An Efficient Algorithm for Mining Maximal Sparse Interval from Interval Dataset

Naba Jyoti Sarmah
Dept of Computer Science
Gauhati University

Anjana Kakoti Mahanta
Dept of Computer Science
Gauhati University

ABSTRACT
Many real world data are closely associated with intervals. Mining frequent intervals from such data allows us to group those data depending on some similarity. A few numbers of data mining approaches have been developed to discover frequent intervals from interval datasets. Here we present a complementary approach in which we search for sparse intervals in data. We present an efficient algorithm with a worst case time complexity of O(n log n) for mining maximal sparse intervals.

General Terms
Data Mining, Algorithm.

Keywords
Interval Data, Sparse Interval, Maximal Sparse Interval.

1. INTRODUCTION
Most of the works in data mining focus on characterizing the similarity of data values in large datasets. This includes pattern mining [1], rule mining [2], classification, clustering etc. The data patterns discovered by these techniques are defined by some measure of similarity or by degree of frequency or occurrence. Few works [3][4] have also been carried out for mining rare item-sets, where the methods try to find out those item-sets which are not frequent in general frequency counting methods but have an importance in the datasets. For example in a market basket datasets items like rice cooker and cooking pan are not as frequent as butter and bread since they are brought less frequently then other items but this type of items may reveal some interesting buying patterns. In our proposed work the less frequent intervals are mined from the interval datasets.

Many real world data are associated with time, describing the occurrence time of the event. For such datasets different methods are developed such as temporal pattern mining [5] [6], temporal rule mining etc. Most of the works done in this field assume that the events are associated with a single timestamp. Whereas in real life, there are events which cannot be described by a single timestamp, for example call records of different customers recorded by a telephone company cannot be represented by single timestamps. This is because each record is associated with two timestamps, starting and ending time of the call. Another example can be the records of the user's login and logout time in an online portal. If DBA wants to record these data then DBA must record both starting and ending time of the event. These types of data where two timestamps are associated are referred as interval data.

Only few works have been carried out for interval datasets, mostly for frequency counting of the intervals [10][12] and for mining temporal relational rules of interval-based data. In this paper, we propose a complementary method for characterizing interval datasets; we focus on mining maximal sparse intervals in the datasets.

Sparse intervals refer to those infrequent intervals all of whose sub intervals are infrequent. Maximal sparse intervals refer to those sparse intervals none of whose super intervals is sparse. Here we propose a method for mining maximal sparse intervals from interval datasets.

Clearly, knowledge about sparse intervals valuable, as it may reveal some unknown correlation between data values which can be exploited in applications. As an example, phone call record of a cellular phone company can be considered, where the company records the starting time and ending time of each phone call. Mining sparse intervals from such data will discover those intervals which are not frequent and by using these intervals company can either prepare some plan or some scheme to attract customers to these intervals or company can take decision to use the channel for some other purpose. Another example of such dataset is, web based learning system where the students login and logout the system. Mining sparse intervals from such datasets will allow the system administrator to ask the faculty and research scholars to use those intervals for uploading the study materials and students to download the study material since less number of users are available at those intervals. In the above systems mining sparse intervals can play an important role in improving the performance of the overall system.

In section 2.1 we discuss different related works available in literature. In section 2.2 we introduce the different definitions related to maximal sparse intervals. In section 2.3 we formally introduce the problem of mining maximal sparse intervals. In section 3.1 we discuss the different properties related to maximal sparse intervals and prove them mathematically and in section 3.2 we propose an algorithm for mining maximal sparse intervals. Correctness of the algorithm has been proved in section 3.3. Complexity analysis of the proposed algorithm is given in section 3.4. We had tested the algorithm for real life datasets and for synthetic datasets and results are given in section 4. We also compare the performance of our proposed algorithm with the existing algorithms. Conclusion and future works are given in section 5.

2. RELATED WORKS AND PROBLEM DEFINITION

2.1 Related Works
To our best knowledge Allen [7] first proposed a method for maintaining knowledge about temporal intervals. In the paper the author had proposed 13 possible relationships between two intervals. A method for mining relationships among the intervals also has been proposed in [7]. Kam and Fu [8] proposed an Apriori based algorithm for mining temporal patterns from interval based events based on the Allen's temporal logic. Further Shin-Yi Wu and Yen-Liang Chen [9]

shows that there are some ambiguity issues in Allen's method and proposed a new representation of interval data and proposed a prefix traversal algorithm for mining temporal patterns from interval data.

Lin [10] has proposed a method for mining maximal frequent intervals from interval datasets. He design a data structure called I-Tree, first all the intervals are inserted into the I-Tree and then finds out the maximal frequent intervals from that I-Tree. Our work is basically a complementary work carried out by Lin [10], here we focus on mining maximal sparse intervals; i.e. we are mining those intervals which are not covered by maximal frequent intervals. Further improvement has been made for construction of the I-Tree in [11] where authors proposed a faster method for construction of I-Tree. A new approach for Mining Maximal Frequent interval has been proposed by M. Dutta in [12], where authors proposed a method for Mining Maximal Frequent intervals without constructing the I-Tree.

Elbassioni [13] proposed an algorithm for mining minimal infrequent multidimensional intervals. In this work author uses the concept of lattice theory and proposed a method for mining multidimensional infrequent intervals. In [14] authors have proposed an algorithm for mining minimal infrequent intervals. The authors use the algorithm for mining maximal frequent interval proposed in [10] and [11] and proposed an algorithm for mining minimal infrequent intervals. In this process a lot of unit size intervals are generated, whereas it is not the case for maximal sparse intervals.

In [15] author has proposed a method for mining maximal sparse intervals. In that work authors use the maximal frequent intervals mined by M. Dutta [12] to mine maximal sparse intervals. For this the list of maximal frequent intervals has to be known beforehand and for mining maximal frequent intervals it takes $O(n^2)$ time and mining maximal sparse intervals from the set of maximal frequent intervals takes O(n) time. But our proposed algorithm takes only O(n log n) time for mining maximal sparse intervals. So the proposed algorithm is clearly an improvement over the existing algorithm.

2.2 Preliminary Definitions

In the following section few definitions related to interval data and maximal sparse intervals is given. These definition have been taken from [15], only for shake of completeness it is given here.

Interval: Given a totally ordered discrete domain $\mathbb{D}$ and $\ell, r \in \mathbb{D}$, a subset $[\ell, r]$ of $\mathbb{D}$ defined by $\{z \mid \ell \le z \le r\}$ is called an interval. Here 'ℓ' is called the left endpoint and 'r' is the called the right endpoint of that interval. In our study we consider only intervals which are closed. That is if [a, b] is an interval and 'x' is a point in the interval then $a \le x \le b$.

Transaction: A transaction '$\mathbf{T}$' consists of a transaction id and an interval $[\ell, r]$. The interval $[\ell, r]$ is said to be the interval associated with the transaction $\mathbf{T}$ and is denoted by $\mathbf{I}(\mathbf{T})$.

Dataset: A dataset $\mathbb{ID}$ is a dataset of n transactions, where each transaction '$\mathbf{T}$' contains a non-empty interval which is represented by a pair of left end point and right end point.

Ordering of endpoints: Let us consider $C = \{a_1, a_2, \ldots a_n\}$ be the collection of all end points corresponding to the intervals in the dataset $\mathbb{ID}$. Let ℓ_{min} denote the smallest left end point and r_{max} denote the largest right endpoint. Since the domain $\mathbb{D}$ is totally ordered, the end points can be ordered in the ascending order.

Containment of Interval ($\mathbf{I_1 \subseteq I_2}$): An interval $\mathbf{I}_1$ said to be contained in another interval $\mathbf{I}_2$ if and only if $\ell_2 \le \ell_1 \le r_1 \le r_2$, where $\mathbf{I}_1 = [\ell_1, r_1]$ and $\mathbf{I}_2 = [\ell_2, r_2]$.

Proper Containment of Interval (I1⊂I2): An interval $\mathbf{I}_1$ said to be properly contained in another interval $\mathbf{I}_2$ if and only if $\ell_2 < \ell_1 \le r_1 \le r_2$ or $\ell_2 \le \ell_1 \le r_1 < r_2$ or $\ell_2 < \ell_1 \le r_1 < r_2$, where $\mathbf{I}_1 = [\ell_1, r_1]$ and $\mathbf{I}_2 = [\ell_2, r_2]$.

Support of an Interval: A transaction '$\mathbf{T}$' supports an interval [a, b], if $[a, b] \subseteq \mathbf{I}(\mathbf{T})$ i.e. $\ell_t \le a \le b \le r_t$ if $\mathbf{I}(\mathbf{T}) = [\ell_t, r_t]$. For a given interval [a, b], sup([a, b]) will denote the number of transactions in $\mathbb{ID}$ that supports [a, b].

Support of a point: Support of any point 'x', where $\ell_{min} \le x \le r_{max}$ is the support of the interval [x, x] in the dataset and the point is called frequent if the support of the interval [x, x] is greater than equal to min_sup.

Frequent Interval: For a given support threshold min_sup with $0 < \text{min\_sup} \le n$ (where $n = |\mathbb{ID}|$), an interval is called frequent if its support is greater than equal to min_sup. Obviously if $[\ell, r]$ is frequent, then $\ell_{min} \le \ell \le r \le r_{max}$.

Infrequent Interval: An interval $[\ell, r]$ will be called infrequent if $\ell_{min} \le \ell \le r \le r_{max}$ and it is not frequent.

Sparse Interval: A sparse interval is an infrequent interval which does not properly contain any frequent interval except the empty intervals which is always considered as frequent, i.e. if an interval $[\ell, r]$ is a sparse interval and $[\ell', r'] \subset [\ell, r]$ where $\ell' \le r'$, then $[\ell', r']$ is infrequent.

Maximal Sparse interval: A maximal sparse interval is a sparse interval which is not contained in any sparse interval, i.e. if an interval $[\ell, r]$ is a maximal sparse interval and $[\ell, r] \subset [\ell', r']$ then $[\ell', r']$ is not a sparse interval.

2.3 Problem Definition

Let ID be the dataset of records or transactions, where each record contains an interval I = [a,b] and if 'x' is a point in the interval [a,b] then $a \le x \le b$. Here we assume that the domain of end points is discrete. The problem is to mine the maximal sparse intervals from the dataset for a given minimum support threshold value.

Let us consider the following example

Table 1: Interval Dataset

TID	1	2	3	4	5
Intervals	[6, 9]	[2,5]	[8,12]	[0,4]	[15,19]
TID	6	7	8	9	10
Intervals	[6,9]	[8,11]	[16,18]	[12,16]	[15,20]

Here the domain of endpoints is the set I (set of integers) which is discrete. So each endpoint has a unique next endpoint (except r_{max}) and a unique previous end point (except ℓ_{min}). Here ℓ_{min} is 0 and r_{max} is 20.

If we consider the minimum support threshold as 2, we have following maximal sparse intervals-

[0, 1], [5, 5], [13,14], [20, 20]

And if minimum support threshold is 3, we have following maximal sparse intervals-

[0, 7], [10, 14], [19, 20]

3. ALGORITHM FOR MINING MAXIMAL SPARSE INTERVALS

3.1 Theoretical Background

In the following section we prove certain theorems related to maximal sparse intervals.

Theorem 1: For any ℓ where $\ell_{min} \leq \ell \leq r_{max}$, ℓ cannot belong to both a sparse interval and a frequent interval.

Proof: This is because if ℓ is in a frequent interval then $[\ell, \ell]$ is a frequent interval and this implies that ℓ cannot be in a sparse interval as all subintervals of a sparse interval are infrequent.

Theorem 2: If 'a' and 'b' are two consecutive endpoints in the dataset then for any two points 'x' and 'y' such that $a < x < y < b$, $sup(x) = sup(y)$.

Proof: If there are no such two distinct points x and y then the result is trivial. If there are, then let $sup(x) > sup(y)$. Then there must be some interval which contain 'x' but not 'y'. If such an interval exist then there must be an endpoint 'e' such that $x \leq e < y$.

Which contradict that 'a' and 'b' are two consecutive endpoints in the dataset.

Similarly we can show that $sup(x)<sup(y)$ is also false.

Theorem 3: If [a,b] is a sparse interval and prev(a) (if $a \neq \ell_{min}$) and next(b) (if $b \neq r_{max}$) are frequent then [a,b] is a maximal sparse interval.

Proof: By the definition, a sparse interval is said to be a maximal sparse interval if none of the super intervals of the interval is sparse.

Here [a,b] is a sparse interval.

Now any super interval of [a,b] must contain either prev(a) or next(b) or both.

It is given that prev(a) (if $a \neq \ell_{min}$) and next(b) (if $b \neq r_{max}$) are frequent.

None of the super intervals of [a, b] is sparse (since a sparse interval cannot have frequent subintervals).

[a, b] is a maximal sparse interval.

Theorem 4: If [a, b] is a maximal sparse interval and then the immediate previous point of 'a' (if $a \neq \ell_{min}$) and the immediate next point of b (if $b \neq r_{max}$) in the underlying domain $\mathbb{D}$ ($\mathbb{D}$ is discrete) belong to frequent intervals.

Proof: Let 'a'' be the immediate previous endpoint of 'a' and $a \neq \ell_{min}$. Then [a, b] $\subset$ [a', b] and by the definition of maximal sparse interval [a', b] can not a sparse interval. i.e. there is a sub-interval of [a', b] which is frequent.

But this subinterval cannot be contained in [a, b] and so [a', a'] is the only possible candidate.

Similarly we can show that [b', b'] is a frequent interval if 'b'' is a immediate next end point of 'b' and $b \neq r_{max}$.

Theorem 5: The maximal sparse intervals are mutually disjoint.

Proof: Let $[\ell', r']$ and $[\ell'', r'']$ be any two maximal sparse intervals having non-empty intersection. Since both the intervals are maximal, none is contained within the other. Without loss of generality we can assume that $\ell'' \leq r'$. Now let us consider the interval $[\ell', r'']$. This interval is infrequent since $[\ell', r']$ is an infrequent subinterval of it. Also all its subintervals are infrequent since any such sub interval is contained in $[\ell', r']$ or in $[\ell'', r'']$ or in both. Therefore $[\ell', r'']$ is a sparse interval. This contradicts our assumption that $[\ell', r']$ and $[\ell'', r'']$ are maximal sparse intervals.

Theorem 6: If $[\ell, r]$ is a maximal sparse interval then 'ℓ' is the immediate next point of a right endpoint if $\ell \neq \ell_{min}$ and 'r' is the immediate previous of a left endpoint in the dataset if $r \neq r_{max}$.

Proof: Let $[\ell, r]$ be a maximal sparse interval and ℓ' be the immediate previous point of 'ℓ' in the domain $\mathbb{D}$.

Then from theorem 3 it follows that 'ℓ'' is in a frequent interval. This implies that there is at least one interval which contain 'ℓ'' but not ℓ. Now since there are no end points between 'ℓ'' and ℓ, 'ℓ'' has to be the right end point of that interval.

Therefore 'ℓ' is the immediate next endpoint of a right endpoint in the dataset if $\ell \neq \ell_{min}$.

Similarly we can prove the result for the right end point 'r'.

3.2 Algorithm

Based on the theorems cited above, we have designed the following algorithm. The correctness of the algorithm is discussed in section 3.3. The algorithm for mining maximal sparse interval has two parts. In the first part, endpoints are sorted. In the process of sorting we follow the following rules-

If P & Q are two endpoints in the dataset then we say P < Q if

1. P < Q (i.e value of P is less than the value of Q) or
2. if P = Q and P is a left endpoint and Q is a right endpoint

In the process of sorting we also keep the information about whether the endpoint is a left endpoint or right endpoint.

Input: Endpoints of the intervals in sorted order and the threshold k. The ordered list of end points is stored in an array of structure named 'ep'. The structure contains an endpoint and information regarding whether the endpoint is left or right endpoint.

Output: Maximal sparse Intervals

```
Step 1    rc= ρ=0, MI=empty
Step 2    l = ep[1].e
Step 3    Repeat forever
Step 4      while(ρ<k)
Step 5         rc++
Step 6         if(rc>2N) GO TO Step26
Step 7         if(ep[rc].e is right)
Step 8           ρ = ρ - 1
Step 9         else
Step 10          ρ = ρ + 1
```

```
Step 11       endif
Step 12     end while
Step 13       r = ep[rc].e
Step 14       if(l<r)   append   <l,prev(r)>
              to MI
Step 15       while( ρ ≥ k)
Step 16         rc++
Step 17         if( rc>2N ) GO TO Step26
Step 18         if( ep[rc].e is right )
Step 19           ρ = ρ - 1
Step 20         else
Step 21           ρ = ρ + 1
Step 22         endif
Step 23       end while
Step 24       l = next(ep[rc].e)
Step 25   end repeat
Step 26   r = ep[2N].e
Step 27   if(l<r) Append <l,r> to MI
```

In the algorithm, maximal sparse intervals are mined by making a single pass over the sorted endpoints in the datasets. Whenever the algorithm encounters a left endpoint it increases the frequency count and whenever it gets a right endpoint it decreases the frequency count. Let us take the example given in table 1. After pre-processing of the dataset we have the sorted array as given bellow-

Table 2: Array after sorting

Array Index	1	2	3	4	5	6	7
Endpoint	0	2	4	5	6	6	8
L/R	L	L	R	R	L	L	L
Array Index	8	9	10	11	12	13	14
Endpoint	8	9	9	11	12	12	15
L/R	L	R	R	R	L	R	L
Array Index	15	16	17	18	19	20	
Endpoint	15	16	16	18	19	20	
L/R	L	L	R	R	R	R	

Let us take minimum support threshold (k) as 3

Initially ℓ = ep[1].e=0, frequency count (ρ) is 0and ρ<k is true.

ep[1].e is a left endpoint so frequency count(ρ) is incremented (ρ = 1)

ep[2].e is also a left endpoint so ρ is incremented (ρ = 2)

ep[3].e is a right endpoint so ρ is decremented (ρ = 1)and it continues

at ep[7] ρ becomes 3, it violates ρ < k and a maximal sparse interval [0, prev(8)] ([0,7]) is generated

ep[8] is a left endpoint so frequency count(ρ) is incremented (ρ = 4) and it continues

at ep[10] ρ becomes 2, it violates ρ >= k is false, set ℓ = next(ep[10]) = 10

Similarly all the endpoints in the dataset are scanned and the maximal sparse intervals in the dataset are generated.

3.3 Correctness Claim

To prove that the proposed algorithm is correct, we have the following theorem.

Theorem 7: The intervals generated by the proposed algorithm are the only maximal sparse intervals.

Proof: To prove the correctness of the algorithm we have to show that the output produced by the algorithm is correct and the maximal sparse intervals generated by the algorithm are the only maximal sparse intervals in the dataset.

When the execution of the algorithm starts, it sets the value of ℓ as $\ell = \ell_{min}$. The execution of the first while loop repeats until ρ ≥ k where 'ρ' is the support of the endpoint ep[rc].e and 'k' is the threshold value i.e. this loop executes until the first frequent endpoint after ℓ_{min} is obtained.

After coming out of the while loop the algorithm adds <ℓ, prev(ep[rc].e)> to the list of maximal sparse intervals. <ℓ, prev(ep[rc].e)> is sparse since all the points x where $\ell \le x \le$prev(ep[rc].e) are infrequent (since ρ < k). Correctness follows from theorem 3 since $\ell = \ell_{min}$ (and hence has no previous end point) and ep[rc].e is frequent.

Execution enters the second while loop with ρ ≥ k and executes until ρ < k, i.e. it executes until we find an endpoint ep[rc].e for which ρ < k is satisfied. The algorithm sets ℓ = next (ep[rc].e) which is an infrequent end point and repeats by entering the first while loop and searches for the next frequent end point and the whole process repeats until we reach the last endpoint in the dataset.

After each execution of the first while loop, the algorithm adds < ℓ, prev(ep[rc].e)> to the list of maximal sparse intervals where ℓ is infrequent and is the immediate next point of a frequent end point (step 24 in the algorithm), ep[rc].e is the first frequent end point after ℓ and so correctness follows from theorem 3.

When we reach the last endpoint in the dataset it exits from the master loop and < ℓ, r> is added to the list of maximal sparse intervals where prev(ℓ) is frequent, r = r_{max} and there is no frequent end point in [ℓ, r].

The algorithm inspects only the endpoints in the dataset, since according to theorem 2 the supports of all points between two consecutive endpoints are equal.

Since the algorithm systematically searches for intervals satisfying the criteria of theorem 4, the algorithm is able to extract all the maximal sparse intervals.

3.4 Complexity Analysis

The method proposed here is composed of two parts. The first part is pre-processing, where we sort the input endpoints. In the process of sorting, information about whether the endpoint is a left endpoint or right endpoint is kept. In pre-processing of data we use merge sort for sorting data, which takes O(n log n) time. After pre-processing, we run the proposed algorithm for mining maximal sparse intervals in the dataset. The algorithm scans each of the endpoints in the dataset only once and produces the output. So the complexity of the algorithm is O(n). Total time Complexity of this algorithm is

O(n log n) + O(n) + O(n) = O(n log n). Which is much more efficient than the algorithm proposed in[15] for mining maximal sparse intervals. The time complexity of the algorithm proposed in [15] is $O(n^2)$.

4. RESULT AND DISCUSSION

To test the proposed algorithm we have developed a C++ program. For comparing the proposed algorithm with the existing algorithms we have used two different datasets. First one is a real life dataset obtained from "Bodhidroom", the online e-learning portal of IDOL, Gauhati University. The dataset contains 10031 records; each record contains an interval describing login and logout time of the users.

The method proposed in [15] is the only method available for mining maximal sparse interval. To compare we run both the methods for different size of data, taking minimum support threshold as 3. Time requirement of both the algorithms for different sizes of datasets are as in the figure below-

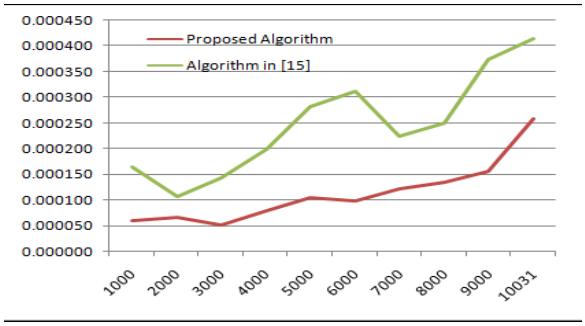

Fig 1: Complexity analysis for real life datasets

Second dataset used is a synthetic dataset. We have developed a syntactic data generator, inputs to the generator are size of the dataset, ℓ_{max} and max_span. We use random number generator function to generate left and right endpoints. For testing we generate dataset of different size by setting ℓ_{max} as 1000000 and mean as 1000. We run these datasets on both maximal sparse intervals from maximal frequent intervals [15] and on our proposed maximal sparse interval taking minimum support threshold as 500. In the following graph time requirement of both the algorithms are given.

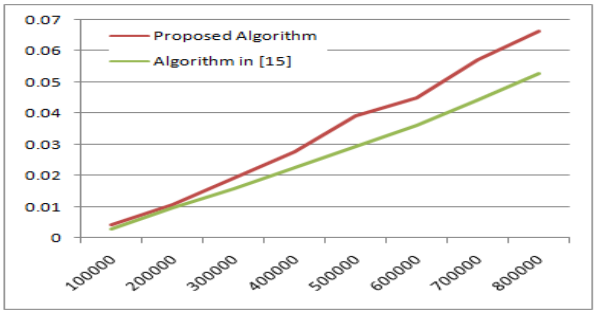

Fig 2: complexity analysis of Synthetic Dataset

From the above results we can clearly say that our proposed algorithm is an improvement over the previously proposed algorithm for mining maximal sparse interval.

5. CONCLUSION AND FUTURE WORKS

Here we proposed an O(n log n) algorithm for mining maximal sparse intervals which is an improvement over the method proposed in [15]. The correctness of the algorithm is also been established mathematically. The algorithm is also tested with real life and synthetic datasets. The future work can be generalizing the method for mining maximal sparse intervals in multidimensional datasets, mining rare intervals in the interval datasets etc.

6. ACKNOWLEDGMENTS

The first author is an INSPIRE fellow, fellowship is granted by Department of Science and Technology for pursuing his Ph. D.

7. REFERENCES

[1] Rakesh Agrawal, Ramakrishnan Srikant, "Mining Sequential Patterns", Proceedings of the Eleventh International Conference on Data Engineering, p.3-14 March 06-10, 1995.

[2] Rakesh Agrawal, Ramakrishnan Srikant, "Fast Algorithm for Mining Association Rules", Proceedings of the 20th VLDB conference, Santiago, Chile, 1994.

[3] L. Troiano, G. Scibelli, C. Birtolo, "A Fast Algorithm for Mining Rare Itemsets" Ninth International Conference onIntelligent Systems Design and Applications, 2009.

[4] L. Szathmary, A. Napoli, P. Valtchev, "Towards Rare Itemset Mining", 19th IEEE International Conference on Tools with Artificial Intelligence, 2007.

[5] A. K. Mahanta, N. H. Son, "Mining Interesting Periodicities of Temporal Patterns" Proceedings of IPMU'08, p.1757- 1764, June 22-27, 2008.

[6] A. K. Mahanta, F. A. Mazarbhuya, H. K. Baruah, "Finding calendar-based periodic patterns" Pattern Recognition Letters, p.1274-1284, Vol 29 Issue 9, July 2008.

[7] J. F. Alen, "Maintaining Knowledge about Temporal Intervals" Communications of the ACM, Vol 26, Nov 1983.

[8] Po-shan Kam, Ada Wai-chee Fu, "Discovering Temporal Patterns for Interval-based Events" Proceedings of the Second International Conference on Data Warehousing and Knowledge Discovery, p.317- 326, 2000.

[9] Shin-Yi Wu, Yen-Liang Chen, "Mining Nonambiguous Temporal Patterns for Interval-Based Events" IEEE Transactions on Knowledge and Data Engineering, Vol 19 No 6, June 2007.

[10] J. Lin, "Mining Maximal Frequent Intervals", Proceedings of 2003 ACM symposium on Applied Computing, p.426-431, ACM, New York, 2003.

[11] M. Dutta, A. K. Mahanta, " An Efficient Method for Construction of I-tree", Proceedings of National Workshop on Design and Analysis of Algorithm(NWDA)2010.

[12] M Dutta, "Development of Efficient Algorithms for Some Problems in Interval Data Mining" Ph D Dissertation, Gauhati University, 2012

[13] Khaled M. Elbassioni, "Finding All Minimal Infrequent Multi-dimensional Intervals", Proceedings of the 7th Latin American conference on Theoretical Informatics, p. 423- 434, 2006.

[14] D. I. Mazumdar, D. K. Bhattacharyya, M. Dutta, "Mining Minimal Infrequent Intervals", Journal of Computer Science and Engineering, communicated.

[15] N. J. Sarmah and A. K. Mahanta. Article: Mining Maximal Sparse Interval. International Journal of Computer Applications 58(5):31-34, November 2012. Published by Foundation of Computer Science, New York, USA