2 = 0.4916)\) between the theoretic (BC) and the measured traffic flow (see Figure 5 (b)).

### 3.2 Shortest Routes based on Time to Travel

In order to further improve our ability to estimate the predicted network flow using the network’s topology, we note that both BC calculation methods (Equations 1 and 2 above) assume that routes are chosen according to shortest path strategy based on hop counting. In this section, we retain the shortest path assumption but use weighted links for calculating the Betweenness score. One option is to use the length of the road

![Flow-in distribution](image-url)

Fig. 3. Exponential distribution of traffic flow through nodes.
segments as their weights for the shortest path calculations (based on the well justified assumption that people prefer short routes over the long ones). However, the road capacity, congestions, and the number of segments also play significant roles when choosing the route to destination. People would prefer highways over sideways when the distance difference is not high.

Shortest path algorithms (such as Dijkstra’s or Bellman-Ford’s) are able to consider only one distance weight on links when computing the shortest path to a destination. We shall therefore assume that the primary heuristic guiding people when they chose a route is the time required to reach their destination. Using this assumption, we recompute the BC on the directed transportation, weighting links by their free-flow travel time.

Let $BC_{ft}(v)$ denote the Betweenness of a node $v$ computed w.r.t. the free-flow travel time. Figure 5 (c) shows significant improvements in the correlation between the measured traffic flow and the theoretical $BC_{ft}$ values computed w.r.t the OD matrix and free-flow travel time link weights ($R^2 = 0.6123$). We can see that there are few nodes whose flow was significantly underestimated by the BC measure. Notice that there are also several nodes whose flow was actually overestimated. This can be explained by the fact that people do not travel strictly via shortest paths, but may have various deviations. In particular the deviations form shortest paths are affected by the day time and the day of week.

3.3 **Peak-Hours Aware Betweenness Centrality**

It is a reasonable assumption that during peak hours travelers will choose to avoid the congested roads and choose their routes based on the congested travel time rather than on the free-flow travel times. Let $BC_{ct}(v)$ denote the Betweenness of a node $v$ com-
Fig. 5. Correlation of flow through nodes and Betweeness Centrality computed w.r.t. the congested time. Computing Betweenness using only the congested travel time weights results in $R^2 = 0.7096$. Although peak hours are relatively small fraction of the day, most vehicles travel at these hours. This is the reason for higher correlation of $BC_{ct}$ with the measured traffic flow.

We shall now combine both the Betweenness centrality computed w.r.t. the free-flow travel time and the congested time by taking a weighted average, namely:

$$BC(v) = \alpha \cdot BC_{ft}(v) + (1 - \alpha) \cdot BC_{ct}(v)$$

where $\alpha$ denotes the relative fraction of vehicles traveling during the free-flow periods. The resulting centrality index can achieve higher correlation with the measured average traffic flow. The maximal correlation of $R^2 = 0.7285$ is obtained for $\alpha = 0.25$ as shown in the Figure 6.

### 3.4 Separating Stubs Nodes from Transit Nodes

Carefully looking at the various nodes we can see that they can be divided into two groups: stub nodes and transit nodes.

A Stub node is a node that is an origin or a destination of the traffic (as seen in the Origin-Destination matrix). These nodes account for approximately 10% of the network’s nodes. All other nodes (namely, nodes that generate insignificant or no outgoing or incoming routes) are called Transit nodes, as they only forward traffic and do not generate or consume it.
Figure 5 (d) presents the correlation that is received when the two groups of nodes are being processed separately. Specifically, the results show a $R^2 = 0.7068$ for the Transit nodes and a $R^2 = 0.7429$ for the Stub nodes.

3.5 Mobility Oriented Betweenness Centrality

As previously mentioned, the transportation network dataset we use contains a “type” attribute for each link, representing the domain-specific “role” of the link in the overall network. For example, links of types 13 and 14 correspond to internal neighborhood roads, whereas links of type 12 correspond to “collectors” — roads that are in charge of aggregating the traffic from neighborhood roads and channeling it to metropolitan roads, and so on. As each type of roads have therefore a different role, we now try to further improve our flow prediction by examining the Betweenness values achieved when calculating it for every group separately.

The results of the correlation that is achieved using this method are presented in Figure 7. We can clearly see that for the more important roads (namely, those with lower type number, representing a more infrastructural role in the transportation network) this technique yields $R^2$ values that are consistently above 0.74, reaching 0.83(!) for road of types 2 and 9 (note that roads of type 90 are fictive roads with infinite capacity that were artificially added in order to connect distinct regions in the network).

It should be noted that each node may have incoming roads of different types. Each plot corresponds to a set of nodes whose max incoming road type is as specified. In addition, the BC calculations were not made for each set of nodes separately — BC was
computed for the complete network, while the correlations were computed separately for each type.

Fig. 7. Correlation of flow through nodes and Betweenness (computed separately for different types of links).

4 Conclusions

In this paper we have discussed the correlation between the Betweenness centrality of a node and its expected traffic flow, in transportation networks. Using a comprehensive dataset that covers the Israeli transportation network we have first performed a simple analysis of the network and its properties, showing that there exists a correlation between the traffic flow of nodes and their Betweenness centrality. We then revised the basic definition of Betweenness centrality, showing that when analyzing the network in a way which takes into account additional known properties of the links (specifically, time to travel through links), a much stronger correlation can be achieved. Taking into account that a large portion of the traffic is being generated during rush hours, and that different roads may have different ‘roles’ in the transportation network, we show that a significantly higher correlation can be achieved when clustering the roads into groups based on their types (a known property of each road), while also giving increased weight to data that is associated with certain hours. Using this method that we call "Mobility..."
Oriented Betweenness Centrality” we demonstrate correlation values of approximately $Z^2 = 0.8$.

This method can now be used in order to generate highly accurate approximations of the traffic flow in the network, based on its topology, the OD matrix, and time to travel without costly simulations. Furthermore, we can also use this method in order to estimate the dynamic changes in traffic flow due to changes in the Betweenness of nodes, caused by events such as car accidents, road detours, etc. This technique can be useful for traffic prediction systems, such as DynaMIT [21].

In addition, based on the correlation between individual flow and Betweenness flow, a similar correlation between Group Betweenness and group flow can be implied. Subsequently, various problems dealing with flow that are relatively hard to solve can now be tackled using their dual Betweenness problems. For example, a knapsack style problem of finding the best group of nodes to put speed cameras at (in order to capture as many speeding drivers as possible) can be translated to a dual problem of finding a group of nodes with the largest group Betweenness. For the latter, however, there exist various efficient approximation heuristics, that can be used in order to derive a solution for the first. Similar approach was taken in [22, 23] for optimizing deployment of traffic inspection systems in communication networks.

References


Abstract. Traffic and mobility analysis are fascinating and fast growing areas of data mining and geographical information systems that impact the lives of billions of people every day. Another well-known scientific field that impacts lives of billions is biological sequence analysis. It has experienced an incredible evolution in the recent past, especially since the Human Genome project. So far, however, both fields never met. This is surprising since both face a similar challenge, namely the identification of relevant patterns in massive sequential information. Indeed, whereas biological sequence analysis has mainly focused on sequences of (few) symbols, traffic and mobility mining often focus on sequences of continuous values. Thus, one may argue that building bridges between them is insurmountable.

In this paper, we show that this is actually not the case. Using well-known discretization techniques such as stay-point detection and map matching, we can turn most — if not all — traffic sequences into a "biological" sequence. Then, we apply the rich toolbox for biological sequence analysis to traffic data. For instance, by just looking at complex traffic data through the biological glasses of sequence logos we get a novel, easy-to-grasp visualization of the data, called "Traffic Logos". Sequence alignment can be used for activity analysis, and profile hidden Markov models are well suited for capturing event persistence during event detection. Actually, our empirical evaluation on three real-world data sets demonstrates that exploiting the link between traffic and DNA can result in state-of-the-art performance.

1 Introduction

Location-based services signify a change in how we as a society use computers to mine data. As Mitchell pointed out [12], we are beginning to analyse 'our reality' — data recording personal activities, conversations, and movements — in space and time in an attempt to improve human health, guide traffic, and advance the scientific understanding of human behaviour in general. In a sense, sensor-equipped computing devices overcome longstanding temporal and spatial boundaries to human perception [15]. Therefore it comes as a no surprise that mining one type of reality data, namely movement and traffic data, is currently receiving a lot of attention. Dozens of mining approaches have been developed for interesting tasks such as 'user activity analysis' — extract the high level routines of traffic users from low-level sensors and GPS devices — and 'traffic
event detection’ — identify unusual bursts of traffic frequencies to detect traffic jams, accidents or gathering at meeting places. However, as we argue in this paper, there is a set of well-known techniques that the community seems to have missed, namely tools for biological sequence analysis.

Similar to traffic data, biology was (and is still) facing the problem of sequence analysis in the wild in order to understand the immense amount of data produced by for instance the Human Genome Project. On this quest, many powerful methods have been developed, often based on principles of probabilistic models. Consider for instance the classical task of aligning to sequences, say HAL and HL. Intuitively, two sequences in an alignment are of same length and similar symbols are matched per position. Getting the sequences to the same length is realized by introducing gaps between consecutive symbols. In our example this could result in H-H,A-gap,L-L.

Indeed, whereas traffic sequences are continuous in time and space, biological sequences are composed of discrete symbols over discrete time. In turn, one may argue that we cannot make use of biological sequences techniques for traffic data. In this paper, we make the somewhat surprising claim that this is not the case. Actually, we demonstrate that standard discretization techniques for traffic data together with biological sequences analysis can result in state-of-the-art performance. Specifically, we

1. establish the first link between traffic mining and biological sequence analysis.
2. Then, using ‘black box’ biological techniques, namely sequence alignments and profile hidden Markov models, we demonstrate that state-of-the-art performance can be achieved for two important traffic analysis tasks: user activity analysis and traffic event detection.
3. Since the choice of similarity score between discretized symbols used by biological sequence analysis algorithms is important and depends on the invariances required by the domain, our main technical contribution here is the design of a data-driven similarity score suitable for traffic data.
4. Finally, again by exploiting the link established, we introduce ‘Traffic Logos’, a novel visualization technique that provides a compact yet descriptive view on the information content of traffic sequences.

Indeed, many more correspondences between biological sequence methods and traffic mining tasks exists, and a biologist would typically use multiple tools and views to solve a task at hand. Examples include conserved regions (consecutive areas of high support), consensus sequences (most probable sequence after alignment) and ‘sequence logo’ i.e entropy at different sequence positions. All of this carries over to the traffic domain.

Before starting off, let us briefly introduce the two traffic mining tasks we focus on. User activity analysis essentially abstracts a person’s movement from raw GPS data to places of interest and then mines relationships among these places [11, 20]. Imagine we want to mine a person’s daily routines, i.e., we ask the following questions: What are interesting and/or frequent stops (stay points) in a person’s travel routines like Banks, Restaurants, Supermarkets, Gyms, Home, Office place etc.? What route does she usually choose for travelling between two stay points? What time does she usually choose to travel from home-to-office and back? What are the shopping routines and weekend routines? And, what time does she usually choose to do sports? Going one step further, one is actually interested in combinations of primitive routines that a user
Biological Sequence Method | Corresponding Tasks in Traffic Mining
---|---
**Pairwise sequence alignment:** Dynamic programming based similarity criteria for variable length sequences | - Similarity criteria between traffic sequences
| - Flexible Pattern matching [17]
| - compression of traffic sequences [5]

**Multiple Sequence alignment (MSA):** Makes all input sequences of the same length by guessing the missing details | - Performing vector like operations
| - Core step for visualization and probabilistic analysis

**Conserved regions:** Consecutive areas of high support in data. | - T-Pattern mining [8]
| - Tagging and compression of sequences [5]
| - Forming hypothesis about functionality

**Consensus sequence/ Profile HMMs:** Representative model of a sequence family | - Probabilistic modelling of relevant traffic sequences [9]

**Sequence Logos:** Information theoretic visualization scheme | - Dense visualization for ‘information content’ in mobility patterns

Table 1. Biological sequence methods and potential applications in traffic

does such as Home- Office- Sports- Home, Home- Office- Shopping- Home, or Home-Bank- Shopping- CityCentre- Home. Such clusters of activities in a user’s movement over the day are clearly interesting in order to identify the user’s most likely activity sequences and profiling of multiple users on the basis of their activities.

The other important mobility mining task we investigate in the present paper is that of *event detection*. Consider e.g. time-series based movement data where a sensor records movements of entities over a short window of time. E.g. an optical sensor placed over a door of an office building reports an ‘estimate of people count’ entering or leaving the building for every 30-minute time intervals. Or, an inductive loop sensor on a highway reports an ‘estimate of vehicles passed’ over the sensor for every 5-minute intervals. This recorded data captures periodical patterns of human activity. e.g. highways are usually busy during morning and early evening time because of traffic ‘towards and from’ work place. Weekdays and weekends can show periodic patterns of their own. Typically, these *periodical activity* patterns are mixed in sensor data with bursts of unusual traffic called *events*; outliers but not noise. Example events include: traffic congestion/jams on a highway, a large meeting in an office or a concert/football game near a highway sensor, etc.. Thus, we have to separate the normal traffic activities from traffic events. Unfortunately, there are no labels which leads us to a problem i.e separation of normal traffic from event. Furthermore, an event is not a single unusually high value, instead it is a chain of sensor reading having its own dynamics. As Ihler et al. [9] nicely described it: "in order to separate normal traffic from events, we need to define a probabilistic model which also incorporate the idea of event persistence i.e the idea that a single, somewhat unusual measurement may not signify anything but several

---

1 http://archive.ics.uci.edu/ml/machine-learning-databases/event-detection/
in a row could indicate the presence of an event”. In this paper, we will employ a hidden Markov model to capture event persistence i.e encoding the dependencies between consecutive readings.

We selected both tasks as they illustrate well the two specific issues we have to deal with when working with traffic data: (a) Traffic data is composed of sequences in time and space of different lengths (with missing data). (b) The amount of raw traffic data is huge. To address both issues, the biological sequence view helps a lot. Actually, we argue that because we abstract the raw data into sufficiently small alphabets using standard discretization techniques, we can instantly solve (a) and in turn (b). Why? Biological sequence analysis will do the rest for us. It was designed to deal with large numbers of variable length sequences. Actually, there is a rich toolbox for all sorts of data analysis tasks including clustering, classification, visualization, and probabilistic modelling of data, among others. However, we have to be a little bit more careful. As already indicated above, the similarity scores used for biological sequences do not carry over to the traffic domain, the invariances in both domains are completely different. Consequently, we have to come up with our own similarity scores. This is not an obvious step in many — if not most — tasks. For this purpose, we come up with a general-purpose, data-driven similarity score.

We proceed as follows. We start off by reviewing biological sequence analysis in terms of traffic symbols. Then, we show how to actually convert traffic data to symbol sequences. Afterwards, we demonstrate that the link established can achieve state-of-the-art performance within our traffic mining tasks. Before concluding, we briefly touch upon related work.

2 Traffic Sequence Analysis

Assume that we have turned a traffic data that is continuous in space and time into a sequence of traffic symbols; we will show how to do this in the next section. We start off by explaining traffic sequence alignment and then continue to profile hidden Markov models, conserved regions and visualization techniques. Before we proceed, let us introduce some notations and definitions required.

An alphabet $\Sigma$ with $|\Sigma| = l$ is a set of symbols in (traffic/biological) sequences. Consider for example a set of stay points from a user’s GPS traces: Home, Work, Shopping, Tennis, Friends, Bank and City-Center i.e $\Sigma = \{H, W, S, T, F, B, C\}$. Let $S$ denote the set of all sequences in a dataset. A single sequence $s \in S$ is a sequence of symbols in $\Sigma$, say $s = HWTHFH$. Say now that we take actually one week of daily raw GPS traces of a user arranged in order from Monday to Saturday: $S = \{HWTHFH, HWSH, HWH, HWSH, HWH, HBSC\}$.

Given two sequences $s_1$ and $s_2$ of length $n$ and $m$ respectively, an alignment $\pi_{s_1, s_2}$ with $|\pi| = p \leq n + m$ defines a correspondence between the elements of $s_1$ and $s_2$ and additional gaps (if required). Gaps are essentially null elements (sometimes we will also denoted them using `-` or $\emptyset$) meaning that we do not match an element of the one sequence with an element of the other sequence. The set of all possible alignments between two sequences $s_1, s_2$ is denoted as $\Pi_{s_1, s_2}$. An alignment problem (sometimes called global alignment) is to find the alignment $\pi^*$ which maximizes a score $\theta$ between
$s_1$ and $s_2$ resulting from the matrix of pairwise similarity scores $\Delta$ between symbols:

$$
\pi_{s_1,s_2}^* = \arg \max_{\pi \in \Pi_{s_1,s_2}} \theta(\pi)
$$

To illustrate, we take two sequence from our running example, namely $s_{mon} = HWTTHFH$ and $s_{thu} = HWSFH$ with following scoring scheme: in case of a match $\pi_{s_i,s_j}(i) = +2$, mismatch $\pi_{s_i,s_j}(i) = -1$, and for gaps $\pi_{s_i,s_j}(i) = -2$. On a more technical level, an alignment $\pi$ is actually a path in a dynamic programming matrix where the score of each individual cell is the maximum of three scores: (i) diagonal above + match/mismatch between corresponding symbols; (ii) horizontal above + gap; or (iii) vertical left + gap.

The well-known Viterbi algorithm finds the path that maximizes the score $\theta(\pi)$ in $O(mn)$ time and outputs the alignment $\pi^*$ along with similarity score. Fig. 1 illustrate applying Viterbi on our running example. It results in the global alignment between $s_1$ and $s_2$:

$$
\pi_{s_1,s_2}^* = HWTTHFH
$$

with $\theta(\pi^*) = (2 + 2 - 2 - 1 + 2 + 2) = 5$.

More important than aligning two sequences is multiple sequence alignment (MSA), i.e., to find the best alignment between multiple sequences under similarity matrix $\Delta$ i.e given $k$ sequences, $s_1, s_2, \ldots, s_k \in S$. This is a fundamental step in many analysis methods for biological sequences as it helps in learning profile HMMs, conserved regions, and visualization, among others. Essentially it builds on top of the pairwise alignment we just described. The exact details, however, are not important for this paper, and instead we present only a result for our running example in Fig. 2. For more details we refer to [6].

Given a multiple sequence alignment, one can compute the so-called ‘consensus sequence’. Fig. 2 shows the consensus sequence HW-S-H of our running example. It describes the most frequent symbols at every position of the alignment (note that a gap is treated as a symbol here). Intuitively, it say the user goes from home to work in the morning. Then, she goes shopping before coming back home again. An alternative view is provided by so-called ‘conserved regions’. Instead of looking at the most likely symbols only, it provides us with the regions (consecutive symbols) with high support in a multiple alignment; the sequence of bars on top of Fig. 2. A conserved region indicates that this region is performing a specific functionality for the set of aligned sequences. In our case, H-W is more frequent than S. In other words, she does not shop every day. However, being at home in the morning and the evening is more likely than going to work due to the weekends where our user is not working. Traffic logos provide a richer and more precise description of traffic sequences than would consensus sequences. They are visualizations which show the variability of symbols at specific positions. Specifically, they consist of stacks of symbols, one stack for each position in the multiple sequence alignment. The overall height of the stack indicates the sequence...
conservation at that position, while the height of symbols within the stack indicates the entropy of each traffic symbols at that position.

However, even logos ignore correlations over time and instead treat the positions independently of each other. To overcome this, one typically uses so-called profile hidden Markov models (HMMs). A profile HMM turns a multiple alignment into a probabilistic model over the edit operations (match, add, delete) for the alignment. In turn, if we want to check how likely a new sequence $s$ matches a set $F$ of sequences, we compute how likely $s$ is according to $F$’s profile HMM. E.g., a traffic sensor reading can be compared to the profile of an accident or a traffic jam to check whether it is an accident or a traffic jam or none of them. Again, we refer to [6] for the details.

3 From Raw Traffic to Symbols

As mentioned earlier, in order to use biological sequence algorithms, one needs to discretize raw traffic into sequences of traffic symbols. Additionally, we need a (symbol) similarity score $\Delta$ in order to align traffic sequences. Stressing the importance of these basic ingredients, we bundle them together under a name ‘translation method’. More formally, let $\mathcal{X}$ be some raw traffic data. We now convert $\mathcal{X}$ into a set $\mathcal{S}$ of traffic sequences using a translation method $M$ i.e $M = (A_M, \Delta_M, \mathcal{F})$ where $A_M = \{a_1, a_2, ..., a_l\}$ is an alphabet (set of symbols the sequences are composed of) and $\Delta_M$ is an $l$-by-$l$ matrix of pair-wise similarities between symbols in $A_M$. Now, a traffic sequence $T_M$ for $M$ is a temporally tagged sequence of symbols chosen from alphabet $A_M$, that is $T_M = \{(a_{t_1}, t_1), (a_{t_2}, t_2), ..., (a_{t_e}, t_e)\}$. Finally, $\mathcal{F}$ denotes a discretization function which maps raw traffic data $\mathcal{X}$ to the set of traffic sequences $\mathcal{S}$ according to $M$, that is $\mathcal{F}(\mathcal{X}) \leftarrow \mathcal{S}_M$. In general, the discretization function to be used in application dependent. Examples include map matching i.e. the process of assigning raw trajectories to street segments, see e.g. [10], region based division of Euclidean space in T-Pattern mining [8], frequency bins from sensor readings and stay point extraction from user trajectories [20], among others.

Let us now touch upon the alphabet and similarity score used in more detail. Every symbol $a \in A$ corresponds to a set of traffic objects. Therefore, it is natural to assume that for any two symbols $a_i, a_j \in A_M, a_i \cap a_j = \emptyset$, that is $a_i$ and $a_j$ correspond to disjoint/non-overlapping sets of traffic objects. Note that the symbols usually represent spatial and unary objects like regions of a city or streets in a street network, however they can also represent non-spatial entities of interest like frequency bins for sensor readings or categories of streets like highway, link road, etc. The condition that symbols...
in $A_M$ represent disjoint sets is sufficient for sequence comparison methods and yet intuitive and powerful enough to capture a very broad range of traffic applications. Indeed, we loose information but gain a more condensed and often more easy-to-grasp view on the data. Specifically, the benefits of using symbols are:

1. Symbols lists are user friendly like street names or regions of a city.
2. Raw traffic data is huge and cumbersome to query and analyse; ’symbol lists’ summarize the data for faster computation.
3. (Disjoint) symbols force the user to define a suitable abstraction level and perform analysis according to her interests; e.g. fig. 4 shows ’Traffic Logos’ for stay point based alphabet after a 99% compression and describes almost all semantic information present in fig. 5 i.e an analysis carried over a 100k GPS readings.

The similarity matrix $\Delta_M$ describes the similarity between symbols in $A_M$. In the context of computational biology, $\Delta_M$ is driven by the following insight: two molecules have higher similarity if they can be converted through chemical reactions readily and vice versa. Therefore, standard matrices have been developed. For traffic applications, the situation is different. There is a multitude of traffic data sets, all with their own characteristics and invariants. Hence, it is unlikely that there is a single good similarity matrix. Instead, it depends upon the application at hand. For example, we have chosen shortest path distances for the model where the input alphabet consists of streets from a street network and the application of interest is ’trajectory clustering’. For cases, where we do not have such domain knowledge available, we now propose a ’data driven’ approach to devise a similarity matrix $\Delta$.

Intuitively, we calculate the average of shortest temporal differences between the symbols in sequences. This is essentially triggered by Buchin et al.’s ’average differences between temporally related points’ [2]. To illustrate, we turn a sequence into a graph in the following way. Each unique symbol in the sequence is a node. Then if two symbols are consecutive in the sequence, there is an edge between the corresponding nodes in the graph. Finally, we weight the edge with the average temporal differences between the two symbols in the sequence. Now, we calculate the shortest path distances between all nodes in the graph. If there are multiple sequences, we simply average all resulting distance matrices. Unfortunately, it may very well happen (in particular for rather small data sets) that there are pairs of symbols which never co-occur in a traffic sequence. In turn, the average temporal difference distance cannot be computed. For example, in the dataset we used for the analysis of user activities, the user never does sports and shopping in a sequence together. In this case, we assign some value larger than the maximum similarity values computed for the ’observed’ symbol pairs. In other words, we just ensure that the two symbols are maximally dissimilar.

We note that now we are in a very similar situation as the well-known IsoMap approach for computing low-dimensional Euclidean embedding [16]. Simply following it, i.e., we embed the weighted graph into Euclidean space $\mathbb{R}^2$ resulting in distances $d_{ij}$ using multi-dimensional scaling [3]. This new distance respects well the intrinsic geometry of the data manifold described by the weighted graph. Finally, we turn the Euclidean distances into similarities by using RBF kernels [10], i.e., $\Delta_{ij} = \exp(-d_{ij})$.

Now, we have everything together to run the traffic sequence analysis techniques introduced in the last section. For instance, we can align a set of sequences. However,
we can do even better. For instance, standard alignment assumes that the time lapsed between two consecutive symbols is constant. This is not true for most traffic data. To accommodate for variable-size steps in time, that means to balance between duration of a time step and the Euclidean distance between the two corresponding symbols, we add a penalty term to the Euclidean distance between them. Specifically, let \( \pi^* \) denote the alignment between two traffic sequences \( s \) and \( s' \) of length \( m \) and \( n \) respectively. Furthermore, let \( d(s_i, s'_j) \) denote the distance after embedding between symbols in \( s \) and \( s' \) at position \( i \) and \( j \) respectively. Now, we define a similarity based on \( d \):

\[
d'(s_i, s'_j) = d(s_i, s'_j) + \lambda \cdot (t_i - t_j)^2
\]

where \( \lambda \in [0, 1] \) denotes the regularizer for variable-size time steps. Its value is application dependent. In case of a gap, we simply fix the gap penalty as a constant i.e \( d'(s_i, -) = d'(-, s'_j) = c \).

Now, we simply use the alignment algorithm to compute \( \pi^* \) and \( \theta(\pi^*) \) (see sec: 2) using the similarity \( \Delta' = \exp(-d') \). Moreover, we can naturally turn the score of the alignment into a similarity score among pairs of whole sequences by normalizing it in the following way.

\[
K_{s,s'} = \frac{\theta(\pi^*_{s,s'})}{\sqrt{\theta(\pi^*_{s,s}), \theta(\pi^*_{s',s'})}}
\]

Now, we finally have everything together to employ the toolbox for biological sequence analysis for traffic mining.

## 4 Traffic Mining using Biology

Our intention here is to investigate the usefulness of biological sequence analysis method for traffic data. Specifically, we investigated the following questions:

- (Q1) Is it possible to solve Traffic problems with the help of out-of-the-box biological sequence analysis methods?
- (Q2) If so, how do they perform compared to state-of-the-art methods?
- (Q3) Can we gain interesting insights into traffic data with the help of biological sequence methods?

To answer Q1-Q3, we choose two traffic mining tasks being investigated by GIS community, namely ‘Traffic event detection’ and ‘Analysis of user activities’.

### 4.1 Analysis of User Activities

We followed two complementary approaches to analysing user routines at different abstraction levels. In the first approach, we extracted daily sequences of the user’s stay points, clustered them using alignments, and analysed the resulting clusters using traffic logos. In the second one, we digged deeper and analysed the user’s routines using ‘map-matched’ trajectories. This helps in grouping functionally relevant trajectories and in turn in identifying specific routes over the street network. Specifically, we used DBscan [7] using the the pair-wise alignment score and then visualized the resulting clusters. Both approaches were applied to the same dataset of 112k recorded position within 363 trajectories.
(Q1, Q3): Stay Points Discretization A stay point is typically defined as a ball of radius \( r \) such that a trajectory stays inside for at least time \( t \). Our goal, however, is to extract more frequent stay points and prune less important ones. For this purpose, first we marked GPS positions from raw trajectories, which stayed within a radius \( r = 100 \) meters for time \( t = 10 \) minutes. Then we clustered these marked points with the help of DBscan [7] to find area which are more dense among these marked positions. In the end, we took the convex hull of each cluster to get the shape of a stay point. To check whether the user is inside a stay point, we took a thresholded distance from the boundary (w.g. 30 meters) to deal with noise in GPS.

Our next step is stay point labelling. To do this, we first looked at the temporal distributions of the stay points in order to label the most important stay points, in our case 'home' and 'work'. Stay points where the user stayed during the daytime (9:00 Am–6:00 Pm) for \( \geq 6 \) hours during weekdays were labelled as work and during night as home. The rest of the stay points are labelled with the help of Google maps (e.g restaurants, post office, bank, shopping markets, Tennis court, etc..). Finally, we pruned stay points that correspond to very short stays with less significance e.g gas stations, etc.

To extract traffic sequences out of the trajectories, we took every daily trajectory and extract the points where user stayed for significantly long time based upon the temporal difference between points. If these points were within a threshold (e.g 30 meters) of a labelled stay point boundary, we added it as a suffix to our traffic sequence and continue till the end of the user’s day. The extracted stay points along with their labellings are shown in Fig. 3; for the sake of keeping privacy, we are omitting the latitudes and longitudes.

After the extraction of stay points, we built the similarity matrix as described above using time regularization. These stay points served as the symbols for activities in our traffic sequences. We calculate distance matrix between daily activity sequences from user with pairwise sequence alignment. Then, the sequences were clustered based using DBscan [7]. The sequences of each clustering were additionally aligned and we produced traffic logos for them shown in Fig. 4(a-d). For the sake of visualization of time information, we labelled the time of stay points in every sequence with M-Morning (before 9:00 Am), D-Day (9:00 am to 6:00 pm) and E-Evening (after 6:00 pm). In the traffic logos we used different colours to indicate these time labels, namely green for morning, yellow for daytime, and red for evening. As one can see, traffic logos show a very dense and illustrative view of clusters for user’s daily routines. Fig. 4(a) describes the largest cluster. It becomes readily clear that this cluster encodes the daily routine of staying at Home in the morning with a higher certainty and then going to office early or staying at home with a very small possibility. In the daytime, the user goes to work with a very higher certainty and comes back around 6pm with a small possibility of staying...
Fig. 4. Traffic logos for stay-point based clustering after approx 99% compression of original data. x-axis denotes sequence positions and y-axis denotes ‘information’ present in each column. Every cluster describes one of the possible routines that user follows in her daily life. The symbols in the figure denote labels of activities based on stay points i.e: H denotes staying at Home; W—Working; A—shopping at ALDI; R—shopping at Real; B—getting cash from Bank; T—playing Tennis and O denotes Other activities for leisure (i.e city center roaming and visiting friends). Colours of symbols denote the time of day i.e green denotes Morning(before 9.am); yellow—daytime(9am-6pm) and blue—evening(after 6pm). The height of a symbol denotes the certainty of an activity in the clusters. For example, staying at work place in the evening is less certain than going Home in the evening (cluster 'a'). Similarly, during weekends, doing other activities in the day after shopping is less certain than coming back home(cluster 'c').

The small ‘O’ at the end of the logo describes a small possibility of going for other leisure activities (city center roaming or visiting friends). Fig. 4(b) describes second cluster with a quite similar daily routine. There is, however, an important difference. The user shops at either of two shopping centres (ALDI, REAL) after work. Fig. 4(c) describes a cluster which is possibly weekend-routine since there is not high W(ork) symbol at all. So on the weekend, the user stays at home in the morning and then gets cash from bank with a small probability. Afterwards the user shops from ALDI-then-Real or only REAL during the day time. Then she comes back home and stays. However, with a small probability, instead of coming back to home after shopping, she chooses to do some Other leisure activity like city center roaming or a visiting a friend.

(d) The last cluster shows a small possibility of Tennis playing after work. This is an affirmative answer to questions (Q1) and (Q3).

(Q1,Q3) Map Matching Discretization In this section, we take our analysis a level deeper and analyse specific map routes that user selects during her travel routines. We do so by choosing street names as symbols in our alphabet and map matching as our discretization function. Map matching is the process which converts raw trajectories to the sequences of street segments travelled. We applied map matching as described in [10]. After map matching, we cluster these map-matched trajectories with the help of biological sequence methods. After clustering on the map matched level, we projected the labelled trajectories back into Euclidean space and visualized them over the street.
network in Fig. 5. We show the different clusters in descending order according of their sizes, namely \{119, 78, 28, 11, 7, 6, 6\}. The clusters were labelled with already extracted stay points, cf. Fig. 3, for user activity. As one can see, the largest cluster is the daily travel from ‘Home to Work’. The runner-up largest cluster is a direct route back home from office. However, as the user may also take alternative routes back home from office, for instance through shopping center ALDI (clusters 3) or REAL (cluster 4), the back home cluster 2 is not as large as the home to work cluster. Cluster 5-8 indicate travelling to Work-to-Tennis, casual shopping from ALDI, weekend shopping results, and city center roaming. So again, we find meaningful clusters. As one can see, pairwise traffic sequence alignment is able to capture very compact clusters by filtering out the noise. We believe that this happens because of high gap costs in the alignment computation. They penalize to group together trajectories with dissimilar sub-parts. Moreover, we contacted the owners of the dataset and they agreed with the possibility of clusters found. This is clearly an affirmative answer to (Q1) and (Q3).

4.2 Q1,Q2—Traffic Event Detection

To further investigate the performance of the biology view on traffic analysis, we considered a classification setting, namely to detect traffic events from sensor data as already described in the introduction. Specifically, we were interested in (a) event persistence and (b) separation of normal traffic from event data. Since events are very rare and usually composed of less than 1% of the dataset, the main step to achieve (b) is subtracting the mean. However, after this normalization choosing a threshold
value for classification will not work well since it does not consider a. Consequently, it would produce a lot of false positives (i.e. noise or unusually high readings due to some temporary phenomena). Fig 6 illustrates this point with the help of sensor readings acquired over a publicly available event based data set. To address a, we propose to learn a profile HMM from the event data. This is a sensible idea for two reasons. (1) Profile HMMs capture event dynamics probabilistically and in turn can be used for soft comparison between event and non-event sensor readings. (2) Most of the events are different in length, e.g. congestion at the end of a concert can last 30-45 minutes. Comparing variable length sequences is one of the strengths of profile HMM. As our experimental results show, they actually perform better (in terms of false positive rates) than state-of-the-art approaches.

We considered two real-world data sets \(^2\) also used by [9]. The first data set is referred to as the building data. It consists of 3 months of count data automatically recorded every 30 minutes at the front door of the Cal-IT2 institute on the UC Irvine campus. The data were generated by optical detectors that measure the presence and direction of objects as they pass through the building’s main doors. The number of ‘counts’ in each direction were stored. The goal here is to predict the presence of a conference on a particular day in the building. The second data set is referred to as the Dodgers traffic data. This loop sensor data was collected for a free-way in Los Angeles. It is close enough to the stadium to see unusual traffic after a Dodgers game, but not so close that the signal for the extra traffic is overly obvious. The observations were taken over 25 weeks with 288 time slices per day in 5-minute counts. Here, the goal is to predict the presence of a baseball game at Dodgers stadium on a particular time. For comparison, we selected two baseline approaches. In baseline 1, we take a small portion of event data as training set, i.e., max(20%, 10 events). Then we mixed it with the same proportions from non-event data. Now, we divided the training data into two groups, i.e., weekend and weekdays. We do so because both of these groups have different trends of traffic. We normalized both groups of training data by subtracting their corresponding means. Finally, with the help of cross validation, we chose a threshold value to classify events that captures all events (irrespective of the number of false positives captured). In baseline 2, we took the state-of-the-art event detection algorithm by Ihler et al. [9], which uses adaptive Poisson processes to identify events. Our profile HMM approach was built upon the output of baseline 1. As there are missing observations, the sequences are of different length. Therefore, we aligned all of the event sequences together and set the length of the profile HMM (a parameter of it) according to the average length of events in the training data. At classification time, we got a complete sequence of sensor readings. To compare it with the much smaller event profile, we used a sliding window equal in length of the event profiles, i.e., profile HMM. Windows with a higher score than a threshold value \(\rho\) (chosen through cross validation) were marked as event traffic. Tab. 2 shows the results. Note that all algorithms capture a high percentage of true positives. However, one can clearly see that just by using out-of-the-box biological techniques, we can get comparable (in one case even significantly lower) false positive rate than the state-of-the-art technique. Having a better recall (low number of false positives) is important in cases when there is a cost attached to a false positive. Consider e.g sending

\(^2\) http://archive.ics.uci.edu/ml/machine-learning-databases/event-detection/
Fig. 6. Sensor readings from Caltech Auditorium Entrance. x-axis denotes time and y-axis denotes sensor readings. ‘Red dots’ denote readings for event days and ‘blue/gray dots’ — non-event days. (a) All readings. (b) Non-event workdays i.e normal movement data with low readings for night/morning times and normal like curve for noon. Unusual spikes are mixture of noise and unreported events. (c) Weekend i.e very low movement in and out. (d) Normalized data after subtracting the mean. Notice that most ‘event days’ have high value however making this a criteria for classification will generate a lot of false positives due to blue dots.

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Training Events</th>
<th>True Positives</th>
<th>False Positives</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>15</td>
<td>76</td>
<td>65</td>
</tr>
<tr>
<td>Poisson Proc.</td>
<td>76</td>
<td>75</td>
<td>23</td>
</tr>
<tr>
<td>HMM profile</td>
<td>15</td>
<td>74</td>
<td>18</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Caltech Auditorium event prediction - Total events= 29</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
</tr>
<tr>
<td>Poisson Proc.</td>
</tr>
<tr>
<td>HMM profile</td>
</tr>
</tbody>
</table>

Table 2. Comparison of event prediction on real world data sets. In both cases, HMM profiles were able to predict almost the same number of events with a better recall (lower number of false positives) and a low training percentage of data. On average, 90 percent of original events are captured by all algorithms. However, out-of-the-box profile HMMs provide a better recall (filtering of false alarms) by capturing event persistence.

a traffic inspector to control a traffic jam when there is no jam. Taking all of our event detection results together, they clearly provide an affirmative answer to (Q1) and (Q2).

5 Related Work

For user activity analysis, recently models have been suggested that abstract a person’s movement from raw GPS data to places of interest and mine relationships among these places [11, 18–20]. Usually the primary goal of activity analysis is modelling of user behaviour to make some recommendations. For example, user’s mobility is modelled to give her advice on path selection during her activities in [11, 18]. Frequent stay points of users are extracted to make travel recommendations in [19] and finding activity sequences in [20] while [14] discusses clustering of raw trajectories and summarizing users routines. Traffic event detection is usually carried out on the basis of traffic streams from sensors [9, 1]. For example [9] discussed an adaptive Poisson process to detect traffic congestion caused by base-ball games near the highway sensor. [13] gives an event detection algorithm for trajectory data. A very nice introduction to biological
sequence methods is [6]. Approaches similar to pairwise sequence alignment for flexible pattern matching and trajectory compression are discussed in [5, 17]. The closest approach to our similarity method is given by [5] which compresses vessel trajectories with alignment kernels. However, they do not discretize the underlying space and hence do not make use of symbolic techniques in biology. More importantly, if two vessels are following the same trajectory but sampled at different time and space, the aligned points show a greater distance than the actual one. [17] used dynamic programming to resolve flexible pattern queries. Examples of discretization of underlying space are: map matching [10], region based division of trajectories for T-pattern mining [8] and stay point extraction for activity analysis [19]. Our data-driven similarity matrix is close in spirit to [2]. However, Buchin et al. do not employ multi-dimensional scaling and hence do not consider the manifold of the data. Weblogos [4] are a variation of sequence logo. A generator is accessible through a website. Profile HMMs and their uses in computational biology are described in e.g. [6].

6 Conclusion and Future Work

In this paper, we have established a first link between ‘biological sequence analysis’ and ‘traffic mining’. Up to now, both fields never met although both face a similar challenge: the identification of relevant patterns in massive sequential information. The main step to establish the link consisted of using discretization methods to map the continuous in time and space traffic data to symbols over time. We exploited the link established in several traffic mining tasks and demonstrated that state-of-the-art performance can be achieved using off-the-shelf biological sequence analysis tools. Specifically, we demonstrated that sequence alignment can be used for activity analysis, and profile hidden Markov models are well suited for capturing event persistence during event detection. In particular, the link allowed us to introduce a novel visualization scheme for traffic data, called Traffic Logos. They provide a condensed, yet illustrative of picture of patterns in traffic sequence data.

There are several attractive avenues for future work. First of all, one should investigate the benefits of out-of-the-box biological sequence techniques for other traffic mining applications. We are currently working on generating more complex profiles and diaries of user’s activities. One should also start comparing such profiles against each other to get user similarity. As the electronic cost and effort for capturing our reality drops, more people are recording what they do, see and hear. To improve the value of such recordings, we should come up with improved content-searching algorithms. In bio-informatics, Basic Local Alignment Search Tool, or BLAST, is an algorithm for comparing biological sequence information and has been proven to be highly efficient and successful. A BLAST search enables a researcher to compare a query sequence with a library or database of sequences, and identify library sequences that resemble the query sequence above a certain threshold. Similarly, one should build up traffic, mobility, and activity databases and employ BLAST-like techniques to gain understanding of human behaviour in the wild.

Personal communication with the authors.
References

Checking out checking in:
observations on Foursquare usage patterns

Martin Chorley, Gualtiero Colombo, Matthew Williams,
Stuart Allen, Roger Whitaker

School of Computer Science & Informatics, Cardiff University, UK, CF24 3AA
{m.j.chorley, g.colombo, m.j.williams, stuart.m.allen,
 r.m.whitaker}@cs.cardiff.ac.uk

Abstract. Location-based services allow access to a rich store of in-
formation on user mobility and behaviour. Such information may reveal
indicators towards user personality which can be used to improve models
of human mobility and create future recommendation systems. Analysis
of Foursquare checkins in venues in two different UK cities over a period
of weeks allows us to draw conclusions about user behaviour, checkin
patterns and similarity of users.

1 Introduction

Mobile location services such as Foursquare and Gowalla allow users to record
spatial and temporal data, keeping a log of the places they have been to and
when. These services use social and game features to encourage users to continue
to participate, producing an ongoing record of spatio-temporal data that can be
mined and analysed to provide a rich picture of user mobility and behaviour.
This picture may reveal indicators to user personality and their disposition to-
wards experiences as suggested by the OCEAN [2] personality model, informa-
tion which could be used to inform the creation and development of mobility
models and recommendation systems.

In these location based mobile services, places are represented by ‘venues’,
which are described as any place a user can visit, such as a shop, a restaurant, a
park, a road and so on. Users ‘check in’ to venues, marking that they have visited
a particular venue at a particular time. Venues can be created by users if they
are not already present in the service, they have location data describing their
position and may also have some user provided data associated with them. This
user created data may at the simplest level be a categorisation of the venue into
one of several pre-defined categories, or it may contain more complex information
such as user tips for the venue, or photographs taken at the venue.

There are two main aims behind the work presented in this paper. It should
be possible to use mobility and social data to identify how well checkins re-
fect predicted user behaviour from social based mobility models. Such mobility
models often include drivers that attempt to force social links and groupings
to influence individual mobility. Analysis of user provided checkin data should
allow comparison to see how mobility in real world is affected by social links and grouping.

Secondly, we aim to use social, temporal and mobility data to inform the creation of future recommendation systems. We envisage a system where users are recommended content and activities based not just on prior behaviour or the patterns of their friends, but on predicted future behaviour. Such a system should be able to recommend activities or content to users without the users necessarily being aware of their desire for such items. In order to create such a system we must first answer some fundamental questions about how personality and behaviour relate to mobility and social ties.

By monitoring a number of venues across two different cities in the UK, we have captured a snapshot of Foursquare checkin information. This checkin information has been analysed to draw conclusions about user behaviour and the patterns that may be found within. In this work we focus primarily on the analysis of Foursquare data relating to categories of venues, and what this can tell us about user behaviour and similarity.

We analyse the distribution of checkins over users and venues to identify any patterns, as well as examining the geographic distribution of checkins over the two cities. We identify different patterns of behaviour in terms of the checkins made in different categories of venue, and the temporal patterns in the same. Finally we use individual category checkins to compute a measure of similarity between users to assess the level of similarity between friends.

2 Related Work

Social aspects have been increasingly used in recent years to develop computational mobility models. For example, [11] links human mobility with social network theory and uses synthetic mobility traces as good approximations of human movement patterns, while [5] stresses the importance of knowing and understanding the social structure beside the use of realistic pairwise statistics produced by analysing contact and inter-contact time distributions. This regularity of movement has been proved to be strictly related to the complex community structure, capturing highly connected circles of friends, families or professional cliques in a social network. In this context, the social and spatial temporal dimensions dynamically evolve and interact thanks to the frequent changes in the activity and communication patterns of individuals [4, 13].

The integration of spatial information with social networks as been largely investigated during the last decade by the creation of prototype social networking systems based on physical co-presence. [8] discusses how location-based social networks services such as Foursquare¹, Gowalla², and Brightkite³, present a number of innovative areas of functionality, either not yet available or fully exploited in other commercial mobile social networking platforms. [10] focuses

¹ https://foursquare.com/
² http://gowalla.com/
³ http://brightkite.com/
on how users connect with friends in relation to how they check-in at different places, finding in general that user activity decays faster than exponentially and so users add friends more quickly than they accumulate check-ins. Similarly, [15] analyses user activities and social graph of a mobile social network using data obtained from Nokia Friend View\(^4\). Results show that user activities are highly correlated and in contrast to from online social networks, where users have a large number of friends but lose weakly-tied subgroups, users of these location-based services tend to have closer, strongly-tied cohesive subgroups. [7] examines the motivations of users of these services, focusing in particular on Foursquare. These include elements of fun, exploration, and coordination with the most popular use being signalling availability to friends, using the history of visited places as a form of presentation of self, and coordinating with the social group.

One of the most distinctive features of these services in comparison with traditional on line social networks is that they allow users to know when their friends are around and provide the ability to meet new people who share their own interests [1]. This could be used for the development of improved recommendation systems in order to suggest places and activities to do at a particular location and time. In fact, current recommender systems are still primarily based on collaborative filtering techniques that measure the similarity between users by considering the rating of common items. Issues such as the social influence on physical proximity in friendship formation can be further introduced to improve these systems beside other factors, such as the similarity of interest and social friendships. [6] proposes three different layers for considering users similarity: a social graph layer (using friendship links from Brightkite), a tags layer based on interests similarity (defined by user profiles), and a spatial layer based on the location based graph of each users updates. [12] uses semantics to propose a clustering method for both the users of location-based services and the corresponding geographic areas. The proposed method is primarily based on the identification of user communities that visit similar categories of places, and on the comparison of urban neighbourhoods within and across cities.

Finally, [9] uses Foursquare as a test bed to conduct a socio-spatial analysis concluding that there exist strong social and geospatial ties among users and their visited locations in the network. The authors propose a recommendation system that, although still based on collaborative filtering with rating, also considers a geospatial variant based on the idea that friends who are close in distance (nearby friends) tend to share more commonly visited locations. The same findings have been used in [3] to propose a mixed-model recommender for leisure activities that uses the well known collaborative filtering model as well as additional ones, such as those based on geographical distance (utility decays with the distance between users) and social learning. An example is the so-called learned preferences model, based on ideas such as that frequent visits to specific categories of places (mexican restaurants in the example) would automatically infer a preference for these locations.

\(^4\) http://betalabs.nokia.com/apps/nokia-friend-view/
3 Foursquare Data

Monitoring of Foursquare checkin data took place in two cities, Cardiff and Cambridge, from Monday 21st March 2011 at 08:49 GMT until Friday 13th May at 11:20 BST. Cardiff is the capital city of Wales with a population of approximately 320,000. It contains campuses for three Universities, with the largest, Cardiff University, having approximately 34,000 students and staff. Cambridge is a relatively small city to the north-east of London, with a population of approximately 130,000. The city contains the University of Cambridge, which has approximately 28,000 students and staff.

An initial search for venues in both Cardiff and Cambridge revealed several thousand registered venues across both cities. A subset of these venues were monitored for checkins, with venues chosen based on geographical location. Each venue was polled for checkins in turn, with the monitoring process cycling through the venue list for each city approximately once an hour, with both cities being monitored in parallel. Although monitoring was carried out continually, there are some gaps in coverage either due to Foursquare API down time or other failures of the monitoring process. Over the monitoring period, checkins were observed in only a proportion of the monitored venues. The numbers of venues found in each city, the number monitored for checkins and the number in which checkins were observed are given in Table 1.

<table>
<thead>
<tr>
<th>All Venues</th>
<th>Venues Monitored</th>
<th>Active Venues</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardiff</td>
<td>4,314</td>
<td>2,219</td>
</tr>
<tr>
<td>Cambridge</td>
<td>3,008</td>
<td>1,884</td>
</tr>
</tbody>
</table>

Venues in the Foursquare database may be categorised under one or more descriptive categories. The types and names of categories themselves are provided by Foursquare, but the categories applied to venues are user suggested and community moderated, so the venue categories become a restricted form of folksonomy. Categories themselves are arranged in a hierarchical tree, with several top-level parent categories being separated into further sub-categories, and so on.

4 Results and Data Analysis

The data collected during the monitoring period showed Cardiff to be considerably more active, both in absolute terms and checkins per venue or per user (see Table 2).
Table 2. Checkin statistics.

<table>
<thead>
<tr>
<th></th>
<th>Active Venues</th>
<th>Checkins</th>
<th>Users</th>
<th>Checkins per Venue</th>
<th>Checkins per User</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardiff</td>
<td>1,234</td>
<td>13,299</td>
<td>1,701</td>
<td>10.78</td>
<td>7.82</td>
</tr>
<tr>
<td>Cambridge</td>
<td>852</td>
<td>6,464</td>
<td>1,196</td>
<td>7.59</td>
<td>5.40</td>
</tr>
</tbody>
</table>

4.1 Geographic Distribution of Checkins

Heatmaps showing the distribution of the checkins observed in the monitoring period over the two cities are given in Figures 1(a) and 1(b). As can be seen, the heavy concentrations of checkins tend to match up to features such as roads and dense areas. Checkins are concentrated within the centre of both cities as may be expected.

Fig. 1. Heatmaps showing the geographic distribution and intensity of checkins in each city.

4.2 User and Venue Checkin Distribution

Figures 2(a) and 2(b) show frequency distributions for the number of user check-ins and the popularity of venues. We can see that most users make very few check-ins. Indeed, we found that the number of users having made only one checkin in the period was 31% in Cardiff and 43% in Cambridge. While the number of very active users is small, these users account for many of the checkins in their cities. For example, in Cardiff the most-active 1% of users are responsible for
15% of the checkins. A similar relationship can be observed with the popularity of venues (Figure 2(b)); that is, each city contains a large number of less popular venues and a small group of very popular venues.

The approximate straight-line shape in Figure 2(a) (shown on a log-log scale) hints at a power-law relationship in the distribution of users check-in activity. The shape is less pronounced in the case of venue popularity (Figure 2(b)), and in both figures there is a great deal of fluctuation in the tail of the distribution.

![Graph showing check-in counts for users and venues](image)

**Fig. 2.** User and venue check-in counts for both Cardiff and Cambridge. Both plots are on a log-log scale.

### 4.3 Checkin Categories

The categories associated with venues make it possible to study the structure of a city and the types of venues in which people tend to make check-ins and record their presence, so possibly revealing something about user behaviour in terms of the types of venue visited. Foursquare’s category hierarchy consists of eight top-level categories, each of which can have a number of sub-categories and sub-sub-categories. These sub-categories are too many and too specific to easily examine, so we have grouped venues by their top-level categories, as shown in Figure 3.

It is clear that the three data points for each category give each city a different fingerprint of activity within the city. It may be possible to classify cities based on these values or the ratios between them, and this data may reveal individual characteristics of the city. For instance, Cambridge has almost as many check-ins in the *College & Universities* category as the *Food* category, while in Cardiff check-ins in the *Food* category outnumber those in the *College & Universities* category by almost 2 to 1. Looking at ratios between the number of venues and the number of check-ins reveals further details, for instance in Cambridge the ratio of venues to check-ins in the *Nightlife Spots* category is approximately 3 : 1, while in Cardiff this ratio is approximately 6 : 1.
Fig. 3. Checkin statistics for venues grouped by top-level category. In particular: the number of active venues in each category, the number of checkins in the venues in each category, and the number of unique visitors to venues in each category.
4.4 Daily Check-in Behaviour

The per-hour distribution of checkins offers insights into how the behaviour of Foursquare users is influenced by the time of day. The plots in Figure 4 show the total number of checkins during each hour of the day, along with the categories of venues that were visited. To limit potential bias due to interruptions in data collection, a 21-day period with the fewest outages was selected. The number of checkins recorded in this period were 5,226 and 2,609 for Cardiff and Cambridge, respectively. Checkins occurring within the same hour of the day were aggregated to produce the figures. We also differentiate between checkins on weekdays (i.e., Monday to Friday) and checkins on weekends (i.e., Saturday and Sunday), as we expect work patterns to have a strong influence on weekday check-in behaviour.

Fig. 4. Number of checkins per hour of the day in Cardiff and Cambridge. The contribution of each venue category to the total checkins for each hour is indicated by colour.
We found that there was a slight increase in the number of check-ins per day on weekends. Cardiff averaged $\mu_1 = 246.2$ check-ins per day on weekdays (with $\sigma_1 = 57.7$) and $\mu_2 = 255.5$ check-ins per day on weekends (with $\sigma_2 = 73.6$). A comparable increase was found in Cambridge, which averaged $\mu_1 = 123.5$ check-ins per day on weekdays (with $\sigma_1 = 29.3$) and $\mu_2 = 126.0$ check-ins per day on weekends (with $\sigma_2 = 51.6$). Within each city we tested whether the difference in the weekday and weekend means were statistically significant by applying the two-sample $T$-test with the null hypothesis $H_0: \mu_1 = \mu_2$. The resulting $p$-values lead us to accept $H_0$ for both cities ($p = 0.79$ for Cardiff and $p = 0.91$ for Cambridge) at the 0.05 significance level. Therefore we conclude that there is no statistically significant change in the number of check-ins on weekdays versus check-ins on weekends.

Both cities also appear to have the same ratio of weekday check-ins to weekend check-ins. The fraction of weekday check-ins is $f_1 = 0.704$ for Cardiff and $f_2 = 0.710$ for Cambridge. We tested the statistical significance of this similarity between Cardiff and Cambridge using the two-proportion $Z$-test with the null hypothesis $H_0: f_1 = f_2$. The test gave a $p$-value of 0.55, indicating that there is no statistically significant difference between the two ratios.

Despite the negligible difference in the number of check-ins per day, the distribution of check-ins on weekdays and weekends are noticeably different. In the case of weekdays, we find that both cities have three bursts of check-ins that occur during the day. In Cardiff there is a morning burst from 07:00 to 10:00, an early-afternoon burst from 12:00 to 14:00, and an early-evening burst from 17:00 to 19:00. Cambridge check-ins follow a similar pattern, but with the morning and early-evening peaks being smaller relative to the early-afternoon peak.

The morning burst on weekdays is likely due to users waking up and checking in to venues as they travel to work. Indeed, in the 07:00 to 10:00 burst the four most-popular venue categories for Cardiff are Home - Work - Others (29%), Food (15%), Shops (15%), and Travel Spots (15%). It is not surprising that the early afternoon (12:00 to 14:00) check-ins are predominantly at Food (24%) and Shops (21%) venues. The percentage of Home - Work - Others check-ins in the early afternoon (11%) is much smaller than in the morning burst (29%), which may indicate that either many individuals do not leave work for lunch or neglect to check-in on their return to work. A notable change in the early evening check-ins is an increase in the percentage of Nightlife Spots check-ins (21%) for 17:00 to 19:00. This category becomes increasingly popular as the evening progresses.

Although we can observe a three-burst pattern for both Cardiff and Cambridge during weekdays, there is no strong similarly between their weekend check-in patterns. The number of check-ins per hour in Cardiff does not drastically change between 11:00 and 21:00. Cambridge has a high rate of checking in between 10:00 and 15:00; however, unlike Cardiff, this rate is not sustained into the evening. These results indicate that weekday user behaviour is predominantly driven by routine, whereas there is scope for more variation and less predictability in weekend patterns.
4.5 User Similarity

The categorisation of venues allows us to look at user checkins and develop a measure of interest for each category over the monitoring period. If we take the number of checkins \( n \) in a category \( m \) over all categories \( M \), we describe a user \( x \)’s interest profile as a vector \( C^x \), where \( C^x_m = n \). Non-zero entries in \( C^x \) for a category \( m \) relate to the users relative interest in that category, while a zero entry describes that the user has no relative interest in that category. This vector can then be normalised such that \( \sum_{m=1}^{M} C^x_m = 1 \).

Using these interest profiles for each user we can then calculate the proportional similarity \( PS(x, y) \) \[14\] between two users \( x \) and \( y \) with interest profiles \( C^x \) and \( C^y \) as:

\[
PS(x, y) = 1 - \frac{1}{2} \sum_{m=1}^{M} |C^x_m - C^y_m|.
\]

It should be noted that this similarity may change over time as and when (or if) users change their preferences or Foursquare checkin behaviour.

In Figures 5(a) and 5(b) we compare the complementary cumulative distribution of proportional similarity between all pairs of users and between friends only, using the category checkin count interest profiles.

From the distributions it is clear that the probability of having a PS of a certain value or higher is much higher between friends than between all users. For instance, in Cambridge the probability of having a Proportional Similarity of more than 0.1 is about 0.5 for friends, but less than 0.2 for all users. Cardiff shows similar results. This suggests that pairs of friends have a much higher average proportional similarity than pairs of random users. The fact that friends have some level of similarity indicates that users share more similarity in checkins with their friends than with all users.

![Graphs](image-url)

**Fig. 5.** Proportional Similarity between all users and friends only using category checkin counts to define interests.
5 Conclusions and Future Work

From analysis of Foursquare checkins in two UK cities we have identified several patterns. In general most users have relatively few checkins over a given time period, while a few users have very many checkins. In Cardiff 31% of users have only 1 checkin, while the top 1% of most active users were responsible for 15% of the checkins. The checkin density in each city is closely related to the centre of the cities.

Each city has different behaviour patterns of users in terms of checkins and number of users in top-level categories, and a physical fingerprint comprised of the number of venues in each category. These patterns may reveal information about the city or the users within the city.

A fluctuation in the number of checkins over a daily period has been observed, with weekdays and weekends having different patterns, based on user behaviour. This information may be used to improve future human mobility models.

Similarity of users can be calculated using the number of checkins in each category, and it has been shown that users are more likely to be similar to their friends than other users.

5.1 Future Work:

We intend to expand analysis to further regions to allow further comparison of different cities. Although visual inspection suggests a potential power law relationship in the distribution between number of checkins for venues and users, we intend to carry out further investigation to validate this.

The weekday patterns identified by the analysis seems to be driven by routine, however our analysis is carried out on aggregate behaviour, individual behaviour of routine and regularity also requires study. The diversity of checkin behaviour across venues and categories will be investigated, and checkin sequences analysed for predictability.

Co-located checkins, where two users register a checkin at the same venue at the same time can be mined from the data to identify the role that social relationships have on mobility, to discover if friends check in together frequently, or to discover at what types of venues co-located checkins occur. Aggregate checkin behaviour over many factors (number of co-located checkins, categories of checkins, diversity of venue choice and so on) may allow classification of users or personalities which can aid the development of future recommendation systems.

6 Acknowledgements

This research has been funded by RECOGNITION grant 257756, an EC - FP7 Future Emerging Technologies project concerning Self-Awareness in Autonomic Systems.
References

Clustering Multiple and Flexible Time Intervals in Sequential Patterns Towards Predictive Modeling of Human Movement Behavior

Roberto Legaspi, Danaipat Sodkomkham, Kazuya Maruo, Kenichi Fukui, Koichi Moriyama, Satoshi Kurihara, and Masayuki Numao

The Institute of Scientific and Industrial Research, Osaka University
8-1 Mihogaoka, Ibaraki, Osaka 567-0047, Japan
{roberto,danaipat,fukui,koichi,kurihara,numao}@ai.sanken.osaka-u.ac.jp
http://www.ai.sanken.osaka-u.ac.jp

Abstract. Human behavior modeling is a pervasive issue in the design of intelligent environments or smart spaces in order to anticipate user needs and provide timely system responses. Untimely responses can be detrimental to an intelligent system’s viability soliciting unfavorable reception from the user. This paper presents our initial effort in utilizing multiple time interval sequential patterns with clusters of flexible time intervals between items in the sequence to model a person’s walking paths and their durations. We experimented with our algorithm using human mobility data gathered by the network of infrared sensors installed in our laboratory. We aim to use our algorithm to support what could plausibly be various predictive modeling tasks when we begin to infer correlations of movement behavior and diverse self-related and environment features.

1 Introduction and Motivation

Knowledge of human behavioral actions is highly viable when designing intelligent environments or smart spaces that anticipate user needs and provide timely responses (e.g., [16, 19, 5, 4]). This is because human behavior typically includes contextual interactions with the environment [17]. The occupant of the environment moves and acts in accordance to a diversity of information, such as his/her cognitive and affective states, attitude, and personality, the location and ambient entities information (e.g., outside and within room conditions, background activities, and ambient state changes) and time or duration contexts. We can consider, for example, that same paths in the environment can be made at different durations by the same person under different conditions such as differences in mood (one walks slower when tired than when anxious), intention (someone who is in a hurry to print obviously walks faster than usual), and gender and/or age (old female people would naturally walk slower than younger males). If the intelligent environment can infer an individual’s behavioral patterns, it can then construct models of effective and timely interactive support provisions. For example, it may issue a notification that the coffee is ready or printing is finished.
so that no time is wasted in going and finding out that coffee or the print-out is not yet ready. This last part is important since untimely responses can prove not beneficial and solicit unfavorable impression and reception of system support.

Behaviors can be viewed within two temporal levels, namely, microscopic (i.e., with shorter time frames; e.g., spike in a brainwave signal, blink of an eye, rapid hand gesture, yawn, facial muscle movement, and affective state changes) or macroscopic (e.g., movements and activities of an individual over a longer period of time) level, but with the problems across the spectrum of these two ends potentially so complex that these are approached with a plethora of recognition techniques [20]. In the microscopic level, we examined in the past how human brainwave productions and affective states behave when exposed to the original compositions of an intelligent music composing system [22], the behaviors of different physiological signals (blood volume pulse, respiratory movement, skin responses, and facial muscle movement) that accompany changes in affective states [12], and affective facial and vocal expression changes [5]. In the macroscopic level, we investigated student learning behaviors towards an intelligent tutoring system based on cognitive development, proficiency level and gender [15], as well as student transitions between learning and non-learning activities as they learn on-line [10]. We have used various machine learning techniques for building different predictive models of human behavior when interacting with an intelligent artifact.

However, in light of our research on an ambient intelligent space that aims to provide empathic support to its occupant [13, 14, 11, 5], we have yet to investigate human walking features (e.g., stride length, cadence, walking path and walking speed) in light of behavior modeling [13]. For instance, there is evidence to suggest that even walking speed is indicative of human traits. Tolea et al. [23] has shown that, generally, women walked slower than men at every age with more marked differences after age 55. In the same study, Tolea et al. [23] also found that extraversion and education were correlated with walking speed. Furthermore, mobility may represent measures of health status [23]. An interesting find is that walking speed can be a useful indicator of how long older adults will live [24]. Our initial attempt to achieve our objective involves inferring the behavior patterns of individuals in terms of their walking paths and walking durations. Our goal is to find walking paths, i.e., paths that an individual would likely walk under given conditions, that are characterized by their walk time intervals. Fig. 1 may help illustrate the significance of this objective on how it can later be put into use. Consider the instance wherein an infrared sensor (IR; indicated by grey-filled circle) network is installed in the space and each sensor node sends information as to which point in the space is active (i.e., a person has been detected by one of the IR sensors). If the intelligent space can infer the usual paths the person takes in the space (e.g., when going to the coffee maker, printer, TV, co-worker’s cubicle, etc.) through the help of the IR sensors and can compute the average times by which the person reaches each point since it kept track of the the times when each sensor was triggered, then the intelligent space can consult later on its predictive models in order to send information about the
states of the amenities in the space. For example, if it will take on the average three minutes for the person to get to the coffee maker, then the intelligent space can notify the person at the time he should be getting his coffee. Or perhaps if the system has knowledge of the person’s usual paths and their durations, then it can anticipate automatically turning on the lights even before the person gets to his destination.

To the above end, we need to have an algorithm that can datamine a person’s usual walking paths and their durations. This paper presents our initial effort in utilizing sequential patterns with clusters of flexible time intervals between items in the sequence, where items indicate various points in the intelligent environment. Many studies have focused on mining sequential patterns, i.e., discovering the set of subsequences, wherein a sequence consists of ordered items, that occur frequently in a sequence database [1, 21, 25, 18]. Since its conception [1], it has drawn attention from various people both in the academe and industry, both researcher and practitioner, that has led to its various extensions and modifications in different directions (refer to [3, 7, 8] for noteworthy surveys). However, data mining experts soon realized that even though sequential patterns could provide information with respect to which items are frequently occurring and in what order, they could not provide the time span between items that could further support decision-making. This was addressed by generalizing the problem into mining time-interval sequential patterns [2], which reveals not only the order of items but also the time intervals between successive items. Time intervals have been dealt in both single- and multi-time interval sequential patterns, but with only the latter taking into account all possible time intervals between all pairs of items [8]. Furthermore, time interval data have been handled as regular or irregular [6]. In a regular time series, data are collected periodically at defined time points, while in irregular time series, data arrive at indeterministic time points.
Lastly, time intervals have been commonly dealt with as having definite ranges (e.g., [3,7], 1-8 weeks, etc.) and can only be either short or long consequently leading to strict or rigid boundary problems (i.e., a time interval that is near the boundary of two adjacent ranges, is either neglected or overemphasized) [8]. What we are proposing, however, is an algorithm that allows multi-time interval sequential patterns with irregular and more flexible time interval data by employing a clustering technique integrated in an Apriori-based algorithm. The use of clustering technique allows non-integral time values to be categorized, neither ignoring nor overemphasizing, effectively and efficiently. Lastly, unlike in [8] where multiple time intervals between items are mainly enumerated, clustering can lead to categorizations, hence characterizations, of time interval data.

2 Concept Representations

Fig. 2 illustrates the basic concepts used in our algorithm. We represent a timed sequence $S$ as an ordered set of discrete items with associated time stamps: $S = \langle (a_1,t_1), (a_2,t_2), (a_3,t_3), \ldots, (a_N,t_N) \rangle$, where $a_i$ refers to the ID of an item (e.g., sensor signal, event, action, and activity, among others) and $t_i$ indicates the time at which $a_i$ occurred, for $1 \leq i \leq N$ where $N$ is the sequence length. The time interval between two items $a_i$ and $a_j$ can then be computed as $|t_i - t_j|$ where $t_i$ and $t_j$ indicate the moments in time at which $a_i$ and $a_j$, respectively, occurred. Furthermore, since our proposed algorithm categorizes the various time intervals into clusters, for every possible ordering of two items in a sequence, our algorithm creates $n_{cl}$ clusters. A time interval $|t_i - t_j|$ is categorized to one of the time interval clusters for $a_i$ and $a_j$, say, $CTI_{a_i,a_j,m}$, where $1 \leq m \leq n_{cl}$.

![Fig. 2. A sequence consists of an ordered set of items with associated time stamps. Time interval clusters are created for every possible ordering of two items. A reference time interval window helps determine the relationship between items in the sequence.](image-url)
Lastly, we use a maximum reference time interval window to determine the relationship between items in the sequence. For any two items that have occurred and their times of occurrence are within this reference time window, these two items are said to be related to each other. This condition is introduced to handle noise, as well as the non-contiguous items, in the sequence. For example, if $RTI_{Max}$ is the maximum reference time interval window value and there exists $< (a_i, t_i), (a_j, t_j) >$ where $|t_i - t_j| \leq RTI_{Max}$, then $a_i$ and $a_j$ are said to be related.

### 3 Our Proposed Algorithm

We consider here that all throughout in time there can be an item newly appended into the sequence database. Our algorithm consists of three steps as follows. In the first step, candidate sequences of length 2 (i.e., there are two items in the sequence) are generated with the newly appended item data as basis. Next, the time interval data of the candidate sequences are categorized to existing clusters. The cluster information is subsequently used to build sequences of length $k$. In the last step, sequences of length $k+1$ are generated recursively from sequences of length $k$, which would eventually form a graph of sequential patterns with clustered time intervals between items depicting a chronicle of events. The following subsections elucidate our algorithm in detail.

#### 3.1 Candidate Sequences

![Diagram](image)

Fig. 3. The maximum time reference window is used to determine which items in the sequence will be put together with the newly appended item $a_p$ to form the candidate sequences of length 2.

Every time a new item is appended, that item and the preceding items that occurred within the reference time window of value $RTI_{Max}$ are selected as candidate sequences, each of length of 2. If $a_p$ is the newly appended item in the timed sequence:

$$S=<(a_1, t_1), (a_2, t_2), (a_3, t_3), \ldots, (a_N, t_N), (a_p, t_p)>,$$

then a set of candidate sequences will be created with the newly appended item as base:
CS = \{(a_p, t_p), (a_p, t_p), (a_{p-1}, t_{p-1}), (a_{p-1}, t_{p-1}), (a_{p-2}, t_{p-2}), (a_{p-2}, t_{p-2}), ... (a_{p-s}, t_{p-s}), (a_{p-s}, t_{p-s})\}
where |t_{p-s} - t_p| \leq RTI_{Max}. Each element in CS consists only of two items, namely, the newly appended item and a previously appended item that occurred within the reference time window. Fig. 3 shows how the RTI_{Max} window works in this given situation. Other items that occurred outside this window will not be considered. The dotted lines drawn between a_p and the previous items within the window indicate the selected items that are included in CS.

3.2 Time Interval Clusters

Each time a new item is appended, the time interval associated to a sequence element \(<(a_i, t_i), (a_p, t_p)>\) in CS, which is |t_i - t_p| will be used to create clusters for that sequence element. Clustering is achieved here by categorizing the time interval |t_i - t_p| to one of the n_{cl} clusters for \(<(a_i, t_i), (a_p, t_p)>\) using either of two methods, namely:
  
  − Categorize the new time interval |t_i - t_p| into an existing cluster C\(\text{TI}_{a_i, a_p}\)m (in Fig. 4-left), or
  
  − Merge existing clusters C\(\text{TI}_{a_i, a_p}\)m and C\(\text{TI}_{a_i, a_p}\)r and create a new cluster C\(\text{TI}_{a_i, a_p}\)v that will contain |t_i - t_p| as one of its instances (in Fig. 4-right)

![Fig. 4. The time interval between items of a candidate sequence is clustered to either one of the existing clusters or to a newly formed cluster.](image)

A mean-square error (MSE) is calculated for each method and the one that has the least error is performed. MSE is calculated using Eq. 1 and Eq. 2 for the first and second methods, respectively, where u_{cl} denotes the average of all time interval instances in C\(\text{TI}_{a_i, a_p}\)cl and n_{cl} denotes the number of instances in that
cluster. In both methods, the average time interval and number of elements are re-calculated each time a new time interval instance is categorized to a cluster.

\[
MSE_1 = \frac{n_{cl} \cdot [(t_i - t_p) - \mu_{cl}]^2}{n_{cl} + 1}. \tag{1}
\]

\[
MSE_2 = \frac{n_{cl_1} \cdot n_{cl_2} \cdot (\mu_{cl_1} - \mu_{cl_2})^2}{n_{cl_1} + n_{cl_2}}. \tag{2}
\]

3.3 Generating the \((k+1)\)-length Sequences

In this last step, the algorithm iteratively builds \((k+1)\)-length sequences from \(k\)-length sequences. The algorithm starts off with creating sequences of length \(k=2\) using the elements in the set of candidate sequences \(CS\) (refer to subsection 3.1). The change here, however, is that instead of looking at the time interval, say \([t_u - t_v]\), between two items, say \(<(a_u, t_u), (a_v, t_v)>\), the algorithm utilizes the information in cluster \(CT I_{a_u,a_v,m}\) to which \([t_u - t_v]\) belongs and creates the sequence \(<a_u,CT I_{a_u,a_v,m},a_v>\). The algorithm performs this using all the elements in \(CS\). Subsequently, if there are two sequences of length \(k=2\), say \(<a_u,CT I_{a_u,a_v,m},a_v>\) and \(<a_v,CT I_{a_v,a_u,r},a_w>\), then a new sequence of length \(k=3\) can be created, i.e., \(<a_u,CT I_{a_u,a_v,m},a_v,CT I_{a_v,a_u,r},a_w>\). Consequently, therefore, given the following two sequences of length \(k\): \(<a_1,CT I_{a_1,a_2},a_2,CT I_{a_2,a_3},a_3,\ldots,a_{k-1},CT I_{a_{k-1},a_k},a_k>\) and \(<a_2,CT I_{a_2,a_3},a_3,\ldots,a_{k-1},CT I_{a_{k-1},a_k},a_k,CT I_{a_k,a_{k+1}},a_{k+1}>\) can be combined to form a new sequence of length \(k+1\):

\(<a_1,CT I_{a_1,a_2},a_2,CT I_{a_2,a_3},a_3,\ldots,CT I_{a_{k-1},a_k},a_k,CT I_{a_k,a_{k+1}},a_{k+1}>>\)

The above scenario shows that two \(k\)-length sequences can be joined to create a \((k+1)\)-length sequence if all the items and clusters between the tail of the first sequence and the head of the second are the same.

For each iteration, the algorithm will create a set of \((k+1)\)-length sequences, \(CS^{k+1} = (S_1^{k+1}, S_2^{k+1}, \ldots, S_p^{k+1})\), from the current set of \(k\)-length sequences, \(CS^k = (S_1^k, S_2^k, \ldots, S_p^k)\). The iteration will stop and not proceed with creating \(CS^{k+1}\) if the number of sequences in \(CS^{k+1}\) will be inefficiently small or the sequences will be insignificantly different to \(CS^k\). To anticipate these two events, we used the support and confidence measures [1], respectively. We adapted these metrics since we are not dealing with transactions for which these are conventionally used. Eq. 3 and Eq. 4 show how these two metrics are calculated. The function \(\text{count}(CS^{\text{length}})\) returns the number of sequences in the set that are \text{length} items long. In Eq. 3, \(\text{count}(CS^2)\) refers to the total number of unique pairs currently in the database. If the support or confidence value becomes less than or equal to the corresponding allowable minimum or threshold, then the iteration stops.

\[
support = \frac{\text{count}(CS^{k+1})}{\text{count}(CS^2)} \tag{3}\]

\[
\text{confidence} = \frac{\text{count}(CS^{k+1})}{\text{count}(CS^k)} \tag{4}\]

46
3.4 Experimenting With Actual Data

Fig. 5. This depicts our experimental site that contains the installed IR sensor network. The various sensor nodes are positioned at desks and along pathways. The actual topology of the network is indicated by the dotted lines.

To experiment with our algorithm, we used human mobility data gathered by the network of IR sensors installed in our laboratory. Fig. 5 shows how the sensors are set-up: the circles with ID labels indicate the various places at which the IR sensors are positioned. The rectangular labels indicate the arrangement of working spaces in this room. The sensors are placed not just on desks but on pathways as well. Furthermore, the dotted lines in Fig. 5 indicate all links in the sensor network, hence, its actual connection topology. However, we assume in our methodology that the network topology is unknown. This is practical since at times sensor nodes can be randomly distributed into the experimental site or the environment, especially when using wireless sensor networks (e.g., in using wireless sensor networks to detect a wildfire, sensor nodes can be randomly spread into the woods). Although we are currently using wired IR sensors, in actual, our algorithm should work with wireless sensors and unknown topologies.

Each time a person walks pass a sensor, the node is activated and the time of activation is logged. When someone walks from one point to another point in the room, data are logged as sequences of sensor IDs and timestamps within the path. We collected this kind of mobility data from July, 2010 to February, 2011. Due to technical problems, however, we could only consider data from the right side of the experiment site (highlighted in grey-filled nodes in Fig. 5).

Fig. 6 depicts clustered time interval sequential patterns that were inferred by our algorithm for two people. The data used to extract these was an hour
long reading of the IR sensor network that consisted of 17,655 instances. Since currently we have no provisions for tracking multiple users whose pathways may be overlapping, we manually filtered the data we used here to distinguish as much as possible the pathways belonging to different occupants. The algorithm had a clustering time of 10 minutes and took less than four seconds to generate the sequence patterns.

The difference in the results shown in Fig. 6 can be attributed to a stricter confidence threshold value (i.e., 0.9; Fig. 6-right). First, as a result, the algorithm was able to find sequences that do not have cycles of degree zero (as with sensors 7 and 2; Fig. 6-left), which can be interpreted as noises. Second, the algorithm found a movement from point 7 to point 31 (Fig. 6-right) which is a shorter route than moving from point 7 to 2 and then back to 31 (Fig. 6-left). This may indicate the pertinent individual realizing later on a shorter path. Lastly, the algorithm was able to find through this stricter threshold more time duration variations as these individuals walked within the space. The values in between nodes indicate the average duration values per cluster.

3.5 Future Work: Applications of the Algorithm

The work we have reported here is part of a bigger effort to construct a space for daily living and working conditions that responds emphatically to the occupants of the space [13, 14, 11, 5]. An empathic response generally involves the
perception of occupant thoughts, affective states, intentions and/or situations, and a response due to this perception that is suitable and beneficial to the occupant. We are employing a wide range of bio and ambient sensors that interact to capture diverse human (e.g., affect, intention, etc.) and ambient (e.g., activity, situation, etc.) characteristics (e.g., in [12, 5, 9]).

What we have shown in the preceding subsection is that the same person may have habitual patterns in moving inside a space and with various walking durations that can be grouped. However, what is significantly missing here is the discussion on the reasons for which these patterns and different walking speeds emerge. As we have pointed out in the first section of this paper, the reasons may include conditions like affective states (e.g., walking slowly when bored and fast when frustrated), intention (e.g., to make a cup of coffee), and habit (e.g., the first thing the person does upon arriving in the space is to make coffee), among others. How we intend to apply this algorithm in this domain are as follows:

- **Find affective state correlates of walk durations.** Most affect recognition research investigate the facial and/or vocal expressions of affect but restrict the human subject from moving freely in a daily-living environment. We build our case on the point that it is viable to investigate the possibly existing correlations between walk durations and affect. To this end, in order to obtain the necessary affect labels, we have been using (e.g., in [9]), the Emotiv EPOC Neuroheadset\(^1\), a commercially available electroencephalogram-based device that automatically measures levels of frustration, engagement and boredom. It collects brainwave signals from 14 sensors touching the scalp and provides affect intensities manifested through brain signals in real time. This can be used even as the person moves within the space (in Fig. 7).

- **Find habitual behavior correlations.** Dominant, i.e., frequently performed, sequences may indicate patterns of habitual behaviors in the space wherein a node can be interpreted as a point where an activity takes place or object is manipulated, habitually. Evidence suggests that low-level IR sensors can be used to extract patterns of human behavior in daily living [13].

\(^1\) [http://www.emotiv.com](http://www.emotiv.com)
– Infer models of timely system responses. If various time intervals in between items can be categorized and predicted, then the system can construct models for various temporal parameters to make efficient responses.

4 Conclusion

We presented in this paper the idea of mining sequential patterns with multiple and flexible time interval data between items in the sequence. Data clustering was employed in this study in order to handle the possibly wide range of time interval data. We intend to further validate our algorithm versus existing techniques. Secondly, we viewed the application of our algorithm in terms of how it can support predictive modeling of human movement behavior, specifically, patterns of walking paths and their durations segmented into multiple time intervals. As a long term plan, we intend to find correlations between movement behavior and affect, intention and habit.

Acknowledgement. This work is supported in part by the Management Expenses Grants for National Universities Corporations through the Ministry of Education, Culture, Sports, Science and Technology of Japan.

References


24. Tse, I.: How fast you walk may predict how long you will live (2011)

From mobility data to social attitudes: a complex network approach

Sabrina Gaito¹, Gian Paolo Rossi², and Matteo Zignani¹

¹ Dipartimento di Scienze dell’Informazione, Università degli Studi di Milano
² Dipartimento di Informatica e Comunicazione, Università degli Studi di Milano

Abstract. This paper focuses on the concepts of community and locations of interest to model social attitudes of people in mobility. We firstly address the problem of representing these two concepts in a single framework built on the basic information about mobility provided by GPS coordinates. Secondly we present the concept of geo-community, that combines both location and community, and the algorithm to extract it from GPS data. Finally we show how to model this concept by means of a bipartite complex network and how to use it to infer social implications.

Keywords: community, location, bipartite graph, graph projection

1 Introduction

In the last decade a great amount of research has been addressed to improve understanding of people mobility and sociality. Very recently it has become evident that these aspects of our daily life are strictly connected and that it is crucial to find a model able to describe both and the way they affect each other. The very large GPS datasets, that are being collected, are a very good starting point to face with the problem. This paper focuses on how to derive useful information from this kind of data. In particular, we consider two relevant concepts: location and community. In the literature, the location is adopted to summarise spatial patterns, while the notion of community is generally functional to describe sociality. A location is loosely meant as a point of human interest, while a community is commonly conceived as a group of people sharing similar interests, needs and behavior.

Unfortunately, so far neither the concept of location nor the concept of community has been univocally defined. Therefore, their specific features and, more importantly the relationship between the two concepts, are still not fully understood and exploited either. It is commonly accepted that people move during the day among locations and communities. Many mobility models are based on the assumption that humans move from one location to another. Nevertheless, locations are just points in a simulation area. Their characteristics, like size, distribution, number, population density, are still unknown in details. Similarly, the concept of community has received different definitions in the literature depending on the particular context; the sociological approach, for example defines
it as people sharing some interests, while the location-aware definition assumes that a community is a group of people meeting at a given location.

In this unclear picture we advocate to approach the problem from a primitive point of view, namely by analyzing real datasets of human mobility traces that provide geographical information and, in particular, GPS data. In this paper we focus on the public available NCSU dataset, whose characteristics are deeply described in [19].

The paper provides the following relevant contributions. We firstly derive a deep understanding of the term “location” and at the same time of the notion of community strictly related to it. Then we merge these two concepts into what we call geo-community. Finally, we propose a complex network-based methodology to extract geo-communities from GPS data. The approach starts on how we discriminate GPS points visited while moving - i.e. during the person’s movement phase - from the points where the person has spent a relevant amount of time. In order to identify the real locations of a larger group of persons, what we call geo-community, we propose a clustering method to aggregate individual geo-locations. Let’s take a train station, for example. Many people go and spend at least a few minutes there, so making it an individual geo-location for each of them. But to extract the location of the train station we must properly overlap all of them in a suitable way. The idea of geo-community we propose is strictly connected to these locations. People mobility is motivated by the need to go somewhere, more than by some social need. Social relationships are created by patronizing the same place: people who hang out at the same location have the potential to establish a social relationship or not, depending on whether or not they meet. But for the aims and purposes of deploying network connectivity and services, they belong to the same geo-community. So, a geo-community is both a point of attraction for people - this includes train stations, workplaces, pubs, etc. - and the set of people who go there, maybe at different times. The formal definition of geo-community and the methodology to extract them from GPS traces are described in Section 2. In [6], by statistically analyzing the available GPS traces, we show how the probability distributions of the main quantities involved in human movements and aggregations can be obtained. Our main results concerned pause-times in geo-community, intra-distances, inter-distances and next geo-community choice. The latter point concerns how distance influences the choice of the next geo-community in the person’s movement.

In Section 3 we present the method to create the complex network that models the structure (individuals, geo-communities). This approach is a powerful tool to derive people mobility, social behaviors and the relevant interactions. It is an undirected bipartite graph whose projections directly provide information about movement patterns, such as group changes of communities or flows between locations, and social interactions, such as people potential links or multiple memberships. This kind of representation enables us to make explicit the social relationships which develop when people spend time in the same location.
2 Extracting geo-communities from GPS data

In this Section we formally define the main concept at the base of our modeling approach, geo-community, and we propose a methodology to extract it from real GPS traces. To this purpose we have to distinguish between individual and aggregated traces. With the term \textit{individual trace} we mean a GPS-point sequence associated to a single node that covers a single day; on the other hand, with \textit{aggregated trace} we mean the whole set of individual traces.

Rhee et al. [19] observed that GPS points do not spread uniformly, but tend to gather in a few limited areas. This characteristic agrees with the intuition that, during the day, people tend to gather in places that attract them. These places can be offices, restaurants, bars and coffee shops, and in general any location they spend time in for a while.

According to this view, we can separate human mobility into two phases; a \textit{static phase} where a person spends some time in a location, and a \textit{movement phase} where he moves towards a place of interest; we quantify and derive such an interest by considering the amount of time spent in it. Since people manifest their interest by staying in a particular location, we focus on the static phase.

2.1 From data to geo-locations

In order to extract the static phase from the whole individual trace, we apply a simple heuristic: if two points \( p_i \) and \( p_{i+1} \) do not satisfy
\[
\|p_i - p_{i+1}\| \leq \Delta
\]
then we delete \( p_{i+1} \) from the original trace, since it belongs to the movement phase. Analyzing walking mobility data, we set the threshold to the very low value of \( \Delta = 27 \text{m} \), according to the fact that we observe that human walking speed is about \( 4-5 \text{ km/h} \) (1.1–1.4 m/s) and, in the traces, time data granularity is 30s. In support of previous remarks we can observe, in Fig.1, how, at speed \( \approx 1.3 \text{m/s} \) (4.6km/h), there is a step in the speed CDF, suggesting a change in the movement behaviour. For each dataset, the parameter \( \Delta = 27 \text{m} \) corresponds to a value that is greater or equal to the 0.9-quantile of the distribution of speeds less than 1.34m/s. It seems a reasonable value as generally, in a location, people do not reach the maximum speed. In this way, we capture points where a person is still located or is moving very slowly inside a small area.

The sequence of points, obtained by applying this heuristic, forms a trajectory \( S = ((p_1,t_1), \ldots, (p_n,t_n)) \) where \( t_i \) is a timestamp and \( p_i \in \mathbb{R}^2 \).

All the points of a trajectory belong to the static phase and are the starting points to extract the clusters of points representing possible regions of interest. Formally, we capture these regions by introducing the concept of \textit{stay-location} \( SL \).

\textbf{Definition 1.} Let \( S \) be a trajectory and \( L \) a partition of \( \{p_1, \ldots, p_n\} \) such that for each \( SL \in L \), for each \( p_u, p_v \in SL \) there exists a sequence \( (p_u = p_w, \ldots, p_{w+j} = p_v) \) of points in \( SL \), such that \( \|p_{w+k} - p_{w+k+1}\| \leq \sigma, k = 0, \ldots, j - 1 \) for a fixed \( \sigma \). A stay-location is an element of \( SL \).
Informally, a stay-location is an area where a person stops, independently of the time he stops there. In daily movements, there are many stay-locations where an individual stays for a short amount of time. These stay-locations are transit-locations and represent small pauses in the movement towards the real destination that we call geo-location.

**Definition 2.** Let $\text{Traj} = (p_1, \ldots, p_n)$ be a trajectory and $SL \in L$ a stay-location. $SL$ is a geo-location if in $\text{Traj}$ there exists a subsequence $(p_i, \ldots, p_{i+k})$ such that $p_{i+j} \in SL$ for $j = 0, \ldots, k$ and $k \geq \phi$. $SL$ is a transit-location if such a subsequence does not exist.

In the following analysis of the dataset we set the threshold $\phi = 10$, which corresponds to 5 min. We must underline that we do not consider the sum of the pause times in a stay-location; rather, we consider the single values. For example, if a person spends three minutes in three different occasions in a stay-location, then we consider it a transit-location.

Let us consider individual traces in order to extract geo-locations and analyze their properties. To find geo-locations we apply a density-based clustering algorithm DBSCAN [4]. We choose DBSCAN because it does not need in input the number of clusters to be found and allows to detect arbitrary shaped clusters. As DBSCAN parameters we use $\delta = 10$ mt. and $\epsilon = 2$ neighbors ($\delta$ represents the maximum distance such that two points are considered neighbors, while $\epsilon$ is the minimum number of neighbors that a node must have to be considered in a cluster). This way we have adopted a stricter definition of cluster than in [19], where they set $\delta = 100$ mt. as a limit for the connection between nodes. We use a fine grain because we can better analyze movements through small locations such as restaurants, offices, coffee shops and bars.

We delete points belonging to a transit-location because we consider points in transit-location noisy. Thus we obtain two important effects: we drastically re-
duce the number of stay-locations and can infer which are the main destinations, the geo-locations, during daily movement.

2.2 Geo-communities: extraction and notation

Our aim is to find the regions of interest of an entire population, capturing those areas shared by a few people albeit not necessarily at the same time. We introduce the concept of geo-community as the overlap of geo-locations which are not disjoint. We take into account a partial overlap to also model those situations where people have a place in common but tend to stay physically far from one another (for example, same office but different workspace).

As a starting point we use points belonging to geo-locations and not all the dataset points. This way we can find the overlap between geo-locations extracted from all the traces. Since an aggregated trace contains many border points between clusters we apply again DBSCAN using a different set of parameters ($\delta = 10, \epsilon = 1$) because of its ability to detect border and outlier points.

By means of this overlapping process we infer geo-communities that, from a geographical point of view, represent the regions of interest of a population while, from a social prospective, identify groups of people who raise a community since they have a place in common. As the consequence, a person is at the same time a member and inside a geo-community. Geo-communities represent a framework to study contact behaviours derived from people’s mobility.

As a further result, we also have a concise trace that summerizes the transitions between geo-communities and the respective pause time in them. In a more formal way, we extend TAS notation introduced in [7]. A TAS can be defined as a couple $T = (S, A)$ where $S$ is a sequence $\langle s_1, \ldots, s_n \rangle$ with temporal annotation $A = \langle a_2, \ldots, a_n \rangle$. In our case $A$ represents the temporal transition or the distance covered between two consecutive elements in $S$, i.e. the sequence of visited geo-communities. In order to introduce the pause time in TAS, for each geo-community $i$ in $S$, we add the entry time $t_i^{IN}$ and the exit time $t_i^{OUT}$ from it. Accordingly, we obtain

$$\forall t_i^{OUT} > t_i^{IN} > t_{i-1}^{OUT} \quad \text{and} \quad i = 2, \ldots, n$$

where

$$S = \{(s_1, t_1^{IN}, t_1^{OUT}), \ldots, (s_n, t_n^{IN}, t_n^{OUT})\} \quad (2)$$

For example the following extended TAS $T = ((1, 34, 67), (2, 78, 84), (9))$ represents a person who remains in the geo-community 1 from time 34 to 67, he moves to the geo-community 2 in 9 time slots and stops for 6 time slots. Adopting the previous compact representation in [6] we showed how the probability distributions of the main quantities involved in human movements and aggregations can be obtained. Our main results concerned pause-times in geo-community, intra-distances, inter-distances and how distance influences the choice of the next geo-community in the person’s movement.

2.3 Position of contacts

By analyzing spatial traces we can obtain all information about encounters typical of direct contact traces (e.g. Bluetooth, iMote, WLAN etc.) such as inter-
contact time, contact duration and contact frequency as shown in [6]. In addition, from GPS traces we can extract another important piece of information: the position of the contacts. Localization allows us to quantify how many interactions happen inside and outside geo-communities and to show the validity of the definition of geo-community.

In order to compute the quantity above we define two users to be in contact if their contact areas overlap. With the term contact area we mean a circle centered in the position of the node and characterized by a radius \( r \) that we set to 15 mt. By contrast, regarding the contact position, it is defined as the pair of GPS coordinates of the users in contact. By taking advantage of the grid used in geo-communities extraction, we can assign each of the two points to a geo-community. This way we can correlate contacts and geo-communities by checking if at least one of the points is inside a geo-community.

A further advantage of this approach can be achieved by studying the relation between contacts durations and contacts inside geo-communities. For each allowable duration value, we compute the ratio of the number of contacts inside geo-communities. Results, shown in Fig.2, can be summarized as follows:

- **KAIST**: 98.4% of all contacts occur in geo-communities as 99% of the contacts lasting more than 1 min
- **NCSU**: 92.8% of all contacts take place in communities as well 99% of those that last more than 2 min.
- **New York**: this dataset presents results very similar to the previous one
- **Disney World** and **State Fair**: due to the different nature of these places (fun fair), respectively 80.6% and 65.2% of the entire contacts occur inside geo-communities, however 90% of the contacts lasting severally 2 and 8 min, happens inside them.

![Fig. 2. Ratio of the number of contacts inside geo-communities as a function of the contact duration (1 unit = 30s)](image-url)
3 Modeling geo-communities as a network

In the previous Section we noticed how the mobility of an entire population causes a high number of contacts that mostly occur in restricted areas. On the other hand we can say that the contact, especially the lasting contact, is the first indicator of a social relationship. Social relationships and social networks are characterized by a structure that is formed by overlapping communities, where each individual belongs to one or more groups with possibly different roles.

In this perspective, the geo-community represents a link between mobility and sociability, and a potential tool with which to model and infer the characteristics of contacts between people. Another advantage is the ability to describe the geo-community and the belonging of people to them through a complex network. In particular we used a bipartite complex network, since we are dealing with two separate entities as individuals and geo-communities. The presence of an edge between a person and a geo-community indicates the membership of that person to a given geo-community. In the analysis below we considered only the membership regardless of the time, but the notation can be extended by considering a time-varying complex network.

Modeling people and geo-communities with a bipartite complex network allows us to reproduce the phenomenon of overlapping communities typical of social networks through the projection of the whole network on the set of individuals. On the other hand, projecting on the geo-community set, allows us to identify communities in motion, that is, groups of people moving from one place to another. As we can see below, projections are also a useful tool in finding a reasonable upper bound on the number of possible contacts inside geo-communities.

3.1 The bipartite graph of Geo-communities and its projections

From the perspective of modeling GPS mobility data by creating a smart complex network, the concepts of geo-community and of individuals (belonging to them) can be represented by a large bipartite graph. This kind of network appears in many settings such as social [22] and co-citation networks [16] or market-basket analysis [18]. Some examples, in which the relations between individuals and communities could be explored, are given by the conference-author network which contains information about the number of papers published by each author in each conference. On this kind of networks, for example, we can infer communities or calculate proximity measures between nodes. Therefore, the bipartite network approach is very meaningful in graph mining and, in our case, it is further enriched by an additional information: the geographical position.

As a consequence we describe the entire framework in terms of an undirected bipartite graph. An undirected bipartite graph $G = (U, V, E)$ is a graph whose vertices can be divided into two disjoint sets, $U \cup V$, such that every edge connects a vertex in $U$ to one in $V$. In our case, we define $U$ as the set of the geo-communities and $V$ as the set of individuals. Consequently, $E$ can be defined
as

\[ E = \{(v, u) | u \in U, v \in V \land v \text{ is inside } u\} \]

A bipartite graph can be also represented by a incidence matrix \( B \). If \( n \) is the number of users and \( g \) is the number of geo-communities, then \( B \) is a \( n \times g \) matrix such that \( B_{ij} = I_j(i) \), where \( I_j(i) = 1 \) if the user \( i \) belongs to the geo-community \( j \) and 0 otherwise.

Bipartite graph representation enables us to infer all possible contacts resulting from the sharing of a community common to some node. In order to reach this goal, we can perform a one-mode projection from the two-mode bipartite form. The simplest projection on the nodes can be obtained by constructing an undirected graph \( G_{\text{proj}} = (V, E') \) where

\[ E' = \{(v_1, v_2) | v_1, v_2 \in V, \exists u \in U, (v_1, u), (v_2, u) \in E\}. \]

When we form such a projection, each geo-community in the bipartite graph results in a \( k \)-clique of vertices in the projection, where \( k \) is the number of users belonging to a geo-community. Thus the overall projection is the union of \( g \) cliques. In this simple way we can reproduce the structure of contacts characterized by overlapping communities typical of social networks.

It is easy to see that, in general, this construction discards a lot of information present in the structure of the contacts. For example, two users could belong to the same geo-community but visit it in different occasions, so a link between them cannot exist. However, already using this type of projection, we can find an upper bound on the number of contacts between nodes that can be taken within the geo-community, as shown in Fig.3. Such upper bound identifies those potential contacts that may occur only because of sharing a geo-community. In an optimistic way it represents a possible prediction on both contacts and social relationships among people.

![Figure 3](image-url)

**Fig. 3.** Rate of actual contacts on the number of possible contacts computed on the eight most visited geo-communities in each dataset.
We also applied a weighted projection on the set of the geo-communities. The result of this projection is a undirected weighted graph $GC$, where the vertices are the geo-communities and the weight $w_{ij}$ represents the number of people belonging to both $i$ and $j$. This kind of tool can be used to highlight the possible flow of people between geo-communities. Furthermore, we can extract the hub-geo-communities (geo-communities who share many people with the other geo-communities), identify geo-communities that can be visited by groups of people or find closely related geo-communities. A visual example is given by the projection $GC$ of the bipartite network inferred from the dataset of the KAIST campus. In Fig.4 we can see how geo-communities 1,3,4,7,41 are closely related and how geo-communities 1,3,41 are hub-geo-communities.

![Fig. 4. Projection on the geo-community set of the bipartite graph taken from KAIST dataset. Vertex size and color are proportional to the weighted degree and edge size is proportional to its weight.](image)

### 3.2 More refined projections

Many direct contact traces have been studied in literature, by applying different strategies in the construction of the contact graph [12], [9], [20]. To the best of our knowledge, the properties of the contact graph induced by the overlap of GPS traces have never been investigated.

As our main purpose is to analyze the structure of the contacts detected only within geo-communities, we discard all coordinates of the contacts that happen outside geo-communities. To maintain uniformity, in our analysis we apply the same definitions and the same contact radius presented in Section 2.3.
We associate to each contact its duration and for each pair of nodes we compute the total duration $d_{ij}$ (sum of the contact durations). After that, we define the contact graph through the matrix $C$ where

$$C_{ij} = \begin{cases} 
1 & \text{if } d_{ij} \geq \theta \\
0 & \text{otherwise}
\end{cases}$$

We set the threshold $\theta = 2\text{min}$ in order to delete rare and brief contacts that do not add information to the relationship between nodes.

By comparing the contact graph $C$ to $G_{\text{proj}}$, we can evaluate the existing relation between real and potential contacts. In particular, for each geo-community we compute the ratio $p_u$ of the number of edges between vertices belonging to it and the maximum number of edges given by the unweighted projection of the same geo-community. By means of this index, we introduce a new type of projection. Let us assume, in the bipartite graph, to assign to each vertex $u \in U$, representing a geo-community, a value equal to the previous ratio $p_u$. Fixed an $\alpha \geq 1$, each geo-community is projected onto a subgraph built according to the Erdős-Rényi model of parameters $n$, equal to the number of vertices belonging to the geo-community, and $p = p_u^\alpha$. As for the other projections, we join subgraphs obtaining a proportional projection.

To enrich the projection, we develop a new way of projecting based on a preferential mechanism. This approach is justified not only by many results in literature [12], [10] showing the presence of a so called “hub node”, but also by the number of neighbors distribution inside geo-communities that we observed in real traces [6]. Our way of projecting builds the subgraphs of the respective geo-communities by using two kinds of construction. If $k$, the number of users belonging to a geo-community, is less than a fixed threshold $\eta$, then we apply an unweighted projection creating a $k$-clique. Otherwise we build a Barabasi-Albert graph [1] on the vertices inside the geo-community. After subgraph constructions we join them in a unique graph obtaining a preferential projection. The use of a threshold can be explained by observing that in real traces, when geo-community cardinality is small, everybody meets.

We can observe that both preferential projection and proportional projections are parametric models; the first depends on the threshold $\eta$ and the number of edges $m$, while the second relies on $\alpha$. Therefore, we can exploit this characteristic with the aim of finding the parameter values whose projection is able to provide the best reproduction of the contact graph properties. Among the previous analyzed features of the graph, we consider vertex degrees and their distribution. For each GPS dataset we compute the parameters of the distribution better fitting sampled degrees. Concerning the preferential projection, we perform a grid search, i.e. a bidimensional numerical search over the parameters space, to identify the pair of values whose projection better approximate the real one. We apply the same method to proportional projection considering a mono-dimensional parameter space.

By analyzing the best projections obtained on different datasets, we can say that the preferential projection reproduces quite closely many characteristics of
Fig. 5. In this figure we synthesize our method. From the extracted geo-communities 5(a) and the bipartite graph form, we build two different projections 5(c), 5(d). We compare them to the contact graph 5(b) inducted by overlapping GPS traces. In this dataset, we can see how geo-communities position reflects on the high modular structure of the related contact graph.
KAIST contact graph ($\eta = 13, m = 3$), whereas the proportional projection better applies on situations where there are several connected components or the graph structure is highly modular, i.e. there exist non-overlapped communities with few elements. For example, as we can see in Fig.5, proportional projection captures many features of Disney World dataset by reducing the number of paths connecting different communities.

To summarize, the bipartite graph and its projections well fit with our approach where nodes move in shared areas (geo-communities) and the sharing of these areas by different nodes generates contacts and social interactions. By applying the best projection on the bipartite graph, we can also infer and reproduce contact graph properties being able to generate synthetic contact traces.

4 Related Work

In this section, we briefly review the related work, which can be categorized into two parts: data mining of GPS data and contact pattern analysis from mobility.

Part of our work, which involves data mining of geographical data, has been devoted to detect the significant location of a user from GPS-based traces. Zhou [24] uses information obtained from GPS to classify the most interesting places of a person, such as home, work, supermar-ket, etc. A set of trajectories is processed by the DJ-Cluster algorithm in order to find the baseline places. GDBSCAN [21] is another extension of DBSCAN, developed for clustering non-spatial attributes. Zheng et al. [23] adopted a density based clustering algorithm to extract significant locations in order to infer transportation modes and to predict users’ preferred locations. Most of these works focus on a single user trajectory, whereas we aggregate traces in order to capture socially interesting places.

Several works for trajectory clustering have been developed to find similar sub-trajectories or dense regions as [15], [8], but they do not distinguish between static and movement phases and do not analysis social aspects of mobility.

Concerning the analysis of mobility contact pattern, many of the datasets on mobility refer to direct contacts between users. They cover several environments and span different periods. For example, contact in conferences [11], [2], on campus [14] or between people affiliated with a department [13], [3] are available. The features of these traces - such as inter-contact and contact duration distributions, number of neighbors in different periods, periodic reappearances in the same places and cyclic contacts - have been explored in several studies.

Beyond all this, complex network analysis [16] has recently been proposed in the analysis of human contacts. In this framework contacts between users are modeled by a undirected graph, called social graph, with an edge representing past encounters. From the social graph we can infer many properties concerning the type and the strength of the contacts and of the social relationships. This brings us to assign to a node a relevance value by measuring some centrality metrics such as degree centrality, betweenness centrality and eigenvector centrality [16]. Many features of the social structure, such as the human tendency to group
in communities, influence the social graph. As a matter of fact in recent years
many community detection algorithms - based on different ideas of community
- have been developed [5] and have been applied on social networks to extract
real communities, even overlapped [17], and to analyze social interactions that
occur in them.

5 Conclusion and future work

In this paper we addressed some issues about modeling human mobility which on
our opinion deserve deeper understanding. The main contributions of the paper
are the following. Firstly, we provide new analysis and modeling methods and
results. We have defined the notion of geo-community which is operational in
describing in a unique framework, both spatial and social, the aspects of human
behavior. By an analysis of real traces we have investigated some contact features
relevant for capturing human social behaviours. They include potential social
relationships extracted from location sensitive quantities by means of a novel
method of projecting a bipartite graph representing the system.

Our future work will focus on the investigation of new metrics to weight the
link representing the social relationships in our graph and its projections, with
the final goal of studying spreading processes on mobile social networks.

References

   (1999)
   human mobility on opportunistic forwarding algorithms. IEEE Transactions on
   Mobile Computing (2007)
   Personal Ubiquitous Computing (2006)
4. Ester, M., Kriegel, H.P., Sander, J., Xu, X.: A density-based algorithm for discov-
   ering clusters in large spatial databases with noise. In: Proceedings of 2nd Inter-
   national Conference on Knowledge Discovery and Data Mining (1996)
   from location-aware traces. Wireless Communications and Mobile Computing (to
   appear)
   sequences. In: Proceedings of the Sixth SIAM International Conference on Data
   Mining (2006)
8. Giannotti, F., Pedreschi, D.: Mobility, Data Mining and Privacy - Geographic
   Knowledge Discovery. Springer (2008)
9. Hossmann, T., Spyropoulos, T., Legendre, F.: Know Thy Neighbor: Towards Opti-
   mal Mapping of Contacts to Social Graphs for DTN Routing. In: Proceedings of
   the 29th IEEE International Conference on Computer Communications (2010)
Generalized network community detection

Lovro Šubelj and Marko Bajec
University of Ljubljana, Faculty of Computer and Information Science, SI-1001 Ljubljana, Slovenia,
[name.surname]@fri.uni-lj.si

Abstract. Community structure is largely regarded as an intrinsic property of complex real-world networks. However, recent studies reveal that networks comprise even more sophisticated modules than classical cohesive communities. More precisely, real-world networks can also be naturally partitioned according to common patterns of connections between the nodes. Recently, a propagation based algorithm has been proposed for the detection of arbitrary network modules. We here advance the latter with a more adequate community modeling based on network clustering. The resulting algorithm is evaluated on various synthetic benchmark networks and random graphs. It is shown to be comparable to current state-of-the-art algorithms, however, in contrast to other approaches, it does not require some prior knowledge of the true community structure. To demonstrate its generality, we further employ the proposed algorithm for community detection in different unipartite and bipartite real-world networks, for generalized community detection and also predictive data clustering.

Keywords: link-density community, link-pattern community, propagation, community detection, data clustering

1 Introduction

Over a decade of research in network analysis has revealed a number of common properties of complex real-world networks [57,13]. Community structure [18]—the occurrence of cohesive modules of nodes—is of particular interest as it provides an insight into not only structural organization but also functional behavior of various real-world systems [35,2]. The analysis of communities has thus been the focus of many recent endeavors [15,38], while community structure analysis is also considered as one of the most prominent areas of network science [8,38].

However, most of the past work was constrained to communities characterized by higher density of links—link-density communities [18] (Fig. 1(a)). In contrast to the latter, recent studies reveal that networks comprise even more sophisticated modules than classical cohesive communities [34,1,37,52]. In particular, real-world networks can also be naturally partitioned according to common patterns of connections among nodes—into link-pattern communities [28,34] (Fig. 1(b)). Link-pattern communities can in fact be related to relevant functional roles in various complex systems [37,52], moreover, they also provide a
further comprehension of real-world network structure that is obscure under classical frameworks. Note that link-density communities could be seen as a special case of link-pattern communities, although several fundamental differences exist [52]. In particular, link-pattern communities do not correspond to densely connected groups of nodes, while generally do not even feature connectedness. The latter actually implies low transitivity—clustering coefficient [57]—for the nodes in link-pattern communities, which contradicts with small-world phenomena [57]. However, recent work suggests that best link-pattern communities might indeed emerge in parts of networks that exhibit low values of clustering (e.g., technological networks), where small-world property does not generally hold [50].

Recently, Šubelj and Bajec [52] have proposed a general propagation algorithm that can reveal arbitrary network modules ranging from link-density to link-pattern communities. Their algorithm does not require any prior knowledge of the true structure, though they introduce a community parameter that models the nature of each community according to the measure of network bottlenecks—conductance [5]. We advance the latter by proposing a more adequate modeling strategy based on node clustering coefficient [57]. The resulting algorithm is evaluated on various synthetic benchmark networks with planted partition, on random graphs and also resolution limit examples. It is shown to be comparable to current state-of-the-art, whereas, the proposed strategy also greatly improves on the approach of Šubelj and Bajec [52] (on these networks). Furthermore, to demonstrate its generality, we also employ the algorithm for community detection in different unipartite and bipartite real-world networks, for generalized community detection and predictive data clustering.

The rest of the paper is structured as follows. In Section 2 we briefly review relevant related work, with emphasis on the community detection literature. Section 3 introduces the proposed algorithm, while the empirical evaluation with formal discussion is done in Section 4. The performance on various real-world examples is presented in Section 5, and conclusions are made in Section 6.
2 Related work

Despite the wealth of the literature on classical communities in recent years [15], only a small number of authors have considered more general link-pattern communities. Nevertheless, authors have recently proposed different algorithms based on stochastic blockmodels [1,20], mixture models [34,47], model selection [28,37], data clustering [27] and other [36]. However, in contrast to the propagation algorithm proposed in this paper, and that in [52], all other approaches require some prior knowledge of the true structure (e.g., the number of communities). The latter indeed seriously limits their use in practice. Note that authors have also analyzed vertex similarity based on common patterns of connections [4,26]—commonly referred to as structural equivalence—whereas, some of the research on classical communities also apply for link-pattern counterparts [19,45].

It ought to be mentioned that link-pattern communities are known as blockmodels [58] in social networks literature. These have been extensively studied in the past, however, their main focus and employed formulation differs from ours.

3 Model-based propagation

Let the network be represented by an undirected graph $G(N,L)$, where $N$ is the set of nodes of the graph and $L$ is the set of its links (edges). Furthermore, let $w_{nm}$ be the weight of the link between nodes $n, m \in N$. Moreover, let $c_n$ denote the community (label) of node $n \in N$, and let $\Gamma_n$ be the set of its neighbors.

The proposed model-based propagation algorithm is, as the algorithm in [52], based on the label propagation principle of Raghavan et al. [39]. In the following, we thus first introduce the latter.

**Label Propagation.** Label propagation algorithm [39] (LPA) reveals link-density communities by exploiting the following procedure. First, each node $n \in N$ is labeled with a unique label (i.e., $c_n = l_n$). Then, at each iteration, the node adopts the label shared by most of its neighbors (with respect to link weights). Hence,

$$c_n = \arg\max_l \sum_{m \in \Gamma^l_n} w_{nm},$$

where $\Gamma^l_n$ is the set of neighbors of node $n$ that share label $l$ (ties are broken uniformly at random$^1$). Due to the existence of many intra-community links, relative to the number of inter-community links, cohesive modules of nodes form a consensus on some label after a few iterations. Thus, when the algorithm converges—a local equilibrium is reached—disconnected sets of nodes sharing the same label are classified into the same community. Due to extremely fast structural inference of label propagation, the algorithm exhibits near linear complexity and can easily scale to networks with millions of nodes and links [53,39].

Note that, to address issues with oscillations of labels in some networks (e.g., bipartite networks), label updates in Eq. (1) occur in a random order [39].

$^1$ When node’s current label is among most frequent, the node retains its label.
General Propagation. Subelj and Bajec [52] have argued that label propagation cannot be directly applied for the detection of link-pattern communities, as the bare nature of propagation requires connected (and cohesive) groups of nodes. However, when one considers second order neighborhoods, and propagates labels through nodes’ neighbors, link-pattern communities indeed correspond to cohesive modules of nodes (see Fig. 1(b)). Based on the above they have proposed general propagation algorithm [52] (GPA) that is presented in the following.

Let $\delta_c$ be a community parameter that models the nature of community $c$, $\delta_c \in [0, 1]$. Assume $\delta_c$ equals 1 and 0 for link-density and link-pattern communities, respectively (to be properly defined later). Label propagation in Eq. (1) is then advanced into a general community detection algorithm as

$$c_n = \arg\max_l \left( \delta_l \sum_{m \in \Gamma_n} w_{nm} b_m d_m + (1 - \delta_l) \sum_{m \in \Gamma_n \setminus \Gamma_s \setminus \Gamma_n} w_{nm} b_m \tilde{d}_m \right)$$

(2)

where $w_{nm} = \frac{w_{mn} w_{nm}}{s_n}$ and $s_n$ is the strength of node $n \in N$ (i.e., $s_n = \sum_{m \in \Gamma_n} w_{nm}$). In the case of link-density communities (left-hand side of Eq. (2)), the labels are propagated among the neighbors as before, whereas, in the case of link-pattern communities (right-hand side of Eq. (2)), the labels are propagated through nodes’ neighbors—between the nodes at distance two. Thus, the algorithm can indeed reveal either link-density or link-pattern communities, or different mixtures of both, when they are clearly depicted in the network’s topology. (Note that in [52] the algorithm was presented for unweighted networks.)

Node balancers $b_n$ [51] and diffusion values $d_n$, $\tilde{d}_n$ [53,52] in Eq. (2) improve the algorithm’s stability and accuracy, respectively. More precisely, random label update orders (see above) severely hamper the robustness of the approach, and consequently also the stability of the identified community structure [54]. In particular, nodes that are updated at the beginning exhibit higher propagation preferences than those that are updated towards the end [51]. Thus, balancers $b_n$ are utilized to counteract for the randomness introduced by update orders—lower and higher preferences are given to the nodes updated first and last, respectively.

Let $i_n$ denote a normalized position of node $n \in N$ in some random order, $i_n \in (0, 1]$. Then, node balancers are set according to

$$b_n = \frac{1}{1 + e^{-\mu(i_n - \lambda)}}$$

(3)

where $\lambda$ and $\mu$ are parameters of the algorithm. Intuitively, we fix $\lambda$ to $\frac{1}{2}$, while $\mu$ is set to 2 based on some preliminary experiments (see Section 4). Node balancers can also be modeled with a linear function as $b_n = i_n$, however, introduction of the above parameters allows for a distinct control over the algorithm. In particular, analysis in Section 4 reveals that increasing $\mu$ improves the stability of the algorithm, although the computational time thus also increases. Note also that setting $\mu$ to 0 yields a classical label propagation where all $b_n$ are equal.

To further boost the community detection strength of the algorithm, defensive preservation of communities is employed through diffusion values $d_n$. 
Here higher diffusion values—propagation preferences—are given to core nodes of each (current) community, while lower values are given to their border nodes. The latter results in an immense ability of detecting communities, even when they are only weakly depicted in the network’s topology [53]. At each iteration, diffusion values are estimated by means of a random walker utilized on each (current) community. Hence,

\[ d_n = \sum_{m \in \Gamma_n^n} d_m / k_{m}^{\text{cn}} \]  

(4)

and

\[ \tilde{d}_n = \sum_{m \in \Gamma_n^n \setminus \Gamma_n | s \in \Gamma_n} \frac{\tilde{d}_m}{\sum_{s \in \Gamma_m} k_{s}^{\text{cn}}}, \]  

(5)

where \( k_{m}^{\text{cn}} \) is the intra-community degree of node \( n \in N \). Besides deriving an estimate of the core and border of each community, the main rationale here is to formulate propagation—diffusion—within each community, to estimate the current state of label propagation, and then to adequately alter the dynamics of the process. Analysis in Section 4 reveals that defensive preservation of communities significantly improves the detection strength of the algorithm, while for further discussion and analysis see [53].

Despite the discussion above, the core of the algorithm is in fact represented by a community modeling strategy implemented through parameters \( \delta_c \). Šubelj and Bajec [52] have proposed to measure the conductance [5] of each community, to determine whether it better conforms with link-density or link-pattern regime. Conductance \( \Phi(c) \) of community \( c \) is defined as a relative size of the corresponding network cut—ratio of inter-community links—thus it is a measure of network bottlenecks. Hence, at each iteration, they simply set \( \delta_c = 1 - \Phi(c) \), while all \( \delta_c \) are initialized to \( \frac{1}{2} \). The main weakness of their strategy is that each community is considered independently of other. Thus, in the following, we propose a more adequate community modeling strategy based on the properties of complex real-world networks.

**Model-based Propagation.** Community modeling strategy of Šubelj and Bajec [52] considers merely the nature of each respective community, whereas all other communities are disregarded. Although no proper empirical study exists, in an ideal case, link-pattern communities would link to other link-pattern communities rather than to other link-density communities. The latter follows from the fact that the concerned links would else obviously decrease the quality of the respective link-density community—make it a link-pattern community. Thus, we propose a community model based on the hypothesis that the neighbors’ communities should be of the same type—either link-density or link-pattern—as the concerned node’s community. Hence,

\[ \delta_c = \frac{1}{|N^c|} \sum_{m \in \Gamma_n^n | n \in N^c} \frac{\delta_{cm}}{k_n}, \]  

(6)
where $k_n$ is the degree of node $n \in N$ and $N^c$ is the set of nodes in community $c$.

We also argue that an adequate initialization of community parameters $\delta_c$ is of vital importance (exact results are omitted). Otherwise, the algorithm can easily get trapped in some local stable—probably suboptimal—fixed point that is hard to escape from. However, Eq. (6) cannot be directly employed at the beginning, as all nodes still reside in their own communities. We thus refine the above hypothesis such that the node’s neighbors should not only reside in the same type of the community, but in the same respective community. The latter immediately implies that the neighbors of the nodes in link-density communities should also link to each other, whereas the opposite holds for the nodes in link-pattern communities. Hence, for each node $n \in N$, one could initially set $\delta_{c_n}$ to $C_n$, where $C_n$ is a node clustering coefficient [57] defined as the probability that two neighbors of node $n$ also link to each other—network transitivity. It ought to be mentioned that recent work suggests that transitivity—rather than homophily—gives rise to the modular structure in real-world networks [17].

However, consider a node with very high degree—a hub node. Hubs commonly appear in link-density communities [19], still, due to a large number of links, they would only rarely experience high values of clustering coefficient (the opposite would in fact imply a large clique). Also, as most networks are disassortative by degree [31], hubs tend to link to low degree nodes that cannot provide for high clustering of the hub node [48]. Indeed, in many real-world networks node clustering coefficient roughly follows $C_n \sim k_n^{-1}$ [56,40,48], where $k_n$ is the degree of node $n \in N$. Hence, we model initial communities as (assume $C_n > 0$)

$$
\delta_{c_n} = \begin{cases} 
1 & \text{for } C_n > \alpha k_n^{-1} + \beta, \\
\rho & \text{otherwise},
\end{cases}
$$

where $\alpha$ and $\beta$ are estimated from the network using ordinary least squares, and $\rho$ is a parameter. We set $\rho$ to $\frac{1}{4}$ based on some preliminary experiments.

Eq. (7) and Eq. (6) define the proposed model-based propagation algorithm (MPA), which is else (almost) identical to the algorithm in [52] (see Alg. 1). However, the evaluation on synthetic and real-world networks in Section 4 and Section 5, respectively, reveals that the proposed approach significantly outperforms that in [52]. For a thorough evaluation, we also analyze two variations of the basic approach that fix all community parameters $\delta_c$ to either 1 or 0. The approaches thus result in a fully link-density or link-pattern community detection algorithms, and are denoted MPA(D) and MPA(P), respectively.

4 Evaluation and discussion

In the following we evaluate the proposed algorithm on different synthetic benchmark networks with planted partition, and also on random networks. The results are assessed in terms of three different measures of community significance, borrowed from the field of information theory and community detection literature.

Let $C$ be a partition extracted by an algorithm and let $P$ be the known partition of the network (corresponding random variables are $C$ and $P$, respectively).
Algorithm 1 Model-based propagation algorithm (MPA)

**Input:** Graph $G(N,L)$ and parameters $\lambda, \mu, \rho$

**Output:** Communities $C$

1. **Community initialization.**
   - For $n \in N$
     - $c_n \leftarrow l_n$ (Unique label)
     - $\delta_{c_n} \leftarrow$ (Model according to Eq. (7))
     - $d_n, \tilde{d}_n \leftarrow 1/|N|$

2. **Model-based propagation.**
   - While not converged
     - Shuffle $N$
     - For $n \in N$
       - General propagation.
         - $b_n \leftarrow 1/(1 + e^{-\mu(i_n - \lambda)})$
         - $c_n \leftarrow \arg\max_l \left(\delta_l \sum_{m \in \Gamma_n \setminus \Gamma_{n,n}} w_{nm} b_m \tilde{d}_m + (1 - \delta_l) \sum_{m \in \Gamma_n \setminus \Gamma_{n,s}} w_{nm} b_m d_m\right)$
       - Re-estimation.
         - $d_n \leftarrow \sum_{m \in \Gamma_n} d_m / k_n$ and $\tilde{d}_n \leftarrow \sum_{m \in \Gamma_n \setminus s \in \Gamma_n} \tilde{d}_m / \sum_{s \in \Gamma_m} k_s$

3. **Community modeling.**
   - For $c \in C$
     - Community modeling.
       - $\delta_c \leftarrow 1/|N_c| \sum_{m \in \Gamma_n, n \in N_c} \delta_{c_m} / k_n$ (Omitted on first iteration.)

4. Return $C$

First—normalized mutual information [8] (NMI)—has become a de facto standard in the recent literature. NMI of $C$ and $P$ is defined as $\frac{2I(C,P)}{H(C) + H(P)}$, where $I(C,P)$ is the mutual information, and $H(C)$, $H(P)$ and $H(C|P)$ are standard and conditional entropies. NMI of identical partitions equals 1, and is 0 for independent ones, NMI $\in [0, 1]$. Second, we also consider normalized variation of information [30, 22] (NVOI), which is a symmetric local measure that has the properties of a distance in the space of partitions. NVOI of $C$ and $P$ equals $\frac{H(C|P) + H(P|C)}{\log |N|}$, therefore, in contrast to NMI, lower values represent better correlation between partitions, NVOI $\in [0, 1]$. Last, for a better comprehension, we also adopt a more intuitive measure—fraction of correctly classified nodes [18] (FCC)—that is commonly adopted within community detection literature. The node is considered correctly classified, if it resides in the same community as at least one half of the nodes in its true community. Again, FCC $\in [0, 1]$.

Community detection algorithms introduced in Section 3 are compared against a greedy agglomerative optimization [32, 7] of modularity $Q$ (denoted MO(G))—a classical link-density community detection algorithm—and a mixture model with expectation-maximization [10] proposed by Newman and Leicht [34] (denoted MM(EM)). The latter can detect arbitrary network modules and is currently among state-of-the-art approaches for generalized community detection [34, 37].
However, it demands the correct number of communities to be known ahead of time, which puts the algorithm in significant advantage compared to others [23]. For simplicity, we limit the number of iterations to 100 for all algorithms.

**GN2 Benchmark.** The algorithms are first applied to a class of benchmark networks [37] that is in fact a generalization of a classical benchmark proposed by Girvan and Newman [18]. Networks comprise four communities of 32 nodes, whereas, two communities correspond to classical link-density modules, while the other two form a bipartite structure of link-pattern communities. Average degree is fixed to 16, while the community structure is controlled by a mixing parameter $\theta$, $\theta \in [0,1]$. When $\theta$ is 0, all links are set according to the designed community structure, while for $\theta$ equal 1, the networks are completely random.

The results are shown in Fig. 2. Observe that for small values of $\theta$ only MPA and MPA(P) can accurately reveal the planted structure in these networks. However, when $\theta$ increases, the performance of MPA is similar to that of a classical community detection algorithm (e.g., MO(G) or MPA(D)). MM(EM) can detect communities to some extent until $\theta \leq \frac{1}{3}$ (dashed lines in Figs. 2, 3)—when, for the nodes within link-density communities, there are twice as many links that conform with the planted structure than randomly placed links. Note also that twice as many links are needed to define a link-pattern community, compared to a respective link-density community, which would yield the same threshold at $\theta = \frac{1}{6}$ for these networks (solid lines in Figs. 2, 3). Thus, MPA accurately extracts planted link-density and link-pattern communities in these networks, as long as they are clearly depicted in the network’s topology. Note also that community modeling strategy within MPA seems more adequate than that of GPA.

**SB Benchmark.** GN2 benchmark provides a rather unrealistic testbed due to homogeneous degree and community size distributions. We address the latter by proposing a class of simple benchmark networks with heterogeneous community sizes. Networks comprise three communities of 16, 32 and 24 nodes, respectively (see network in Fig. 8(a)). The latter two again form a bipartite structure of link-pattern communities, while the third community corresponds to a classical cohesive module. Links are placed according to the designed community structure such that the average degree of the nodes in the first and third community is fixed to 16. The latter implies an average degree of 8 for the nodes in the second community. Furthermore, we also add some number of links uniformly at random for each node—denoted node confusion degree $\kappa$, $\kappa \geq 0$.

The results appear in Fig. 3. The performance of the algorithms is rather similar to that on GN2 benchmark (note different scales in Figs. 2, 3). Only MPA can accurately reveal the planted structure for small values of $\kappa$, while the model within GPA again seems to fail. Observe that MM(EM) can extract communities equally well, even when $\kappa$ equals 16—only $\frac{1}{3}$ of the links for the nodes in the second community still agrees with the intrinsic structure, thus, the communities are only marginally defined. The latter clearly demonstrates that knowing an exact number of communities indeed presents a significant advantage.
Fig. 2. Analysis on GN2 benchmark networks [37]. The values are estimates over 100 network realizations, while error bars show standard error of the mean.

Fig. 3. Analysis on SB benchmark networks. The values are estimates over 100 network realizations, while error bars show standard error of the mean.

Fig. 4. Analysis on LFR benchmark networks. The values are estimates over 10 network realizations, while error bars show standard error of the mean. To ensure convergence, $\mu$ is set to $\frac{1}{2}$.
**LFR Benchmark.** To enable easier comparison with previous literature on community detection, we also apply the algorithms to a class of standard benchmark networks with scale-free degree and community size distributions proposed by Lancichinetti et al. [25]. The size of the networks is set to 1000, while community sizes range between 10 and 50 nodes. Note that all communities here correspond to a link-density regime. As before, the quality of the planted structure is controlled by a mixing parameter θ, θ ∈ [0, 1]. For comparison, we also analyze two variations of MPA that do not employ either balanced propagation or defensive preservation of communities (denoted MPA-D and MPA-B, respectively).

Results in Fig. 4 show that MPA most accurately reveals the planted structures in these networks, while it also significantly outperforms the other generalized community detection algorithm MM(EM). Observe also that defensive preservation of communities greatly improves the algorithm’s community detection strength. Comparing the results with an analysis of over ten state-of-the-art approaches for classical community detection conducted in [24], we conclude that, at least on these networks, MPA performs similarly as the best algorithms analyzed there. These are hierarchical modularity optimization of Blondel et al. [3], model selection technique of Rosvall and Bergstrom [44], spectral algorithm proposed by Donetti and Muñoz [11] and multi-resolution spin model of Ronhovde and Nussinov [43].

**HN Benchmark.** Next, we also analyze the proposed algorithm on a class of benchmark networks with a hierarchical structure [6]. In particular, networks are constructed according to a community dendrogram in Fig. 5(a), where leafs correspond to eight modules of 16 nodes, while each node d of the dendrogram is also associated with a probability pd, pd ∈ [0, 1]. The nodes of the network are linked with the probability associated with the lowest common ancestor in the community dendrogram. Varying the values of pd can infer (almost) arbitrary hierarchical structure of either link-density or link-pattern communities. However,
Due to simplicity, we associate each level of the nodes with the same probability \( p_d \). Thus, denote \( p = [p_1, p_2, p_3, p_4] \) to be the vector of respective probabilities for the nodes from the lowest to the highest level of the hierarchy, respectively.

The performance of MPA on five realizations of the above benchmark can be seen in Fig. 5. Values of NMI were estimated such that each revealed partition was compared against (only) three intrinsic community structures—represented by dashed lines in Fig. 5(a)—and the best correspondence was reported. (Note that the results are thus actually rather pessimistic.) Observe that MPA can accurately reveal the planted structure in all five cases—see legend in Fig. 5(b)—which further confirms the adequacy of the proposed community model. More precisely, in the first case, the intrinsic network structure results in a hierarchy of link-pattern communities, whereas, in the second case, communities are in fact defined on two levels of the designed hierarchy. In each of the last three cases, the communities corresponds to a single level of the hierarchy. Thus, MPA can indeed be employed for the detection of arbitrary community structure.

Random Graphs. We also apply the algorithms to Erdős-Rényi random graphs [12] that presumably have no community structure. We fix the number of nodes to 128 and vary the average degree from 4 to 32. The results are shown in Fig. 6(a). Note that, in contrast to MO(G), neither MPA nor GPA reports any community structure for these networks—all nodes are classified into a single community.

Resolution Limit. We further analyze the algorithms on a resolution limit [16]—existence of an intrinsic scale within the algorithm, below which the communities are no longer recognized—test benchmarks networks [16]. Hence, the networks consist of cliques with 4 nodes that are linked into a ring. Results in Fig. 6(b) reveal that neither MPA nor GPA is seriously attributed to the resolution limit issues, whereas, the opposite holds for MO(G). Although some fluctuations are indeed observed for MPA, these are not as severe as in the case of modularity [16].

Fig. 6. Analysis on (a) random graphs [12] and (b) resolution limit test networks [16]. The values are estimates over 100 network realizations, while error bars are smaller than the symbol sizes.
Algorithm Stability. As previously discussed, random label update orders severely hamper the stability of label propagation, and thus also the robustness of the revealed community structure [54]. Hence, balanced propagation [51] is employed, yet this introduces two parameters $\lambda$ and $\mu$ (Section 3). Value of $\lambda$ is intuitively fixed to $\frac{1}{2}$ (see Eq. (3)), while parameter $\mu$ in fact controls the stability of the algorithm. In Fig. 7 we analyze MPA with respect to stability parameter $\mu$ on three real-world networks from Table 1. Plots show pair-wise distance between revealed community structures, and also the number of iterations for the algorithm to converge (note different scales). Observe that increasing $\mu$ improves the stability of MPA in all three networks, however, the number of iterations also increases. Furthermore, as one would expect, when $\mu$ exceeds a certain threshold, pair-wise distance between community structures notably increases—some number of nodes already gets completely disregarded due to propagation preferences close to 0 (see Eq. (3))—while the number of iterations can also increase substantially (see Fig. 7(b)). The transition occurs at around $\mu \approx 4$ for these networks, thus, for the analysis throughout the paper, $\mu$ is set to 2 (if not stated otherwise). It ought to be mentioned that balanced propagation can also improve community detection strength of the basic label propagation [51] (see above).
Community Modeling. For a comprehensive analysis, we also directly analyze the proposed community modeling strategy of MPA on SB benchmark networks with confusion degree set to 2 (see above). In particular, we measure the average value of community parameter $\delta_c$ (Section 3) for the nodes in each of the planted network communities. Results in Fig. 8 show that, at least for these networks, values of community parameter $\delta_c$ clearly distinguish between link-density and link-pattern regime—average $\delta_c$ is close to 1 and 0 for the nodes in link-density and link-pattern communities, respectively. Note also that, due to lower average degree, values of $\delta_c$ are initially higher for the larger of the two link-pattern communities. However, before the algorithm converges—average number of iterations is shown by a horizontal line in Fig. 8(b)—community model in MPA infers the same average value of $\delta_c$ for both link-pattern communities. Note also that GPA cannot properly model communities planted in these networks (Fig. 3).

Computational Complexity. Basic label propagation and its advances exhibit near linear time complexity $O(|L|)$ [39], where $|L|$ is the number of links in the network. In particular, the exact complexity was estimated to around $O(|L|^{1.2})$ [53]. Similarly, the proposed model-based propagation MPA exhibits complexity near $O(k|L|)$, where $k$ is the average degree in the network. Although a thorough empirical analysis is out of scope of this paper, based on the results in [53] (and above), we estimate that MPA should scale up to networks with a million links—accessible on a standard desktop computer within an hour.

Final Remarks. The above analysis on different benchmark networks and random graphs indeed confirms that MPA can reveal arbitrary composites of either link-density or link-pattern communities, as long as they are clearly depicted in the network’s topology. Moreover, the proposed community modeling strategy also seems more adequate than the approach proposed by Šubelj and Bajec [52].
Table 1. Analysis on real-world networks subject to NMI estimated over 1000 runs (10
runs for software networks). Corporate network is reduced to the largest component,
while the known partition is also limited to 86 corporate nodes—we thus set \( \mu \) to \( \frac{1}{2} \).

<table>
<thead>
<tr>
<th>Network</th>
<th>Nodes</th>
<th>Links</th>
<th>Comm.</th>
<th>MO(G)</th>
<th>GPA</th>
<th>MM(EM)</th>
<th>MPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Zachary’s karate club [59]</td>
<td>34</td>
<td>78</td>
<td>2</td>
<td>0.6925</td>
<td>0.7155</td>
<td>0.7870</td>
<td><strong>0.8949</strong></td>
</tr>
<tr>
<td>American college football [18]</td>
<td>115</td>
<td>616</td>
<td>12</td>
<td>0.7547</td>
<td>0.8769</td>
<td>0.8049</td>
<td><strong>0.8919</strong></td>
</tr>
<tr>
<td>Davis’s southern women [9]</td>
<td>32</td>
<td>89</td>
<td>4</td>
<td>0.7338</td>
<td>0.8332</td>
<td>0.8084</td>
<td></td>
</tr>
<tr>
<td>Scottish corpor. interlocks [46]</td>
<td>217</td>
<td>348</td>
<td>8</td>
<td><strong>0.6634</strong></td>
<td>0.5988</td>
<td>0.6411</td>
<td></td>
</tr>
<tr>
<td>Java (org namespace) [49]</td>
<td>709</td>
<td>3571</td>
<td>47</td>
<td>0.5029</td>
<td><strong>0.5190</strong></td>
<td>–</td>
<td><strong>0.5187</strong></td>
</tr>
<tr>
<td>Java (javax namespace) [49]</td>
<td>1595</td>
<td>5287</td>
<td>107</td>
<td>0.7048</td>
<td><strong>0.7369</strong></td>
<td>–</td>
<td><strong>0.7386</strong></td>
</tr>
</tbody>
</table>

for all networks considered. Further note that, although MPA is mostly outperformed by MM(EM) on the benchmarks above, the latter should be attributed to the fact that MM(EM) is advised about the number of communities. However, this currently cannot be properly estimated for large networks [23]. Moreover, MPA also performs significantly better on real-world networks in Section 5.

5 Real-world examples

In the following we further employ the proposed algorithm for community detection in different unipartite and bipartite social networks—classical and fully link-pattern community detection, respectively—and also for a generalized community detection and predictive data clustering. All of the networks considered below are regarded as unweighted and undirected.

Community Detection We first consider two classical networks for community detection—a network of social interactions between members of a karate club analyzed by Zachary [59], and a network of interplays in the 2000 NCAA American football schedule proposed in [18]—and two well-known bipartite networks—a network of social collaborations between women in Natchez, Mississippi collected by Davis [9], and a network of corporate interlocks in Scotland between 1904 and 1905 introduced in [46] (see Table 1). All these networks have known natural community structures that results from earlier studies (see also Fig. 1).

Propagation algorithms—MPA and GPA—most accurately reveal the true community structure for main of these networks (Table 1), whereas, community modeling strategy of MPA again seems more adequate than that of GPA. Note also that most values of NMI for MPA in Table 1 are considerably high.

Next, we also consider two software class dependency networks representing org and javax namespaces of Java language compiled in [49]. Here, the natural community structure should coincide with respective software packages [49], while these are expected to conform with link-density and also link-pattern
Fig. 9. Community structure of Java software network revealed with MPA (b). Only communities with more than 24 nodes are shown, still, the structure contains 1020 nodes and 4184 links. Link colors correspond to high-level software packages—javax.swing, javax.management, javax.xml, javax.print, javax.naming, javax.lang and other—while each dot was enlarged five times for better visibility.

Data Clustering

To apply community detection algorithm for data clustering, the respective dataset must first be represented by a network using some measure of similarity. According to [42], we adopt the inversed Chebyshev distance, with initial [0,1]-normalization. In order to obtain a sparse network, links must also be thresholded accordingly. (Due to simplicity, we consider only unweighted versions of the algorithms.) Note that the resulting network thus commonly decomposes into several connected components, however, community detection algorithm can still be employed to further partition these components (see Table 2).

We employ community detection to predict class variables of two famous datasets—Iris plants dataset introduced by Fisher [14], and Ecoli protein localization sites dataset [21]. For comparison, in Table 2 we also report the results for a classical partitional clustering algorithm K-Means [29] (denoted KM). Observe that MPA obtains extremely promising results on these datasets, while it also significantly outperforms MM(EM) and KM that are both advised about the

regime [52]. Again, propagation algorithms most accurately extract the true network structures (Table 1), whereas MM(EM) fails completely. In Fig. 9 we also show the community structure of javax network revealed with MPA that obtains NMI = 0.7431. Observe how communities rather agree with high-level software packages, whereas, the majority of the links in the network is consistent with the revealed structure. Interestingly, some packages contain mainly link-pattern communities (e.g., javax.swing), while others are composed of only link-density communities (e.g., javax.xml).
Table 2. Analysis of data clustering on two real-world datasets subject to NMI and FCC, respectively (estimated over 100 runs).

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Items</th>
<th>Classes</th>
<th>Links</th>
<th>Comp.</th>
<th>KM</th>
<th>MM(EM)</th>
<th>MPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Iris plants dataset [14]</td>
<td>150</td>
<td>3</td>
<td>2405</td>
<td>2</td>
<td>0.8234</td>
<td>0.8113</td>
<td>0.8264</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.8227</td>
<td>0.8196</td>
<td>0.8983</td>
</tr>
<tr>
<td>Ecoli protein dataset [21]</td>
<td>336</td>
<td>8</td>
<td>14685</td>
<td>4</td>
<td>0.5835</td>
<td>0.0797</td>
<td>0.6251</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>0.2530</td>
<td>0.0277</td>
<td>0.4164</td>
</tr>
</tbody>
</table>

number of communities. Still, the results could be further improved in various ways. (Note that low NMI for MM(EM) on Ecoli dataset is not entirely evident.)

6 Conclusions

The paper proposes an enhanced community modeling strategy for a recently introduced general propagation algorithm [52]. The resulting algorithm can detect arbitrary network modules—ranging from link-density communities to link-pattern communities—while, in contrast to most other approaches, it requires no apriori knowledge about the true structure (e.g., the number of communities). The algorithm was evaluated on various benchmark networks with planted partition, on random graphs and resolution limit test networks, where it is shown to be at least comparable to current state-of-the-art. Moreover, to demonstrate its generality, the algorithm was also employed for community detection in different unipartite and bipartite social networks, for generalized community detection and data clustering. The results imply that the proposed community model provides an adequate approximation of the real-world network structure, although, recent work suggests that network clustering and degree mixing could be even further utilized within the model [48, 17, 55, 41]. The latter will be considered for future work. (For supporting website see http://lovro.lpt.fri.uni-lj.si/.)

Acknowledgments

This work has been supported by the Slovene Research Agency ARRS within Research Program No. P2-0359.

References

Pedestrian Route Prediction from GPS Logs using Augmented Cover Trees

Gavin Smith and James Goulding
Horizon Digital Economy Research
School of Computer Science
University of Nottingham
{gavin.smith,james.goulding}@nottingham.ac.uk

Abstract. Pedestrian movement prediction can provide valuable information to location-aware services. Recent approaches have mined logs of historic movement data, encoding frequently observed patterns within mathematical models such as universal predictors and markov models. However, a characteristic of many such models is that temporal information is not treated explicitly as a variable, but instead encoded within the structural scaffolding of the model (predominantly as a state transition). This leads to non-trivial problems when handling predominantly noisy datasets such as GPS data, where missing data points abound and sample rates are inconsistent.

In this paper we present an alternative approach. While we employ matching algorithms that find the historical pattern representing the closest match to the input trajectory (using the remainder of the historic trajectory as the prediction), we encode temporal data as a first-order feature of the model directly alongside spatial information. This facilitates out-of-sequence matching capabilities that are particularly effective in the face of noisy, inconsistent data.

The algorithm we present avoids the computational intractability that would normally be associated with this sort of technique through the employment of Augmented Cover Trees. Taking advantage of this, a number of novel trajectory encodings are formulated, with the system being thoroughly evaluated using a data set of real GPS logs from mobile phones. Results highlight the logarithmic runtime of the new algorithm and show statistically significant gains in prediction accuracy compared to contemporary approaches.

1 Introduction

In this work we address the problem of route prediction from GPS location data. Given initial segmentation of each individual’s GPS data streams into logical trajectories (trajectories that start and end at known or inferred locations), the prediction problem at hand may be formalized as:

The inference of such locations, and hence segmentation, is not addressed in this work. However, many techniques exist including the use of loss of signal [22], pauses in movement [14] or other more involved learning methods [21].
Given a set of historic trajectories: \( H = t_0, \ldots, t_n \) where a trajectory \( t_i = p_0, \ldots, p_{m_i} \) and each point is equal to some feature vector, for instance: \( p_j = [x, y, t] \).

And given an observed input trajectory \( I = p_0, \ldots, p_n \)

Provide a resulting prediction \( R = p_{n+1}, \ldots, p_{n+q}, q \geq 1 \) of arbitrary length indicating the route the observed pedestrian is expected to take.

Early attempts to solve this sort of movement prediction problem were motivated by applications such as cell service provisioning (e.g. [8, 25]).

In such applications, datasets are characterized by inherently high levels of spatial and temporal quantization, and quite reasonably this led to the implementation of state based models with probabilistic transitions, e.g. Markov Models [30] and Universal Prediction. However, these foundations have led to the same models [6] dominating even recent literature despite the fact that application datasets have increased in both spatial and temporal granularity, moving from broad cell tower positions to fine-grained GPS data from consumer grade devices [3, 32].

We note that, while solid results have been achieved, these approaches are based on two implicit assumptions: correct symbol assignment (that the noise involved is not enough to raise concerns about data points to being attached to incorrect quantization points) and correct sequence assignment (that sampling rates are high enough to assume that intermediate symbols in a trajectory are not missed). However, neither of these two assumptions can be guaranteed when applying GPS data.

In GPS datasets missing data points are common and the sample rate of devices is generally unknown or inconsistent, particularly across the vast array of different devices that now generate trajectories in practice. Additionally, location accuracy is affected by a large range of factors, including satellite positions, signal obstructions (such as buildings), sensor quality, atmospheric conditions and multipathing signal interference [16].

If left unaccounted for, these violated assumptions can lead to a significant degradation in prediction performance. Plausible solutions might seem to be to either perform higher levels of quantization (reducing the impact of noise) or to uniformly resample the dataset by interpolating in a linear fashion (filling in gaps). However, neither of these processes is desirable. Firstly, high quantization undermines the utility of the route prediction result and can remove the ability to discriminate between trajectories. Secondly, blind interpolation can introduce many points of error from a single erroneous reading, dependent on the resample rate being employed.

In light of this, we propose a prediction method that does not make any such assumptions. The basic prediction mechanism we employ is similar to the approach presented in [29]: an input trajectory is matched against historic trajectories, with the remainder of the best matching trajectory being returned as the final prediction. In our work, if there exists two equally appropriate matches, the most frequent trajectory is chosen\(^2\).

\(^2\)This is equivalent to taking the prediction with the highest conditional probability since the denominator in the calculation of the conditional probability is fixed between all choices after matching.
However, in contrast to [29], we specifically do not encode sequence as part of the model structure, hence avoiding any assumption that the temporal dimension of the data is more accurate than any other. Instead we are motivated by the premise that time should be treated just as another dimension allowing a user-defined trade off between spatial and temporal error. Of course, there has traditionally been good reason to encode sequence as a structural component of a model: computational tractability. However, as shown in the literature surrounding the indexing of moving object trajectories, it is possible to construct efficient indexes in order to answer similarity queries for trajectories with time encoded as an additional dimension, given a specific notion of similarity (see, for example [13,26]).

1.1 A New Algorithm

The approach we present is tailored to the characteristics of route prediction from GPS logs and underpinned by a novel indexing and search algorithm based on an augmented Cover Tree data structure [5]. The algorithm first indexes individual point observations (between which a defined metric distance exists), in order to define a global (potentially asymmetric) trajectory similarity function. Having a non-metric similarity function is actually very appealing in this problem-space, because it reflects the notion that an input trail is only meant to match part of the historic trail and not vice-versa (hence we are looking for an injective relationship).

Finally, the approach does not place any constraints on the features we use to compose an individual point observation, allowing the incorporation of domain specific information. In this work we propose two novel feature encodings and compare them to two baseline encodings, one that does not encode time and a second that directly enforces sequence. In summary the proposed approach has the following benefits:

– **The Modelling of Time as a Feature.** Since time is modelled as a feature, its importance can be weighted just as any other feature is, allowing a user-defined level of trade off between spatial error and temporal error.

– **Match Guarantees.** Resampling and Interpolation are not required because multiple points in an input trajectory can be matched against single points in a historic trajectory. This prevents excessively long inputs not matching any trajectory while still providing a penalty mechanism. Additionally it can help prevent excessive penalization of inputs with a higher frequency rate depending on the encoding.

– **Reduced Parameterization.** Traditional matching algorithms penalize gaps in trajectories. In our approach any penalization is data-driven, with gaps being implicitly penalized by the distance to the nearest point within the historic trajectory, rather than a parameterized penalty. Additionally since the approach does not perform any spatial quantization or resampling no choices with respect to the level of quantization need to be made.
- **Computational Tractability** The algorithms employed offer tractable complexity and fast empirical performance (see sections 4.2 and 6), that is easily parallelizable in the length of the input.

- **Arbitrary feature encoding** Each point observation is simply a feature vector of arbitrary format, with the exception that a distance metric must be defined over it. In this work the \(L_2\) distance metric is almost exclusively used (see section 5).

The rest of the paper is structured as follows. In section 2 related work is discussed. Then in section 3 the trajectory similarity function and the search algorithm are discussed. In sections 4 and 5 the augmentation of the Cover Tree data structure and the proposed feature encodings are discussed respectively. Finally the experimental evaluation of the proposed encodings and similarity functions are presented comparing the performance with the movement prediction approach from [23] and the two baseline feature encodings. The results show significant improvements in prediction accuracy when using the proposed approach, particularly highlighting the gains made when and sequence is not enforced.

## 2 Related Work

Current predictive approaches typically encode time via sequential structures within their models. Because of this an input trajectory is compared to historic data point by point, with point matches occurring sequentially and in a pairwise fashion. When a point observation is encoded as a feature set, temporal information is rarely included in that set, with ordinality instead being expressed through the mechanism that the model employs to link those observations together. An example of this would be a fixed-order *Markov Model* (such as that utilized by [3]) where sequence is encapsulated within the transitions between states, but where the states themselves hold no temporal data. Work by [23] stands out in that they extend a variable length markov model to encode timing information in addition to the ordinal transitions. Using a standard prefix tree data structure, they provide additional time based discrimination through a penalty function that is invoked if timings on the input trajectory do not fall into the bounds defined at that transition. There are still problems here: time is only modelled at a very coarse level and the use of maximum and minimum bounds is only useful if such tight bounds actually exist. But moreover, because time is being modelled structurally, and via a different mechanism to other dimensions within the feature set (such as spatial position), alterations in the representation of time require a wholesale reworking of the model.

A similar situation occurs within *Universal Prediction* approaches where sequence is again encapsulated within transition encodings [12,6]. This trend continues into prediction via *Principal Component Analysis* - Eagle & Pentland [9], for example, use a binary encoding scheme to encode spatial features with time used to line up features in *post-hoc* fashion. The *Alignment Prediction* approach [29] also incorporates time though sequence as part of its underlying model, utilizing the sequential information for matching, albeit in a relaxed way where distance functions are
used allowing points within the trajectories being matched to be skipped and a match still made. However gaps must be assigned some arbitrary cost, the value of which is left to the implementer. [28, pg. 234] suggests the use of $\frac{1}{2} \sqrt{n}$ as a fixed cost for the distance between a gap and any symbol, where $n$ is the dimensionality of the feature space and $r$ is a user-defined parameter for which no further comment is made.

Other measures of trajectory similarity, particularly from the time series and database communities, have also been proposed and applied to varying problems with differing notions of optimal alignment - examples are Dynamic Time Warping (DTW), Longest Common Subsequence (LCS) and various Edit Distance (ED) algorithms (see [18] for a review of such time series approaches). However, none of these have not been employed in the movement prediction literature to date. It is of note that most of these approaches have been aimed at indexing and comparing whole trajectories, such as the Edit Distance on Real Sequences (EDR) [7]. A notable exception is that proposed in [13], particularly since it also treats time as simply another dimension. However, the accuracy with respect to partial matching is not evaluated and interpolation is required, a requirement this work aims to avoid due to the nature of the GPS data.

3 Trajectory similarity

Given a set of pairwise distances between all unordered features in two trajectories (the input trajectory and one of the historic trajectories) a global function is required to arrive at a final similarity score between input and historic trajectories. In this work we contrast the use of two such functions: Expected Minimum Distance Criteria (EMC), which minimizes the expected value of the pairwise comparisons, and Max Minimum Distance criteria (MMC), although we only provide an efficient indexing and search algorithm for MMC in this work. While the MMC may initially appear less intuitive, it turns out to be a computationally rewarding global criteria when coupled with the cover tree implementation proposed additionally achieving higher prediction accuracy in many cases. We note that both criteria are not symmetric which reflects the notion that the input trajectory is meant to match to part of the historic trajectory and not vice versa. In order to formally compare these functions the following symbols are defined:

- Let historical trajectory $P$, be encoded as a set of feature points, $\{p_0, \ldots, p_m\}$.
- Let the input query trajectory, $Q$, also be a set of feature points, $\{q_0, \ldots, q_n\}$.
- Let $\delta(p, q)$ be a dissimilarity (distance) function that provides a numeric level of dissimilarity between two arbitrary feature points.

3 Adaptation of these approaches so that they match partials trajectories, along with application to movement prediction has been marked for future work.
3.1 Expected Minimum Distance Criteria (EMC)
Formally the EMC can now be defined as:

\[ EMC(P,Q) = \frac{1}{|Q|} \sum_{q \in Q} \min_{p \in P} \delta(p,q) \]  

(1)

Hence, for each point, \( q \), in the input trajectory, \( Q \), we find the distance to the closest point, \( p \), in the historical trajectory, \( P \), and finally average the results.

3.2 Max Minimum Distance Criteria (MMC)
MMC is formally defined as:

\[ MMC(P,Q) = \max_{q \in Q} \left( \min_{p \in P} \delta(p,q) \right) \]  

(2)

This is identical to the EMC, taking each point, \( q \), in the input trajectory, \( Q \), finding the distance to the closest point, \( p \), in the historical trajectory, \( P \), but instead of averaging the results, it returns the furthest distance of these. This requirement skews the choice of patterns based on the most different point between the trajectories being compared. In situations considered to have limited symbol corruption noise this may be of benefit if small deviations in the historic patterns are what discriminate between them. An example highlighting this is shown in figure 1(a), where the first point is exactly what discriminates two possible matches to the input. In noisy environments, however, the reliance on a single point can artificially exclude patterns. This is highlighted in figure 1 (b).

4 Enabling efficient search: A multiple Cover Tree algorithm
Now that a function representing the (dis)similarity between two trajectories has been established, given an input trajectory we are able to...
search across the historical dataset in order to find its nearest neighbour. Such a search is computationally expensive, but this can be ameliorated through the use of Tree-based Indices, which utilize the structure within the data itself in order to speed up nearest neighbour queries. In cases where matching consists of comparing multiple features (in this case individual points in the trajectories), where those features must be invariant with respect to some model defined consistency function, multiple tree approaches allow features to be explored simultaneously. As such the trajectories can be indexed by a pointwise distance function (which is a true distance metric) while still enforcing a broader range of criteria such as MMC and EMC which are not symmetric. In many cases this leads to greater pruning opportunities, and consequently reduced computation. In the context of matching trajectories, we propose that the use of multiple iterators over a single tree of all historic points can be used instead of multiple trees, with each iterator searching for the closest historic point to one of the points within the input. During the search the consistency (trajectory similarity) function ensures that the iterators converge globally to points contained as part of a single historic trajectory rather than multiple different trajectories.

As discussed in-depth in the doctoral thesis of Jeremy Kubica [20], multiple tree-based algorithms have been used to great effect to address a wide range of problems such as amortizing the cost of searching for individual nearest neighbours for multiple points [4], locating asteroid tracks [20], applying dual trees to ‘N-body’-like problems and spatial intersect queries [1,17], spatial join queries [15] and complex queries [10,11]. In contrast to this prior work, while still following the general multiple tree paradigm, the proposed approach to movement prediction does not require structured queries which refer to specific features (rather requiring each point in one set to match arbitrarily with any point in the other set) and utilizes a large number of iterators on a single index with a consistency function tailored to the movement prediction problem.

The use of multiple iterators over a tree implies the choice of a tree-based index, for which we select the Cover Tree for our predictor. The choice of the Cover Tree over standard indices such as the R-Tree, KD-Tree or M-Tree is motivated by two factors:

- Movement prediction can potentially benefit from using an array of contextual features (such as weather, mood, social context, etc.) that are not appositely represented as numerical values in a multidimensional space, but for which a complex distance function can be generated. Cover Trees, like M-trees, are hence favourable because they assume a Metric and not a Euclidian space.

- It is envisioned that a large number of features will be incorporated, and hence structures that are designed to perform well with high dimensional features is desired. Additionally it is expected that a large number of trajectories will be indexed. Unlike Cover Trees, the M-Tree and its variants are not guaranteed to have logarithmic runtime performance in the number of points, and thus the former is preferred.

The Cover Tree [5] is a tree based index of a set of features constructed such that at the $i^{th}$ level of the tree the nodes in that level are all points
that have a distance greater than $2^i$ between each other (separation property). Stepping down a level in the tree the children for each node include the point represented by the parent (nesting property) and have the property of being separated by a distance of at least $2^{i-1}$ with respect to each other (separation property at the next level down) while being within a distance of $2^i$ with only one point at the $i$th level, its parent (covering tree property). As such only one path exists to each node in the tree (down from each child’s unique parent) with the parents being close to the children. For a more detailed explanation of the Cover Tree and graphical examples see [19,5,4]. Intuitively, at the higher levels of the tree only a few, spread out, data points are visible. As one proceeds to lower levels of the tree more and more points come into view, dispersed relative to the minimum inter-point distance permitted at that particular level.

Searching for the nearest neighbour in this data structure works by first calculating the distance between the point and the root, $\alpha$. It can be shown (see [5] for the proof) that the closest point must either be the root, or be a point at the current level within $\alpha$, or a child of a point at the current level within $\alpha + 2^i$ where $i$ is the current level. Therefore the search progresses by, at each level, calculating $\alpha$ as the minimum distance between the query point and any node under consideration at that level and then selecting all points within $\alpha + 2^i$. The children of these points (remembering each parent is a child of itself) then become the nodes under consideration at the next level. As $i \to -\infty$, $2^i \to 0$ and only a single point is left.

In other words an upper bound on the minimum distance to the nearest neighbour is constantly reduced as more and more points are brought into view, with an exponentially shrinking term ensuring that no, currently unrevealed, points are missed. It is the monotone decreasing property of the minimum distance that is utilized in the multiple iterator approach to perform nearest neighbour search over the historic movement trajectories.

4.1 Prediction via an augmented cover tree

In this section we discuss the use of the augmented cover tree to effectively index, search and finally predict trajectories based on the MMC. The tree is constructed by indexing each point from each historic trajectory independently. In addition the cover tree structure is modified, storing at each node a pointer to the historic trajectories that the node and its children are part of. When performing the nearest neighbour search one iterator is created per point within the input. Normal point-based nearest neighbour search is then conducted simultaneously per iterator with all iterators globally coordinated during the descent, of the tree. Global coordination involves replacing the distance used to determine which nodes will be expanded in each iterator by the largest of these over all iterators (the global distance). Before each descent the nodes that will be considered at the next level are checked to ensure that at least one trajectory is represented by at least one node (or their children) in each iterator. This is achieved by a simple intersect operation across
the pointers held at each node over all iterators. These important enhancements to the individual iterators nearest neighbour search means that the iterators converge to a set of trajectories of which all have points within the defined global minimum distance. In general this is a single (or set in the case of identical global distances) trajectory. It is possible, however, that all input points are each close to points from different historic trajectories and hence the global distance used at that level results in no trajectories being supported at the next level. This is because each iterator acts locally when recursing down the tree at each level, with the global condition checked after descending to the next level. This is shown visually in figure 2.

**Fig. 2.** Example illustrating that multiple historic observations can be returned by the cover tree matching.

It is possible to envisage an algorithm that avoids this. For example, if no trajectories were left in the set after an intersect operation the point that contributed the minimum could be removed from consideration and a new global minimum obtained with no additional distance computations, repeating the process until the set of trajectories left after the intersect operation was non-empty. In practice, however, the set of possible trajectories identified one level in the tree before the intersect set is reduced to zero is typically very small\(^4\) and so in practice, in the implementation used in this paper, the final solution from this point is solved via a brute force calculation.

After the complete nearest neighbour procedure has finished it is possible that multiple predictions are returned, which simply means multiple historic paths matched equally well. If this occurs the probabilities of the prediction sequences conditional on the input sequence is used to distinguish the most likely prediction. This involves the pre-calculation

\(^{4}\) Average 2.3 on a random subset of 102 tests with 58% of the time only one trajectory returned, 93% fewer than 5. Only 2 returned more than 10. The most returned was 23 and the size of the training set was 918 historic trajectories.
of support values for each historic trajectory. These pre-computed values are then used to quickly generate conditional probability scores via lookup. In order to determine the support (done at the same time as the tree construction) the same global matching criteria is used, in the case of the augmented cover tree approaches this is the MMC. In evaluation when comparing to the Expected Minimum Distance criteria, predictors using the EMC also use EMC in determining the support.

4.2 Complexity analysis

The augmented Cover Tree utilizes the same representation as the normal Cover Tree, using multiple iterators. As such the $O(n)$ space complexity is maintained, where $n$ is the number of unique points in all historic trajectories\(^5\). The time complexity bounds (in terms of the number of distance comparisons) is $k$ times the original time complexities, where $k$ is the number of points in the input. This results in $O(ke^{10} \log(n))$, where $c$ is the expansion constant detailed in [5].

5 Feature Set Encodings for Cover Trees

Within augmented cover trees an individual observation can now be encoded as a wide range of feature sets - importantly we can also optionally include time/sequence information alongside spatial dimensions, so long as we supply an appropriate custom distance functions, $\delta(x, y)$. In our experiments we examine four potential encodings, although many more are possible:

1. **XY**: No encoding of time. Only two-dimensional spatial coding is used. The standard $L_2$ distance metric is used.
2. **XY-SEQ**: Strict encoding of sequence. Two-dimensional spatial coding is used, along with a third sequence indicator. A custom distance function is used penalizing the alignments for which the sequence numbers do not match, effectively enforcing sequence. When the sequence numbers match the standard $L_2$ distance is used.
3. **XY-START**: No time encoding, start point emphasised via encoding the change in distance from the start point as a feature. Additionally two-dimensional spatial coding was used. The distance function was the $L_2$ across all three features.
4. **XY-TD** Relaxed encoding of sequence by encoding the total distance travelled in conjunction with a two-dimensional spatial coding. The standard $L_2$ distance metric is used.

The first, XY, represents the most basic case, neglecting time completely. This has the effect that points are allowed to be matched completely out of order. When the EMC is used the second, XY-SEQ, is analogous to the encodings discussed in 1 where the sequential structure is entwined within the algorithms themselves. Differing from previous approaches (in particular the approach from [23] which is later used as a baseline for

\(^5\) Depending on the use of spatial quantization or otherwise the number of unique points may be equal to or less than the number of points in all trajectories.
our experimental evaluation) the distance function is not normalized and the matching function is performed as a completely separate step. The third encoding, XY-START, encapsulates the notion that the start point of a trajectory is discriminative. A motivating example is someone living on a major road joining two major destinations on the other side of a town. If order is unimportant then the trajectory that starts halfway along the road and proceeds along it for a number of samples is quickly matched to trajectories that travel along the whole route to the other side of town, even though in every case the person who lives at this house only drives along the road for ten blocks then turns off at the local supermarket. Therefore the encoding relates to the notion that the start location is significant so trajectories that start in one location should preferentially match other trajectories that start at the same location. The final encoding, XY-TD, relates to a relaxed notion of sequence where the total distance along the trajectory is used as a feature.

The encodings detailed above show a number of different ways to model sequence rather simply assuming strict ordinal accuracy. As is shown in the next section, this modelling extension, when combined with an appropriate trajectory similarity function, can lead to increase prediction accuracy and provide state-of-the-art results.

6 Experimental Evaluation

In order to evaluate the effect of modelling order and the trajectory similarity function eight different predictor variants were evaluated. Within the variants the encodings that strictly ignore (encoding XY) or encode (encoding XY-SEQ) sequence information act as baselines with respect to the importance of sequencing information. In addition one baseline predictor from the literature was selected, the predictor from [23]. While other baselines are possible (such as Dynamic Time Warping), this predictor was chosen for comparison as it represents a prevalent approach within the route prediction literature.

- **MMC**: MMC using a Cover Tree. Four variants. One per encoding: XY, XY-SEQ, XY-START and XY-TD.
- **EMC**: A brute force algorithm implementing EMC for comparison. Four variants. One per encoding: XY, XY-SEQ, XY-START and XY-TD.
- **WN** The predictor from [23]. To be comparable, no spatial quantization was performed and the location points used directly. The matching algorithm parameters (for which no generally optimal values are known) were experimentally varied for the evaluation data set and those achieving the best performance reported here. This potentially inflates the predictor’s results (compared to what would achieved if the parameters were selected based on an independent data set). This is considered acceptable as the method is only used as a baseline. The parameters selected were: temporal weighting = 0.75, spatial weighting = 0.5, temporal threshold = 6 seconds.

The dataset used in the evaluation was the D-SCENT dataset [27]. The D-SCENT dataset was generated from an augmented reality simulation
that was developed as part of the D-SCENT project funded by EPSRC at the University of Nottingham in the UK. While tasks undertaken by participants were artificial, their movement across the real world playing area was completely unconstrained and reflects real spatial behaviour. The dataset was collated over a year, with the simulation featuring 12 locations and covering a 80,000m² spatial area. Sixty participants interacted with the game via G1 smart phones and their (assisted) GPS data was collated every 5 seconds by a central server resulting in over 30,000 position records.

For the experiment presented in this work each individuals path was broken into trajectories, by splitting the GPS stream when a gap of six seconds or more occurred. Trajectories were then only kept if they were at least 20 meters in length and had at least five points. In total 1031 individual trajectories were created with an average length of 128.7 meters. No spatial quantization was performed, however, the longitude and latitude readings were converted into OSGB (Ordnance Survey of Great Britain) grid references using Jcoord for Java⁶.

6.1 Runtime performance results

Analysis of the runtime performance is primarily measured with respect to the number of distance calculations as it provides an insight into the performance of the algorithm abstracted from the specific implementation. In addition the number of intersect operations are reported, although it must be noted that the number of intersect calculations were not optimized⁷ in the implementation and as such represent an upper bound. In general these are expected to be cheaper than the distance calculations. Only the prediction phase is measured. This reflects the assumption that training, in other words building the tree, would be performed offline. As a baseline a brute force approach was implemented using the MMC. In this algorithm each point in the input was compared to each point in each historic path and the maximum, minimum distance for each point in the input used as the distance for the (input, historic) pair.

Performance measurements were taken by withholding a test set of 100 observations and then randomly selecting a training set without replacement of sizes 100 to 900 in increments of 100. For each training set size the resampling was repeated 10 times and the average taken. Finally an overall average for each training set size was taken. It is of note that the number of distance calculations recorded from the augmented cover tree represent the amount when distance caching is used per prediction. The caching is strictly per prediction with all caches being cleared as the first step of each prediction. The baseline brute force algorithm never

⁶ Available from: http://www.jstott.me.uk/jcoord/
⁷ Intersects were calculated at each level, even if there was no change in any cover sets. This did not influence the distance calculation count as distance calculations involved were simply repeats of previously calculated distances and hence were taken from the cache and did not contribute to the count of distance calculations.
repeats a distance calculation per prediction and therefore no cache is implemented.

The results are presented in figure 3. The first of the two graphs shows a comparison of the number of distance calculations (combined distance and intersect calculations in the case of the cover tree) made on average per observation for both the brute force approach and the cover tree, clearly showing the benefit of using the tree based data structure. Zooming in on only the cover tree performance (the second graph) the logarithmic nature can be seen.

Fig. 3. Graphs showing the performance between the brute force and the augmented cover tree approach to prediction under the MMC and the number of distance and intersect operations separately for the cover tree approach.

6.2 Prediction accuracy results

The prediction accuracy is evaluated using 10 × 10-fold cross-validation. A prediction is made by splitting a trajectory from the test set into an input and ground truth segment, making a prediction based on the input segment and comparing the prediction to the ground truth via the average Fréchet distance [2]. For each test set the Fréchet distances are then converted into a proportion based on a user-defined level of an acceptable distance. This is done since simply taking the mean of the distances between each prediction is highly influenced by outliers. In this section the parameter is set to 5 meters corresponding to the level of granularity chosen in [27]. Despite having pre-selected this granularity for further investigation, a range of other granularities were initially examined and a line graph plotting the parameter vs. performance is shown in figure 4 (a). The graph paints a broad picture of each of the predictor’s performance.
A first point of interest is the poor performance exhibited by the baseline predictor, WN. Comparing the baseline to the somewhat similar EMC:XY-SEQ it is of note that the latter shows much better performance. While both assume correct sequence information the latter does not match based on a normalized distance measure and does not integrate frequency counts of the historic path as part of the matching strategy. Rather, the matching is performed first and the the frequency information used if more than one prediction is returned.

The graph also shows the general superiority encodings that relaxes time but include, as an additional feature either the total distance travelled so far or the distance to the start point from that point. This observation does not emerge until the admissible precision is set to three metres or greater, however.

Within the better performing encodings the MMC shows better performance. This applies more generally once the admissible precision is set to 5 metres or greater with all encodings showing either equal or better performance than under the EMC.

The case where the relevance parameter is set to 5 meters is now considered. Figure 4 (b) shows a side-by-side box plot of each fold proportion. From the plot it is clear that statistically significant differences exist, at least between the WN and XY encodings and other encodings. Of additional interest is the comparison of the encoding that respects sequential encoding, EMC:XY-SEQ and MMC:XY-SEQ, and the other methods (excluding WN and XY) that do not, which also show signs of the existence of a significant difference. The plot does not seem to indicate large differences...
differences between the two global distance criteria providing rough conclusions similar to the previous graph.

Paired t-tests are performed using the proportion means from the $10 \times 10$-fold cross validation with the variance estimated conservatively via the approach detailed in [24]. This approach prevents liberal inference from the non-independent nature of the repeated test and training sets in cross validation. The p-values are then adjusted via the Holm procedure. Before performing the statistical tests the distribution of each predictor was checked using normal probability plots, confirming that they satisfied the required assumptions of normality.

Of the 36 comparisons, 25 show significant differences after adjustment. Of the 25, 18 ($p < 0.005$) confirm what was expected from the box plot, that the time relaxed encoding where no other features are used (CT:XY, EMC:XY) and the baseline WN predictors perform significantly worse than all other methods. Of these three predictors none significantly outperform any other.

Considering the trajectory similarity functions, no significant differences were observed when only this function was altered between MMC and EMC. The best performing encoding was XY-TD although no significant difference was observed between it and the XY-START encoding. Compared to the encoding which assumed correct sequence information, XY-SEQ, the XY-TD encoding always performed significantly better ($p < 0.005$) as did the XY-START encoding ($p < 0.05$), except when the EMC was used compared to the XY-SEQ encoding also under the EMC.

A potential explanation for the superior performance of the total distance feature encoding is that, while deviations in accuracy regularly occur in varying magnitudes for individual points, over the full input sequence the total magnitude may be similar. Another potential reason is that the encoding penalizes trajectories within the historic set which have points which are a long way from the overall path created from the trajectory due to the disproportional increase in total length they contribute. Since these points are generally recorded in error, historic trajectories with errors of these kinds are avoided as prediction candidates, potentially improving prediction performance.

Regardless of the exact reason, the result highlights (1) that the proposed approaches perform significantly better than the baseline approach, (2) that under proposed encodings a relaxed approach to modelling sequence leads to improved performance, and (3) that choice of the trajectory similarity function is not so important, although in conjunction with the better performing encodings it can bring performance gains. It is of note that the cover tree framework can be used to model many other feature variants with relaxed notions of sequence, some of which may perform better still.

---

8 In calculating the conservative estimator the parameter $M$ was set to seven.
9 We used the implementation available in the R package muToss [31]
7 Conclusion

In this paper a computationally efficient new approach for prediction was presented based on using multiple iterators over augmented cover trees. Addressing issues arising from the use of noisy, inconsistent data the approach relinquishes the use of sequence as part of the algorithm structure, in contrast to current state-of-the-art proposals. Instead the approach models sequence as a first-order feature as part of a feature vector along side the spatial information. Allowing a wide range of previously unexplored encodings four of these (two acting as baselines) were investigated under two trajectory similarity functions, additionally comparing to a predictor recently proposed in [23].

Between predictors the experiments showed statistically significant improvements when using encodings that provide a relaxed encoding of sequence compared to encodings that assume correct sequence information or conversely assume none at all. In all cases the new encodings showed significant performance increases over all baselines, highlighting not only that the relaxed encoding is important but that the specifics in the predictor, such as keeping the matching as a independent step and the exact distance function used must be carefully considered. Overall the results highlight the superiority of the proposed predictors in the investigated context compared to previous work and the potential of the approach of relaxed sequence encoding in general. Furthermore it must be noted that many other unexplored encoding schemes can be implemented in this framework, providing a platform for interesting future work. With respect to the computational complexity the results inherited from the use of the cover tree data structure provide solid theoretical results with the runtime complexity logarithmic in the number of unique points in all historic trajectories, albeit with a large constant. This logarithmic runtime performance was clearly demonstrated in practice, with significantly less distance calculations required with respect to a brute force approach, even at small numbers of training observations. This result demonstrates the practical applicability of this approach to large data sets.

References


