Effective Aggregation and Querying of Probabilistic RFID Data in a Location Tracking Context

RAZIA HAIDER, FEDERICA MANDREOLI, RICCARDO MARTOGLIA
FIM - University of Modena and Reggio Emilia
Department of Applied Mathematics
Via Campi 213/b, 41125 Modena
ITALY
<name.surname>@unimo.it

Abstract: RFID applications usually rely on RFID deployments to manage high-level events such as tracking the location that products visit for supply-chain management, localizing intruders for alerting services, and so on. However, transforming low-level streams into high-level events poses a number of challenges. In this paper, we deal with the well known issues of data redundancy and data-information mismatch: we propose an on-line summarization mechanism that is able to provide small space representation for massive RFID probabilistic data streams while preserving the meaningfulness of the information. We also show that common information needs, i.e. detecting complex events meaningful to applications, can be effectively answered by executing temporal probabilistic SQL queries directly on the summarized data. All the techniques presented in this paper are implemented in a complete framework and successfully evaluated in real-world location tracking scenarios.

Key–Words: RFID data streams, Data summarization, Probabilistic Data Management, Object tracking, Probabilistic Database

1 Introduction

In the last several years, RFID technology has gained significant popularity due to its ability of detecting objects and people. RFID applications usually rely on RFID deployments to manage high-level events such as tracking the location that products visit for supply-chain management [13, 15], monitoring the location and status of patients in hospital environment [23], localizing intruders for alerting services [7], and so on. In an RFID system, an environment is deployed with the RFID readers and antennas, while users and objects carry RFID tags. RFID readers detect the presence of tags in their vicinity and generate streams of low-level observations in the form of TREs (Tag Read Events) $(tag\_id, antenna\_id, time)$ that show when and where tags are being sighted. Since the nature of RFID data stream is noisy, redundant and unreliable, streams of low-level tag-reads such as "Tag 101 was seen at antenna 12 at 10:00" must be transformed into meaningful relation instances such as "Alice entered office 1-10 at 10:00".

The nature of an RFID data stream is noisy, redundant and unreliable. Thus, RFID data it unsuitable for direct use in applications, and the process of transforming low-level streams into high-level events poses a number of challenges [5, 19]. RFID deployments, generally, produce imprecise data because of: (a) *Conflicting Readings*: Readings in the presence of contradiction i.e., when an RFID tag is simultaneously detected by two antennas that cover adjacent areas, it becomes difficult to establish the actual location of tag [22]; *Missing Readings*: Loss of reading instances in which RFID tags are not detected by the antenna while actually being present within its coverage area [11, 22]. A common approach to effectively solve these problems for real-time applications is to use models (e.g. an Hidden Markov Model, HMM) that continuously infer location data based on sensor readings (such as in the filtering and uncertainty management approach proposed in [7, 17]). In this way, the stream can be transformed into a probabilistic data stream $(tagID, location, time, prob)$: an example is `(101,1-10,10:00,0.7)`, which indicates that tag 101 at time 10:00 was in office 1-10 with probability 0.7.

However, two main problems still need to be solved in order to make the generated stream suitable to be usefully exploited in applications:

1. *Extreme redundancy and huge size of data*: RFID tags continually send out their IDs at pre-programmed intervals (few seconds) and for each tag read, the number of probabilistic tuples

equals the number of reference locations. Therefore, an HMM for RFID deployments produces huge volumes of uncertain data that can reach in practical cases the size of gigabytes in a day. Storing all these probabilistic tuples is extremely expensive and, even more important, most often not necessary. For instance, Figure 1 (a,b) depicts one sample scenario, having a total duration of 2 hours and 20 minutes. In Figure 1(a), Paul, a user wearing an RFID tag that transmits every second, works in his office O1 for two hours. Then, Paul goes to the research lab (R2) by passing through the hall (H1), where he stays for some minutes talking with one of his colleagues. Since the number of locations in this scenario is three, $21,600$=(60 seconds * 120 minutes * 3 locations) probabilistic tuples are produced for the first two hours which report more or less the same location information for him (stay in office). This represents a rather realistic scenario, as usually person or good movements are noticeably slower than RFID transmission rates;

2. *Data-Information Mismatch*: Mismatch between the information to which the application is concerned and the available data stream. Typically an application is particularly interested in high-level information such as "Find when Paul was seen last time at his office", "Did it happen that Paul and Suzy were together in one of the recreation rooms?". In order to be able to effectively and efficiently solve such kinds of queries, it is very important to rely on strong data manipulation systems that can simplify the information extraction process while preserving the probabilistic nature of the data.

In this paper, we deal with the two above mentioned problems and close the circle for RFID data management in a location tracking scenario by:

- exploiting a newly introduced on-line summarization mechanism, which is able to provide small space representation for massive RFID probabilistic data streams while preserving the meaningfulness of the information;
- promptly storing the result of the summarization in a probabilistic database (we use MayBMS [20]). In such a way, we show that common information needs, i.e. detecting complex events meaningful to applications, can be effectively answered by executing simple temporal probabilistic SQL queries.

The simple on-line summarization mechanism we propose draws inspiration from the field of clustering

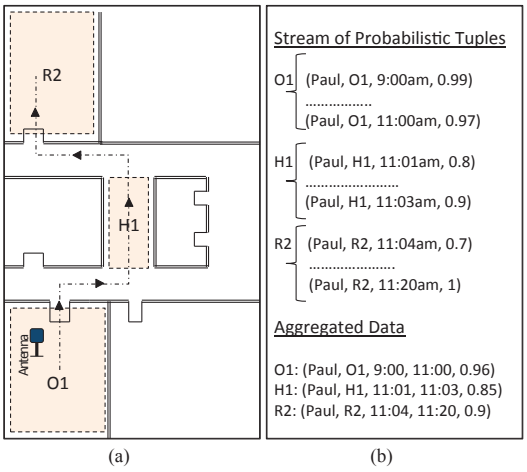


Figure 1: (a) A visual representation for Paul movements; (b) The stream of probabilistic tuples before and after applying the summarization mechanism

[21]. The main idea is to keep on aggregating tuples until a state transition is detected. Our data aggregation algorithm processes probabilistic tuples as they arrive, i.e. directly taking a probabilistic data stream generated by our filtering and uncertainty management techniques [7, 17] as its input, hence avoiding the use of expensive and offline disk based operations such as sorting and summarization. Finally, the use of a probabilistic database for storing and querying the resulting stream greatly simplifies and bridges the gap between stream data and required information.

All the techniques presented in this paper are implemented in a complete framework and evaluated under real-cases in the context of location tracking. However, they are general enough to be applicable to other RFID data management application contexts.

The rest of the paper is organized in the following way: in Section 2 we shortly describe the RFID deployment and the data filtering techniques which, even if not in the scope of the paper, are needed to understand the context in which we operate. Section 3 discusses the summarization/aggregation algorithm, while in Section 4 we deepen how to store and query the aggregated probabilistic tuples. In Section 5 we present extensive experiments in real object tracking scenarios, showing a very good reliability of the proposed techniques. Finally, Section 6 analyzes related works and gives some concluding remarks.

2 Background

In this section, we will shortly describe the background information that is needed to understand the

context in which the techniques presented in this paper are applied. First of all, in Section 2.1 we shortly describe the reference RFID deployment for acquiring the raw data; then, in Section 2.2 we show the filtering techniques that are employed in order to transform the raw stream into a probabilistic stream. This stream will be the actual input of the data aggregation and querying techniques which are at the focus of this paper and we will describe in detail in Section 3.

2.1 Acquiring Raw RFID Data

As discussed in the introduction, RFID readers detect the presence of tags in their vicinity and generate streams of low-level observations in the form of TREs (Tag Read Events) $(tag\_id, antenna\_id, time)$ that show when and where tags are being sighted. In order to acquire such raw data, we exploit an RFID deployment for location tracking purposes, populated by RFID devices including RFID tags and readers. RFID tags are attached to the objects and people that have to be tracked, while RFID readers receive data from these tags in the form of radio signals and convert them in digital form to pass it to the upper levels of the framework. The following are some details on our hardware configuration (to which the results presented in Section 5 will refer):

- ***Reader***: we use a fixed reader that can interrogate tags at distances of up to 300 feet (100 meters). The reader establishes the connection to the host system by using the RS422 interface. For data exchange, a simple master/slave protocol is used by the reader. The protocol also gives us some additional information such as time of data reception, signal strength and number of times the tag has been read by the reader;
- ***Antennas***: the choice of antennas depends on the type and requirement of the application. An *Elliptical Polarized Antenna* has a wide apex angle of $(120°)$, which enables it to cover large read zone. Therefore, it is capable of reading a large number of tags at one time even at fast speeds. The orientation of the tags relative to the antenna is not important. On the other hand, a *Linear Polarized Antenna* is more suitable for applications in which read zones are restricted and data collection must be selective. This antenna has smaller apex angle of $(60°)$. The field of antenna is either horizontally or vertically polarized depending on the mounting direction, thus requiring the tag to have the same orientation. Elliptical antennas are the ones most suited to our purposes and are the ones used for final experimentation;
- ***Tags***: we employ active RFID tags based on UHF radio frequency. The tags are capable of providing long range for wireless applications and can transmit data at distances of up to 300 feet (100 meters) to readers. The tags continuously send static data written in their memory at pre-programmed intervals known as ping rate. Ping rate can be one second to four minutes (one second in our setup). Due to the ultra-low power consumption of the active tags, an operational lifetime of up to 6 years can be expected, making them suitable for identification and tracking applications.

2.2 Filtering Acquired Data

The Data Filtering techniques receive raw data and perform online filtering and uncertainty management on it. In particular, by exploiting a specially designed data model which is based on a Hidden Markov Model (HMM) and ad-hoc particle filtering techniques [7, 17], we take the raw RFID data stream as input and produce as output a probabilistically cleaned and filtered RFID data stream.

In our location tracking context, given m tags $T_1, T_2, \ldots, T_m$ and n locations $\lambda^1, \lambda^2, \ldots, \lambda^n$, data filtering produces a stream of timestamp ordered probabilistic tuples:

$$X_1^{T_1}, X_1^{T_2}, \ldots, X_1^{T_m}, X_2^{T_1}, \ldots, X_2^{T_m}, \ldots$$

where each tuple $X_t^{T_i}$ has the form:

$$(T_i, t, P(L_{T_i,t} = \lambda^1), P(L_{T_i,t} = \lambda^2), \ldots, P(L_{T_i,t} = \lambda^n))$$

where $P\left(L_{T_i,t}\right)$ is the probability distribution of the random variable $L_{T_i,t}$ over locations $\lambda^1, \lambda^2, \ldots, \lambda^n$, one for each tag T_i. In other words, for each location λ_k, $P\left(L_{T_i,t} = \lambda^k\right)$ represents the probability that tag T_i is in λ^k at time t. Please note that, for ease of presentation and without loss of generality, we assume that tuples arrive in tag order and that the discrete probability distribution of the location random variable is represented as one tuple instead of n different tuples.

3 Aggregating Probabilistic RFID Data

An HMM filtering technique, as the one shortly described in the last section, produces huge volumes of uncertain data that can become really difficult to manage. Moreover, high level events such as a location

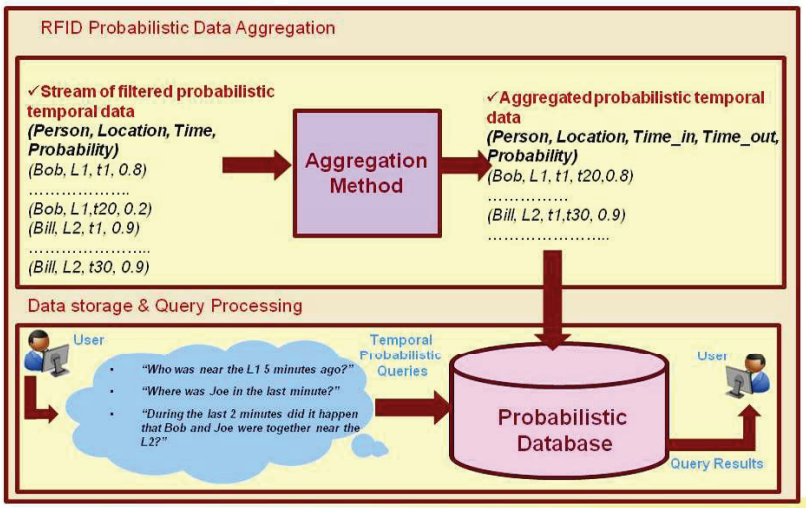


Figure 2: Block Diagram of our Data Aggregation and Query Processing Approach

change by a subject are still implicit in the data and could require expensive data scans in order to be identified. Our aggregation/summarization technique (see top part of Figure 2 for a block diagram) gives an answer to such problems. It receives the probabilistic RFID data streams and aggregates them by applying an on-line aggregation algorithm.

The aggregation algorithm outputs a stream of probabilistic tuples of the form:

$$X^{T_i}_{[t_s,t_e]} = (T_i, t_s, t_e, P(L_{T_i,[t_s,t_e]} = \lambda^1), P(L_{T_i,[t_s,t_e]} = \lambda^2), \ldots, P(L_{T_i,[t_s,t_e]} = \lambda^n)$$

such that:

- for each pair of tuples on the same tag T_i, $X^{T_i}_{[t_{s_1},t_{e_1}]}$ and $X^{T_i}_{[t_{s_2},t_{e_2}]}$, $[t_{s_1}, t_{e_1}] \cap [t_{s_2}, t_{e_2}] = \emptyset$;
- for each source tuple $X^{T_i}_t$, a result tuple $X^{T_i}_{[t_s,t_e]}$ exists such that $t \in [t_s, t_e]$.

The aggregation algorithm (Algorithm 1) works on the intuition that if a person wearing a tag T_i is stationary or resides at the same location for a period of time $[t_s, t_e]$, the corresponding probabilistic tuples $X^{T_i}_{t_s}, \ldots, X^{T_i}_{t_e}$ should show "similar" probability distributions. Therefore, in order to derive $X^{T_i}_{[t_s,t_e]}$ it draws inspiration from the large dataset clustering field [16] in that it incrementally groups together consecutive "similar" tuples. To this end, at each timestamp t the algorithm maintains at most m clusters, one for each tag T_i, and for each cluster $c^{T_i}_t$ it treats the tuple region collectively through some statistics $stat^{c^{T_i}_t}$, providing a summarized description for the cluster. When a new tuple $X^{T_i}_{t+1}$ arrives, the algorithm tries to add it to the cluster associated to the corresponding tag $c^{T_i}_t$ by updating the corresponding $stat^{c^{T_i}_{t+1}}$ values (see lines 3–5 of Algorithm 1). Then, a boundary condition is checked (line 6) and, if it is the case, the tuple is inserted into the cluster by replacing its statistics with the newly computed ones $stat^{c^{T_i}_{t+1}}$ (line 7). On the other hand, if a violation is detected:

- $c^{T_i}_t$ is closed and discarded from the set of current clusters S (line 10);
- a tuple $X^{T_i}_{[t_s,t]}$ describing the behavior of the tag T in the period in which the cluster $c^{T_i}_t$ was active is stored in the database (line 11);
- a new cluster for T_i is created including tuple $X^{T_i}_{t+1}$ only, its statistics is computed and it is added to S (lines 12 and 13).

Until now, we intentionally left our aggregation model generic. In the following, we show how the clusters are represented and how cluster statistics are computed.

3.1 Cluster Representation

In many clustering applications, the resulting clusters have to be represented or described in a compact form to achieve data abstraction. Basically, the most typical compact description of a cluster is given in terms of cluster prototypes or representative patterns such as the *centroid* [21]. The centroid is the logical center

Algorithm 1 Tuple aggregation algorithm

Require: n number of locations, m number of tags , B critical boundary

1: $S =$ current set of clusters; *//S contains at most m elements*
2: **repeat**
3: receive the next stream point $X_{t+1}^{T_i}$
4: $c_t^{T_i}$ = identifyCluster($X_{t+1}^{T_i}$, S) *//$stat^{c_t^{T_i}}$ is extracted from $c_t^{T_i}$*
5: $stat^{c_{t+1}^{T_i}}$= updateStatistics($stat^{c_t^{T_i}}$,$X_{t+1}^{T_i}$)
6: **if** testBoundaryCondition($stat^{c_{t+1}^{T_i}}$) **then**
7: $c_{t+1}^{T_i}$ = add($X_{t+1}^{T_i}$,$c_t^{T_i}$); *//$stat^{c_t^{T_i}}$ is replaced with $stat^{c_{t+1}^{T_i}}$*
8: update S with $c_{t+1}^{T_i}$;
9: **else**
10: close and discard $c_t^{T_i}$ from S;
11: insert $X_{[t_s,t]}^{T_i}$ in the database;
12: $c_{t+1}^{T_i}$ = createNewCluster($X_{t+1}^{T_i}$);
13: add $c_{t+1}^{T_i}$ to S;
14: **end if**
15: **until** data stream ends

of the cluster, usually computed as the average of all cluster points. The use of the centroid to represent a cluster is a very popular schema and works well when the clusters are compact, as in our case.

Therefore, we represent tuples in the n-dimensional Cartesian space as points whose coordinates are the probability values for the n locations. For instance, going back to our reference example, the number of locations is three, therefore each tuple would be a point in a 3-dimensional space, whose coordinates are the probability values for the locations O1, H1 and R2. Since Paul is residing at a same place (his office) for a long period, a large number of points would be concentrated in a specific region of the plane ("O1 region"); all these points could be aggregated in one point which will be representative of the behavior of all of them. As Paul moves from O1 to H1 and consequently to R2, there is a transition that could be seen in the form of some scattered points on the graph plane. Hereinafter, whenever the context is clear, we will use $X_t^{T_i}$ to denote either a probabilistic tuple $(T_i, t, P(L_{T_i,t} = \lambda^1), P(L_{T_i,t} = \lambda^2), \ldots, P(L_{T_i,t} = \lambda^n))$ or its representation in the Cartesian space $(P(L_{T_i,t} = \lambda^1), P(L_{T_i,t} = \lambda^2), \ldots, P(L_{T_i,t} = \lambda^n))$.

Then, we incrementally compute the centroid $V_{c_t^{T_i}}$ of each cluster $c_t^{T_i}$ while it evolves and, when it is closed, we store $X_{[t_s,t]}^{T_i}$ as $(T_i, t_s, t, V_{c_t^{T_i}})$.

3.2 Determining When to Close a Cluster

The main objective of the boundary condition test is to be able to discriminate when a cluster has to be closed in order to avoid distortion. To this end, we draw inspiration from techniques at the state of the art for cluster validity measurement [26]. Two measurement criteria are typically used for evaluating a clustering schema [26]: compactness and separation. While the former expresses the requirement that the members of each cluster should be as close to each other as possible, the latter refers to the fact that the clusters themselves should be widely separated and it is not particularly interesting for our scenario; we thus focus on compactness and consider three different methods for quantifying it. The three models, which provide different indices that can be used in the boundary condition test, are:

- *Maximum Probability Change (MPC)*: it monitors the probability distribution trends. To this end, let $\overline{L}_{X_t^{T_i}}(\overline{L}_{c_t^{T_i}})$ be the location with the maximum probability value in $X_t^{T_i}(c_t^{T_i})$. For each cluster $c_t^{T_i}$, MPC maintains $\overline{L}_{c_t^{T_i}}$ as statistics, and the boundary condition is satisfied when $\overline{L}_{c_t^{T_i}} = \overline{L}_{X_{t+1}^{T_i}}$. The main disadvantage of this method is that it is very sensitive to noise and thus makes more clusters with fewer points in it;

- *Diameter-oriented (DM)*: it measures how large the cluster shape is. To this end it uses the cluster diameter as statistics and checks whether the latter is within a threshold B: $\max_{X,Y \in c_{t+1}^{T_i}}\{d(X,Y)\} \leq B$. The main disadvantage of this approach is the time and space complexity, due to the fact that the distance between all pairs of points have to be computed and constantly kept updated on the arrival of new data elements. This function is also very sensitive to noise, since the maximum cluster diameter can quickly become large in a noisy environment;

- *Centroid Vs Latest Reading Comparison (CLRC)*: it gives a measure of the mutual distance between the centroid $V_{c_t^{T_i}}$ and the latest point $X_{t+1}^{T_i}$. To this end, it checks whether $d(V_{c_t^{T_i}}, X_{t+1}^{T_i}) \leq B$. The main advantage of this method w.r.t. the DM model is that computations are less time and space consuming, as $V_{c_t^{T_i}}$ can be computed incrementally.

Regarding distance $d(\cdot,\cdot)$ between tuples, our approach is independent from the actually adopted func-

tion. Several alternatives are possible for its implementation since we only require it is applicable in a n-dimensional space. In our experiments we adopted the Euclidean distance. Finally, note that for both DM and CLRC, we can control the quality of the clustering process by properly selecting the threshold B: low values of B produce a high number of small and tight clusters, while we have an opposite behavior for high values of B.

4 Querying Probabilistic Aggregated Data

In this section, we will describe how the summarized probabilistic tuples produced by the aggregation algorithm can be effectively and efficiently managed. Given their probabilistic nature, we show that they can be directly stored and queried in a probabilistic database management system.

A probabilistic database (we use MayBMS [1]) stores data by means of special U-relational tables, providing a complete and concise representation of the large number of possible worlds that are generated in the presence of probabilistic tuples [3]. It also provides an expressive query language that supports the entire set of capabilities offered by SQL and extends it with features designed to support the probability and to work with uncertainty. For instance, special functions such as $conf()$, $aconf()$ are available for calculating the confidence of the tuples, $argmax()$, $esum()$, $ecount()$ are approximate aggregation functions, and so on. This language is sufficiently general because it adopts semantics independent of the details related to the mode of data representation and composition.

Due to its compatibility with the relational algebra and standard SQL, a comprehensive set of constructs for data transformation can be easily exploited. In this way, complex high-level events can be successfully extracted by means of standard (probabilistic) SQL queries. In the following, we show some significant examples of possible queries that can be issued on the probabilistic data we generate. The queries contain constraints (interval or snapshot) over the temporal history of the RFID data and are used to identify and track RFID objects in the test environment. The queries are expressed directly on the summarized version of the data: in Section 5, we will also experimentally prove that the effectiveness of the obtained answers is the same as the one achieved by executing the queries on the whole unsummarized data. `start_time()` and `cur_time()` are user-defined functions for retrieving the startup time of the used data set and the current system time. The employed relational schema is the following:

```
   cluster_data(tag_id,
time_in, time_out, location_id,
              probability),
```

which directly reflects the contents of the output tuples as discussed in Section 3.

Q1. *Find who was at location 'L1' 10 seconds ago?*

```
SELECT tag_id, conf()
FROM cluster_data
WHERE location_id='L1'
AND time_in < SELECT cur_time()
    - interval '00:00:10'
AND time_out > SELECT cur_time()
    - interval '00:00:10'
GROUP BY tag_id;
```

Q2. *Find where was person 'P1' at time 't'?*

```
SELECT location_id, conf()
FROM cluster_data
WHERE tag_id='P1'
AND time_in < 't'
AND time_out > 't'
GROUP BY location_id;
```

Q3. *Find when 'P1' was seen last time at location 'L1'?*

```
SELECT time_out, conf()
FROM cluster_data
WHERE location_id= 'L1'
AND tag_id = 'P1'
AND time_out=
   (SELECT max(time_out)
    FROM cluster_data
    WHERE location_id = 'L1'
    AND tag_id = 'P1'
    AND probability > 0.5)
GROUP BY time_out;
```

Q4. *Find where and which persons are detected at the first moment by the system?*

```
SELECT location_id, tag_id,
   conf()
FROM cluster_data
WHERE time_in=
   (SELECT start_time())
GROUP BY location_id, tag_id;
```

Q5. *Whether it happened that 'P1' and 'P2' are together at the same location at the same time? Where?*

```
SELECT c1.location_id, conf()
FROM cluster_data c1,
   cluster_data c2
WHERE c1.tag_id = 'P1'
AND c2.tag_id = 'P2'
AND c1.location_id =
   c2.location_id
GROUP BY c1.location_id;
```

Q6. *Find when 'P1' moved from location 'L1' to 'L2'?*

```
SELECT c2.time_in,conf()
FROM cluster_data c1,
   cluster_data c2
WHERE c1.tag_id = 'P1'
AND c1.tag_id = c2.tag_id
AND c1.location_id = 'L1'
c2.location_id = 'L2'
AND (c2.time_in-c1.time_out) <=
   interval '00:00:02'
AND (c2.time_in-c1.time_out) >=
   interval '00:00:00'
GROUP BY c2.time_in;
```

5 Experimental Evaluation

In order to evaluate the performance of the presented approach, we have conducted several experiments in different location tracking scenarios, collecting data from persons wearing RFID tags. The experimental scenarios are all set in three indoor locations (denoted L1, L2 and L3) and capture different possible movement behaviors: (i) "No Stay", where people rapidly move between locations without staying on any specific one; and (ii) "Stay", where people move between locations and spend some time on each of them. Both types of scenarios have been tested with one/multiple tags. In all the experiments, we apply the aggregation methods we propose to the stream of tuples generated by the *RFID Online Filtering and Uncertainty Management Module*.

The goal of our evaluation studies is two-fold: (i) to validate and compare the effectiveness of each method in precisely summarizing the movement behaviors which actually took place in the scenarios (Section 5.1); and (ii) to evaluate the best performing method on realistic target applications, i.e. to compare the results which can be obtained by querying the RFID data via a temporal probabilistic database with and without applying the aggregation method to the involved data (Section 5.2).

5.1 Effectiveness of Aggregation Methods

We analyze the performance of the presented aggregation methods by means of five experiments conducted on different movement scenario types (stay/no stay) and with a varying number of actually visited locations and tags. The experimental setup and the obtained results are summarized in the left and right parts of Table 1, respectively. For each experiment, we measure the effectiveness of the methods based on four parameters: (a) number of output clusters (#Cluster); (b) fraction of occupied space w.r.t. non-aggregated data (SP); (c) percentage of time at actual location (%TAL); and (d) average location error (AvgLocError) between clustered and actual locations. The basic intuition for (a) is that the nearer it is to the number of actually visited locations, the more effective is the method; (b) provides a clear quantification of the space required by the aggregated tuples (the smaller the fraction the higher the saved space). Beyond these "overview" approaches, (c) and (d) provide us with more detailed information on the actual contents of the generated clusters. More specifically, the %TAL is the percentage of time for which aggregated data reports the same location as of ground truth; besides correctness, this gives us an idea about the promptness of each method to adjust the output to the ground truth over the experiment duration (the higher the value the better). Finally, average location error takes into account how much the summarized description of each generated cluster is near to the actual ground truth values. We devised the measure so to highlight how long and how much each method differs from the ground truth: it is calculated by means of an average Euclidean distance between the ground truth and the aggregated summarized descriptions over the total time span, only considering those time instants when a "wrong" location is reported. Values of AvgLocError are between 0 and 1, therefore the lower the value the better the estimate.

Table 1: Performance Evaluation of (a) MPC, (b) DM and (c) CLRC

(a) MPC								
EXP	Scenario	#Tags		#Locs	#Clusters	%SP	%TAL	AvgLocError
1	No Stay	1		3	3 (=)	0.033	98.91	0.0136
2	Stay	1		5	13 (+160%)	0.026	95.93	0.0452
3	No Stay	2	Tag 1	5	9 (+80%)	0.080	86.61	0.1549
			Tag 2	5	11 (+120%)	0.097	83.04	0.1957
4	Stay	2	Tag 1	4	8 (+100%)	0.033	92.89	0.0707
			Tag 2	4	10 (+150%)	0.041	95.82	0.0453
5	Stay	2	Tag 1	4	16 (+300%)	0.053	82.16	0.1929
			Tag 2	4	13 (+225%)	0.043	86.96	0.1495
Mean					+141%	0.050	90.29	0.108

(b) DM								
EXP	Scenario	#Tags		#Locs	#Clusters	%SP	%TAL	AvgLocError
1	No Stay	1		3	3 (=)	0.033	98.91	0.0136
2	Stay	1		5	5 (=)	0.010	96.95	0.0383
3	No Stay	2	Tag 1	5	5 (=)	0.044	88.39	0.1419
			Tag 2	5	5 (=)	0.044	85.71	0.1739
4	Stay	2	Tag 1	4	5 (+25%)	0.020	94.14	0.0608
			Tag 2	4	8 (+100%)	0.033	95.82	0.0445
5	Stay	2	Tag 1	4	6 (+50%)	0.020	84.95	0.1696
			Tag 2	4	8 (+100%)	0.026	87.96	0.1374
Mean					+34%	0.040	91.60	0.0975

(c) CLRC								
EXP	Scenario	#Tags		#Locs	#Clusters	%SP	%TAL	AvgLocError
1	No Stay	1		3	3 (=)	0.033	98.91	0.0136
2	Stay	1		5	5 (=)	0.010	96.95	0.0383
3	No Stay	2	Tag 1	5	5 (=)	0.044	88.39	0.1410
			Tag 2	5	5 (=)	0.044	85.71	0.1739
4	Stay	2	Tag 1	4	4 (=)	0.016	94.14	0.0596
			Tag 2	4	5 (+25%)	0.020	96.65	0.0393
5	Stay	2	Tag 1	4	4 (=)	0.013	88.63	0.1190
			Tag 2	4	5 (+25%)	0.016	89.97	0.1185
Mean					+6%	0.024	92.41	0.0879

Besides the complete report shown in the right part of Table 1, Figures 3(a-d) offer an immediate graphical comparison between the three aggregation methods on the basis of the experimental results. The values shown in the graphs are the mean values between all the different experiments.

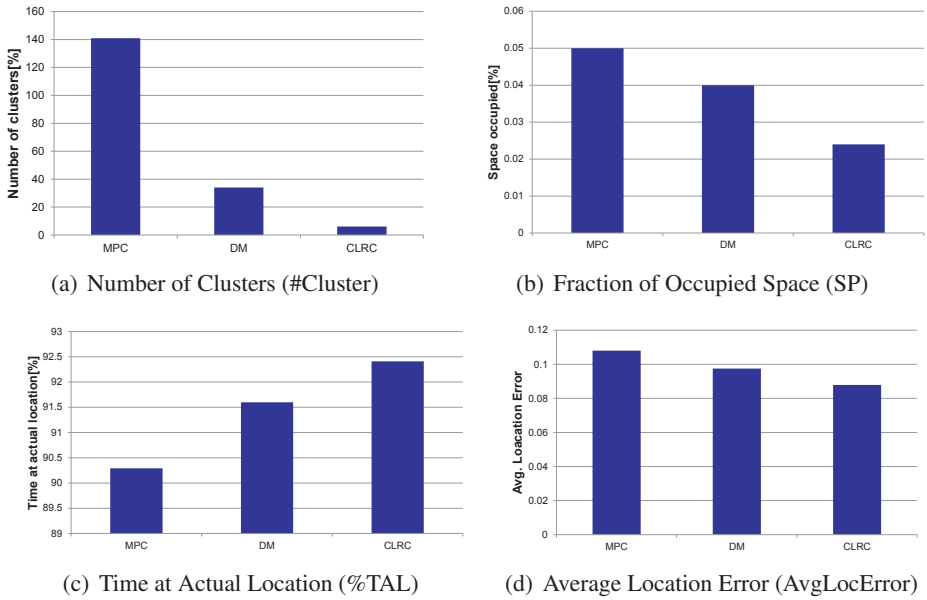
From the obtained experimental results, we see that MPC is very sensitive to noise and thus performs poorly in the presence of noisy data. On average, it makes 141% more clusters than expected (up to 300% more in EXP5), while the average location error is quite high, for instance with values of 0.19 for EXP3 and EXP5 (0.108 on mean for all the experiments). TAL is about 90% on mean, with the lowest values being 83% (EXP3) and 82% (EXP5).

DM performs better than MPC but its diameter can quickly become very large in presence of noisy data. DM has an average location error of 0.0975 and average TAL of approximately 92%, while it makes 34% more clusters than expected.

CLRC shows superior performance to MPC and DM, giving good results even in noisy environments. The average TAL is about 92%, whereas the average location error is approximately 0.0879; on average, it only makes 6% more clusters than expected, which, together with the other figures, represents a very encouraging result. The same holds for the very consistent space savings produced by all methods (ranging from 0.05% of the space required by non-aggregated data to the most compact 0.024%, given by MPC and CLRC, respectively).

5.2 Aggregation Effects on Temporal Probabilistic Query Processing

After having evaluated the goodness of the output data *per se*, we now want to assess the performance of a probabilistic DBMS in answering some typical queries over the summarized versus non-summarized data of our five experiments. For the tests in this sec-



(a) Number of Clusters (#Cluster)

(b) Fraction of Occupied Space (SP)

(c) Time at Actual Location (%TAL)

(d) Average Location Error (AvgLocError)

Figure 3: Comparison between aggregation methods

tion we will exploit the CLRC method, since it has been proven the best performing one (see Section 5.1).

As described in Section 4, in order to handle the uncertainty associated to our probabilistic streams, we use the MayBMS database management system [20]. In our experiments, we validated the results obtained on the aggregated and complete data over a number of queries. The ones we selected are those discussed in Section 4; the obtained results are summarized in Table 2. In particular, the table shows, for each of the five experiments (columns) and of the six queries (rows), from left to right, the actual (expected) and computed output results over aggregated and non-aggregated data. Note that, in some of the experiments (EXP1, EXP4 and EXP5) the actual answer to Q1 should be "no one" ("-" in Table 2). Furthermore, Q5 is not applicable to EXP1 and EXP2, since only one tag is used.

In all cases, we can see that the results on summarized data are correct and with a confidence which is very near (almost identical) to the non-aggregated data results; this shows that, even if data in aggregated form contain less detailed information, they provide accurate answers to the queries. Moreover, in some cases the confidence of the correct answer is higher on the summarized data, due to the noise that is present in the non-summarized data (see for instance Q2 in experiments 1-4).

Finally, Q6 is an interesting case of high-level event detection, in this case a transition between locations. As expected, the results we got from the DBMS experimentally prove that transitions are much easier to identify on the aggregated data, since the complete data contain a lot of "noise", thus producing a very large quantity of irrelevant and/or incorrect results ("*" in Table 2).

6 Related Works and Concluding Remarks

One of the main concerns for data management is that the rate of RFID data streams is quite high and, therefore, the resulting volume of the stream is quite huge. For RFID data compression, several proposals have been recently discussed in literature. A graph-based model is discussed in [6] for providing compression in RFID systems. This model captures the possible object locations and their containment relationships. However, high detection rates at the RFID readers are required in order to have accurate results. In [14], a new model for warehousing RFID data has been proposed. The proposed model provides significant data compression and path-dependent aggregates while preserving the object transitions. The proposed work basically takes advantage of object movements in bulk, of data generalization and the merge or collapse of the path segment that RFID objects follow.

In [4] the authors present an aggregation mechanism for RFID data streams based on temporal and spatial aggregations. The proposed algorithm exploits the time and space dimension to reduce the volume of input RFID data streams. A special data cube termed as Flowcube is introduced in [12] for RFID systems. The Flowcube is a data cube computed for a large col-

Table 2: Probabilistic Query Results for Aggregated and Non-Aggregated (Complete) Data

	EXP1					EXP2				
	Actual	*Aggregated Data*		*Non-Aggregated Data*		*Actual*	*Aggregated Data*		*Non-Aggregated Data*	
			conf		*conf*			*conf*		*conf*
Q1	P1	P1	0.983	P1	0.996	_	P1	0.027	P1	0.004
Q2	L2	L1 L2 L3	0.105 0.831 0.062	L1 L2	0.482 0.518	L3	L2 L3	0.213 0.786	L2 L3	0.606 0.394
Q3	2:09:34	2:09:34	0.983	2:09:34	0.78	5:20:55	5:20:55	0.984	5:20:55	1
Q4	L1,P1	L1,P1	0.983	L1,P1	0.994	L1,P1	L1,P1	0.975	L1,P1	1
Q5	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Q6	2:09:35	2:09:35 2:09:46	0.817 0.022	*	*	5:14:44	5:14:48 5:16:47 5:19:50	0.879 0.005 0.0001	*	*

	EXP3					EXP4				
	Actual	*Aggregated Data*		*Non-Aggregated Data*		*Actual*	*Aggregated Data*		*Non-Aggregated Data*	
			conf					*conf*		*conf*
Q1	P1 P2	P1 P2	0.981 0.988	P1 P2	1 1	_	P1 P2	0.033 0.004	P1 P2	0.016 0.04
Q2	L2	L1 L2 L3	0.026 0.944 0.03	L2 L3	0.976 0.024	L2	L1 L2 L3	0.053 0.898 0.048	L1 L2	0.216 0.784
Q3	4:41:24	4:41:24	0.94	4:41:24	0.516	6:05:10	6:05:10	0.989	6:05:10	0.801
Q4	L1,P1 L1,P2	L1,P1 L1,P2	0.988 0.981	L1,P1 L1,P2	1 1	L1,P1 L1,P2	L1,P1 L1,P2	0.996 0.989	L1,P1 L1,P2	1 1
Q5	L1 L2 L3	L1 L2 L3	0.998 0.974 0.686	L1 L2 L3	1 1 0.999	L1 L2 L3	L1 L2 L3	0.987 0.991 0.654	L1 L2 L3	1 1 1
Q6	4:40:03	4:40:04 4:40:20 4:41:03	0.701 0.028 0.001	*	*	6:05:11	6:05:12 6:06:08	0.882 0.001	*	*

	EXP5				
	Actual	*Aggregated Data*		*Non-Aggregated Data*	
			conf		*conf*
Q1	_	P1 P2	0.037 0.021	P1	0.002
Q2	L3	L2 L3	0.166 0.833	L2 L3	0.146 0.854
Q3	6:26:45	6:26:39	0.988	6:26:56	0.518
Q4	L1,P1 L1,P2	L1,P1 L1,P2	0.998 0.988	L1,P1 L1,P2	1 1
Q5	L1 L2 L3	L1 L2 L3	0.988 0.969 0.73	L1 L2 L3	1 1 1
Q6	6:26:46	6:26:40 6:28:02	0.842 0.022	*	*

lection of paths. The Flowcube computes the movement trends of each specific item instead of computing aggregated measurements like in a traditional data cube. Basically, the Flowcube examines item flows in an RFID system. Since RFID data has different flow of information from the traditional data, data storage and query processing tasks are difficult. Lee et al. has discussed this aspect of RFID data in [25]. They proposed an efficient storage scheme and query processing for supply chain management. They used an effective path encoding method to represent the flow information representing movements of products. A storage scheme is developed to process tracking queries and path oriented queries efficiently based on path encoding scheme and numbering scheme.

Another approach for RFID data compression in a supply chain scenario is presented in [8]. This approach takes advantage of the property that objects move together. In particular, this work represents an incremental aggregation approach based on various combinations of attributes describing RFID data other than paths and locations. Using this compression approach, the authors develop a lossless, relational-based storage model which preserves information about both path dependent and path independent items. In [10], a lossy compression technique is proposed for RFID data streams. In particular, the authors define a data structure to represent compressed RFID warehouses. Moreover, they proposed an architecture that gathers readings from RFID readers and

store them in a compact way.

Also, in the database community, various algorithms have been proposed for a number of clustering problems and several methods working on very large amounts of data gained popularity, such as DBSCAN [9], CURE [16] and BIRCH [30]. Besides purely deterministic approaches, the vague and uncertain nature of the data stream has recently captured a lot of research attention and many clustering algorithms have been proposed which also take into account the probabilities associated to the involved data. In this context, a fuzzy version of DBSCAN has been presented as FDBSCAN [24]. This algorithm, instead of finding regions with high density, identifies regions with high expected density, based on the probability distributions of the objects.

Another probabilistic extension is P-DBSCAN [28], which takes advantage of the probability distribution information of the object locations in the definition and computation of probabilistic core object and probabilistic density-reachability.

In [27], an extension of the K-means algorithm is proposed, named as UK-means algorithm, which considers expected distance between the object and the representative of the cluster.

As UK-means is based on classical K-means algorithm, it can be sensitive to noise. UMicro [2] uses a general model of the uncertainty and keeps track of the standard errors of each dimension within each cluster, showing that the use of even general uncertainty model during the clustering process is enough to improve the quality of results over purely deterministic approaches. Other similar related approaches are the two-phase clustering algorithm discussed by Zhang et al. in [29], named as LuMicro, and PW-Stream [18], which has been proposed for the specific problem of sliding windows.

The objective of most of the methods discussed above is to analyze incoming data and judge on their "certainty", thus producing the highest quality possible clusters both in terms of compactness and high probability, discarding low quality ones. Further, they work on the assumption of knowing specific information characterizing the uncertainty, such as having the entire probability density function or standard error data available. The number of clusters to be produced is also usually known in advance.

On the other hand, the method presented in this paper is targeted for a different objective, i.e. a summarization task in a location tracking context, and is thus designed to work on a different perspective. More specifically, our ultimate goal is to correctly identify and highlight state transitions, while avoiding redundant information produced in stable states. In this context, not only one active cluster per tag suffices but, even more importantly, we never have to judge on the quality (probability) of the created clusters; instead, we purely and "objectively" summarize the received data in order to make it available to upper level applications in a more compact but equally meaningful way. In this way, as experimentally proven, a probabilistic database such as MayBMS can effectively answer a wide range of probabilistic queries on the summarized version of the data, which only take up a fraction of the original space.

In the future, we will test our summarization and querying approach in larger settings, also involving open environments and/or a higher number of antennas. Moreover, we will also consider other application scenarios beyond location tracking and see how the method can be customized for them in order to maintain its high level of effectiveness.

References

[1] http://www.cs.cornell.edu/bigreddata/maybms/.

[2] C.C. Aggarwal and P.S. Yu. A framework for clustering uncertain data streams. In *Proceedings of the 24th international conference on Data Engineering*, pages 150–159. IEEE, 2008.

[3] L. Antova, T. Jansen, C. Koch, and D. Olteanu. Fast and simple relational processing of uncertain data. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 983–992. IEEE, 2008.

[4] D. Bleco and Y. Kotidis. Rfid data aggregation. *GeoSensor Networks*, pages 87–101, 2009.

[5] S. S Chawathe, V. Krishnamurthy, S. Ramachandran, and S. Sarma. Managing RFID data. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 1189–1195, 2004.

[6] R. Cocci, T. Tran, Y. Diao, and P. Shenoy. Efficient data interpretation and compression over rfid streams. In *Data Engineering, 2008. ICDE 2008. IEEE 24th International Conference on*, pages 1445–1447. IEEE, 2008.

[7] R. Cucchiara, M. Fornaciari, R. Haider, F. Mandreoli, R. Martoglia, A. Prati, and S. Sassatelli. A Reasoning Engine for Intruders' Localization in Wide Open Areas using a Network of Cameras and RFIDs. In *Proceedings of 1st IEEE Workshop on Camera Networks and Wide Area Scene Analysis*. IEEE, 2011.

[8] R. De Virgilio, P. Sugamiele, and R. Torlone. Incremental aggregation of rfid data. In *Proceedings of the 2009 International Database Engineering & Applications Symposium*, pages 194–205. ACM, 2009.

[9] M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data mining*, volume 1996, pages 226–231. Portland: AAAI Press, 1996.

[10] B. Fazzinga, S. Flesca, E. Masciari, and F. Furfaro. Efficient and effective rfid data warehousing. In *Proceedings of the 2009 International Database Engineering & Applications Symposium*, pages 251–258. ACM, 2009.

[11] C. Floerkemeier and M. Lampe. Issues with RFID usage in ubiquitous computing applications. *Pervasive Computing*, pages 188–193, 2004.

[12] H. Gonzalez, J. Han, and X. Li. Flowcube: constructing rfid flowcubes for multi-dimensional analysis of commodity flows. In *Proceedings of the 32nd international conference on Very large data bases*, pages 834–845. VLDB Endowment, 2006.

[13] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analyzing massive RFID data sets. In *22nd International Conference on Data Engineering, ICDE'06*. IEEE Computer Society, 2006.

[14] H. Gonzalez, J. Han, X. Li, and D. Klabjan. Warehousing and analyzing massive RFID data sets. In *Data Engineering, 2006. ICDE'06. Proceedings of the 22nd International Conference on*, page 83, 2006.

[15] Yu Gu, Ge Yu, and Na Guo. Triggered moving range queries over rfid monitored objects. *J. Inf. Sci. Eng.*, 29(3):401–416, 2013.

[16] S. Guha, R. Rastogi, and K. Shim. CURE: an efficient clustering algorithm for large databases. In *ACM SIGMOD Record*, volume 27, pages 73–84. ACM, 1998.

[17] Razia Haider, Federica Mandreoli, and Riccardo Martoglia. Online Filtering and Uncertainty Management Techniques for RFID Data Processing. *WSEAS Transactions on Information Science and Applications*, 11:231–241, 2014.

[18] W. C. Hu and Z. L. Cheng. Clustering algorithm for probabilistic data streams over sliding window. In *Proceedings of the 9th International Conference on Machine Learning and Cybernetics (ICMLC)*, pages 2065–2070. IEEE, 2010.

[19] Y. Hu, S. Sundara, T. Chorma, and J. Srinivasan. Supporting rfid-based item tracking applications in oracle dbms using a bitmap datatype. In *Proceedings of the 31st international conference on Very large data bases*, pages 1140–1151. VLDB Endowment, 2005.

[20] J. Huang, L. Antova, C. Koch, and D. Olteanu. MayBMS: a probabilistic database management system. In *Proceedings of the 35th SIGMOD international conference on Management of data*, pages 1071–1074. ACM, 2009.

[21] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Compututing Survey*, 31:264–323, September 1999.

[22] S. R Jeffery, M. Garofalakis, and M. J Franklin. Adaptive cleaning for RFID data streams. In *Proceedings of the 32nd international conference on Very large data bases*, pages 163–174, 2006.

[23] D.S. Kim, J. Kim, S.H. Kim, and S.K. Yoo. Design of RFID based the Patient Management and Tracking System in hospital. In *Engineering in Medicine and Biology Society, EMBS. 30th Annual International Conference of the IEEE*, pages 1459–1461. IEEE, 2008.

[24] H.P. Kriegel and M. Pfeifle. Density-based clustering of uncertain data. In *Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 672–677. ACM, 2005.

[25] C.H. Lee and C.W. Chung. Efficient storage scheme and query processing for supply chain management using rfid. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 291–302. ACM, 2008.

[26] Csaba Legány, Sándor Juhász, and Attila Babos. Cluster validity measurement techniques. In *Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases*, pages 388–393, 2006.

[27] W.K. Ngai, B. Kao, C.K. Chui, R. Cheng, M. Chau, and K.Y. Yip. Efficient clustering of

uncertain data. In *Proceedings of the 6th International Conference on Data Mining(ICDM),*, pages 436–445. IEEE, 2006.

[28] H. Xu and G. Li. Density-based probabilistic clustering of uncertain data. In *Proceedings of International Conference on Computer Science and Software Engineering*, pages 474–477. IEEE, 2008.

[29] C. Zhang, M. Gao, and A. Zhou. Tracking high quality clusters over uncertain data streams. In *25th International Conference on Data Engineering, ICDE'09.*, pages 1641–1648. IEEE, 2009.

[30] T. Zhang, R. Ramakrishnan, and M. Livny. BIRCH: an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114. ACM, 1996.