ISSN: 2321-7782 (Online)
Impact Factor: 6.047

Volume 4, Issue 6, June 2016

International Journal of Advance Research in Computer Science and Management Studies

Research Article / Survey Paper / Case Study

Available online at: www.ijarcsms.com

# A New Parallel Algorithmic Approach for Sequential Pattern Mining using Binary Representation

**K. Subramanian**[1]
Assistant Professor,
VSS Government Arts College,
Pulankurichi – 630405,
Tamil Nadu – India

**E.Elakkiya**[2]
Research Scholar,
PG & Research Department of computer Science,
J.J College of Arts and Science,
Pudukkottai-622004, Tamil Nadu – India

***Abstract:*** ***Data mining techniques are used in many areas in the world to retrieve the necessary information from huge data. Sequence Pattern Mining is the most often used technique in Data mining concepts. The few applications of sequence pattern mining are Product Sales analysis, Web log click sequence, DNA sequence, Phone Calling pattern and etc,. In this paper, the proposed algorithm is a Parallel algorithm for Sequence Pattern Mining using Direct Bit Position Method to increase the performance of the previous SPAM using DBP algorithm. The proposed algorithm uses separate processing element for block of sequences in DB. The main purpose is to reduce time complexity and the power of computation cost.***

***Keywords: Parallel Algorithm; Sequence; Data Mining; Frequent Pattern; sequential Pattern; bitmap presented.***

## I. INTRODUCTION

The Sequential Pattern Mining was introduced by Agarwal [1]. From a given Sequence Database, the problem was to find out all the frequent and sequential patterns by user defined minimum support threshold. The actual challenge is to generate candidate patterns in a huge datasets by minimum computational cost.

The much research work took place in this area and among the related works, after mid's 1990's following Agarwal and Srikant many scholars are provided many efficient algorithms for Pattern Mining[2][3]. In addition, these work carried out to extend the mining of related patterns. Apriori like method find the frequent sequence in beginning itself. By using multiple iterations, the candidate patterns with length $l$ are generated from the frequent patterns with length ($l$-1) using iteration. Then the supports of these candidate patterns are checked to discover frequent patterns with length $l$. The Apriori-like sequential pattern mining methods suffer from the costs to handle a potentially huge set of candidate patterns and scan the database repeatedly. To eliminate the snags present in the before mentioned algorithms, Prefix Span [4] algorithm developed by Pei et.al, developed from Free Span [5], was proposed and the design of prefix Span was based on divide and-conquer approach. Recursive method was employed to create sequences where each sequence has the same prefix subsequence. By developing local frequent prefix subsequence in each projected database recursively, all the sequential patterns were discovered without any candidate generation. Although Prefix Span prevented from generating unnecessary candidate patterns, the cost of constructing projected databases recursively was not affordable when dealing with large databases.

A sequential pattern mining algorithm was called Direct Bit Position method using SPAM, is to mining frequent sequential patterns efficiently. By extending the structures of bitmap data representation, the item PresentIn table are constructed first and the corresponding binary position table is constructed next. Based on the binary data presented, several heuristic mechanisms are proposed to speed up the efficiency of support count and the S-Patterns and I-Patterns are found [6]. The item in sequence pattern can be in different ways that: The small value can make thousands of patterns or the big value can generate a less number of patterns in sequence DB.

 

However, the DBP-SPAM is not that much effective for the huge sequence database. In this paper, the new modified and improved DBP-SPAM is proposed to increase the power of computation and efficiency by adopting the parallel approach with DBP-SPAM. The Parallel approach is to have sufficient processing elements *P* to scan each sequence *S* in DB. The proposed algorithm is the most efficient algorithm for a huge DB.

## II. PROBLEM STATEMENT

The actual problem of mining sequential pattern was initiated [1]. The following definitions are referred to [2],[6]. Let I = {i1, i2, i3, … . in} be a unique item set. A sequence S is an ordered list of events, denoted as <e1, e2, e3… en> where ei is an item, (i.e.) ei ⊆ I for 1 ⊆ i ⊆ n. For brevity, the brackets are omitted if the element has only one element, (i.e.) (a) is written as a. An item can occur multiple times in different event of a sequence. Number of events in a sequence is called the length of a sequence and a sequence of l length is l-sequence. A sequence SA= {a1, a2,a3.. an} is contained in another sequence SB={b1,b2,b3,.. bm}, if there exist integers 1 ⊆ i1< i2< i3 … < in ⊆ m such that a1=bi1, a2=bi2, …. an= bin. If sequences SA contains in another sequence SB, then SA is called subsequence of SB and SB a super-sequence of SA, denoted by SA ⊆ SB.

From the table 1(a),1(b) and 1(c) shows the input sample dataset and sample sequence dataset S has taken by synthetic data generated by the IBM synthetic market basket data generated.

**Table I (a):** Sample Dataset1

| TID | ITEMS BOUGHT |
|---|---|
| T1 | Bread, Butter, Milk, Bread, Milk |
| T2 | Bread, Butter, Bread, Milk |
| T3 | Butter, Bread, Milk, Jam, Butter,Milk |
| T4 | Jam, Jam, Butter |
| T5 | Jam, Butter, Milk, Milk |

**Table I (b):** Sample Dataset2

| TID | ITEMS BOUGHT |
|---|---|
| T1 | Scale, pencil, book, Eraser, pen |
| T2 | Scale, pencil, scale, book |
| T3 | Pencil, scale, book, Eraser |
| T4 | Eraser , Eraser, pencil |
| T5 | Book, scale,pen, scale |

**Table I (c):** Sample Sequence Dataset

| Sid | Sequence |
|---|---|
| S1 | <A(CD)AD> |
| S2 | <ACAE> |
| S3 | <CAD(BCD)> |
| S4 | <BBC> |
| S5 | <(BCD)D> |

The input sequence database S is a set of Items (Sid, S), where Sid is the sequence identifier and X is the input sequence. The number of Items in S database are called the base size of the database S, and denoted as |S|. A tuple (sid, X) is said to contain in sequence SA, If SA is a sub-sequence of X. The support of a sequence Sa in the database S is the number of tuples in the database containing SA, denoted as sup (SA).[6] For a given positive integer min-sup, as the support threshold, a sequence SA is called frequent sequential pattern in database S, if sup (SA) ⊆ min-sup. Otherwise, the pattern is not frequent.

 

### A. PARALLEL APPROACH

The Problem is divided into sub-problem and is executed in parallel to get individual outputs. Later on, these individual outputs are combined together to get the final desired output. This approach is to develop parallel algorithms for clustering and classification for large data sets.

## III. PROBLEM DEFINITION

In DBP-SPAM, constructs an item bit position table for each sequence in the database S. The candidate generation happens by direct bit position from PresentIn table. The building PresentIn table is too large, when the database is too huge. And the binary data representation also has to be done sequentially. It may take long time because of the sequence operations. The length of the binary represented row in the position table is equal to the length of the sequence in the database S. If the item A is in the $i^{th}$ position of the sequence from left, the $i^{th}$ position of that item A is put to 1, otherwise it is put to 0. Table2 shows the binary representation in a Sequence Bit position and Table3 illustrates the PresentIn Table for all the sequences. If the Database S have huge sequences, it is difficult to have a too big PresentIn table to generate candidates.

**Table II:** Item Bit Position Table.

| Sequence < A (CD) AD > | | | | |
|---|---|---|---|---|
| **S1** | **A** | **CD** | **A** | **D** |
| A | 1 | 0 | 1 | 0 |
| C | 0 | 1 | 0 | 0 |
| D | 0 | 1 | 0 | 1 |

**Table III:** Sample PresentIn Table.

| Item | S1 | S2 | S3 | S4 | S5 | Sup. |
|---|---|---|---|---|---|---|
| A | 1 | 1 | 1 | 0 | 0 | 3 |
| B | 0 | 0 | 1 | 1 | 1 | 3 |
| C | 1 | 1 | 1 | 1 | 1 | 5 |
| D | 1 | 0 | 1 | 0 | 1 | 3 |
| E | 0 | 1 | 0 | 0 | 0 | 1 |

## IV. PROPOSED APPROACH

In this proposed Parallel DBP-SPAM, Consider the input sequence database S with its size of *Sn* (number of sequences *Sn*). Let the user defined minimum support threshold is min_sup, and numbers of processing element is *Pm*. Let it be the sequence S1, S2,...Sn. Processing elements are Pi to Pm. The sequences are evenly distributed to the processing elements. The processes could have been done the allocated sequences with the existing algorithm of DBP-SPAM. Finally, all the processing elements communicate each other to concatenate the S-Extended and I-Extended patterns which will be found. All the processing element starts at the same time to process its own sequences accordingly. In this approach, the time reduced enormously, compared to previous normal DBP-SPAM algorithm.

## V. THE PARALLEL DBP-SPAM ALGORITHM

In parallel DBP-SPAM algorithm, is defined the input Sequence Database *S*, to come up with an appropriate min_support: one needs to have prior knowledge about the query and the specific task and number of processing elements *Pm*. The parallel DBP-SPAM algorithm is defined step1: first of all find the length (size) of the given sequence Database S, and it is stored in Sn. In Step2: divide the portions of the database into equal number of sequences by the given number of processing elements (Pm). Each processing elements have its own block of sequences are =Sn/Pm. (If the number of sequence is in Odd. The Last or First process can be reduced by one sequence.) In Step3: allotted block of sequences scanned by the corresponding processing elements at the same time. Generates the base table and do the same tasks in DBP-SPAM algorithm to find S-Extended and I-Extended Patterns. In Step4: S-Extended Patterns and I-Extended Patterns from all the processing elements are concatenated.

 

A. *PSEUDO CODE FOR PARALLEL DBP-SPAM*

**Parallel DBP-SPAM (S, min_sup, Pm)**

**INPUT:** S – Sequence Database, min_sup – Minimum Support, Pm – Parallel Processing elements
**OUTPUT:** Sequential patterns

```
SN As Number of Sequences

BS As Block of Sequences

/*Divide the Number of sequences into equal number block of sequences (BS)*/

BEGIN:

BS=Round(Sn/Pm)

BSCount=0

For each [Pi <=Pm] begin // each processes

For each [Sj <= Sn] begin // Sequences

BSi=BSi+Sj  //block of sequences for each Pi

BSCount++

If BSCount == BS then

BSCount=0

Next Sj

Break //Exit inner for

End if

End for

End for

For each [Pi to Pm] begin // process P1 to Pm

in parallel do

PresentIn+=DBP-SPAM(BSi, min_sup)

Done

Patterns= IS-patterns(PresentIn, min-sup)

End For

END

Function DBP-SPAM(S, min_sup)

Initialize bitPositions sequences,

PresentIn sequences and Scount as zeroes

For each [sidi, s]<= D begin

For each Element sj of s begin
```

 

For each item i<=sj begin

If PresentInI (i) = 0, Mark PresentInI (i) = 1

Set $j^{th}$bit in POSI(i) =1

End for

End For

End For

END

Function IS-Pattern(PresentIn,min-sup)

INPUT: Present In table, min-sup

BEGIN:

For each item i <= Present In Table

Form base itemsets by applying AND operation

If base Itemsets>= min-sup

Store base itemsets in base table

End for

For each Itemsets K <= Base table

Fetch POS tables according to the items <= K

Find S-Extended patterns based on position

Count the S-extended patterns

If S-extended patterns >= min-sup

Store in Results

Find I-Extended patterns based on equal position

Count I-extended patterns

If I-Extended patterns >=min-sup

Store in Results

End For

Return Results

END

---

**Example1:** Consider the sample Database from table1. Let the given input for minimum support threshold is min_sup=2 and the number processing elements Pm=2. The size of the database sequence is Sn=5.

Processing blocks of sequences for each processing elements are Sn/Pm => 5/2=2.5 rounded up to 3 sequences for each processing elements. Processing Elements are P1 and P2. Each Processing Elements should have 3 sequences. Accordingly, P1(S1,S2,S3) and P2(S4,S5) because no more sequence in given DB. The PresentIn tables 4, 5 and 6 are shows that, the parallel

 

processing is possible to generate and find the frequent patterns in the fastest way. It helps to reduce the time for the huge data item sequences.

**Table IV:** PresentIn Table for P1

| ITEM | P1 | | | Supp. |
|---|---|---|---|---|
| | S1 | S2 | S3 | |
| A | 1 | 1 | 1 | 3 |
| B | 0 | 0 | 1 | 1 |
| C | 1 | 1 | 1 | 3 |
| D | 1 | 0 | 1 | 2 |
| E | 0 | 1 | 0 | 1 |

**Table V:** PresentIn Table for P2

| ITEM | P2 | | Supp. |
|---|---|---|---|
| | S4 | S5 | |
| A | 0 | 0 | 0 |
| B | 1 | 1 | 2 |
| C | 1 | 1 | 2 |
| D | 1 | 1 | 2 |
| E | 0 | 0 | 1 |

**Table VI:** PresentIn Table for P1 and P2

| ITEM | P1 | | | P2 | | Supp. |
|---|---|---|---|---|---|---|
| | S1 | S2 | S3 | S4 | S5 | |
| A | 1 | 1 | 1 | 0 | 0 | 3 |
| B | 0 | 0 | 1 | 1 | 1 | 3 |
| C | 1 | 1 | 1 | 1 | 1 | 5 |
| D | 1 | 0 | 1 | 0 | 1 | 3 |
| E | 0 | 1 | 0 | 0 | 0 | 1 |

The table 7 shows the base table construction for frequent sequence pattern. It could use the same existing Direct Bit Position Method using SPAM.

 

**Table VII:** Construction of Base Table

| Pattern | AND Operation | Result | Count | PRUNE |
|---|---|---|---|---|
| AB | A=11100 B=00111 | AB=00100 | 1 | PRUNED |
| AC | A=11100 C=11111 | AC=11100 | 3 | |
| AD | A=11100 D=10101 | AD=10100 | 2 | |
| BC | B=00111 C=11111 | BC=00111 | 3 | |
| BD | B=00111 D=10101 | BD=00101 | 2 | |
| CD | C=11111 D=10101 | CD=10101 | 3 | |
| ABC | AB=00100 C=11111 | ABC=00100 | 1 | PRUNED |
| ABD | AB=00100 D=10101 | ABD=00100 | 1 | PRUNED |
| ACD | AC=11100 D=10101 | ACD=10100 | 2 | |
| BCD | BC=00111 D=10101 | BCD=00101 | 2 | |

## VI. Experimental Evaluation and Test Environment Datasets

The proposed Parallel DBP-SPAM is implemented on Visual C# programming with use of multi-thread on a personal computer of Intel Dual core 2.66 GHz processor, 2 GB RAM on Windows7 Ultimate. The experimental evaluation performed on synthetic market-basket data generator. It is generated from market-basket by IBM (table8). The comparison experimental table is in table9.

**Table VIII:** IBM Synthetic Data

| *Parameters* | *Description of parameter* | |
|---|---|---|
| D | Total number of sequences in the dataset | |
| C | Average elements per sequence | |
| S | Average length of potentially frequent sequential patterns | |
| I | Average length of itemsets in maximal potentially frequent patterns | |
| T | Average number of items per transactions. | |
| N | Number of different items in 000s. | |
| K | 000s, which covers the range of typical values | |

**Table IX:** Comparison with its running time

| Running Time(Sec) - D3KC8T5S5I5N1K | | | | | |
|---|---|---|---|---|---|
| | Min_sup | | | | |
| **Algorithm** | 0.01 | 0.015 | 0.02 | 0.03 | 0.035 |
| SPAM | 99 | 58 | 26 | 19 | 12 |
| DBP-SPAM | 62 | 41 | 22 | 18 | 11 |
| Parallel DBP-SPAM | 21 | 14 | 8 | 6 | 4 |
| | Data Size D | | | | |
| | 4K | 5K | 6K | 7K | 8K |
| SPAM | 21 | 33 | 58 | 76 | 132 |
| DBP-SPAM | 20 | 31 | 43 | 63 | 109 |
| Parallel DBP-SPAM | 10 | 15 | 16 | 31 | 53 |
| | Average Element C | | | | |
| | 4 | 5 | 6 | 7 | 8 |
| SPAM | 9.2 | 23 | 43 | 71 | 93 |
| DBP-SPAM | 9 | 22 | 40 | 54 | 66 |
| Parallel DBP-SPAM | 3 | 7 | 13 | 18 | 22 |

These graphs show the different type of analysis, the experimental evaluation concerning the running time is compared on different synthetic datasets. The results as the minimum support are changed from 1 to 0.2 percentages.

 

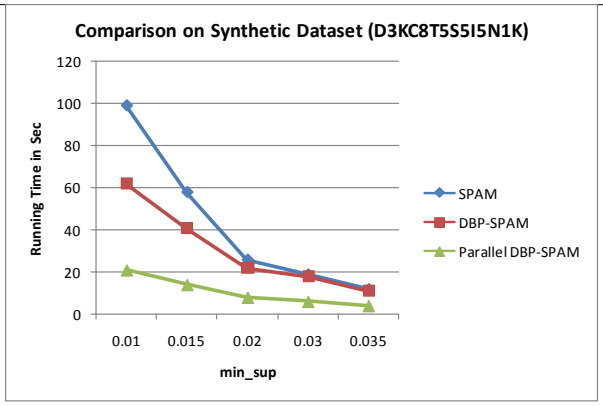


Fig.1 min_sup and Running time Comparison

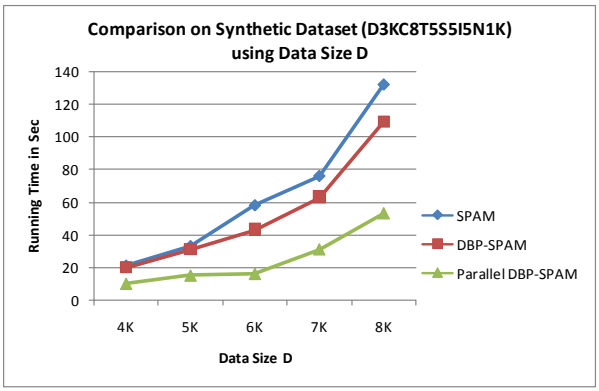


Fig.2 Comparison with Data Size D

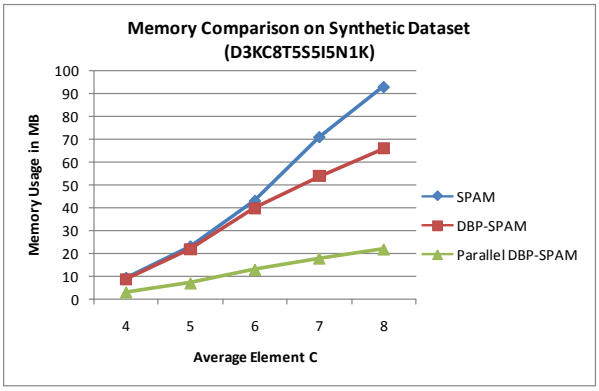


Fig.3 comparison with Average Element C

## VII. CONCLUSION

The new proposed algorithm Parallel DBP-SPAM is the finest algorithm when the database sequences are very large in size. The actual primary challenge in sequential pattern mining depends on the size of the candidates generated and squeezes the computations involved for the support count. The shared memory needed to store the data is much less or equal to the existing DBP-SPAM algorithm. The new results table.10 shows that the proposed algorithm outperformed the DBP-SPAM on very large datasets and smaller min-sup threshold values.

**Table X:** Sequential Pattern Output.

| *Patterns* | *Count* | *Patterns* | *Count* |
|---|---|---|---|
| AA | 2 | BCD | 2 |
| AC | 3 | BD | 2 |
| ACA | 2 | CA | 3 |
| A(CD) | 2 | CAD | 2 |
| AD | 2 | CD | 3 |
| ADD | 2 | (CD)D | 2 |
| BC | 2 | | |

 

## References

1. R. Agarwal and S.Arya, .Mining multiple level Association Rules to mining Multiple level Correlation to discover complex patterns. In Proc. 2012, International Journal of Computer Science, 2012.
2. Alpa Reshamwala and Dr. Sunita Mahajan, Improving Efficiency of Apriori Algorithms for sequential Pattern mining. International Journal of data mining, In March 2014.
3. KMVM Kumar, PVS Srinivas, CR Rao – MS-SPADE: Sequential Pattern Mining with multiple minimum supports. In Proc. 2012, International Journal of Computer science 2012.
4. HJ Shyur, C Jou, K Chang: A data mining approach to discovering reliable sequential patterns, Journal of Systems and software, 2013.
5. Manan Parik, Barat Caudari and chetna Chand: A comparative Study of Sequential Pattern Mining Algorithms, International Journal of Application or Innovation in Engineering & Management. Vol 2, Feb 2013.
6. K. Subramanian, E. Elakkiya, Modified Sequential Pattern Mining Using Direct Bit Position Method", International Journal of Science and Research (IJSR), ISSN (Online): 2319-7064, 2016.
7. J. Pei, J. Han, Pattern Growth Methods: Frequent Pattern Mining. In Proc. 2014 Springer.
8. S.MuthuSelvan, Ks. Sundaram, "A survey of Sequence Patterns in Data Mining Techniques. International Journal of Applied Engineering Research 2015.
9. Youssef Bassil , Aziz Barbar, "Sequential & Parallel Algorithms For the Addition of Big-Integer Numbers", International Journal of Computational Science, vol. 4, no. 1, pp. 52-69, 2010.

## AUTHOR(S) PROFILE

**Dr. K. Subramanian,** received the B.Sc degree in Computer Science from Madurai Kamaraj University, Madurai, M.Sc degree in Computer Science from Alagappa University, Karaikudi, M.Phil degree in Computer Science from Bharathidasan University, and Ph.D degree in Computer Science and Engineering from Alagappa University, Karaikudi, in 1991, 1995, 2005 and 2012 respectively. In 1995, He had started his career as Lecturer at APSA College, Thiruppathur, He worked as Head, in the Department of Computer Science later he had work as Vice-Principal at JJ College of Arts and Science, Pudukkottai, during the year of 1999 to 2015. From July 2015, He joined as Assistant Professor in Government Arts College, Kulithalai, and he shifted to V.S.S. Govt. Arts College, Pulankurichi from November 2015 onwards. He guided 27 M.Phil Scholars in Computer Science and guiding 8 Ph.D Research Scholars.

**E. Elakiya,** received the B.Sc degree in Information Technology, M.Sc degree in Information technology and M.Phil degree in Computer Science from JJ College of Arts and Science, Pudukkottai, in 2007, 2009 and 2011 respectively. In 2013, she has started her Research on Data Mining by Registering the Doctor of Philosophy (PhD) in JJ College of Arts and Science, Affiliated to Bharathidasan University, Pudukkottai. She is a Research Scholar.

 