Pythia: Distributed Pattern-based Future Location Prediction of Moving Objects

Panagiotis Tampakis[1], Nikos Pelekis[2]

[1]Department of Mathematics and Computer Science, University of Southern Denmark, Denmark
[2]Department of Statistics & Insurance Science, University of Piraeus, Greece

Abstract

Predictive analytics over mobility data is a domain that has received a lot of attention by the research community the past few years and encapsulates a wide range of sub-problems aiming to predict e.g. the future location of a moving object, the future trajectory of a moving object, the traffic flow, the expected time of arrival of a moving object to its destination etc.. These are all quite challenging problems from their nature and what makes them even more challenging is the massive production of mobility data, which sets some limitations over training such predictive models. In this paper we propose *Pythia*, a framework able to predict simultaneously, the exact future location of an extremely large set of moving objects, given a look-ahead time, by employing massive historical mobility patterns. In order to achieve this we build a predictor for each moving object, in the form of a directed acyclic graph, by taking into account not only its past movement but also collective historical patterns. Our experimental study shows that our approach can predict accurately the future location of moving objects in an efficient way.

Keywords

predictive analytics, big mobility data, future location prediction

1. Introduction

The propagation of GPS-enabled devices that has been observed during the recent years has led to an unprecedented rate of mobility data generation which in turn has posed new challenges in terms of storage, querying, analytics and knowledge extraction out of such data. The explosion of mobility data generation has led the research community to devise new interesting methods to analyse them. One of these directions is predictive analytics over mobility data, which aims at the utilization of historical data in order to predict future behavioural patterns and trends. An important operation with a wide range of applications, such as collision detection, traffic estimation and service recommendation, is future location prediction (*FLP*). What is even more challenging is how to deal with this problem in the Big Data era, where new positions arrive at frequent rates and the accumulated ones scale to petabytes of data, and in application scenarios where latency and scalability matter.

Inspired by the above, in this paper we propose *Pythia* , a framework able to predict simultaneously, in real-time, the exact future location of an extremely large set of moving objects, given a look-ahead time, by employing historical mobility patterns. In order to achieve this we adopt a *hybrid* approach, where we build a predictor for each moving object by taking into account not only its *individual* past movement but also *collective* historical patterns. By doing so, we increase the predictive ability of our system, meaning that the amount of cases where our system can make a prediction is significantly increased, as compared to only using the *individual* history of each moving object. Furthermore, the accuracy of our systems predictions is increased, as compared to only using *collective* historical patterns, since individual moving objects tend to follow the same routes.

Clearly, solving the above problem is quite challenging, since one has to take into account not only the inherent complexity of the *FLP* problem but also the challenges posed by the Big Data era, in terms of volume and velocity of the incoming data. In more detail, the problem can be broken down in an *Offline* and a *Query* module. The *Offline* part is responsible for identifying patterns of movement, either *individual* or *collective*, while the *Query* part is responsible for predicting the future location of a moving object, given a look-ahead time and the set of patterns that were identified during the *Offline* module.

Several efforts try to deal with this problem by applying spatial discretization and generalization and then try to find frequent locations or sequences of locations ([1, 2, 3, 4, 5, 6]). However, such approaches provide generalized and thus inaccurate predictions. Moreover, a large number of approaches do not take into account the temporal information during the mobility behavior extraction and/or during the prediction ([7, 3, 8, 6, 1, 9, 10]). Another line of research, that takes into account both time and the exact location of the moving objects, includes efforts that try to deal with this problem by grouping entire trajectories, identifying patterns of movement and then using them to predict the future location ([11, 2, 12, 4]). However, identifying patterns that are valid for the entire lifespan of the moving objects can overlook significant patterns that might exist only for some portions of their lifespan.

Furthermore, all of the above approaches are centralized and do not scale with the size of today's trajectory data, which calls for parallel and distributed algorithms in order to address the *FLP* problem in a scalable and efficient way. Towards this direction, [13] utilize the work done by [14] on distributed subtrajectory clustering in order to be able to extract individual subtrajectory patterns from big mobility data. These patterns are subsequently utilized in order to predict the future location of the moving objects in parallel. Despite the fact that this solution takes advantage of subtrajectory patterns and is Big Data compliant, it suffers from the fact that it takes account only *individual* patterns, thus decreasing the system's predictive ability. Furthermore, due to the fact that subtrajectory patterns are patterns that are valid for smaller portions of the trajectories lifespan, this might lead to stumbling into "dead ends" (i.e. reaching the end of a pattern and not having the ability to predict further ahead in time), which, in turn, would lead to decreased look-

Published in the Proceedings of the Workshops of the EDBT/ICDT 2024 Joint Conference (March 25-28, 2024), Paestum, Italy
ptampakis@imada.sdu.dk (P. Tampakis); npelekis@unipi.gr (N. Pelekis)
0000-0003-1274-3306 (P. Tampakis); 0000-0001-7205-5703 (N. Pelekis)

ahead prediction ability. In turn, this might lead to either decreased predictive ability or decreased accuracy, depending on whether the system provides a prediction ([11] return the last point of the pattern as the predicted point if the look-ahead time exceeds the lifespan of the pattern) or not ([13] does not provide a prediction if the look-ahead time exceeds the lifespan of the pattern).

Our main contributions are the following:

- We propose an efficient and scalable tailor-made prediction-oriented subtrajectory pattern extraction technique.
- We propose a distributed pattern network reconstruction technique. These networks have the potential to serve as predictors for other predictive analytics, such as trajectory prediction and ETA.
- We perform an experimental study, where the effectiveness of the proposed algorithms is evaluated.

The rest of the paper is organized as follows. In Section 2 we provide an overview of the relevant literature. Subsequently, in Section 3 we introduce some preliminaries and provide the problem definition. In Section 4 we present our proposed solution to the problem of *distributed pattern-based future location prediction* and in Section 5, we present the results of our experimental study. Finally, in Section 6 we conclude the paper and discuss about the future work.

2. Related Work

The fact that the *FLP* problem has been thoroughly studied brings up its importance and applicability in a wide range of applications. There are several studies in the past that dealt with the *FLP* problem that can be categorized in some groups. There is a group of studies that adopt a time series analyses approach [15] and [16]. However, this approaches are individual and perform spatial discretization which can affect the accuracy of the system. In another line of research, probabilistic models are employed and more specifically Hidden Markov Models (HMM). The general approach includes discretization of the spatial domain in order to identify sequences of "important" locations [17, 18]. Nevertheless, both of these approaches are collective-only, do not take into account the temporal information and perform some kind of spatial generalization, which can affect the accuracy of the system. Some studies use methods like HMM [10], Continuous-Time Markov Processes [3] or Dynamic Bayesian Networks models [9]. However, all of the above approaches are individual-only, do not take into account the temporal information and perform some kind of spatial generalization, which can lead to prediction with decreased accuracy.

A different way of addressing the *FLP* problem includes machine learning approaches. [19] model the trajectory of sea vessels and provide a service that predicts in near-real time the position of any given vessel by employing multilayer perceptrons (MLPs), while in [20, 21, 22], the authors employ LSTM networks to predict the future location of vessels. [23] introduce a machine-learning model which exploits geospatial time-series surveillance data generated by sea-vessels, to predict future trajectories based on real-time criteria. Nevertheless, most of the machine learning approaches use collective-only patterns and are not "cost" efficient due to the "expensive" training step. Furthermore, explainability of the results is always an issue, especially in neural network based approaches. [7]

Another interesting viewpoint is to take advantage of historical movement patterns in order to predict the future location. There are several approaches that use either collective-only patterns [6, 5, 1, 24] or individual-only patterns [4, 25]. As already discussed this might affect either the predictive ability of the model or the accuracy pf the predictions. [11] propose MyWay, a hybrid, pattern-based approach that utilizes individual patterns when available, and when not, collective ones, in order to provide more accurate predictions and increase the predictive ability of the system. In more detail, MyWay clusters trajectories of individuals in order to identify patterns that are valid for the entire lifespan of each individual object. Moreover, the same procedure is repeated for the entire set of trajectories of all moving objects, in order to identify collective patterns. Finally, when available, the individual patterns are employed, otherwise the collective patterns are utilized in order to perform the future location prediction. Similarly, [26] a mobility graph that depicts the most likely movements among two ports. Nonetheless, as already mentioned, identifying patterns that are valid for the entire lifespan of the moving objects can overlook significant patterns that might exist only for some portions of their lifespan.

To deal with this, there is a group of clustering methods [27, 28, 29] that aim to segment trajectories to subtrajectories, based on some criteria, and then discover clusters of subtrajectories. However, all of the above approaches are centralized and do not conform with the requirements of the Big Data era in terms of speed and velocity, which, in turn, calls for parallel and distributed algorithms in order to address the *FLP* problem in a scalable and efficient way. Towards this direction, [30, 13] utilize the work done by [14] on distributed subtrajectory clustering in order to be able to extract individual subtrajectory patterns from big mobility data. These patterns are subsequently utilized in order to predict the future location of the moving objects in parallel. Despite the fact that these solutions take advantage of subtrajectory patterns and are Big Data compliant, they suffer from the fact that they take into account only *individual* or *collective* patterns, respectively, thus decreasing the system's predictive ability and accuracy. Furthermore, due to the fact that subtrajectory patterns are patterns that are valid for smaller portions of the trajectories lifespan, this might lead to decreased look-ahead prediction ability.

3. Problem Definition

3.1. Preliminaries

In this section we are going to provide some preliminary definitions. In more detail, given a set D of trajectories moving in the xy-plane,

Definition 1. ***(Trajectory)*** *A trajectory $r \in D$ is a sequence of timestamped locations $\{r_1, \ldots, r_N\}$. Each $r_i = (x_i, y_i, t_i)$ represents the i-th sampled point, $i \in 1, \ldots, N$ of trajectory r, where N denotes the length of r (i.e. the number of points it consists of). The pair (x_i, y_i) and t_i denote the 2D location in the xy-plane and the time coordinate of point r_i respectively.*

Definition 2. *(**Subtrajectory**) A subtrajectory $r_{i,j} \in r \in D$ is a subsequence $\{r_i, \ldots, r_j\}$ of r which represents the movement of the object between t_i and t_j where $i < j$.*

Definition 3. *(**Distance**) Further, let $d_s(r_i, s_j)$ denote the spatial distance between two points r_i, s_j, which is defined as the Euclidean distance in this paper, even though other distance functions are also applicable. Also, let $d_t(r_i, s_j)$ denote the temporal distance, defined as $|r_i.t - s_j.t|$.*

Regarding the similarity between (sub)trajectories, in this paper we chose to employ the Longest Common Subsequence (LCSS) for trajectories, as defined in [14], which holds several desirable properties, such as the support of trajectories with variable sampling rate, variable length and lack of alignment and the fact that it allows trajectories that have some temporal displacement to be considered as "similar". Moreover, this similarity measure is symmetric and computationally efficient. More specifically, the LCSS utilizes two parameters, the parameter ϵ_t indicating the temporal range wherein the method searches to match a specific point, and the ϵ_{sp} parameter which is a distance threshold to indicate whether two points match or not.

Definition 4. *(**Similarity**) Hence, the similarity between two (sub)trajectories r and s is defined as:*

$$Sim(r,s) = \frac{\sum_{k=1}^{min(|r|,|s|)} (1 - \frac{d_s(r_k, s_k)}{\epsilon_{sp}})}{min(|r|, |s|)} \quad (1)$$

where (r_k, s_k) is a pair of matched points and $|r|$ ($|s|$) is the length of r (s respectively). Moreover, it holds that $Sim(r,s) = Sim(s,r)$.

3.2. Properties of Subtrajectory Pattern Extraction

The goal of this module is to identify frequent patterns of movement that will maximize the prediction accuracy and the predictive ability of our system. The desired properties that a prediction-oriented (sub)trajectory pattern extraction algorithm should hold are the following:

Discovering patterns of subtrajectories. As argued in Section 1, this will enable us to identify more patterns, which would be otherwise "lost".

Spatiotemporal pattern extraction. It is of great importance for the task of prediction to discover patterns that take into account time, hence implicitly the speed.

Clusters with small spatial extent. In order for the predictions to be more accurate, the identified patterns should represent groups of objects that are not spatially dispersed. Obviously, this, rules out a large number of approaches that perform density-based clustering which might lead to spatially extended clusters through expansion.

Coverage. In order to increase the predictive ability, the identified patterns need not to be concentrated in a specific region, but to cover the datasets extent as much as possible.

"Dead ends" minimization. As already mentioned in Section 1, due to the fact that subtrajectory patterns are patterns that are valid for smaller portions of the trajectories lifespan, this might lead to stumbling into "dead ends", which leads to decreased look-ahead prediction ability.

Distributed. In order for this solution to be able to extract patterns from big trajectory data, parallel and distributed algorithms need to be utilized.

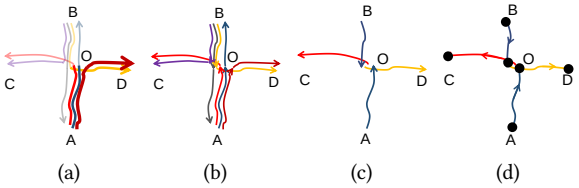


Figure 1: Example of (a) Subtrajectory Join, (b) Trajectory Segmentation, (c) Subtrajectory Pattern Extraction and (d) Pattern Network Reconstruction.

3.3. Distributed Subtrajectory Pattern Network Extraction (D-SPaNE)

Our approach towards building a framework that meets with the aforementioned specifications can be split into four sub-problems. To help us, let us consider the following example.

Example 1. *Figure 1 illustrates six trajectories moving in the xy-plane, where each one of them has a different origin-destination pair. More specifically, these pairs are $A \rightarrow B$, $A \rightarrow C$, $A \rightarrow D$, $B \rightarrow A$, $B \rightarrow C$ and $B \rightarrow D$. The goal of a subtrajectory pattern extraction method is to identify 4 patterns ($A \rightarrow O$, $B \rightarrow O$, $O \rightarrow C$, $O \rightarrow D$), as depicted in Figures 1(c).*

Problem 1. *(**Subtrajectory Join**) The first step is the so called* subtrajectory join*, a well-defined problem in the literature of mobility data management, where the goal is to retrieve for each trajectory $r \in D$, all the moving objects, with their respective portion of movement, that moved close enough in space and time with r, for at least some time duration [31]. Actually, the subtrajectory join will return for each pair of (sub)trajectories, all the candidate longest common subsequences. Figure 1(a) illustrates the* subtrajectory join *result for trajectory $A \rightarrow D$ of the Example 1.*

Problem 2. *(**Trajectory Segmentation**) The second step takes as input the result of the first step, which is actually a trajectory r and its neighboring trajectories and aims at segmenting each $r \in D$ into a set of subtrajectories. The way that a trajectory is segmented into subtrajectories is neighbourhood-aware, meaning that a trajectory will be segmented every time the density of its neighbourhood changes significantly. In this paper, we are going to adopt the approach followed in [14]. Returning to Example 1, trajectory $A \rightarrow D$ should be segmented to $A \rightarrow O$ and $O \rightarrow D$, since at O the density of its neighbourhood changes significantly, as illustrated in Figure 1(b).*

Problem 3. *(**Subtrajectory Pattern Extraction**) Given the output of Problem 1, applying a trajectory segmentation algorithm for the trajectories D will result in a new set of subtrajectories D'. The third step takes as input D' and the goal is to identify a set of "representative" subtrajectories (patterns), whose cardinality is unknown. More specifically, let $R = \{R_1, \ldots, R_K\}$ denote the set of "representatives", where K is the number of the identified patterns. Actually, the problem can be viewed as a subtrajectory sampling problem, where the goal is, given D', to select the most "representative" subset R of subtrajectories, in terms not only of the number of subtrajectories that are represented but also to cover the entire extent of D' as much as possible. Towards this direction, we employ the work presented in [14].*

Here, (a) actually guarantees the large coverage of R by allowing a subtrajectory to be added in R iff it is dissimilar with the already existing members of R and (b) ensures that D' is maximally represented given the restrictions imposed by (a). Figure 1(c) depicts the four identified patterns, namely $A \rightarrow O$, $B \rightarrow O$, $O \rightarrow C$ and $O \rightarrow D$, of Example 1.

Finally, the fourth step takes as input the set of identified patterns R and aims at constructing a spatiotemporal directed graph $G = (V, E)$, where V is a set of vertices and E is a set of edges, that represents a network of movement through which a moving object can be routed. At this point, we should mention that V is a set of spatiotemporal points and E is a set of subtrajectories, as defined in Definition 2. By this, we can minimize the number of "dead ends" and increase the look-ahead prediction ability.

Due to the hybrid nature of out system, there is the possibility that an individual pattern might be identical with a collective pattern. Furthermore, a pattern might be the continuation of another pattern, irrespective of whether they are individual or collective. Hence, an algorithm that takes as input a set of patterns and tries to construct a directed graph, would have to perform a series of "merge" and "append" operations. More specifically, a "merge" operation between two patterns r' and s' takes place iff r' is an individual pattern and s' a collective one (or vice-versa) and $Sim(r', s') > \alpha$, where α is a similarity threshold. Moreover, an "append" operation two patterns r' and s' takes place iff $0 \leq s'_1.t - r'_N.t \leq \epsilon_t \wedge dist_s(s'_1, r'_N) \leq \epsilon_{sp}$ or vice-versa, where s'_1 is the first point of s' and r'_N is the last point of r'. The spatiotemporal gap that might arise between r'_N and s'_1 gets filled by the shortest path between these two points when the underlying network of movement is known or by linearly interpolating in any other case. This can be considered as the "connection edge" between the two patterns. Finally, G can be constructed by considering as vertices all the initial and ending points of the patterns and as edges all the resulting patterns along with the connection edges. Figure 1(d) illustrates the reconstructed pattern network, given the patterns of Figure 1(c).

Problem 4. ***(Pattern Network Reconstruction)*** *Given a set of patterns R, a spatial threshold ϵ_{sp}, a temporal tolerance ϵ_t and a similarity threshold α, construct a spatiotemporal directed graph $G = (V, E)$ after performing all the appropriate "merge" and "append" operations.*

In this paper, we address the challenging problem of prediction-oriented subtrajectory pattern network extraction in a distributed setting, where the dataset D is distributed across different nodes, and centralized processing is prohibitively expensive.

Problem 5. ***(Distributed Subtrajectory Pattern Network Extraction)*** *Given a distributed set of trajectories, $D = \cup_{i=1}^{P} D_i$, where P is the number of partitions of D, perform the prediction-oriented subtrajectory pattern extraction task in a parallel manner.*

Actually, Problem 5 can be broken down to solving Problems 1, 2, 3 and 4 (in that order) in a parallel/distributed way. Finally, the Future Location Prediction problem can be defined as follows:

Definition 5. ***(Future Location Prediction)*** *Given a desired look-ahead time t^{pred}, a pattern network G and the recent k positions $\{r_{N-k+1}, \ldots, r_N\}$ of moving object r, where, r_N is the latest reported position, predict the position of r at $r_N.t + t^{pred}$.*

4. The *Pythia* Framework

4.1. Overview

The *Pythia* framework consists of the *distributed subtrajectory pattern network extraction* (*D-SPaNE*) and the *Prediction* component, as illustrated in Figure 2. The *D-SPaNE* component is responsible for extracting the subtrajectory pattern networks in a distributed manner, given a large set of accumulated historical data. The *Prediction* component receives the k-recent positions of a trajectory and the look-ahead time for each moving object, it retrieves its corresponding *hybrid* pattern network, "matches" its recent history on the network and routes through it until it finds the future location of the moving object at the desired look-ahead time t^{pred}.

4.2. D-SPaNE Component

Concerning the *D-SPaNE* component, it consists of a distributed storage file system, such as *HDFS*, which contains accumulated historical mobility data and *D-SPaNE* itself, which takes as input a distributed trajectory dataset from the distributed file system and constructs a set of *hybrid* subtrajectory pattern networks $SPN = \{SPN_1, \ldots, SPN_N\}$, where N is the number of moving objects. As already mentioned, the term *hybrid* indicates that we build a predictor for each moving object by taking into account both its individual past movement and collective historical patterns.

Algorithm 1 $D - SPaNE(D)$

1: **Input:** D
2: **Output:** set SPN of subtrajectory pattern networks
3: **Preprocessing:** *Align* and *Repartition D*;
4: **for each** partition $D_i \in \cup_{i=1}^{P} D_i$ **do**
5: **perform** *Point-level Join*;
6: **group by** *Trajectory*;
7: **for each** Trajectory $r \in D$ **do**
8: **perform** *Subtrajectory Join*;
9: **perform** *Trajectory Segmentation*;
10: **group by** D_i;
11: **perform** *Pattern Extraction* $\forall$ D_i;
12: **perform** *Refine Results*;
13: **group by** *Trajectory*;
14: **for each** Trajectory $r \in D$ **do**
15: **perform** *Network Reconstruction*
16: **return** SPN;

An overview of *D-SPaNE* is presented in Algorithm 1. In more detail, initially, we temporally *Align* the first points of each trajectory starting at $t = 0$, in such a way that the temporal dimension depicts the duration since the start of a trajectory, and then *Repartition* the data into P equi-sized, temporally-sorted temporal partitions (files), which are going to be used as input for the join algorithm in order to perform the subtrajectory join in a distributed way (line 3). Note that this is actually a preprocessing step that only needs to take place once for each dataset D. However, it is essential as it enables load balancing, by addressing the issue of temporal skewness in the input data. Subsequently, for each partition $D_i \in \cup_{i=1}^{P} D_i$ and for each trajectory we

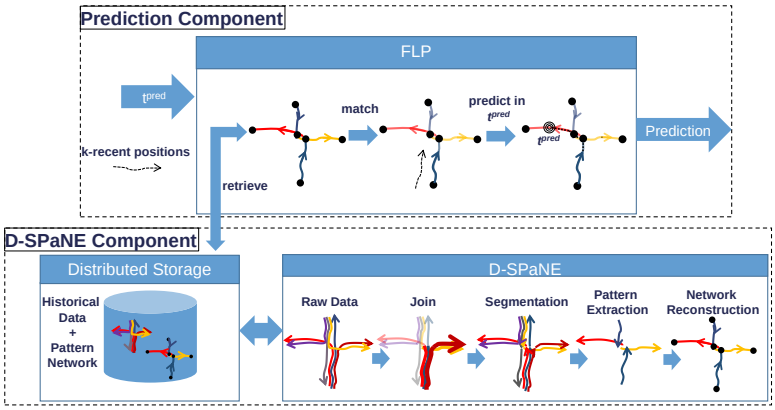


Figure 2: The *Pythia* Architecture.

discover parts of other trajectories that moved close enough in space and time (line 5). Successively, we group by trajectory in order to perform the subtrajectory join (line 8). At this phase, since our data is already grouped by trajectory, we also perform trajectory segmentation in order to split each trajectory to subtrajectories (line 9). In turn, we utilize the temporal partitions created during the *Repartition* phase and re-group the data by temporal partition. For each $D_i \in \cup_{i=1}^{P} D_i$ we perform the pattern extraction procedure (line 11). At this point we should mention that if a subtrajectory intersects the borders of multiple partitions, then it is replicated in all of them. This will result in having duplicate and possibly contradicting results. For this reason, as a final step, we treat this case by utilizing the *Refine Results* procedure (line 12). We should also mention that lines 5-12, will be executed twice, once for identifying *collective* and once for *individual* patterns. The actual difference between the two executions lies at the *Point-level Join* (line 5), where during the extraction of *collective* patterns, we discover for each trajectory parts of other trajectories that moved close enough in space and time that belong to different moving objects, while during the extraction of *individual* patterns, they need to belong to the same moving object. Finally, we perform the *Network Reconstruction* (line 15) procedure by grouping the identified *individual* patterns by trajectory, hence each processing node receives one such group, while we distribute the global patterns to all of the processing nodes. Finally, the set SPN is emitted.

Clearly, tackling the above problem is quite challenging in a distributed setting. For this reason, we outline a solution that follows the popular MapReduce paradigm. Figure 3 illustrates the *D-SPaNE* algorithm in terms of MapReduce. Up to the *Refine Results* step we employ the distributed solutions presented in [31] and [14]. The only difference for the *Pattern Extraction* step is that we do not have to calculate the similarity between a representative r' and all the other non-representative subtrajectories. As for the *Refine Results* step, apart from the possible duplicates that we might get, due to the fact that each subtrajectory that temporally intersects multiple partitions is replicated to each one of them, there might exist non-intersecting patterns from different partitions that represent the same pattern of movement. This is due to the temporal tolerance parameter of ϵ_t of the similarity function presented in [14]. For this reason, apart from the duplicate elimination problem that needs to be addressed here, we also need to investigate whether non-intersecting patterns, whose temporal distance from the borders of a temporal partition is less than ϵ_t (also referred to as intersecting from now on), are significantly similar with intersecting patterns coming from other partitions. Among those similar patterns, the ones that are selected to be removed are the ones that have less support.

Regarding the *Network Reconstruction*, the goal is to take as input the set R of *individual* and *collective* patterns, identified by the previous step, and construct a set of directed graphs $SPN = \{SPN_1, \ldots SPN_N\}$, where N is the number of moving objects and $SPN_i = (V_i, E_i)$ corresponds to the graph of the i-th moving object. As already mentioned in Section 3, the network reconstruction algorithm, will have to perform, a series of "merge" and "append" operations, as depicted in Figure 4 and at the same time calculate the "weight" of each edge E_i and reconstruct the network. Algorithm 2 describes the network reconstruction procedure.

Algorithm 2 $ReconstructNetwork(R)$

```
1:  Input: R
2:  Output: set SPN of networks
3:  for each moving object i do
4:    for each pattern r ∈ R_i do
5:      CalcEdgeWeight(r);
6:      A = A + r;
7:      for each element s ∈ A do
8:        if r.t^start − ε_t > s.t^end then
9:          SPN_i ← s;
10:         A = A − s;
11:       else
12:         if r is individual ∧ s is collective then
13:           if Sim(r, s) ≥ α then
14:             A = A − s;
15:         else if s is individual ∧ r is collective then
16:           if Sim(r, s) ≥ α then
17:             A = A − r;
18:         if r.t^start > s.t^end then
19:           if d_s(r.p^start, s.p^end) ≤ ε_sp then
20:             SPN_i ← {s.p^end, r.p^start});
21:   SPN_i ← A;
22: return SPN;
```

In order to achieve this, for each pattern r, we calculate its "weight" and add it in a new set A (lines 4-6). The "weight" of each pattern is calculated by taking into account both the voting $V(r)$ of each pattern (i.e. support) r and whether r is an *individual* pattern or not. In more detail,

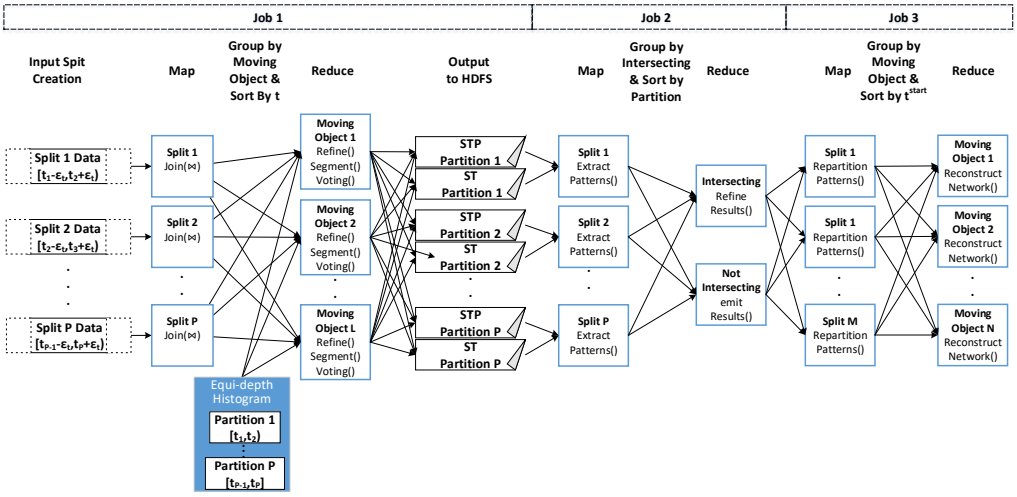
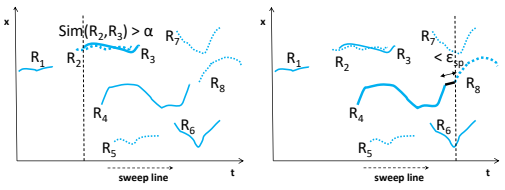


Figure 3: The *D-SPaNE* algorithm. (Job 1) *Subtrajectory Join* and *Trajectory Segmentation*, (Job 2) *Pattern Extraction* and *Refine Results* and (Job 3) *Network Reconstruction*.

$$w(r) = 1 - (\alpha * isIndividual + (1 - \alpha) * V(r)) \quad (2)$$

where $isIndividual$ is a boolean that equals to 1 if r is an *individual* pattern and 0 otherwise and $V(r)$ is the normalized voting (support). In this way, edges that correspond to *individual* and/or highly "voted" patterns take values "closer" to zero and thus become more likely to be traversed by a routing algorithm. We should remind here that the patterns arrive in each reducer sorted by t^{start}, as illustrated in Figure 3. Subsequently, for each other pattern s in A, we examine whether $r.t^{start} - \epsilon_t > s.t^{end}$. If this is the case, it means that the end of s is so far in the past that is impossible for s to participate in any future "merge" or "append" operations, as depicted in Figure 4(a). For this reason, we add s to the final SPN_i set and remove it from A (lines 7-10). Otherwise, we examine if r is an *individual* and s a *collective* pattern or vice versa. If so, we calculate their similarity and if $Sim(r, s) \geq \alpha$ then we merge the two patterns by keeping in A only the *individual* one (lines 12-17), as delineated in Figure 4(b). Further, we investigate if $r.t^{start} > s.t^{end}$. In case this is true, this indicates that an "append" operation might take place, hence we further examine if $d_s(r.p^{start}, s.p^{end}) \leq \epsilon_{sp}$ and if this is also true then perform the "append" operation by adding the "connection edge" $\{s.p^{end}, r.p^{start}\}$ into SPN_i (lines 18-20), as illustrated in Figure 4(c).When, all patterns are traversed, then A gets flushed to SPN_i and finally, after all moving objects get processed SPN is returned (lines 21- 22).



(a) (b)

Figure 4: Example of 8 patterns, 4 *individual* and 4 *collective* (dotted), illustrating the cases of (a) "merge" and (b) "append".

4.3. Prediction Component

Regarding the *Prediction* component, it receives the k most recent positions of moving objects and the goal is to predict its future location at the given look-ahead time t^{pred}. This is done in a straightforward way, as depicted in Figure 2, by retrieving the corresponding subtrajectory pattern network, matching its k-most recent positions on the network and predict its future location at the given look-ahead time t^{pred}. In case there are multiple candidate positions to be returned, we retrieve the one that there is actually a path between the current position and the candidate destination on the network. Finally, in case we have multiple candidate destinations where a path exists, we select the one with the shortest path, according to the edge weights (Equation 2) that we have assigned.

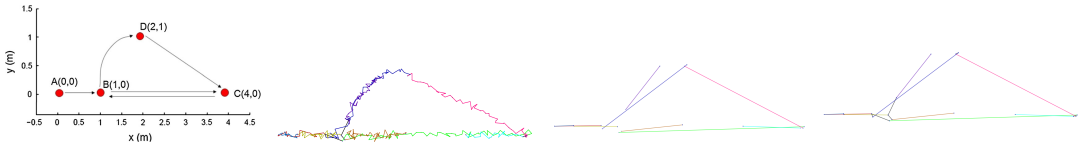
5. Experimental Evaluation

In this section, we present some preliminary findings. The experiments were conducted in a 49 node Hadoop 2.7.2 cluster. The master node consists of 8 CPU cores, 8 GB of RAM and 60 GB of HDD while each slave node is comprised of 4 CPU cores, 4 GB of RAM and 60 GB of HDD.

For our experimental study, we employed a real and a synthetic dataset. Concerning the real dataset, we used a dataset from the urban domain of 1 week duration , called VFI[1], consisting of 25019834 records. Furthermore, we utilized a synthetic dataset in order to verify that our solution operates as anticipated, given a dataset with a known ground truth.

In more detail, the synthetic dataset (SMOD) consists of an object which has performed 400 trips (trajectories) and is used for the ground truth verification. The scenario of the synthetic dataset is the following: the object moves upon a simple graph that consists of the following destination nodes (points) with coordinates: A(0,0), B(1,0), C(4,0) and D(2,1), illustrated in Figure 5(a). Half of the times the object moves with normal speed (2 units per second) and another half moves with high speed (5 units per second). When an object arrives at a node, it ends its trajectory with a

[1]This private dataset was kindly provided by Vodafone Innovus



(a) (b) (c) (d)

Figure 5: (a) The 2-D map of SMOD, (b) Discovered patterns, (c) Network edges and (d) Reconstructed network

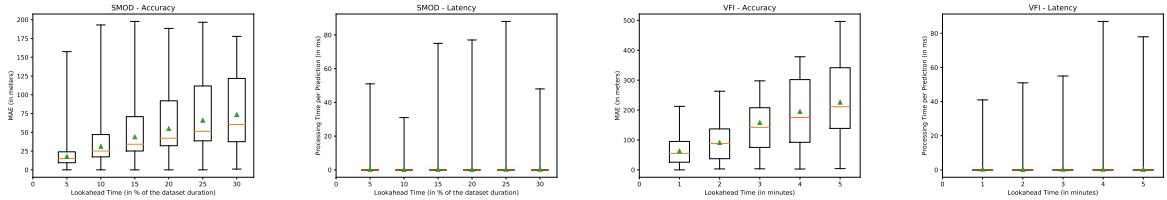


Figure 6: Prediction Accuracy of (a) SMOD and (c) VFI and Latency of (b) SMOD and (d) VFI

probability of 15%. Finally, there is 1% of the trajectories that follow a random movement in space (other than these roads) with a speed that is updated randomly.

Our experimental methodology is as follows: Initially, we verify that our solution operates as anticipated by applying it to a dataset with a known ground truth. Subsequently, we evaluate the quality of the predictions by varying the look-ahead time and measuring the accuracy of the predictions in terms of MAE. Finally, we measure the performance by varying the look-ahead time and measuring the latency.

In order to verify the correctness of our solution we utilized the aforementioned SMOD dataset. To begin with, we are going to verify that the pattern extraction component of our solution operates as anticipated. In more detail, the ground truth of the patterns that are hidden in SMOD can be inferred by the description of the dataset itself. In particular, eight clusters of subtrajectories need to be identified. The following Table lists the eight clusters along with their spatial (2nd column) and temporal projection (3rd column).

Cluster	Path	Time periods (clusters)
$\#1, \#2$	$A \rightarrow B$	[0, 0.2], [0.2, 0.7]
$\#3, \#4$	$B \rightarrow C$	[0.2, 0.8], [0.7, 1.2]
$\#5, \#6$	$B \rightarrow D$	[0.2, 0.52], [0.7, 1.2]
$\#7$	$C \rightarrow B$	[0.8, 1]
$\#8$	$D \rightarrow C$	[0.52, 1]

Indeed, the pattern extraction module discovers these eight clusters, as illustrated in Figure 5(b). The next step is the network reconstruction component, where the goal is to construct a directed graph G from the subtrajectory patterns. The challenge here is to restore the connectivity of movement of the specific objects by applying "stitches" when appropriate. In more detail, Figure 5(c) illustrates the edges (each pattern is an edge) of the graph before the graph reconstruction procedure and Figure 5(d) depicts the constructed graph, where the applied "stitches" are in black colour.

Quality of the predictions. At this point we have verified that the offline component functions as expected. Regarding the accuracy of the predictions over the synthetic dataset we performed a set of experiments where we vary the look-ahead time and measure the Mean Average Error (MAE) of the predictions. In more detail, Figure 6(a) shows that the accuracy achieved is high, considering that the dataset diameter is approximately 500 meters and the look-ahead time ranges from 5%-30% of the dataset duration (each trajectory 'lives" for 100 seconds). Furthermore, as anticipated the larger the look-ahead time, the larger the MAE between the predictions and the actual positions. Regarding the VFI dataset, as depicted in Figure 6(c), the findings regarding the behaviour of MAE w.r.t. the look-ahead time are equivalent, the larger the MAE between the predictions and the actual positions.

Performance. Finally, we investigate the performance of our solution in terms of latency (i.e. how much time it takes to make a prediction). Initially, we utilized the SMOD dataset and measured the latency of our solution. Figure 6(b) presents our findings, where we can observe that our system, for the majority of the cases, can make a prediction at about 1 millisecond. In addition to that, we can see that the look-ahead time does not affect the processing time per prediction. We run the same set of experiments, but this time for the VFI dataset. As expected, the findings are again similar, which demonstrates the capability of our solution to handle large volumes of data in timely fashion. Figure 6(d) presents our findings, where we can observe that our system, for the majority of the cases, can make a prediction at about 1 millisecond.

6. Conclusions and Future Work

In this paper, we presented *Pythia*, a big data framework able to predict the exact future location of an extremely large set of moving objects, given a look-ahead time, by employing massive historical mobility patterns. Our preliminary results demonstrate the potential of our proposal. As for future work, we aim perform a more in-depth experimental study, utilizing more datasets and comparing with the state of the art. Furthermore we plan to evaluate the performance of the *D-SPaNE* component, even though we anticipate to follow the performance of the work presented in [31] and [14]. We also aim to consider the same problem in an online scenario where streams of data arrive at high rate. Another interesting direction, that stems from the above, is to investigate how the proposed networks can be updated/maintained when new data arrive. Finally, as already mentioned, this framework shows potential for other

predictive analytics also, like trajectory prediction and ETA. So, we would like to investigate this possibility of extending this framework to a wide range of predictive analytics.

Acknowledgments

This work was supported in part by the Horizon Europe research and innovation programme under grant agreements 101070416 (Green.Dat.AI) and 101093051 (EMERALDS) funded by the European Union.

References

[1] A. Monreale, F. Pinelli, R. Trasarti, F. Giannotti, Wherenext: a location predictor on trajectory pattern mining, in: ACM SIGKDD, 2009, pp. 637–646.

[2] P. Lei, T. Shen, W. Peng, I. Su, Exploring spatial-temporal trajectory model for location prediction, in: IEEE MDM, 2011, pp. 58–67.

[3] G. Gidófalvi, F. Dong, When and where next: individual mobility prediction, in: ACM SIGSPATIAL MobiGIS, 2012, pp. 57–64.

[4] H. Jeung, Q. Liu, H. T. Shen, X. Zhou, A hybrid prediction model for moving objects, in: ICDE, IEEE, 2008, pp. 70–79.

[5] G. Yavas, D. Katsaros, Ö. Ulusoy, Y. Manolopoulos, A data mining approach for location prediction in mobile environments, Data Knowl. Eng. 54 (2005) 121–146.

[6] E. H. Lu, V. S. Tseng, P. S. Yu, Mining cluster-based temporal mobile sequential patterns in location-based service environments, IEEE Trans. Knowl. Data Eng. 23 (2011) 914–927.

[7] T. Anagnostopoulos, C. Anagnostopoulos, S. Hadjiefthymiades, Mobility prediction based on machine learning, in: IEEE MDM, 2011, pp. 27–30.

[8] Y. Jia, Y. Wang, X. Jin, X. Cheng, Location prediction: A temporal-spatial bayesian model, ACM TIST 7 (2016) 31:1–31:25.

[9] M. Nishino, Y. Nakamura, T. Yagi, S. Muto, M. Abe, A location predictor based on dependencies between multiple lifelog data, in: LBSN, 2010, pp. 11–17.

[10] D. Qiu, P. Papotti, L. Blanco, Future locations prediction with uncertain data, in: ECML PKDD, 2013, pp. 417–432.

[11] R. Trasarti, R. Guidotti, A. Monreale, F. Giannotti, Myway: Location prediction via mobility profiling, Inf. Syst. 64 (2017) 350–367.

[12] H. Georgiou, N. Pelekis, S. Sideridis, D. Scarlatti, Y. Theodoridis, Semantic-aware aircraft trajectory prediction using flight plans, International Journal of Data Science and Analytics (2019) 1–14.

[13] P. Petrou, P. Tampakis, H. V. Georgiou, N. Pelekis, Y. Theodoridis, Online long-term trajectory prediction based on mined route patterns, in: MASTER@ECML-PKDD, 2019, pp. 34–49.

[14] P. Tampakis, N. Pelekis, C. Doulkeridis, Y. Theodoridis, Scalable distributed subtrajectory clustering, in: IEEE (Big Data), 2019, pp. 950–959.

[15] S. Scellato, M. Musolesi, C. Mascolo, V. Latora, A. T. Campbell, Nextplace: A spatio-temporal prediction framework for pervasive systems, in: Pervasive, 2011, pp. 152–169.

[16] M. Ceci, A. Appice, D. Malerba, Time-slice density estimation for semantic-based tourist destination suggestion, in: ECAI, 2010, pp. 1107–1108.

[17] S. Ayhan, H. Samet, Aircraft trajectory prediction made easy with predictive analytics, in: ACM SIGKDD, 2016, pp. 21–30.

[18] S. Ayhan, H. Samet, Time series clustering of weather observations in predicting climb phase of aircraft trajectories, in: IWCTS@SIGSPATIAL, 2016, pp. 25–30.

[19] A. Valsamis, K. Tserpes, D. Zissis, D. Anagnostopoulos, T. A. Varvarigou, Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction, J. Syst. Softw. 127 (2017) 249–257.

[20] P. Tampakis, E. Chondrodima, A. Tritsarolis, A. Pikrakis, Y. Theodoridis, K. Pristouris, H. Nakos, P. Kalampokis, T. Dalamagas, i4sea: a big data platform for sea area monitoring and analysis of fishing vessels activity, Geo spatial Inf. Sci. 25 (2022) 132–154.

[21] P. Tampakis, E. Chondrodima, A. Pikrakis, Y. Theodoridis, K. Pristouris, H. Nakos, E. Petra, T. Dalamagas, A. Kandiros, G. Markakis, I. Maina, S. Kavadas, Sea area monitoring and analysis of fishing vessels activity: The i4sea big data platform, in: IEEE MDM, 2020, pp. 275–280.

[22] E. Chondrodima, N. Pelekis, A. Pikrakis, Y. Theodoridis, An efficient lstm neural network-based framework for vessel location forecasting, IEEE TITS 24 (2023) 4872–4888.

[23] N. Zorbas, D. Zissis, K. Tserpes, D. Anagnostopoulos, Predicting object trajectories from high-speed streaming data, in: IEEE TrustCom/BigDataSE/ISPA, 2015, pp. 229–234.

[24] M. Morzy, Mining frequent trajectories of moving objects for location prediction, in: MLDM, 2007, pp. 667–680.

[25] F. Terroso-Saenz, M. Valdes-Vela, A. F. Skarmeta-Gomez, Online route prediction based on clustering of meaningful velocity-change areas, Data mining and knowledge discovery 30 (2016) 1480–1519.

[26] N. Zygouras, A. Troupiotis-Kapeliaris, D. Zissis, Envclus*: Extracting common pathways for effective vessel trajectory forecasting, IEEE Access 12 (2024) 3860–3873.

[27] N. Pelekis, P. Tampakis, M. Vodas, C. Panagiotakis, Y. Theodoridis, In-dbms sampling-based sub-trajectory clustering, in: EDBT, 2017, pp. 632–643.

[28] N. Pelekis, P. Tampakis, M. Vodas, C. Doulkeridis, Y. Theodoridis, On temporal-constrained sub-trajectory cluster analysis, Data Min. Knowl. Discov. 31 (2017) 1294–1330.

[29] P. Tampakis, N. Pelekis, N. V. Andrienko, G. L. Andrienko, G. Fuchs, Y. Theodoridis, Time-aware sub-trajectory clustering in hermes@postgresql, in: ICDE, 2018, pp. 1581–1584.

[30] P. Petrou, P. Nikitopoulos, P. Tampakis, A. Glenis, N. Koutroumanis, G. M. Santipantakis, K. Patroumpas, A. Vlachou, H. V. Georgiou, E. Chondrodima, C. Doulkeridis, N. Pelekis, G. L. Andrienko, F. Patterson, G. Fuchs, Y. Theodoridis, G. A. Vouros, ARGO: A big data framework for online trajectory prediction, in: SSTD, 2019, pp. 194–197.

[31] P. Tampakis, C. Doulkeridis, N. Pelekis, Y. Theodoridis, Distributed subtrajectory join on massive datasets, ACM Trans. Spatial Algorithms Syst. 6 (2020) 8:1–8:29.