IJMMS, *Vol. 13, No. 2, (July-December 2017) : 345-356*

ISSN: 0973-3329

SEQUENCING OF RFID TAG MOVEMENTS IN SUPPLY CHAIN USING DYNAMIC PROGRAMMING TECHNIQUES

***Sabu Augustine*[*], *M. S. Samuel*[**] *and Sajimon Abraham*[***]**

Abstract: *Radio Frequency Identification (RFID) has played a pivotal role in developing technologies for supply chain managements. Over the years, tremendous amount of research inputs has been invested to develop novel technologies to improve the efficiency of product tracking and supply. Due to wide availability of RFID tags at reduced cost, this technology has been widely spread in industrial applications like tracking of product movements in supply chain. This technological development opens up new avenues of tracking the objects movements in order to analyze pattern of movement. As this huge amount of data generated from RFID devices, managing and finding useful pattern from this data of large size become a challenging exercise. We address this issue by proposing suitable mathematical model using path encoding scheme, Unique factorization theorem, Chinese remainder theorem and Euler's algorithm. The paper also discusses the similarity based data analysis of RFID tags movements in over the proposed data structure by analyzing the sequence similarity metrics obtained by the dynamic programming techniques.*

1. INTRODUCTION

Radio Frequency Identification (RFID) is a technology [1,2] widely spread in industrial applications like tracking of product movements in supply chain, due to wide availability of RFID tags at reduced cost. The manufacturer's objective in Supply Chain Management [3] is to analyze product and logistic information in order to ensure that the optimum quantity of products arriving at the proper time to the right destinations. However there are many challenges that RFID data management requires as it has to record data from multiple RFID readers, which comes data in peta bytes.

RFID is the reading of physical tags on single products, cases, pallets, or re-usable containers which emit radio signals to be picked up by reader devices like transponder and an antenna. These devices and software must be supported by a sophisticated software architecture that enables the collection and distribution of location-based information. The data collected by RFID devices are automatically

[*] Department of Mathematics, Marian College, Kuttikkanam, India, *E-mail: sabaug@rediffmail.com*

[**] Department of Computer Applications, MACFAST, Tiruvalla, India, *E-mail: ktymmssamuel@gmail.com*

[***] School of Management & Business Studies, M.G University, Kottayam, India, *E-mail: sajimabraham@rediffmail.com*

updated with global standardized databases, ensuring real time access to up to date information about relevant products at any point which will add benefit to the supply chain system. Tags contain a unique identification number called an Electronic Product Code (EPC), and potentially additional information of interest to manufacturers, healthcare organizations, military organizations, logistics providers and retailers, or others that need to track the physical location of goods or equipment [3]. All information stored on RFID tags accompanies items as they travel through a supply chain or other business process. All information on RFID tags, such as product attributes, physical dimensions, prices or laundering requirements, can be scanned wirelessly by a reader at high speed and from a distance of several meters.

2. RFID DATABASE MODELING

We design an RFID database model and implement an extensible storage system for RFID data to support further searching and extracting hidden knowledge. Basically an RFID system [4] consists of three components like Radio frequency tag, antenna and receiver. A tag T in a location reflects RF signals within its allowed detection region. When the product that carry the tag detected by the antenna, the reader understand the signal and stores the EPC and the current timestamp into the RFID database. We represent paths, tags, times and other information in the model and develop path coding schemes for the movement of T in a stored data model. We also incorporate the internal hierarchies and relevant product information into the product data model and the logistic hierarchy data model respectively, which are depicted in Figure 1 based on the data model proposed in paper [4]. This serves as a foundation for the database implementation.

To explain the database model in Figure 1, we start by assuming a simple model of RFID raw data in SCM. An RFID reader detects a tag and generates *Raw Data Record* having three attributes (EPC, LOC, TIME) where EPC is a unique ID of a tag and LOC is the location of the (unique) RFID reader and TIME is the time instant of detecting the tag. We then collapse the raw data into the *Stay Record* entity having four attributes (tag id, loc code, time in, time out) where time in and time out are the time instants of the first detection and the final detection of the tag.

A stay record represents an RFID tag moving through a location during a specific time interval. However, using a stay record as a stored data model is too simplistic to support the evaluation of path queries that track objects, particularly for those path queries involving many locations, which need to perform self-joining of the table many times.

Thus, we develop various sophisticated coding techniques and include *Path Record* and *Path Code* entities into the model.

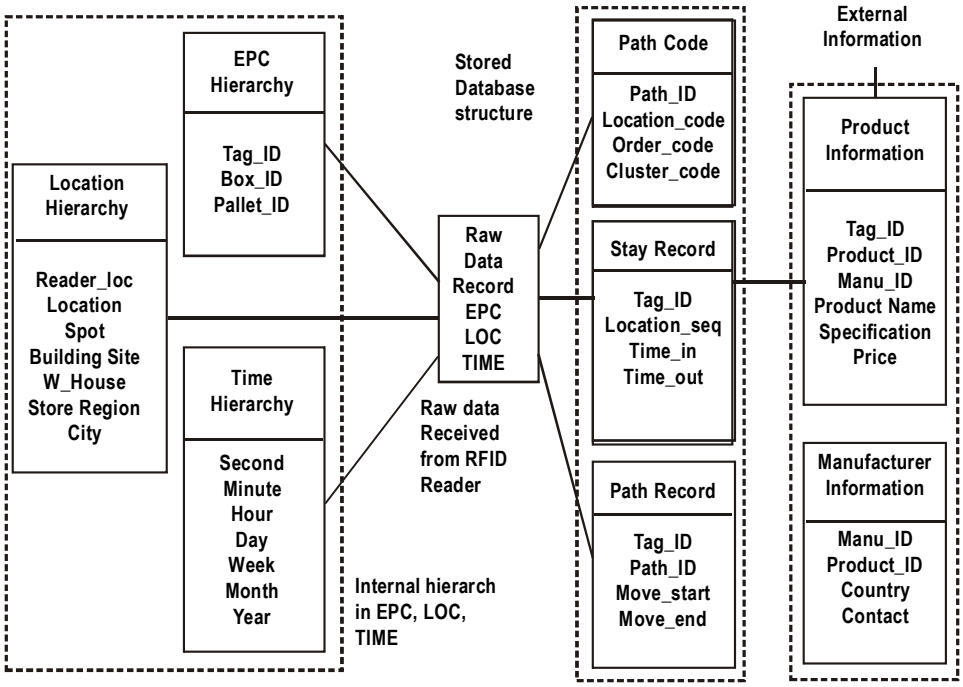

Figure 1: RFID Database Model with Stored data structure and External data

2.1. Using Path Encoding Scheme

The basic idea of the path encoding scheme is that a tag movement path "L1 → L2 → · · · → Ln" can be coded by assigning a unique prime number, Pl, to represent all locations (or all readers) and another prime number, Po, to represent the location order. To generate a unique integer pair (Pl, Po), we rely on the *Unique Factorization Theorem* for coding locations and the *Chinese Remainder Theorem* for coding their order. However, due to the scarcity of prime numbers, the state-of-the-art method in [3] which supports neither long path coding (e.g. encode a long path of more than 8 locations) nor cyclic path coding (e.g. encode a path in which a tag passed a location twice). The first problem is due to the fact that most programming languages use unsigned integers (32 bits)that only support 2^{32}•1, which is less than the product of the first nine prime numbers $2 \times 3 \times 5 \times 7 \times 11 \times 13 \times 17 \times 19 \times 23$. Even for 64 bit unsigned integers (2^{64}-1),it can only support the first 15 prime numbers. In addition, suppose a tag goes through the path "L3 → L5 → L7 → L3 → L5 → L7". In this case, the coding method in [3]fails, since the system of simultaneous *congruences* of Chinese remainder theorem is not applicable here and thus (Pl, Po) fails to code 3→5→7→3→5→7.

The following section summarizes how we approach the two problems discussed earlier

2.2. Problem with long path

We quote the following theorems from theory of numbers [4] in addressing the above problem.

(a) **Theorem 1: "The Fundamental Theorem of Arithmetic:** Any natural number greater than 1 is uniquely expressed by the product of prime numbers"

(b) ***Theorem 2: The Chinese Remainder Theorem:*** **"***Suppose that* $m_1, m_2 . . .$,and m_k *are pairwise relatively prime numbers. Then, we can find a unique X such that* $0 \leq X < m_1 m_2 . . . m_k$ which solves the following system of simultaneous congruences X mod $m_p \equiv a_p$, $1 \leq p \leq k$"

(c) ***Theorem 3: Euler Formula for Prime Generation:*** **"***For every integer* x *between* 0 *and* 40, $x^2 - x + 41$ *is a prime number"*

To solve the long path coding problem, we first partition the whole set of locations into different clusters having roughly the same number of locations. Using the result of finite continued fraction we are able to represent a cluster code denoted as C (having a unique positive integer as its id) together with its respective (Pl, Po). Notably, the clustering can be straightforwardly extended into more than one level within each cluster, which therefore removes the constraint of having no more than 8 prime numbers for coding a path in a cluster. Suppose there are two clusters coded by two positive integers C1 and C2. The sub path in cluster 1 can be computed as loc_code1 and order code 1 and similarly notations for the sub path in cluster 2. If a path goes from cluster 1 to cluster 2, we generate the

Full path code P [4] as

1/(c1+1/(loc_code1+1/(order_code1+1(c2+1/(loc_code2+1(order_code2))))))

When decoding P, we first check whether it is smaller than 1. If this is the case, then the path covers more than one cluster. We then decompress P to extract loc_code and order code in each cluster.

2.3. Problem with cyclic path

To solve the cyclic path coding problem, we apply Euler's prime number generation formula in Theorem 1. For example, the cyclic path 5→7→11→5→7→11 is coded as "5→7→11→61(x=5) → 83(x=7)→151(x = 11) which can be used to form the system of congruences required by the Chinese remainder theorem(CRT).

We are now ready to present our algorithms to handle long and cyclic paths. For simplicity in presentation, we assume one reader for each location and one level of clustering. We divide the whole set of locations into n clusters, each of which has less than 8 locations. We then code each cluster by an integer and within each cluster a location is coded by a unique prime number np. We now represent a

path with three integers (loc code Pl, order code Po, cluster code Nc) only. Loc code can be computed by using the *Fundamental Theorem of Arithmetic* in Theorem 1, which by definition Pl is the product of all prime numbers associated with the path. Order code exists according to the *CRT* and Po can be computed by Euclid's algorithm [5].

For example, consider $\{m_1 = 3, m_2 = 5, m_3 = 7\}$ and (Po mod 3) ≡ 2, (Po mod 5) ≡ 3, (Po mod 7) ≡ 4, then Po can be computed using Euclidian algorithm as

$$2*-1*35 +3*1*21 + 4*1*15 = 53$$

If order code Po congruent to the same location twice, we cannot generate Po using CRT. In such cases we need to assign more than one prime number to those repeatedly visiting locations. To address this problem, firstly, we code each location with a prime number as said and this number is not an *Euler Formula Prime*. We call this set of location prime numbers the *Fundamental Location Set [4]* and denote the set by F. Given that the *Stay Records* can be sorted by time in, if a specific location occurs twice, the first occurrence will be the prime code n from F and then the second occurrence can be coded by applying Euler Formula by putting x = n. The new generated Euler prime numbers do not belong to F.

3. LOCATION SEQUENCING OF RFID TAGS

In the context of market basket analysis in a supermarket each basket is imbedded with an RFID tag, the sequence of these basket movements due to customer preferences can used as a tool for predicting the frequent moving path of the customer. This may be useful for the administrators to re-arrange the locations of related products nearby in order to maximize the sales and also to provide better customer satisfaction.

In order to find out the most similar basket movements based on location sequence alignment of RFID tag attached, we consider a known sequence by assuming a possible set of locations sequence where more customers preferred in a heuristic way. Then we make a query sample which consists of all such locations selected and then this query tag movement path is being used to compare the sequence alignment of all available customer basket movement paths to find whichever is more similar to the query path in terms sequence alignment.

We represent each RFID tag movement path as a string consisting of sequence of locations. Here length of a path is the number of locations in the string. For two strings of length rand s, optimal sequence alignment has zero or more gaps inserted into the sequence to maximize the number of positions in the aligned strings that match. Dynamic programming techniques [[74]Gunduz 2003,[75]Weinan 2002] is used in finding out the optimal sequence alignment which is being discussed in the following section.

3.1. Sequence Alignment

Sequence alignment is one of the fundamental operations performed in computational biology research. It is at the heart of the Human Genome Project, where sequences are compared to gather evidence for a common function or biological origin. The goal is to produce the best alignment for a pair of DNA or protein sequences (represented as string of characters).

Generally there are two popular methods for sequence alignment such as global alignments and local alignments. Global alignment computation is the form of global optimization that align the entire length of all query sequences. On the other hand, local alignments divide long sequences into widely divergent sub sequences and then identifying the regions of similarity. The practical difficulty with local alignments is the complex calculation involved in finding out sub region which are divergent in nature. A number of computational algorithms are available to the sequence alignment problem. These include slow but formally correct methods like dynamic programming and, heuristic algorithms or probabilistic methods designed for large-scale database search, that do not guarantee to find best matches. In this paper we have chosen the dynamic programming as the most accurate method for sequence alignment[8].

3.2. Dynamic Programming for Sequence Alignment

Dynamic Programming is a mathematical technique well suited for the optimization of multistage decision problems

Iterative Methodology

A good alignment has zero or more gaps inserted into the sequences to maximize the number of positions in the aligned strings that match. For example, consider aligning the sequences “ATTGGCT” and “AGGAC”. By inserting gaps (“-”) in the appropriate place, the number of positions where the two sequences agree can be maximized:

ATTGG-CT

A—GGAC—

Here, the aligned sequences match in four positions. Algorithms for efficiently solving this type of problem are well known and are based on dynamic programming [[74]Gunduz2003]. Aligning the sequences “ATTGGC” and “AGGAC” reduces to finding the maximum cost path through an array of size m + 1 and n + 1 (m = 5, n = 6, adding an extra row and column to include the gap).

We use a scoring system which helps to find the optimal matching between two trajectory sequences generated from RFID tag movement. An optimal matching is an alignment with the highest score. The score for the optimal matching is then

used to calculate the sequence similarity between two tag movement path obtained from market basket movements. The various principles followed in matching the sequences are shown below.

(i) The location sequences can be shifted right or left to align as many pages as possible.

For example, path1 includes a sequence of locations L_1, L_2, L_3, L_4, L_5 Here each location is represented by its corresponding bit string as described previously. Similarly another path path2 assumes a sequence of locations L_2, L_3, L_4 The best matching between the two session sequences can be achieved by shifting path 2:

path 1 : L_1, L_2, L_3, L_4, L_5

path 2 : –, L_2, L_3, L_4, L_5

(ii) For achieving better matching gaps(-) are allowed to be inserted into the beginning, middle, or end of location sequences. For example, for the following two trajectory sessions, a gap in path 3 helps getting the best matching.

path 1 : L_1, L_2, L_3, L_4, L_5

path 3 : L_1, L_2, -, L_4, L_5

(iii) We do not simply count the number of identical locations when we are aligning session sequences. For each pair of locations, the scoring function gives a similarity score where higher score indicates higher similarity between locations. We should have a scoring system to find the optimal matching when we use dynamic programming techniques to compute the similarity between travelling locations. For each identical matching, i.e. a pair of locations with similarity 1.0, the similarity score is 2; for each mismatching, i.e. a pair of locations with similarity 0.0 or match a location with a gap, the similarity score is -1.

Illustration

path 1 = L_1, L_3, L_5, L_3, L_4, L_2, L_6

path 2 = L_1, L_3, L_4, L_2, L_6

The path sequence becomes, after inserting gaps

path 1 = L_1, L_3, L_5, L_4, L_2, L_6

path 2 = L_1, L_3, — , L_4, L_2, L_6

The Figure 2 contains a matrix which shows an example for alignment and finding optimal path and optimal score. The optimal path is shown in the arrows and the score in the bottom right most cell encircled is the optimal score.

	-	L_1	L_3	L_5	L_4	LL L_2	L_6
-	0	-1	-2	-3	-4	-5	-6
L_1	-1	**2**	1	0	-1	-2	-3
L_3	-2	1	**4**	3	2	1	0
L_4	-3	0	3	3	**5**	4	3
L_2	-4	-1	2	2	4	**7**	6
L_6	-5	-2	1	1	3	6	**9**

Figure 2: Optimal path based on Optimum score

One sequence is placed along the top of the matrix and the other sequence is placed along the left side. There is a gap added to the start of each sequence which indicates the starting point of matching. The process of finding the optimal matching between two sequences is actually finding an optimal path from the top left corner to the bottom right corner of the matrix. Any step in any path can only go right, down or diagonal. Every diagonal move corresponds to matching of two locations. A right move corresponds to the insertion of a gap in the vertical sequence and matches a location in the horizontal sequence with a gap in the vertical sequence. A down move corresponds to the insertion of a gap in the horizontal sequence and matches a location in the vertical sequence with a gap in the horizontal sequence. The method for sequence alignment using dynamic programming technique is given in Algorithm 1.

Algorithm 1: DPROG(R_i,R_j)

Finding Optimal Sequence Similarity score for two RFID tag movement path R_i and R_j using Dynamic Programming Techniques

Input: Pair of session sequences which constitutes each movement path R_i,R_j

Output: Sequence Similarity Measure (SSM)

Begin

1. Matrix M is created with m+1 columns and n+1 rows where m and n corresponds to the sizes of R_i and R_j sequences respectively.
2. Align the sequences by providing gap between the sequences so that two sequences can be matched as much as possible.
3. Place sequences R_i in top of Matrix and sequences R_j on left side of the matrix.
4. Assign top left corner of matrix =0

   /*Find optimal path */

```
    For each pair of locations
5.  if location matches
6.          Score = 2
7.      else
8.          Score = -1
9.  endif
10. M cell entry=max(left entry+score of top entry+score above left en
    try+score) /* Find path for every two sequence of locations as follows */
11.  If both the pages are matching
                Diagonal move
    Else if gap on top horizontal sequence
                Right move
    Else if gap on left vertical sequence
                Down move
    End if
End for
12. SSM = value at the lower right corner of the matrix M.
13. Return SSM
    End
```

After finding the score for the optimal session alignment, the similarity between Paths is computed by considering the final optimal score and the length of the two paths. For this, we first get the length of the shorter path referred by the variable ‘‘shorter path’, then the similarity between the two paths is achieved by dividing the optimal matching score by 2* shorter path. Note that the optimal score cannot be more than 2* shorter path in this method. Thus the sequence based similarity measure between these two paths is defined

We can find that $Sim_s(R_i,R_j)$ satisfies the following properties

i) $Sim_s(R_i,R_j)$ is in [0,1]

ii) $Sim_s(R_i,R_j) = Sim_s(R_j,R_i)$

3.3. Sequence Similarity Algorithm

The similarity measurement scheme as detailed in Algorithm 2, enables us to search for those trajectories which are similar to the trajectory of a specific moving object (query trajectory) and hence to create a similarity set. The query trajectory is generated using the given set of locations.

The procedure accepts the 5 parameters as detailed below.

i) SP_{IN} : The set of input tag sequences which are formed from stay records

ii) P_Q : Query path which is formed intuition using given set of locations.

iii) ρ : threshold for sequence similarity measure

iv) SSS is the sequence similarity set obtained after going through all the phases. For each trajectory P_r in SP_{IN} the member unit P_r.s contains the sequence similarity measure.

Algorithm 2: **To Create Sequence Similarity Set based on a given set of locations PathSimilar (SP_{IN}, P_Q, ρ)**

Input: Input user session trajectories SP_{IN}, query trajectory P_Q , spatial similarity threshold δ
Output: Sequence similarity set SSS containing paths similar to query path P_Q

Begin

1. SSS = Empty set
2. n_1 = Number of locations in P_Q
3. *For* each P_r in SP_{IN}
4. n_2=no of locations in P_r
5. shorterpath = smaller of n_1 and n_2
6. pscore= DPROG(P_r,P_q) /* Algorithm 1 */
7. Pr.s = pscore/ (2*shorterpath)
8. *If* Pr.s>δ *then*
9. SSS = SSS U{P_r}
10. *End For*
11. *Return* SSS
12. *End*

4. EXPERIMENTAL EVALUATION

We have used a proximity card data set [9] which consists of thousands of individual proximity card data records each of which record the outcome of when a user presented a proximity card at a reader. One record will be generated when the RFID reader attempts to read a proximity card each time. The format of the raw RFID data recorded is as shown below

Date time time-period Access reader-code tag-id

Here a raw record is a space-separated set of attributes and corresponding data values. For example: 23/08/20106:45:02 PM granted 30210.

This data record specifies that tag number 210 was granted access to doorway controlled by reader 30 on date 23/08/2010 at 6:45:02 PM.

This experimental set up focuses on RFID tag data set taken from above data set. Experiments were taken with query trajectory having 50,100,150 and 200 locations of interest. Scalability is measured in terms of how the search time grows with respect to increase in number of LoI's. The results of experiments confirm that (Figure 3) the average search time of our proposed sequence similarity algorithm

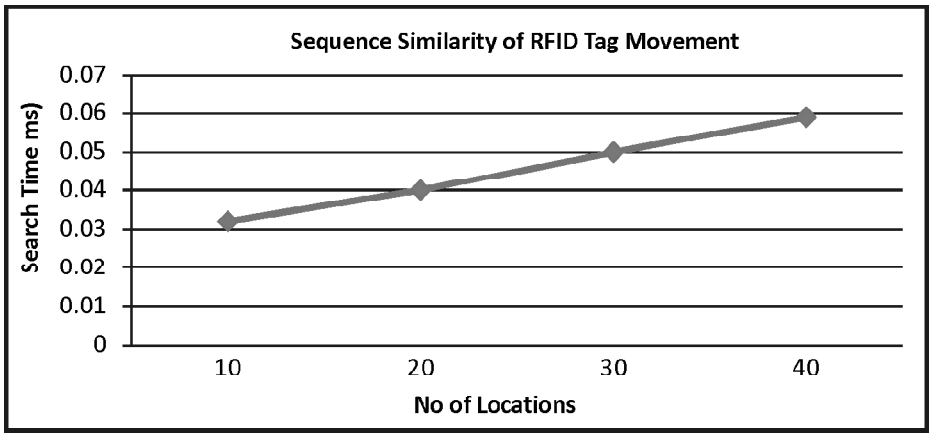

Figure 3: Search time for Proximity Card set

(***Path Similar***) is linearly increasing as the increase in number of LoI's which supports the fact that the algorithm is scalable.

5. CONCLUSION

Organizations, worldwide in this era of globalization are continuously focusing on strategies of improving their competitiveness. Applying RFID technology is one such strategy which has been made revolutionary changes in supply chain domain by providing visibility for companies that want to track movements of physical items from manufacturer to the point of end customer. This paper introduces a model for the optimization of the storage structure of RFID database using theory of numbers and continued fraction. The paper also discusses the sequence similarity of RFID tags movement by using dynamic programming techniques.

References

[1] Bai, F. Wang, P. Liu, C. Zaniolo, and S. Liu. RFID Data Processing with a Data Stream Query Language. In: Proc. of ICDE, 2007.

[2] Palmer M, Principles of Effective RFID data management, Enterprise System, March 2004.

[3] H. Lee, C-W Chung, Efficient Storage Scheme and Query Processing for Supply Chain Management using RFID, In: Proc. of SIGMOD 2008.

[4] Wilfred Ng, Developing RFID Database Models for analyzing Moving Tags in Supply Chain Management, LNCS 6998pp. 204-218, 2011, @springer-Verlag, Berlin Heidelberg 2011.

[5] Hardy, G. H.; Wright, E. M. , An Introduction to the Theory of Numbers, 1979.

[6] R. Sedgewick, P. Flajolet, An Introduction to the Analysis of Algorithms.

[7] Chun-Hee Lee and Chin-Wan Chung, "RFID data processing in Supply Chain Management Using a Path Encoding Scheme", IEEE Transactions on Knowledge and Data Engineering , Vol. 23, No. 5, May 2011.

[8] Sajimon Abraham, Sojan Lal P., 2010. Trajectory similarity of Network Constrained Moving Objects and Applications to Traffic Security, Pacific Asia International Workshop On Security Informatics(PAISI 2010) held in 21 June, Hyderabad, India, LNCS 6122, pp. 31–43, Springer-Verlag Berlin Heidelberg.

[9] Luke.Mirowski, Jacqueline Hartnett, Raymond Williams, Tony Gray, RFID Proximity Card Data Set, Technical report for the School of Computing and Information Systems at the University of Tasmania, Version June 25, 2008.