Fast On-Line Summarization of RFID Probabilistic Data Streams

Razia Haider, Federica Mandreoli, Riccardo Martoglia and Simona Sassatelli

DII - University of Modena and Reggio Emilia
Via Vignolese, 905, 41125, Modena, Italy
{firstname.lastname}@unimore.it

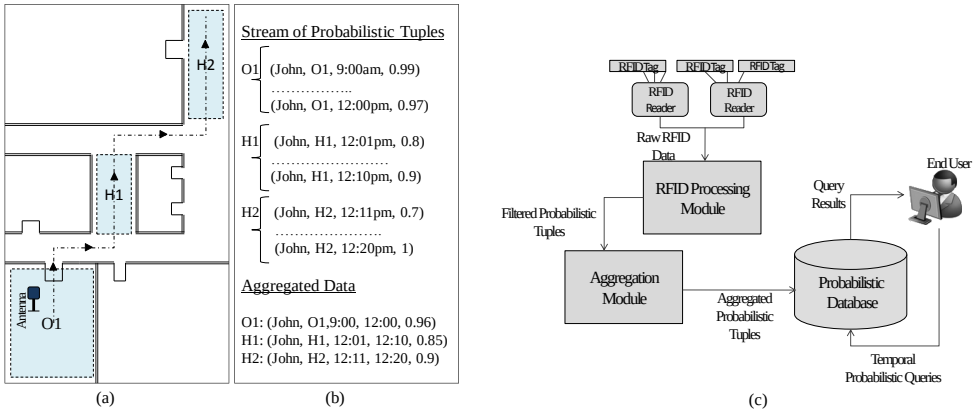
Abstract. RFID applications usually rely on RFID deployments to manage high-level events. A fundamental relation for these purposes is the location of people and objects over time. However, the nature of RFID data streams is noisy, redundant and unreliable and thus streams of low-level tag-reads can be transformed into probabilistic data streams that can reach in practical cases the size of gigabytes in a day. In this paper, we propose a simple on-line summarization mechanism, which is able to provide small space representation for massive RFID probabilistic data streams while preserving the meaningful information. The main idea behind the proposed approach is to keep on aggregating tuples in an incremental way until a state transition is detected. Probabilistic tuples are processed as they arrive, hence avoiding the use of expensive offline disk based operations, and the output is stored in a probabilistic database in such a way that, as we also experimentally prove, a wide range of probabilistic queries can be applicable and answered effectively.

Keywords: RFID, probabilistic tuple aggregation, location detection

1 Introduction

In the last several years, RFID technology has gained significant popularity due to its ability of detecting objects and people carrying small RFID tags in an environment equipped with RFID readers. RFID applications usually rely on RFID deployments to manage high-level events such as tracking the location that products visit for supply-chain management [4], monitoring the location and status of patients in hospital environment [9], localizing intruders for alerting services [2], and so on.

A fundamental relation for these purposes is the location of people and objects over time. However, the nature of RFID data stream is noisy, redundant and unreliable and thus 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 "Tag 101 entered office 1-10 at 10:00". To this end, a common approach for real-time applications is to use an Hidden Markov Model (HMM) that continuously infers locations based on sensor readings [2]. Such a relation, therefore, is a probabilistic relation `At(tagID,location,time,prob)` that is usually stored



(a) (b) (c)

Fig. 1. (a) A visual representation for John movements; (b) The stream of probabilistic tuples before and after applying the summarization mechanism; (c) Architecture for RFID Probabilistic Data Stream Aggregation System

in a (probabilistic) database table and queried to detect complex events meaningful to applications [13]. An example tuple 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.

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 in the probabilistic database is extremely expensive and, even more important, it is not always useful. For instance, Fig.1 (a,b) depicts one sample scenario, having a total duration of 3 hours and 20 minutes. In Fig.1(a), John, a user wearing an RFID tag that transmits every second, works in his office for three hours. Then, John goes to the coffee room (H2) 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, $32,400$=(60 seconds * 180 minutes * 3 locations) probabilistic tuples are produced for the first three 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.

In this paper, we propose a simple on-line summarization mechanism, which is able to provide small space representation for massive RFID probabilistic data streams while preserving the meaningful information. The mechanism draws inspiration from the field of clustering [8]. The main idea behind the proposed approach is to keep on aggregating tuples until a state transition is detected. This can be seen in Fig.1(b): only one tuple shows John location from 9:00am to 12:00pm i.e. in his office O1. An object or person is said to have state transition if its location changes from one to other, as in Fig.1(b) where John moves from O1 to H1 and consequently to H2. In this case, the proposed summarization

method stores only 3 probabilistic tuples instead of $36,000$=(60 seconds * 200 minutes * 3 locations) probabilistic tuples, while these 3 stored probabilistic tuples give enough information about John's movements. The approach has been implemented in an RFID probabilistic data management system whose architecture is shown in Fig.1 (c). The aggregation module processes probabilistic tuples as they arrive, hence avoiding the use of expensive and offline disk based operations such as sorting and summarization, and promptly stores the output in the probabilistic database MayBMS [7] in such a way that a wide range of probabilistic queries can be applicable and answered effectively.

The rest of the paper is organized as follows. In Section 2, we introduce our aggregation method with its implementation details and different boundary condition tests. Section 3 reports the experimental results. Finally, in Section 4, we briefly discuss about related works and give some concluding remarks.

2 Aggregating tuples

In this section, we describe the details of our on-line aggregation algorithm (see Algorithm 1) that is implemented in the *aggregation module* shown in Fig.1 (c).

Given m tags and n locations, the *RFID processing module* performs inference on an HMM to produce a stream of timestamp ordered probabilistic tuples:[1]

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

where each tuple X_t^T has the form:

$$(T, t, P_{T,t}(L^1), P_{T,t}(L^2), \ldots, P_{T,t}(L^n))$$

and each $P_{T,t}(L^i)$ is a score representing the probability that the considered tag T is in location L^i at time t. This is received in input by the aggregation algorithm that in turn outputs a stream of probabilistic tuples of the form:

$$X_{[t_s,t_e]}^T = (T, t_s, t_e, P_{T,[t_s,t_e]}(L^1), P_{T,[t_s,t_e]}(L^2), \ldots, P_{T,[t_s,t_e]}(L^n))$$

such that:

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

The aggregation algorithm works on the intuition that if a person wearing a tag T is stationary or resides at the same location for a period of time $[t_s, t_e]$, the corresponding probabilistic tuples $X_{t_s}^T, \ldots, X_{t_e}^T$ should show "similar" probability distributions. Therefore, in order to derive $X_{[t_s,t_e]}^T$ it draws inspiration from the large dataset clustering field [5] 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, and for each cluster c_t^T, it

[1] For ease of presentation and without loss of generality, we assume that tuples arrive in tag order. For the same reason, the discrete probability distribution of the location random variable is represented as one tuple instead of n different tuples.

treats the tuple region collectively through some statistics $stat^{c_t^T}$ providing a summarized description for the cluster. When a new tuple X_{t+1}^T arrives, the algorithm tries to add it to the cluster associated to the corresponding tag c_t^T by updating the corresponding $stat^{c_{t+1}^T}$ 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 $stat^{c_{t+1}^T}$ (line 7). On the other hand, if a violation is detected:

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

Algorithm 1 Tuple aggregation algorithm

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

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

2.1 Output tuples

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* [8]. The centroid is the logical center 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. This tuple

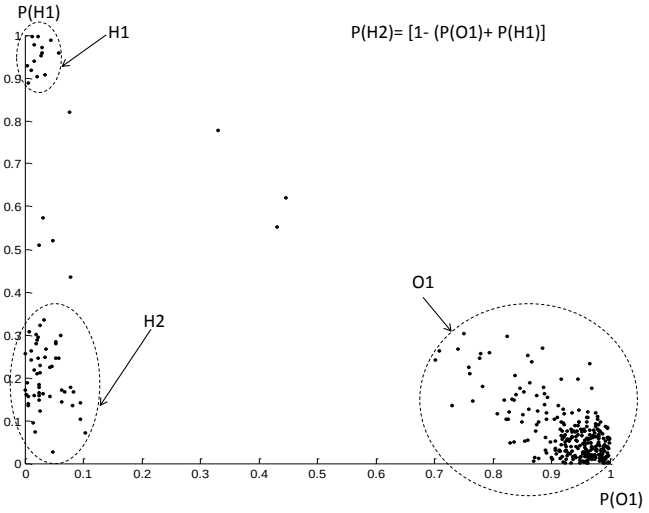


Fig. 2. Cartesian Space representation of the probabilistic tuples of our sample scenario

representation actually exhibits tight clustering as long as the state does not change and a good separation in case of state transition. Fig.2 shows the cartesian plane representation of the sample scenario discussed in Section 1. Since the number of locations is three in this scenario, each tuple generated by the *RFID processing module* is a point in a 3-dimensional space whose coordinates are the probability values for the locations O1, H1 and H2 (the graph shows only the first two dimensions since the third is linearly dependent from the others). We can see that, since John is residing at a same place (his office) for a long period, a large number of points are concentrated in the O1 region; all these points can be aggregated in one point which will be representative of the behavior of all of them. Instead, as John moves from O1 to H1 and consequently to H2, there is a transition that can 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 to denote either a probabilistic tuple $(T, t, P_{T,t}(L^1), P_{T,t}(L^2), \ldots, P_{T,t}(L^n))$ or its representation in the Cartesian space $(P_{T,t}(L^1), P_{T,t}(L^2), \ldots, P_{T,t}(L^n))$.

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

2.2 Boundary conditions

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 [11]. Two measurement criteria are typically used for evaluating a clustering schema [11]: 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 $\overline{L}_{X_t^T}(\overline{L}_{c_t^T})$ be the location with the maximum probability value in $X_t^T(c_t^T)$. For each cluster c_t^T, MPC maintains $\overline{L}_{c_t^T}$ as statistics, and the boundary condition is satisfied when $\overline{L}_{c_t^T} = \overline{L}_{X_{t+1}^T}$. 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: $\max_{X,Y \in c_{t+1}^T}\{d(X,Y)\} \leq B$. 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 $V_{c_t^T}$ and the latest point X_{t+1}^T. To this end, it checks whether $d(V_{c_t^T}, X_{t+1}^T) \leq B$. The main advantage of this method w.r.t. the DM model is that computations are less time and space consuming, as $V_{c_t^T}$ can be computed incrementally.

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

3 Experimental Evaluation

In order to evaluate the performance of the presented approach, we have conducted several experiments in different 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 Processing 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

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

which actually took place in the scenarios (Section 3.1); and (ii), to evaluate the best performing method on a possible target application, 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 3.2).

3.1 Effectiveness of Aggregation Methods

In this subsection, we analyze the performance of the presented aggregation methods by means of five experiments conducted on different movement scenarios 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) percentage 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

Table 2. Probabilistic Query Results for Aggregated and Non-Aggregated 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	*	*

space required by the aggregated tuples (the smaller the percentage 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). Moreover, 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 what we really think is crucial in this evaluation, i.e. 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).

From the obtained experimental results (right part of Table 1 (a, b, c)), we found 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 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).

3.2 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 section we will exploit the CLRC method, since it has been proven the best performing one (see Section 3.1).

As mentioned earlier, the output of the *RFID Processing Module* is a probabilistic stream of tuples where we keep the probabilities of each object being on a specific location at a specific instant. In order to handle the uncertainty associated to these probabilistic streams, we use the MayBMS database management system [7] and validate the results obtained on the aggregated and complete data over a number of queries. 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.

In the following, we discuss six of the most significant queries, named Q1 to Q6, that we used in the tests. For each query, we will show its plain text form, its MayBMS (SQL) form shown in Fig.3, and discuss the obtained results as 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. The MayBMS SQL code of each query is shown in Fig. 3.

Q1. *"Find who was at loacation 'L1' 10 seconds ago?"* Note that `conf()` is the MayBMS function for calculating the confidence of the answer. Further, in

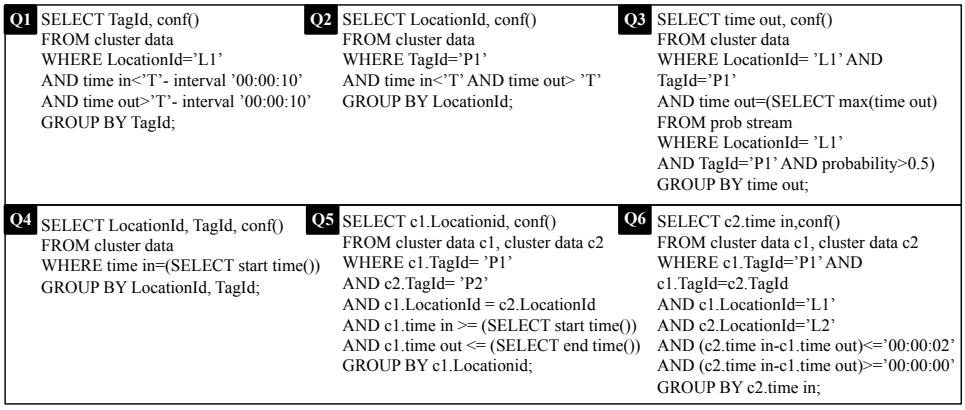


Fig. 3. MayBMS (SQL) form of selected Queries

some of the experiments (EXP1, EXP4 and EXP5) the actual answer to this query should be "no one" ("-" in Table 2). 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.

Q2. *"Find where was person 'P1' at time 'T'?"* Again, all the answers on the aggregated data are correct. Moreover, from this and some of the following queries we can see that the confidence of the correct answer is higher on the summarized data, due to the noise that is present in the non-summarized data.

Q3. *"Find when 'P1' was seen last time at location 'L1'?"* Note that `start_time()` is a user-defined function for retrieving the startup time of the used data set.

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

Q5. *"Whether it happened that two persons are together at the same location at the same time? Where?"* Note that this queryis not applicable to EXP1 and EXP2, since only one tag is used.

Q6. *"Find when 'P1' moved from location 'L1' to 'L2'?"* This is an interesting case involving transition detection between two 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" producing a very large quantity of irrelevant and/or incorrect results ("*" in Table 2).

4 Related Works and Concluding Remarks

The efficient management of RFID data involves a large number of issues in a wide range of applications. One of the main concerns for data management is that the rate of RFID data streams is quite fast and, therefore, the resulting volume of the stream is quite huge. For these reasons, clustering becomes one

of the more challenging tasks to perform. While, to the authors' knowledge, no specific works exist on RFID data stream aggregation, 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 [3], CURE [5] and BIRCH [16].

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 [10]. 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 [14], 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 [12], 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 [1] 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 [15], named as LuMicro, and PWStream [6], which has been proposed for the specific problem of sliding windows.

The objective of most of the methods discussed above is to analyze the 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, our methods are targeted for a different objective, i.e. a summarization task in a location tracking context, and are 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 subsequent modules 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.

References

1. Aggarwal, C., Yu, P.: A framework for clustering uncertain data streams. In: Proceedings of the 24th international conference on Data Engineering. pp. 150–159. IEEE (2008)
2. Cucchiara, R., Fornaciari, M., Haider, R., Mandreoli, F., Martoglia, R., Prati, A., Sassatelli, S.: 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)
3. Ester, M., Kriegel, H., Sander, J., Xu, X.: 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. vol. 1996, pp. 226–231. Portland: AAAI Press (1996)
4. Gonzalez, H., Han, J., Li, X., Klabjan, D.: Warehousing and analyzing massive RFID data sets. In: 22nd International Conference on Data Engineering, ICDE'06. IEEE Computer Society (2006)
5. Guha, S., Rastogi, R., Shim, K.: CURE: an efficient clustering algorithm for large databases. In: ACM SIGMOD Record. vol. 27, pp. 73–84. ACM (1998)
6. Hu, W.C., Cheng, Z.L.: Clustering algorithm for probabilistic data streams over sliding window. In: Proceedings of the 9th International Conference on Machine Learning and Cybernetics (ICMLC). pp. 2065–2070. IEEE (2010)
7. Huang, J., Antova, L., Koch, C., Olteanu, D.: MayBMS: a probabilistic database management system. In: Proceedings of the 35th SIGMOD international conference on Management of data. pp. 1071–1074. ACM (2009)
8. Jain, A.K., Murty, M.N., Flynn, P.J.: Data clustering: a review. ACM Compututing Survey 31, 264–323 (September 1999)
9. Kim, D., Kim, J., Kim, S., Yoo, S.: 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. pp. 1459–1461. IEEE (2008)
10. Kriegel, H., Pfeifle, M.: Density-based clustering of uncertain data. In: Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining. pp. 672–677. ACM (2005)
11. Legány, C., Juhász, S., Babos, A.: Cluster validity measurement techniques. In: Proceedings of the 5th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering and Data Bases. pp. 388–393 (2006)
12. Ngai, W., Kao, B., Chui, C., Cheng, R., Chau, M., Yip, K.: Efficient clustering of uncertain data. In: Proceedings of the 6th International Conference on Data Mining(ICDM),. pp. 436–445. IEEE (2006)
13. Ré, C., Letchner, J., Balazinksa, M., Suciu, D.: Event queries on correlated probabilistic streams. In: Proceedings of the ACM SIGMOD international conference on Management of data. pp. 715–728 (2008)
14. Xu, H., Li, G.: Density-based probabilistic clustering of uncertain data. In: Proceedings of International Conference on Computer Science and Software Engineering. pp. 474–477. IEEE (2008)
15. Zhang, C., Gao, M., Zhou, A.: Tracking high quality clusters over uncertain data streams. In: 25th International Conference on Data Engineering, ICDE'09. pp. 1641–1648. IEEE (2009)
16. Zhang, T., Ramakrishnan, R., Livny, M.: BIRCH: an efficient data clustering method for very large databases. In: ACM SIGMOD Record. vol. 25, pp. 103–114. ACM (1996)