TELKOMNIKA, Vol.15, No.2, June 2017, pp. 709~717
ISSN: 1693-6930, accredited **A** by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v15i2.5098

Progressive Mining of Sequential Patterns Based on Single Constraint

Regina Yulia Yasmin*[1], Putri Saptawati[2], Benhard Sitohang[3]
School of Electrical Engineering & Informatics, Institut Teknologi Bandung, Jl. Ganesha 10, Bandung, Indonesia
Corresponding author, e-mail: regina.yasmin@gmail.com*[1], putri.saptawati@stei.itb.ac.id[2], benhard@stei.itb.ac.id[3]

Abstract

Data that were appeared in the order of time and stored in a sequence database can be processed to obtain sequential patterns. Sequential pattern mining is the process to obtain sequential patterns from database. However, large amount of data with a variety of data type and rapid data growth raise the scalability issue in data mining process. On the other hand, user needs to analyze data based on specific organizational needs. Therefore, constraint is used to impose limitation in the mining process. Constraint in sequential pattern mining can reduce the short and trivial sequential patterns so that the sequential patterns satisfy user needs. Progressive mining of sequential patterns, PISA, based on single constraint utilizes Period of Interest (POI) as predefined time frame set by user in progressive sequential tree. Single constraint checking in PISA utilizes the concept of anti monotonic or monotonic constraint. Therefore, the number of sequential patterns will decrease, the total execution time of mining process will decrease and as a result, the system scalability will be achieved.

Keywords: *sequential pattern mining, progressive mining of sequential patterns based on single constraint, progressive sequence tree, big data*

1. Introduction

Scalability is an important criterion of any data mining algorithms dealing with progressive increasing data, including sequential pattern mining. Many researches have resulted in scalable algorithms for sequential pattern mining [1-4]. Moreover, technology on distributed data processing such as MapReduce/Hadoop also contributes in increasing scalability. Some researches which apply Mapreduce framework to increase the scalability are [5-10].

Beside scalability problem, sequential pattern mining also deals with huge search space during generating candidate sequence. It can decrease the mining efficiency, especially when it faces high dimension of distinct item in sequence database. One way to handle this problem is by incorporating user specified constraint into sequential pattern mining. As a result, the mining process could reduce the search space and consider only patterns which are of interest [1]. Some algoithms that have incorporated constraints are: SPIRIT [11], EXT-PrefixSpan [12], and Prefix Growth [13].

Based on the above explanation, we propose to incorporate constraints into existing scalable algorithm in order to increase its performance when dealing with progressive & high dimension sequence data. The performance includes decreasing the number of sequential patterns to evaluate, and execution time. Progressive mIning of Sequential pAtterns, PISA based on single constraint is the scalable algorithm of our interest, since it is a progressive sequential pattern mininng that searches for sequential patterns which satisfy minimum support and single constraint within certain window length. Since the window length is flexible to be shifted, it makes PISA is flexible in adding or subtracting data in sequence database [1].

This paper is organised as follows. Section I contains Introduction. Section II explains about related work on progressive sequential pattern mining. Section III explains about progressive mining of sequential patterns based on single constraint which explains in detail about progressive mining of sequential patterns, types of constraints and progressive mining of sequential patterns based on single constraint: the proposed approach. Section IV explains

Received January 1, 2017; Revised March 15, 2017; Accepted April 2, 2017

about experiments and its results. Section V contains conclusion of this paper and future research.

2. Related Work

Sequential pattern mining is one of the development of frequent itemset mining [14], as well as structured pattern mining, correlation mining, associative classification, and frequent pattern-based clustering. Denoted by $I = \{i_1, i_2, \ldots, i_k\}$ which is a set of all items, subset of I is called itemset. Sequence $\alpha = \langle t_1, t_2, \ldots, t_m \rangle$ where ($t_i \subseteq I$) is ordered list. Each itemset in a sequence represents the set of events that occur at the same time (same timestamp). Different itemset appears at different time [15]. Sequence $\alpha = \langle a_1, a_2, \ldots, a_n \rangle$ is a subsequence of sequence $\beta = \langle b_1, b_2, \ldots, b_m \rangle$ if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$ [16, 17].

Previous studies have tried to address the scalability problem, such as progressive sequential pattern mining and sequential pattern mining in distributed system. Progressive sequential pattern mining includes incremental sequential pattern mining [2-4]. Moreover, frequent pattern mining was also developed in distributed systems [18]. Several researches have been developed to mine sequential patterns from large database [19-20]. Some sequential pattern mining algorithms that conform to user's needs were developed based on existing algorithm such as, (1) SPIRIT (Sequential pattern mining with regular expression constraints) algorithm to satisfy regular expression constraint [11] that was based on Apriori algorithm, (2) EXT- Prefix Span algorithm [12] and Prefix Growth [13] that was based on prefix span algorithm. Execution time performance of Prefix Span was found better than Apriori [21] while Prefix Growth [13] can handle monotonic and anti-monotonic constraints.

In Prefix Growth algorithm, the concept of monotonic and anti-monotonic constraint on subsequence is inherited into prefix property. Since prefix is also a subsequence, prefix inherits monotonic and anti-monotonic constraint property. A constraint, denoted as Cpa is a prefix anti-monotonic if for every sequence α satisfies the constraint, then the subsequence of α also satisfies the constraint. While, a constraint denoted as Cpm is prefix monotonic if for every sequence α satisfies the constraint, then each sequence containing α as a prefix also satisfies the constraint. Constraint is called prefix-monotone if it satisfies the prefix anti-monotonic or prefix monotonic. Prefix growth approach was developed from the concept of prefix monotonic and anti-monotonic [13].

3. Progressive Mining of Sequential Patterns Based on Single Constraint

The objective of this research is to modify PISA algorithm in order to incorporate single constraint criteria. Obviously PISA was scalable and flexible in data addition or subtraction [1]. Meanwhile, the use of constraint on mining large-scale data was also proven to generate sequential patterns according to user's need effectively [13]. Therefore, this study is expected to improve the performance of total execution time and lower the number of sequential patterns. Moreover, sequential patterns that were found conform to user's need.

3.1. Progressive Mining of Sequential Patterns

PISA introduced the concept Period of Interest (POI). POI was a sliding window which window length was determined by the user and moved continuously according to time. Basically PISA represents a sequence database in the form of Progressive Sequential Tree, PS-tree. By applying the POI on PS-tree, PISA creates flexibility to add or subtract data to be mined without having to reconstruct the sequence database. [1] As shown in Figure 1, sequence ID consists of itemset where each itemset is grouped by time. POI is a user defined time interval which every itemset that is within the POI is subject to be constructed in PS-tree. For example, if we choose to set POI is 4, then PS-tree is built from t1 until t5 and shifted to the next timestamp, i.e from t2 until t6. This sliding window will continously shift untills the last timestamp.

S01	A	BC		D				
S02	AB			C				
S03		A	B			C		
S04	BC		D					
S05		AB		D				
SID	t_1	t_2	t_3	t_4	t_5	t_6	t_m	...time

Figure 1. Sequential database representation

PISA looks for sequential patterns in POI time interval that satisfy the given minimum support. PISA gets sequential patterns by traversing the PS-tree. The depth of PS-tree depends on the specified POI range [1]. PISA performs more efficiently than previous algorithms such as Apriori or Prefix Span since it does not generate candidate sequential patterns nor repetition to form postfix database [1, 22]. Progressive sequence tree (PS-tree) with PISA algorithm [7] records sequence ID, label and timestamp. Figure 2 shows about PISA algorithm that consists of main program and procedure traverse. Procedure traverse builds the PS-tree, do the traversing and count the occurence frequency of itemset and compare it with minimum support.

```
Input    : dataset, minimum support, POI
Output   : sequential patterns
Method   :

PISA (support, POI)
begin
 Var PS; //PS-tree
 Var currentTime; //timestamp now
 Var eleSet; //used to store elements ele
 While (there is still new transaction)
 eleSet = read all ele at currentTime;
 traverse(currentTime, PS);
 currentTime++;
end

Procedure traverse(currentTime,PS)
For(each node of PS in post order) do
   If(node is Root)
      For(ele of every seq in eleSet) do
         For(all combination of elements in the ele) do
            If(element==label of one of node.child)
               If(seq is in node.child.seq_list)
                  Update timestamp of seq to currentTime;
               Else Create a new sequence with currentTime;
            Else //create a child
               Create a new child with element, seq and currentTime;
   Else //the node is a common node
      For(every seq in the seq_list) do
         If(seq.timestamp<=currentTime-POI
            Delete seq from seq_list and continue to the next seq;
            If(there is new ele of the seq in eleSet)
               For(all combination of elements in the ele) do
                  If(element is not on the path from Root)
                     If(element==label of one of node.child)
                        If(seq is in node.child.seq_list)
                           Child.seq_list.seq.timestamp = seq.timestamp;
                        Else Create a new sequence with seq.timestamp;
                     Else //create a child
                        Create a new child with element, seq and
                        seq.timestamp;
      If(seq_list.size==0)
         Delete this node and all of its children from its parent;
   If(seq_list.size>=support*sequence number)
      Output the labels of path from Root to this node as a SP
End.
```

Figure 2. PISA algorithm [1]

3.2. Types of Constraints

Constraint or user constraint is limitation set by user in accordance with user's need & organizational characteristic. Constraint can reduce number of short/trivial sequential patterns. Constraint was predefined before mining process is begun. By applying constraint in searching for sequential patterns, number of discovered sequential patterns will be reduced but the sequential patterns still conform with user needs [16]. Many researches on constraint-based sequential pattern mining has proven that constraint improve interestingness of sequential pattern and also mining performance [11-12], [23-26].

There are many types of user constraints that are described as follows [13]:

1. Item constraint is a limitation where sequential pattern should contain certain item.
2. Length constraint is a length limitation of the sequential pattern. It limits the number of occurrences of distinct items or the number of items in the transaction. Length constraint is divided into 2 types, length less constraint and length more constraint. Length less constraint requires that the length of sequential patterns should be shorter than the given value. While length more constraint requires that the length of sequential patterns should be longer than the given value.
3. Super-pattern constraint looks for patterns that contain a particular pattern as a set of sub-pattern.
4. Aggregate constraint is a constraint on the aggregate of items in the pattern, for example, an item with aggregate functions sum, average and so on.
5. Regular expression constraint is a constraint with regular expression, such as disjuntion and Kleene closure of the set of items.
6. Duration constraint requires sequential patterns to satisfy the condition that the time difference between first and last transactions should be longer or shorter than given period.
7. Gap constraint as well as duration constraint, requires timestamp for each transaction data. Sequential patterns should satisfy the condition that the time difference between two adjacent transactions should be longer or shorter than given period.

Constraint can be categorized into monotonic or anti-monotonic constraint. A constraint, denoted by C_A, is said as anti-monotonic constraint if there is a sequence of α satisfies the constraint C_A then any subsequence of α also satisfies constraint C_A. While, a constraint, denoted by C_M, is said as monotonic constraint if there is a sequence α satisfies C_M constraint then any super-sequence of α also satisfies constraint C_M. Some monotonic constraints are item constraint, super pattern constraint, duration constraint, whilst anti-monotonic constraint is gap constraint. Some constraints can be classified into both of monotonic and anti-monotonic constraints, such as item constraint, length constraint, aggregate constraint such as max, min, sum constraint, and also duration constraint [13, 16].

3.3. The Proposed Approach

The proposed approach adds constraint checking so that it can check constraints such as item, length constraint within POI. Constraint checking on the sequential patterns is made after the list of sequential patterns candidates was obtained in POI time frame. In order to speed up the constraint checking, constraint is checked on its subsequence or super sequence based on its property, whether the constraint belongs to anti-monotonic or monotonic constraint and apply it to all sequential patterns. Therefore, the process of obtaining sequential patterns that satisfy the minimum support and a single constraint is expected to be faster.

1. For anti-monotonic constraint such as length less constraint, if sequence of α satisfies the constraint C_A then any subsequence of α also satisfies constraint C_A. If the SPExisting satisfies the constraint and if SP is a subsequence of SPExisting then SP also satisfies the constraint.
2. For monotonic constraint, if there is a sequence α satisfies C_M constraint then any super-sequence of α also satisfies constraint C_M. Monotonic constraints are item constraint, length more constraint, super pattern constraint, duration constraint where $Dur(\alpha)>\Delta t$. If the SPExisting satisfies the constraint and if SP is a super-sequence of SPExisting then SP also satisfies the constraint.

As shown in Figure 3, constraint checking is added after checking minimum support.

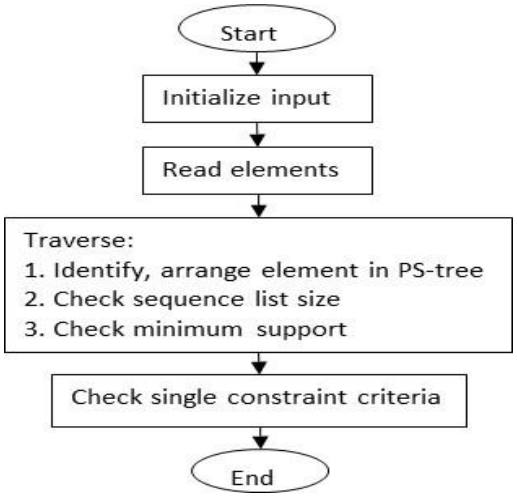


Figure 3. PISA* scheme

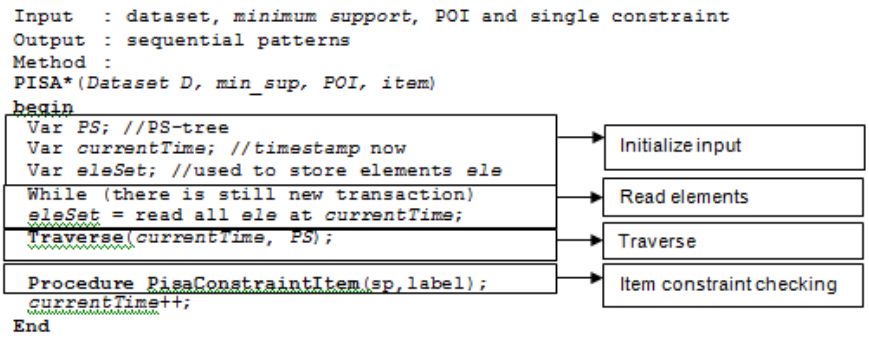
Algorithm of PISA* is explained in Figure 4.

4. Experiment Results

The experiments were conducted on one year e-commerce sales data with total 22,699 transactions, and have attributes: product id, sales date, sales hour and customer id. The objective of experiments is to compare the performance of the proposed method between PISA* (PISA based on single constraint) and PISA based on the number of sequential patterns and the total execution time parameters. The total or accumulated execution time is total time needed to get sequential patterns. The experiment uses item constraint or length less constraint or length more constraint.

Data were represented in sequence database based on sales date and time. Fig 1 represents the sequence database. Sequence id or SID represents sequences based on user's point of view to analyze. In this case, sequence ID is based on sales date. Each SID consists of series of itemset. Each itemset consists of product ID. Itemset in each SID is grouped by the same timestamp. In this experiment, time or timestamp is in sales hour unit. The experiment was aimed to get the sequential patterns of product id from all sales date on the sales time range between 00.00 and 24.00 hours. POI is a user defined time sliding window between specified sales hours interval. Therefore, based on sales hours, maximum number of POI is 23. For example POI can be set of 4, that means time sliding window is set between 4 hours.

The first experiment was aimed to compare the accumulated execution time between PISA* and PISA. The experiment was conducted at minimum support 0.5. As shown in Figure 5 PISA* results in less execution time than PISA. Although in this experiment the accumulated execution time difference between PISA* and PISA is relative small, but for a large amount of data, this difference will be significantly larger.



```
//This is the start of PISA* algorithm
Procedure Traverse(currentTime,PS)
For(each node of PS in post order) do
    If(node is Root)
       For(ele of every seq in eleSet) do
          For(all combination of elements in the ele) do
             If(element==label of one of node.child)
                If(seq is in node.child.seq_list)
                   Update timestamp of seq to currentTime;
                Else Create a new sequence with currentTime;
             Else //create a child
                Create a new child with element, seq and currentTime;
             End if
    Else //the node is a common node
       For(every seq in the seq_list) do
          If(seq.timestamp<=currentTime-POI
             Delete seq from seq_list and continue to the next seq;
             If(there is new ele of the seq in eleSet)
                For(all combination of elements in the ele) do
                   If(element is not on the path from Root)
                      If(element==label of one of node.child)
                         If(seq is in node.child.seq_list)
                            Child.seq_list.seq.timestamp=seq.timestamp;
                         Else Create a new sequence with seq.timestamp;
                      Else //create a child
                         Create a new child with element, seq and
                         seq.timestamp;
                      Endif
                   Endif
                EndFor
             Endif
          Endif
       If(seq_list.size==0)
          Delete this node and all of its children from its parent;
       If(seq_list.size>=support*sequence number)
          If(PisaConstrainExt.Item(node, :prmItem) == true)
             Output the labels of path from Root to this node as a SP;
          Else remove;
          Endif
       Endif
End //This is the end of traverse procedure


        //This is the procedure of constraint checking
        Procedure PisaConstraintItem(sp, label)
        if(sp contains label)
         If(subsequence(spList) ∈ sp)
          continue; Else remove;
         Endif
         return sp; Endif

        Procedure PisaConstraintLengthLess(sp, length)
        If (sp.treeDepth < length)
         If(sp ∈ subsequence(spList))
          continue; Else remove;
         Endif
         return sp; Endif

        Procedure PisaConstraintLengthMore(sp, length)
        If (sp.treeDepth >= length)
         If(subsequence(spList) ∈ sp)
          continue; Else remove;
         Endif
         return sp; Endif

        Procedure PisaConstraintSubPattern(sp,subPattern)
        If(sp.label contains(subPattern))
         If(sp ∈ subsequence(spList))
          continue; Else remove;
         Endif
         return sp; Endif
         //This is the end of constraint checking procedure. }
```

Figure 4. PISA* algorithm

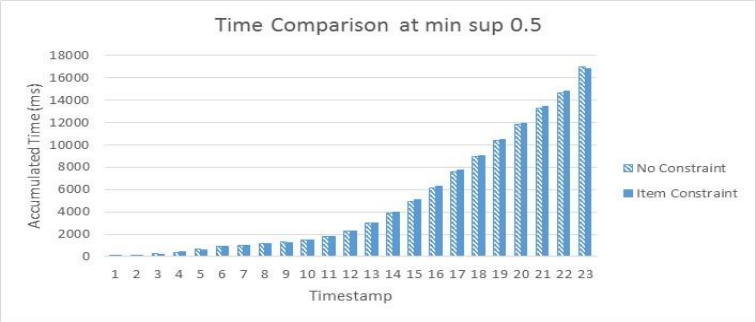


Figure 5. Comparison of accumulated execution time at minimum support 0.5

Second experiment was intended to compare the total accumulated execution time at last timestamp, which is 23, between PISA* and PISA at minimum support 0.5. It is shown in Figure 6 that total accumulated execution time from PISA* is lesser than PISA because PISA* only processes sequential patterns candidates that satisfy constraint. It meets the expectation that constraint checking reduces total accumulated execution time of PISA*, that adopts either item or length less or length more constraint.

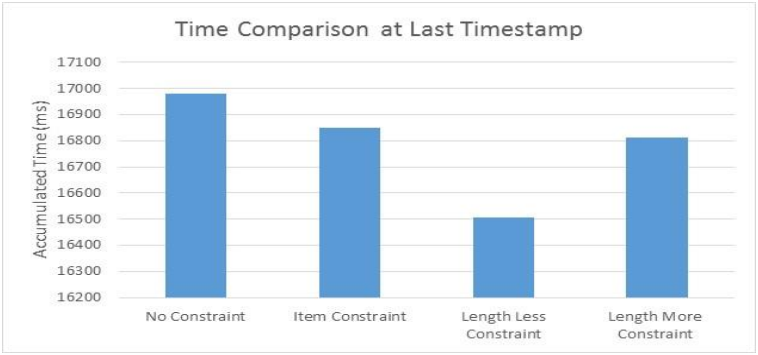


Figure 6. Comparison of accumulated execution time at last timestamp

The third experiment was conducted to compare the total accumulated execution time between PISA* and PISA at several minimum support. The result was shown in Figure 7 that PISA* consumes lower accumulated execution time than PISA at minimum support 0.6. This result meets the expectation that single constraint checking in PISA*, whether with item or length less or length more constraint, can reduce total accumulated execution time.

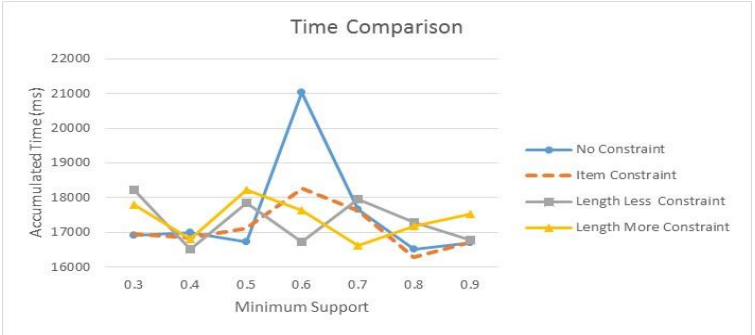


Figure 7. Comparison of accumulated execution time at different minimum support

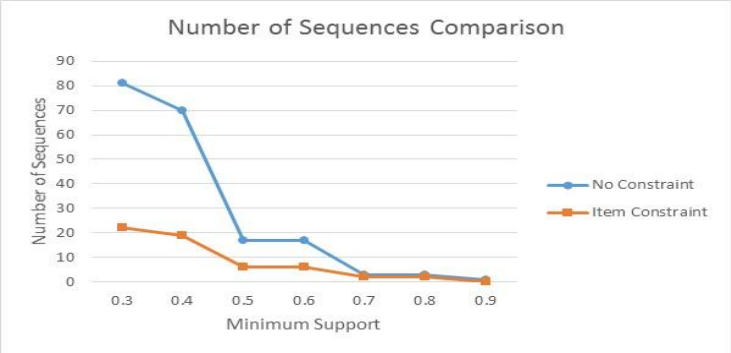


Figure 8. Comparison of number of sequences at different minimum support

The forth experiment was conducted to calculate the number of sequences patterns produced by PISA* and PISA. The experiment was also conducted at some minimum supports. The result was shown in Figure 8 by which that the number of sequences from PISA* is less than PISA. It is also concluded that constraint checking in PISA* reduces the number of sequential patterns. This is due to the fact that PISA* checks the sequential patterns candidates against the predefined constraint so that PISA* yields only sequential patterns that satisfy user's need.

5. Conclusion & Future Research

The experimental results proves that PISA* based on single constraint, either item, length less or length more constraint, gives better results in total execution time and the number of sequential patterns than PISA. Experiments using a limited test data have delivered good performance. PISA* can reduce the number of short, trivial and less meaningful sequential patterns. Moreover, those sequential patterns still satisfy user constraint.

This feature will be useful for further work to develop PISA based on multiple constraints that can accommodate more various types of constraints including time-based constraints. Moreover, there is an expectation that PISA* can also be utilized in preprocessing stage to support data mining main task for progressive data.

References

[1] J Huang, C Tseng, J Ou, M Chen. *A General Model for Sequential Pattern Mining with a Progressive Database.* In IEEE Transactions on Knowledge and Data Engineering. 2008; 20(9): 1153-1167.
[2] M Zhang, B Kao, D Cheung, CL Yip. *Efficient Algorithms for Incremental Update of Frequent Sequences.* In Pacific-Asia Conference on Knowledge Discovery and Data Mining. 2002: 186-197.
[3] H Cheng, X Yan, J Han. *IncSpan: Incremental Mining of Sequential Patterns in Large Database.* In 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2004: 527-532.
[4] D Yuan, K Lee, H Cheng, G Krishna, Z Li. CISpan*: Comprehensive Incremental Mining Algorithms of Closed Sequential Patterns for Multi-Versional Software Mining.* In 2008 SIAM International Conference on Data Mining. 2008: 84-95.
[5] J Woo. Apriori-Map / Reduce Algorithm.
[6] J Dean, S Ghemawat. *MapReduce: Simplied Data Processing on Large Clusters.* Proc. 6th Symp. Oper. Syst. Des. Implement. 2004: 137-149.
[7] K Shim. MapReduce algorithms for big data analysis. *Comput. Sci.* 2013; 7813: 44-48.
[8] PN Sabrina. *Multiple MapReduce and Derivative Projected: New Approach for Supporting PrefixSpan Scalability.* In 3rd ICODSE. 2016.
[9] MN Akbar, GAP Saptawati. *Scalable Sequential Pattern Mining Based on PrefixSpan for High Dimensional Data.* In 2016 International Conference on Data and Software Engineering (IcoDSE), 2016.
[10] X Yan. Design and Analysis of Parallel MapReduce based KNN-join Algorithm for Big Data Classification. *Indones. J. Electr. Eng. Comput. Sci.* 2014; 12(11): 7927-7934.
[11] MN Garofalakis. *SPIRIT: Sequential Pattern Mining with Regular Expression Constraints*. In 25th VLDB Conference. 1999: 223-234.

[12] S Vijayalakshmi. Mining Constraint-based Multidimensional Frequent Sequential Pattern in Web Logs. *Eur. J. Sci. Res.* 2009; 36(3): 480-490.
[13] J Pei, Han Jiawei, W Wang. *Mining Sequential Patterns with Constraints in Large Databases.* In Proceedings of the eleventh international conference on Information and knowledge management. ACM. 2002: 18-25.
[14] RY Yasmin, GAP Saptawati, B Sitohang. *Survey on sequential pattern mining.* In International Conference Information Technology and Business Application (ICIBA). 2013: 218-228.
[15] J Han, H Cheng, D Xin, X Yan. Frequent pattern mining: current status and future directions. Data Min. Knowl. Discov. 2007; 15(1): 55-86.
[16] RY Yasmin, GAP Saptawati, B Sitohang. *Classification Based on Constrained Progressive Sequential Pattern Mining : A Proposed Model.* In 2016 International Conference on Data and Software Engineering (IcoDSE). 2016.
[17] R Srikant, R Agrawal. *Mining Sequential Patterns : Generalizations and Performance Improvements.* In 5th International Conference on Extending Database Technology (EDBT '96). 1996: 1-17.
[18] DC Anastasiu, J Iverson, S Smith, G Karypis. Big Data Frequent Pattern Mining. *Freq. Pattern Min.* 2013: 225-259.
[19] D Wu, J Ren. Sequence Clustering Algorithm Based on Weighed Sequential Pattern Similarity. *Indones. J. Electr. Eng. Comput. Sci.* 2014; 12(7): 5529-5536.
[20] RY Yasmin, GAP Saptawati, B Sitohang. *Survey on Sequential Pattern in Preprocessing Phase for Knowledge Data Discovery.* In International Conference Information Technology and Business Application (ICIBA). 2013: 80-87.
[21] J Pei, J Han, P Helen, Behzad Mortazaviasl, Q Chen, U Dayal, MC Hsu. *PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth.* In 17th International Conference on Data Engineering. 2001: 215-224.
[22] M Mathankar, N Jathe, M Mathankar. Comparison of Algorithms to find Optimum Time For Sequential Pattern Mining with Progressive Database. *Int. J. Mod. Commun. Technol. Res.* 2014; 2(3): 17-20.
[23] S Jose. Constraint-Based Rule Mining in Large. *Dense Databases.* 2000; 4(2): 217-240.
[24] F Masseglia, P Poncelet, M Teisseire/ *Pre-Processing Time Constraints for Efficiently Mining Generalized Sequential Patterns.* In 11th International Symposium on Temporal Representation and Reasoning. 2004: 87-95.
[25] M Leleu, C Rigotti. *Constraint-Based Mining of Sequential Patterns over Datasets with Consecutive Repetitions.* In European Conference on Principles of Data Mining and Knowledge Discovery. 2003: 303-314.
[26] J Pei, J Han, W Wang. Constraint-based sequential pattern mining: the pattern-growth methods. *J. Intell. Inf. Syst.* 2007; 28(2): 133-160.