Simplified clustering algorithms for RFID networks

Vinay Deolalikar, Malena Mesarina, John Recker, Salil Pradhan
HP Laboratories Palo Alto
HPL-2005-163
September 16, 2005*

clustering, RFID, sensors

The problem of clustering in networks of RFID readers in particular and active sensors in general is addressed using techniques from sensor data fusion. Two algorithms are provided that are simple and do not need extensive computation. For smaller networks these algorithms can be implemented without any computational assistance. The constraints these algorithms place on the time scheduling of readers in the network are discussed.

* Internal Accession Date Only
Approved for External Publication

Simplified clustering algorithms for RFID networks

Vinay Deolalikar[1], Malena Mesarina[2], John Recker[2], and Salil Pradhan[2]

[1] Information Theory Research Group
vinayd@hpl.hp.com
[2] Mobile and Multimedia Systems Labs
{mesarina, jrecker, salil}@hpl.hp.com
Hewlett-Packard Labs
Palo Alto CA 94304

Abstract. The problem of clustering in networks of RFID readers in particular and active sensors in general is addressed using techniques from sensor data fusion. Two algorithms are provided that are simple and do not need extensive computation. For smaller networks these algorithms can be implemented without any computational assistance. The constraints these algorithms place on the time scheduling of readers in the network are discussed.

1 Introduction

The development and deployment of Radio Frequency Identification (RFID) has generated several problems that need to be solved in order to operate larger networks of RFID readers in an optimal fashion. Perhaps the two foremost problems in this regard are the clustering and the scheduling problems for networks of RFID readers. Briefly, the clustering problem asks the question - how should one combine the outputs of the various RFID readers so as to obtain the best estimate of any event. The scheduling problem asks the question - what is the optimal time sequence for firing the various RFID readers in the network. In this paper, we focus on the clustering problem, although we do discuss the interplay of its solution with the scheduling algorithm.

Both these problems are rendered nontrivial because of two major types of interactions between the readers. The first is correlation. Readings of various readers are correlated not just to the event, but to each other. The second interaction is collision. Readers lying in each other's field can collide with each other, resulting in faulty or missing reads. The challenge then is to optimize the functioning of the network across both these interactions.

2 The clustering problem

Clustering is a major challenge in the optimal functioning of RFID networks. Roughly, clustering is the problem of finding, given certain constraints, the set

[1] Contact author.

of sensors whose outputs can be combined to provide the best estimate of the event that is being sensed. If there were no constraint, then clearly we could do no better than use the outputs of every sensor to estimate each event. But often we operate under a set of constraints that forces us to choose a relatively smaller subset of all the sensors and then combine their outputs to obtain our estimate.

The clustering problem has received attention in various fields. There are several algorithms proposed to solve it. We provide two simplified algorithms that need almost no computation, yet provide good clusters in many practical applications. Thus the emphasis of this article is on providing easy to use clustering algorithms based on sound principles of data analysis that can be used in concrete application scenarios.

Let us begin by examining a formalization of the clustering problem for networks of generic sensors. We are given a set of sensor outputs $\{S_i\}_{i=1,\ldots,n}$ and a cost function f of n arguments, each of which corresponds to a sensor. All of the sensors give us information about an event E. Formally, for each sensor, we are given the correlation $corrS_i, E$ with the event to be estimated. The clustering problem then is to identify the subset of $\{S_i\}_{i=1,\ldots,n}$ that provides the best estimate $\widehat{E}$ of E under certain constraints on f. Specifically, we wish to maximize the correlation $corr(E, \widehat{E})$. It is the cost constraint that makes the problem non-trivial. Without any constraint, clearly we could do no better than choose the entire set $\{S_i\}_{i=1,\ldots,n}$.

In order to approach networks of RFID readers in the framework above, we must fix a notion of correlation could in the context of RFID readers. Things are somewhat complicated because we are dealing with sets of readings here, and not just one random variable. We provide one such notion of correlation. Assume that the readers in the RFID network all have the same capacity for reading tags. We pass several test boxes of tags, each of which has more tags than the capacity of the readers, through the network. We record for each box

1. the fraction of tags present in the box that is read by each reader
2. the fraction of tags read in common by each pair of readers.

The mean values of Steps 1 and 2 provide us with numerical quantities that lie in $[0, 1]$ and satisfy all the properties of correlations. The averages of readings in step 1 will provide us with the correlations between each reader and the box of tags, while those of step 2 will provide us with the cross correlations between each pair of readers.

Finally, the ratio of the number of tags read by the readers in the cluster to the actual number of tags present in the box will provide us with the value of $corr(E, \widehat{E})$.

2.1 A Min-Max algorithm for clustering

The basic intuition for this algorithms is that if two sensors give a similar view of an event, then picking one of these readings is sufficient. On the other hand, if two sensors give a very different view of the event, then it is important to get

both readings in order to have a good description of the event. It just remains to formalize this intuition using the values of correlations defined in the previous section. We do this as follows. Given event E and sensors $\{S_i\}_{i=1,\ldots,n}$, include S_j in the cluster if

1. $corr(S_j, E)$ is high and
2. $corr(S_i, S_j)$ is low for all sensors $S_i,\ i \neq j$ that lie in the cluster.

Viewed in this manner, clustering becomes a Min-Max problem that seeks to find subset of $\{S_i\}_{i=1,\ldots,n}$ such that reader S_i lies in this subset if $corr(Si, E)$ is high and $corr(S_i, S_j)$ is low for $S_j,\ i \neq j$ in the subset. In other words, we want to find clusters with the property that correlations within a cluster are minimized, but correlations intracluster are maximized. There are sophisticated algorithms in data analysis that perform this min-max for data sets. However, the algorithm we provide below is considerably simpler, and still retains the basic skeleton of these more sophisticated algorithms.

We provide our algorithm for multiple readers as a recursive algorithm with two performance parameters that will quantify the notions of high and low in the discussion above.

Algorithm 1 Min-Max sensor clustering algorithm

1: Order $\{S_i\}_{i=1,\ldots,n}$ in descending order of correlation with event.
2: Pick two thresholds $0 < \gamma, \delta < 1$. These will be the performance parameters.
3: From remaining sensors in $\{S_i\}_{i=1,\ldots,n}$, delete sensors whose cross-correlation with sensor picked in Step 1 exceeds γ and correlation with event is below δ.
4: Goto Step 1.

The stress here is on simplicity. As one can see, this algorithm can be performed by hand for small networks.

2.2 An eigen-analysis based algorithm for clustering

Next we provide another simple algorithm to perform clustering based on an eigenvalue analysis of the correlation matrix of the sensors.

First we compute the cross-correlation matrix R of the set of readers $\{S_i\}_{i=1,\ldots,n}$ as follows. Cross correlation between two readers is modelled simply as the fraction of tags that are read by *both* the readers. We can extend this to correlation between any number of readers similarly.

$$R = \begin{pmatrix} corr(S_1, S_1) & \ldots & corr(S_1, S_n) \\ \vdots & \vdots & \vdots \\ corr(S_n, S_1) & \ldots & corr(S_n, S_n) \end{pmatrix}$$

The matrix R is symmetric, invertible and positive definite. It follows that all its eigenvalues $\lambda_1, \ldots, \lambda_n$ are positive. R is diagnonalizable. Let T be the diagonalizing matrix. In other words $R' = TRT^{-1} = \text{Diag}(\lambda_1, \ldots, \lambda_n)$ is a diagonal matrix. Without loss of generality, let us assume that $\lambda_1 > \ldots > \lambda_n$.

T is thus a linear transformation that transforms the input sensor set $\{S_i\}_{i=1,\ldots,n}$ into a set of virtual sensors $\{S'_i\}_{i=1,\ldots,n}$. The virtual sensors $\{S'_i\}_{i=1,\ldots,n}$ are pairwise uncorrelated. From this it follows that the correlation matrix R' of the set of virtual sensors set is diagonal. If we arrange the diagonal eigenvalue matrix elements in the order of magnitude, the virtual sensor corresponding to the highest eigenvalue is the virtual sensor with the most information. Given our definition of correlation between a reader and the event, this implies that this virtual sensor will read more tags than the other virtual readers. This is a very useful piece of knowledge.

This knowledge immediately leads to an algorithm for selecting sensors for the cluster. Based on the performance that is desired, we pick a threshold and select the virtual sensors from the set $\{S'_i\}_{i=1,\ldots,n}$ whose corresponding eigenvalues are larger than the chosen threshold. The tranformation T then specifies the required sensors in terms of the original set of sensors set $\{S_i\}_{i=1,\ldots,n}$. These then form the desired cluster.

Algorithm 2 Eigenvalues based clustering algorithm

1: Transform set $\{S_i\}_{i=1,\ldots,n}$ into uncorrelated set $\{S'_i\}_{i=1,\ldots,n}$
2: Let $\lambda_1 > \ldots > \lambda_n$ be corresponding eigenvalues (reorder if necessary)
3: Use performance measure to pick threshold κ
4: Discard sensors from $\{S'_i\}_{i=1,\ldots,n}$ whose correlation is less than κ
5: Transform the remaining sensors from $\{S'_i\}_{i=1,\ldots,n}$ into their corresponding original set $\{S_i\}_{i=1,\ldots,n}$.
6: This set forms the desired cluster.

3 Interplay between clustering and scheduling

In this section we explore the scheduling constraints imposed by the clustering algorithms of the previous section. First we start with the observation that the algorithms provided in this article do not take into account the collision between readers. They are only concerned with the correlations between the various readers. Collision imposes its own constraints on the scheduling of readers in RFID networks. For more details on the collision problem, the reader is referred to [1]. While using the algorithms provided in this article therefore, we must also take into consideration the extra constraints posed by collision avoidance.

While the Min-Max algorithm provides clusters directly as a subset of original set of sensors $\{S_i\}_{i=1,\ldots,n}$, the eigenvalue based algorithm yields clusters of virtual sensors $\{S'_i\}$. Each of these virtual sensors is a linear combination of the sensors $\{S_i\}_{i=1,\ldots,n}$. The sensors that appear with a non-zero coefficient in the expression for virtual sensor S'_i will be said to be the *support* of the S'_i. We now state the constraint posed by a choice of virtual sensors in a cluster.

Proposition 1. *For each virtual sensor that appears in a cluster, the real sensors that lie in its support must not fire simultaneously.*

Proof. We eventually seek to combine the readings of the sensors in a virtual sensor to gain information about a particular event of interest. There are two cases to consider. If the readers collide, then they should not read simultaneously. On the other hand if they do not collide, then their fields do not overlap and therefore it is not possible for the event of interest to lie in all their fields simultaneously. In either case, firing all the sensors simultaneously is counter-productive. □

In the first case of the proof above, the clustering algorithm does not place any additional constraint over collision avoidance. But in the second case it does. As an example, if the virtual sensor $S_1' = S_1 - 2S_2 + S_5$ then the real sensors S_1, S_2 and S_5 must not fire simultaneously. This is significant because it may be the case that S_1, S_2 and S_5 have no collision. Hence the graphical algorithms to design scheduling schemes presented in [1] would not preclude their simultaneous firing. However the additional constraint imposed by the clustering algorithm does preclude this. The net result is that the time cycle for the completion of a firing sequence increases.

Coming back to the partitioning of the RFID readers into partitions, we then see that a cluster for a given event would have to be chosen from readers such that no two are in a single partition. In other words, a cluster would be chosen from across partitions. One strategy would be to choose a cluster such that it has exactly one reader from each partition. This idea of choosing a cluster across partitions is illustrated in Fig 1.

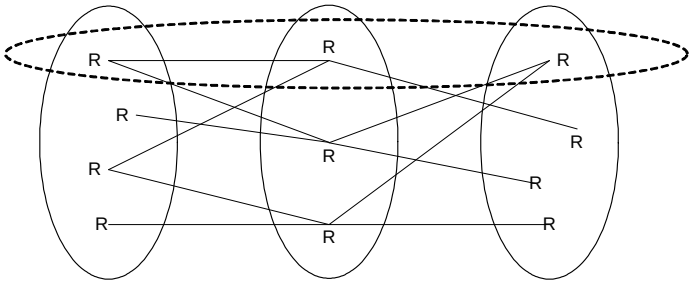

Fig. 1. Tripartite graph, edges denote reader collision

The question that immediately poses itself then is: do we allow clusters to overlap? In other words, can the same RFID reader represent its partition in two different clusters? When would one prefer one approach over the other? The two scenarios are depicted in Figures 2 and 3.

In our experience with deploying such systems, we have found several cases where the application demands the choice of one over the other. Overlapping clusters might be preferred in a situation where some significant feature of the sensors motivates such a scenario. One such case would be when a high power reader is shared with lower powered local readers. Another case would be when

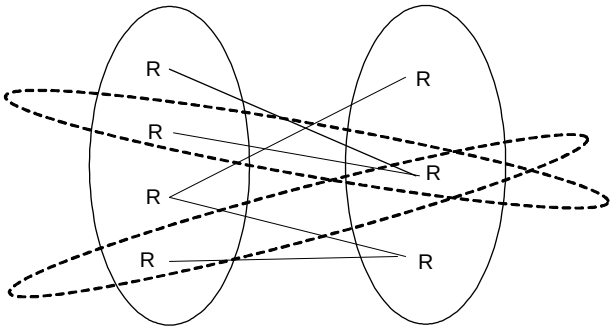

Fig. 2. Bipartite system with overlapping clusters. Overlaps are advantageous in certain scenarios.

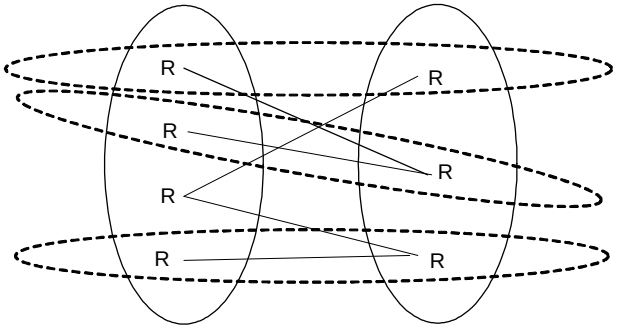

Fig. 3. Bipartite system with non-overlapping clusters. This eliminates single points of multiple cluster failure

a clustered sensor is a multi-sensor device, for example a camera/RFID reader combination.

In other cases, overlaps are not desirable. This is particularly the case where we do not want any single point of multiple cluster failure, as may happen in critical real time applications. This brings us to the question of whether a non-overlapping cluster exists at all. To answer this question, we need the notion of a perfect matching.

Definition 1. *A perfect matching in a bipartite graph* $G = (V_1 \cup V_2, E)$ *is an injective mapping* $f : V_2 \to V_1$ *such that for every* $x \in V_2$ *there is an edge* $e \in E$ *with endpoints* x *and* $f(x)$.

For any subset $A \subset V_2$, define ΔA to be the set of all vertices $y \in V_1$ that are endpoints of edges with one endpoint in A. Now the answer to our question is provided by Hall's perfect matching theorem. Intuitively, it states that the obviously necessary condition for a perfect matching to exist also turns out to be sufficient.

Theorem 1. *Let* $G = (V_1 \cup V_2, E)$ *be a bipartite graph. There exists a perfect matching* $f : V_2 \to V_1$ *if and only if for every subset* A *of* V_2, $|\Delta A| \geq |A|$.

There exist low complexity polynomial time algorithms that find a perfect matching in a bipartite graph. Most of these are variants of what is known as the "Hungarian algorithm" and can be found in most standard books on graph algorithms, see for instance [3].

4 Conclusion

We have written this paper for field engineers who are actually deploying RFID reader networks and need simple algorithms to perform initial testing or even final configuration. Thus the accent is on simplicity of the algorithms and the preclusion of any extensive computation. The algorithms can be implemented without the assistance of any sophisticated software, and in smaller networks can even be implemented by hand.

5 Acknowledgment

We thank Devaraj Das, Christophe Gouguenheim, and Mehrban Jam for useful discussions.

References

1. V. Deolalikar, M. Mesarina, J. Recker, D. Das, and S. Pradhan Perturbative time and frequency allocations for RFID reader networks, preprint 2005.
2. V. Deolalikar, M. Mesarina, J. Recker, and S. Pradhan Optimal scheduling for networks of RFID readers, preprint 2005.

3. J. Dossey, A. Otto, L. Spence, and C. Eynden, Discrete Mathematics, 3rd ed. Reading, MA, Addison-Wesley, 1997.
4. D. Engels, The Reader Collision Problem, Technical Report, available at http://www.autoidcenter.org/research/MIT-AUTOID-WH-007.pdf
5. F. Harary, Graph Theory, Addison-Wesley, 1969.