International Journal of Electrical and Computer Engineering (IJECE)
Vol. 7, No. 4, August 2017, pp. 2142~2151
ISSN: 2088-8708, DOI: 10.11591/ijece.v7i4.pp2142-2151

Classification with Single Constraint Progressive Mining of Sequential Patterns

Regina Yulia Yasmin, Putri Saptawati, Benhard Sitohang
School of Electrical Engineering & Informatics, Institut Teknologi Bandung, Jl. Ganesha 10, Bandung 40132, Indonesia

Article Info

Article history:

Received Dec 4, 2016
Revised Apr 4, 2017
Accepted Apr 18, 2017

Keyword:

CBS_CLASS*
Classification-by-sequence with single constraint pisa
PISA*
Sequential pattern mining
Single constraint progressive mining of sequential patterns

ABSTRACT

Classification based on sequential pattern data has become an important topic to explore. One of research has been carried was the Classify-By-Sequence, CBS. CBS classified data based on sequential patterns obtained from AprioriLike sequential pattern mining. Sequential patterns obtained were called CSP, Classifiable Sequential Patterns. CSP was used as classifier rules or features for the classification task. CBS used AprioriLike algorithm to search for sequential patterns. However, AprioriLike algorithm took a long time to search for them. Moreover, not all sequential patterns were important for the user. In order to get the right and meaningful features for classification, user uses a constraint in sequential pattern mining. Constraint is also expected to reduce the number of sequential patterns that are short and less meaningful to the user. Therefore, we developed CBS_CLASS* with Single Constraint Progressive Mining of Sequential Patterns or Single Constraint PISA or PISA*. CBS_Class* with PISA* was proven to classify data in faster time since it only processed lesser number of sequential patterns but still conform to user's need. The experiment result showed that compared to CBS_CLASS, CBS_Class* reduced the classification execution time by 89.8%. Moreover, the accuracy of the classification process can still be maintained.

Corresponding Author:

Regina Yulia Yasmin,
School of Electrical Engineering & Informatics,
Institut Teknologi Bandung,
Jl. Ganesha 10, Bandung 40132, Indonesia.
Email: regina.yasmin@gmail.com

1. INTRODUCTION

The classification process is needed to categorize data with various types for various purposes, for instance classifying web click data to find out the website search patterns of active internet users, classifying face or voice biometric data to recognize its type of source and classifying e-commerce data to categorize prospective customers etc. The large amount of data with high increasing growth rates and various data types, especially big data, require the development of more efficient classification method [1]. Moreover, the time order of data appearance is expected to bring knowledge that can be utilized to classify data. Classification method of data that appear in time sequence can be used to categorize the class of poker step sequences [2]. The study was conducted to determine whether a particular poker step will result in victory or defeat [2]. Classification of data that appear in time sequence is called a sequential data classification.

Sequential data as the input for classification process can be provided directly or preprocessed using a sequential pattern mining. Various studies on sequential pattern mining had been done with a variety of mining techniques [3], [4]. Several approaches in sequential pattern mining are Apriori [5], GSP [6], SPADE [7], Prefix Span [8], PISA [9] etc.

Research on integrating classification process with preprocessing using sequential pattern mining had been done, as well as integrating sequential pattern mining with clustering process. By integrating

sequential pattern mining in preprocessing with the next mining process, the performance was improved [10]. Research on classification of sequential patterns has been done by combining AprioriLike sequential pattern mining with the classification process. The approach was called CBS, Classify-by-Sequence [11]. Basically, AprioriLike algorithm searches for sequential patterns by generating sequence candidates that are constructed from the shortest sequences to get longer sequences iteratively.

However, AprioriLike algorithm consumed long time because the iterative process was repeated for lots of data. Moreover, the number of generated sequential patterns was overmuch and mostly they were short and trivial to users so that they burdened the next classification process. Hence, the classification process using AprioriLike sequential pattern mining took longer time. Moreover, from business point of view, users of data mining sometimes only require the data analysis from a certain perspective. This viewpoint is adopted from the organization needs. For example, users are generally more interested in getting knowledge of the latest data than of the old ones. Therefore, there's a necessity to improve sequential pattern mining performance in preprocessing by improving execution time to get sequential patterns and reducing the number of sequential patterns to only those that satisfy the user's need.

Constraint in sequential pattern mining is expected to answer this necessity. The sequential pattern mining algorithm is developed based on Progressive Mining of Sequential Patterns, PISA. PISA algorithm arranges a sequence database in the form of Progressive Sequence Tree, PS-tree, and does not generate sequence candidates so that the mining process becomes faster. We named the single constraint progressive mining of sequential patterns, as PISA*. In addition to its ability to generate sequential patterns that satisfy the users' need, constraint can reduce the number of short and trivial sequential patterns [13]. The lesser number of sequential patterns will reduce workload of the next classification process. It is believed that the classification process will be faster with better accuracy level. Therefore, this study is expected to provide solutions to the problem (1) whether sequential pattern mining, especially PISA* can lower the number of sequential patterns and, (2) improve classification time performance and maintain accuracy level of the classification process. In order to solve the problem, we propose a method Classify-By-Sequence_Class*, CBS_CLASS* which integrates PISA* to find sequential patterns and classification process.

The objective of this research is to classify data based on sequential patterns that were found using PISA*. Therefore, CBS_CLASS is modified to integrate PISA* algorithm and classification process. This study is expected to improve the performance of classification's total execution time while maintaining the accuracy level.

This paper is organised as follows. Section I contains introduction, related work, CBS approach and PISA*. Section II explains about CBS_CLASS* with PISA*: the proposed approach. Section III explains about the results and analysis. Section IV contains conclusion of this paper and future research.

1.1. Related Work

Sequential pattern mining is one of the development of frequent itemset mining [5]. 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 [14]. 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 $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$ [6].

Research on the mining sequence as features for classification had been done to classify steps sequence in order to predict whether the steps will lead to victory or defeat [2]. The study introduced a Feature Mining approach. In this study, there were 3 steps taken, i.e. (1) the repeated simulation step to generate execution traces, (2) mining the simulated execution traces to obtain features and (3) train classifier using features to predict the success or failure of simulated traces. This study used two processes, sequential pattern mining process to obtain sequential patterns that correlate with target class and the classification process using features obtained from the previous process [2]. However, CBS was proven to have better accuracy performance than the Feature Mining since CBS integrates AprioriLike sequential pattern mining with probabilistic induction so that sequential patterns can be used as a classifier rule [11].

1.2. Classify-by-Sequence, Cbs Approach

Classify-by-Sequence, CBS, is a classification process that builds model from sequential patterns. As shown in Figure 1, CBS performs two processes sequentially, (1) searches for sequential patterns through AprioriLike sequential pattern mining, (2) classifies based on the sequential patterns which act as a classifier rule [11]. Sequential patterns that are obtained from AprioriLike sequential pattern mining are also called as Classifiable Sequential Patterns, CSP. AprioriLike seeks sequential patterns with length-(n+1) from length-n. By the iterative process, AprioriLike requires a long time to get sequential patterns.

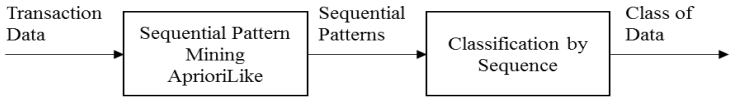

Figure 1. CBS Scheme [13]

Data that are processed in CBS are categorical data. Each record is associated to a particular class label. Denote D_i that consists of a time series of data associated with class n, $D = \{D_1, D_2, D_3, \ldots, D_n\}$. AprioriLike searches for Classifiable Sequential Patterns (CSP) as classifier rules. Therefore, a sequential pattern i corresponds to class m, $SP_i \rightarrow C_m$, with SP_i is a sequence of $a_1 \rightarrow a_3 \rightarrow a_7$ and C_m is the class m [11].

CBS algorithm is divided into two types, namely CBS_ALL and CBS_CLASS. CBS_ALL mines all data in the database while CBS_CLASS first divides the data based on corresponding class to be mined further. Since CBS_CLASS builds classification model from each class, CBS_CLASS was proven to achieve better accuracy compared to CBS_ALL [11]. The CBS_CLASS method is as follows, (1) divides the data based on the corresponding class or label, (2) from each class, AprioriLike searches the sequential patterns, (3) classifies data using the sequential patterns that act as Classifiable Sequential Patterns, CSP. The CSP is used as a classifier rule. To find sequential patterns, AprioriLike uses the minimum support. Each CSP will be given a score based on the length of sequential pattern. Then, CSP score is normalized so that the maximum score is 1 [11].

Figure 2 shows about CBS_CLASS algorithm that consists of AprioriLike algorithm and classification process [11].

```
Input:       dataset, minimum support
Output:      class of sequential patterns
Method:

CBS_CLASS(Dataset D, min_sup)
{
For each c_i ∈ class_set(D) do
D_i = class_dataset(D,c_i);
CSP_i = FindSP(D_i, min_sup) //to get CSP using AprioriLike
End
}


//Start program of AprioriLike algorithm for finding
sequential patterns
FindSP(Dataset D, min_sup)
{
Sp_1 = {large-1 items}
 For (i=2; Sp_{i-1};i++) do
       SP_C_i = gen_candidateSP(Sp_{i-1});
       For each data d ∈ D do
             SP_s = SP_C_i ∩ subseq(d);
             For each sp ∈ SP_s do
                   Sp.sup++;
             End
       End
 Sp_i = {sp | sp ∈ SP_s, sp.sup ≥ min sup}
End
 Return ∪_i SP_i
}
//This is the end of AprioriLike algorithm

//Start program for classifying test data
Class_of_sequence(x)
{
Total_score = new array[class_count(D)];
For each csp_i ∈ CSP do
Total_score[csp_i.class] += csp_i.sp.length;
End
Score_array[]=new array[class_count(D)];
For each csp_i ∈ CSP do
If csp_i ∈ subseq(x)
       For each c_m ∈ belong_class_set(csp_i)
             Score_array                y[m]                +=
csp_i.sp.length/total_score[m]; //count score for test data
       End
End if
End


k = index_of(max{score_array[]});
return c_k;
}
//End program for classifying test data
```

Figure 2. CBS_CLASS algorithm [11]

1.3. Progressive Mining of Sequential Patterns, PISA

A constraint is a limitation set by user before the mining process begins. Constraint in sequential pattern mining is expected to reduce the number of sequential patterns [13]. There are many types of user constraints, such as item, length, super-pattern, aggregate, regular expression, duration and gap constraint [15]. Constraint can be either (1) item constraint, is a limitation where a sequential pattern should contain or not contain certain item, (2) length less constraint, requires that the length of sequential pattern should be shorter than the given value, (3) length more constraint, requires that the length of sequential pattern should be longer than the given value, (4) super-pattern constraint, is a constraint adopted to look for sequential pattern that contains a particular pattern as a set of sub-pattern, (5) aggregate constraint, is a constraint on the aggregate of items in the pattern, (6) regular expression constraint, is a constraint with regular expression, such as disjuntion and Kleene closure of the set of items, (7) duration constraint, is a constraint on the sequence database that requires sequential pattern to satisfy the time difference between predetermined start and end transactions, (8) gap constraint, requires to satisfy predefined time difference between two adjacent transactions [15].

Based on how to check constraint on sequential patterns, constraints can be categorized into monotonic or anti-monotonic constraint. Constraint checking utilizes characteristic of subsequence and super sequence. A constraint, denoted by C_M, is said as a monotonic constraint if there is a sequence α meets C_M constraint then any super-sequence of α also satisfies constraint C_M. Meanwhile, a constraint, denoted by C_A, is said as anti-monotonic constraint if there is a sequence of α satisfy the constraint C_A then any subsequence of α also satisfies constraint C_A [15].

PISA represents a sequence database into a progressive sequential tree, PS-tree [9]. PISA uses the concept of a time frame named as Period of Interest, POI, which is a sliding window and move continuously in time [9]. POI made PISA to be flexible in adding or deleting data [9]. PISA processes sequence database that is composed from itemset from each sequence ID based on occurrence timestamp. PS-tree with PISA algorithm [9] records sequence ID, label and timestamp. PISA algorithm traverses from the root node to the leaf one and calculate the occurrence frequency of item. If the occurrence frequency of item is larger than minimum support then it will be considered as sequential pattern. However, original PISA approach has not accommodated constraint checking when searching for sequential patterns. Therefore, PISA* has been developed so that it is able to check single constraint.

2. CBS_CLASS* WITH PISA*: THE PROPOSED APPROACH

Classification CBS* with Single Constraint Progressive Mining of Sequential Patterns or PISA*, is a classification approach that integrates sequential pattern mining PISA* with classification process. Classification process needs sequential patterns as input. Figure 3 shows that sequential patterns were obtained from sequential pattern mining PISA* that was conducted before the classification starts. Single constraint PISA, PISA* searches for sequential patterns that meet a minimum support, a single constraint within the user-defined POI. Minimum support, single constraint and POI were set before sequential pattern mining process starts. The advantages of PISA* are (1) the flexibility in adding or subtracting data, (2) the conformance of sequential patterns with user's need, (3) the removal of short and trivial sequential patterns.

Sequential pattern will be used as a CSP, Classifiable Sequential Pattern that act as a classification rule, which is denoted as $Sp_i \rightarrow C_m$, sequential pattern-i belongs to the class m. Classification rules are used to determine the class of new data [11].

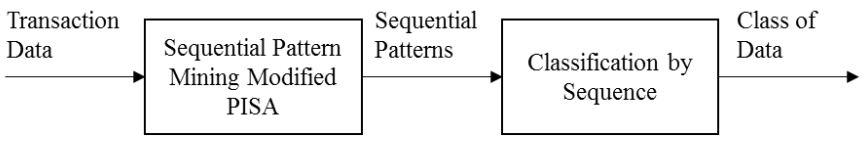

Figure 3. Classification by sequence with prior process PISA* [13]

This approach proved to perform better in searching for sequential patterns based on a minimum support and a single user-defined constraint. Single constraint that has been tested are constraint item, length less constraint and length more constraint. Moreover, this approach was also proven to decrease the number of short and trivial sequential patterns and performed in faster time than the original PISA algorithm.

Like CBS_CLASS, CBS_CLASS* also splits database based on the corresponding class. Denotes, D_m contains sequential data that corresponds to class m. Database is represented as

$D = \{D_1, D_2, D_3, D_4, \ldots D_n\}$, with n classes. Data sets in D_m consists of time-series data instances $\{i_{m1}, i_{m2}, i_{m3}, \ldots, i_{mn}\}$, where i_{mn} represents the value of D_m at the n^{th} time point. At each database that relates to corresponding class, PISA* seeks sequential patterns that meet minimum support and a single constraint. Sequential pattern obtained is called CSP, Classifiable Sequential Pattern. CSP is weighted based on the length of sequential pattern. Longer sequential pattern is given a greater weight. Maximum weight of each CSP is 1.

CBS_CLASS* with PISA* is explained as follows:

1. For each database D_m per class m, minimum support, single constraint and POI are predefined based on user requirements. Single constraint may be an item or length less or length more constraint. Find sequential patterns using PISA* as follows:
   a. Transform the database into sequence database that has property of sequence id, timestamp and label. Label is also called as itemset.
   b. Using Procedure Traverse: Develop and traverse PS-tree of itemset elements within POI. Every new element is recorded at the root node, while every element that has predecessor is recorded at common node. While traversing, check whether sequential pattern satisfies minimum support, which is the sequence list size ≥ minimum support * number of sequence.
   c. Using Procedure PisaConstraintItem: To obtain sequential patterns that satisfy the single constraint, check sequential pattern against single constraint in POI time frame. Check constraint based on property anti monotonic or monotonic constraint and apply it to all sequential patterns.
      For anti-monotonic constraint such as length less constraint, if sequence of α satisfies the constraint anti-monotonic C_A then any subsequence of α also satisfies constraint C_A. If the $SP_{Existing}$ satisfies the constraint and if SP is subsequence of $SP_{Existing}$ then SP also satisfies the constraint.
      For monotonic constraint such as item, length more, super pattern constraint, if there is a sequence α satisfies constraint monotonic C_M then any super-sequence of α also satisfies constraint C_M. If the $SP_{Existing}$ meets the constraint and if $SP_{Existing}$ is subsequence of SP or SP is super-sequence of $SP_{Existing}$ then SP also satisfies the constraint.
2. Using Procedure Class_of_sequence: Classify the data
   a. For each obtained sequential pattern or CSP, compute CSP score by accumulating length of sequential pattern.
   b. For each CSP, normalization of scores is counted. If CSP is a subsequence of the existing CSP array, then score = accumulated of (length of sequence / total score).
   c. Check the sequential patterns found from test data against list of CSP. If the corresponding class is failed to get, then check the subsequence of sequential patterns from test data, from the longest to the shortest length, against the list of CSP recursively until its correspondence class is found.
   d. Based on the scores obtained in point 4, index k will be obtained, where k is the maximum score. Class obtained is a class with index k.
      CBS_CLASS* with PISA* algorithm is described in Figure 4:

```
Input:       dataset, minimum support, POI and single constraint criteria, for example
item constraint
Output:      class of sequential patterns
Method:

CBS_CLASS*(Dataset D, min_sup, POI, label)
{
 For each ci ϵ class_set(D) do
       Di = class_dataset(D,ci);
       CSPi = PISA*(Di, min_sup, POI, label)
 End
}

PISA*(Dataset D, min_sup, POI, label)
{
1.     Var PS; //PS Tree
2.     Var currentTime; //timestamp now
3.     Var eleSet; //used to store elements ele
4.     While (there is still new transaction)
5.     eleSet = read all ele at currentTime;
6.     traverse(currentTime, PS);
7.     PisaConstraintItem(sp, label); //item constraint checking
8.     currentTime++;

//This is the start of modified 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)
                                   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)

       Child.seq_list.seq.timestamp=seq.timestamp;
                                          Else
                                                 Create a new sequence with
seq.timestamp;
                                          Endif
                                   Else //create a child
                                          Create a new child with element,
seq and seq.timestamp;
                                   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)
              Output the labels of path from Root to this node as a SP
End
```

Figure 4. The proposed CBS_CLASS* algorithm

3. RESULTS AND ANALYSIS

Experiment uses e-commerce sales data which contains categorical data. The amount of data is about 22,699 transactions with attributes are product ID, date of sales, customer ID, order ID. The objective of the experiments is to compare the classification time performance and the accuracy between CBS_CLASS and CBS_CLASS*. In order to get the accuracy level, k-fold stratified cross validation is used as testing scenario. In this experiment, CBS_CLASS uses original PISA method for searching sequential patterns, and CBS_CLASS* uses PISA* with single constraint.

Constraint used is either an item constraint or length less or length more constraint. Item constraint requires sequential patterns to contain 1 specific item. Length less constraint requires that length of sequential patterns must be less than a certain number. On the contrary, length more constraint requires that length of sequential patterns must be more or the same than a certain number.

First, data in each database D_m that corresponds to class m, were represented in sequence database based on sales date and time. Figure 5 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. Moreover, POI is used as a user defined time sliding window between specified sales hours interval. Therefore, based on sales hours, maximum number of POI is 23. In this experiment, POI is set of 5, means that time sliding window is set between 5 hours.

S01	A	BC		D			
S02	AB		C				
S03		A	B		C		
S04	BC		D				
S05		AB		D			
Sn							
SID	t1	t2	t3	t4	t5	tm	time

Figure 5. Sequential database representation in each D_m [13]

Sequential patterns are sequence of itemsets that contain:

a. Sequential pattern 1: <(product$_{11}$), <(product$_{12}$), ..., <(product$_{1n}$)>,
b. Sequential pattern 2: <(product$_{21}$), <(product$_{22}$), ..., <(product$_{2n}$)>,
c. Sequential pattern m: <(product$_{m1}$), <(product$_{m2}$), ..., <(product$_{mn}$)>.

In this experiment, classification process categorizes sequential patterns in two classes, prospect and non prospect products. For example, sequential patterns generated by PISA* that act as features for the classification process are as follows:

a. Class of prospect product :
   1) Sequential patterns 1: <(Samsung Galaxy V SM-G313HZ White), (Samsung Galaxy Tab S 8.4 "- SM-T705NT - Titanium Brown)>,
   2) Sequential patterns 2: <(Preorder Microsoft Lumia 535 Black), (Samsung Galaxy V SM-G313HZ White)> and so on.
b. Class of non prospect product :
   1) Sequential patterns 1: <(Emtec USB 2.0 Flash Drive 8 GB Black Panther), (V-Gen Micro SDHC 16 GB)>
   2) Sequential patterns 2: <(PNY Dual USB Flash Drive - OU1 16GB), (uNiQue Laptop Backpack i-Protect Biru), (V-Gen Micro SDHC 16 GB)> and so on.

The objective of the first experiment was to compare the number of sequential patterns from PISA in CBS_CLASS and PISA* from CBS_CLASS*. The experiment was conducted at minimum support 0.2. The result is shown in Figure 6 that number of sequential patterns obtained from PISA* in CBS_CLASS* are lower than in CBS_CLASS. From the result, the hypothesis was proven that CBS_CLASS* with prior PISA* process can reduce the number of sequential patterns to only those that satisfy user constraint.

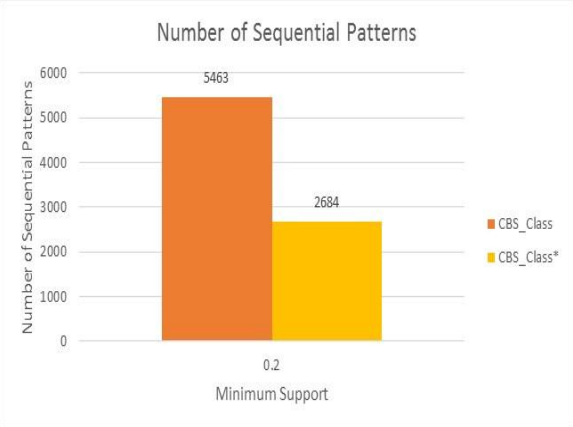

Figure 3. Comparison of number of sequential patterns at minimum support 0.2

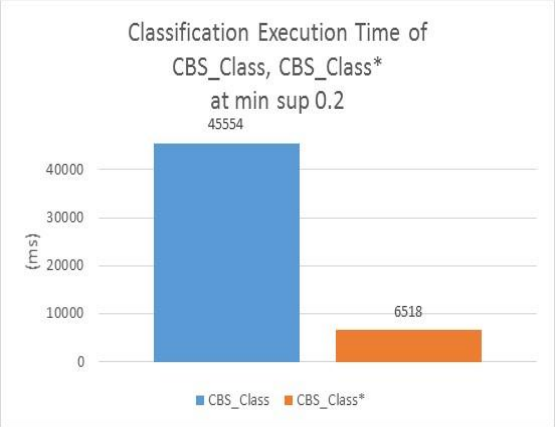

Figure 4. Comparison of classification execution time at minimum support 0.2

The objective of the second experiment was to compare the classification execution time between CBS_CLASS and CBS_CLASS*. The experiment was conducted at minimum support 0.2. The result is shown in Figure 7 that classification execution time of CBS_CLASS* is much lower than CBS_CLASS. From the result, the hypothesis was proven that CBS_CLASS* with prior PISA* process can reduce the classification execution time since classification processes lesser number of sequential patterns so that they do not burden the classification process.

The third experiment was aimed to compare the accuracy level between CBS_CLASS* and CBS_CLASS at minimum support 0.2. As shown in Figure that the accuracy level of CBS_CLASS* is higher than CBS_CLASS. From the result, it can be concluded that the lesser number of sequential patterns that satisfy user's constraint increases the accuracy of classification. Moreover, subsequence checking of test data to the list of CSP increases classification's accuracy level. It meets the expectation that besides reducing classification execution time, CBS_CLASS* is capable of maintaining the accuracy level of classification process.

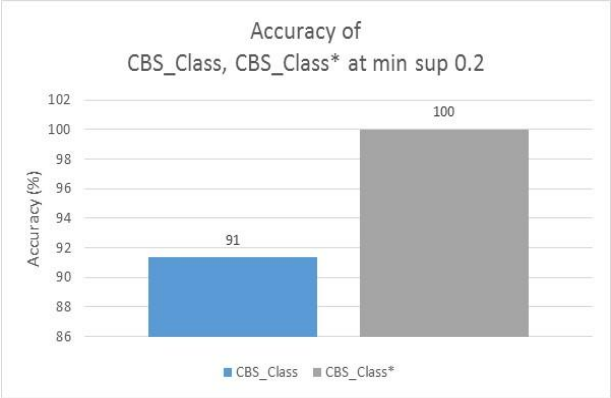

Figure 8. Comparison of classification's accuracy at minimum support 0.2

The forth experiment was conducted to know the comparation of classification execution time between CBS_CLASS* and CBS_CLASS at several minimum supports. The result was shown in Figure 9 that CBS_CLASS* consumes lower classification execution time than CBS_CLASS at several minimum support. It meets the expectation that CBS_CLASS* is more efficient than CBS_CLASS.

Figure 9. Comparison of classification execution time at different minimum support

The fifth experiment was aimed to compare the accuracy level between CBS_CLASS* and CBS_CLASS at several minimum support. It is shown in Figure 10 that CBS_CLASS* can maintain the

accuracy level, compared to CBS_CLASS. It is concluded that CBS_CLASS* can lower the execution time and maintain the accuracy level as well.

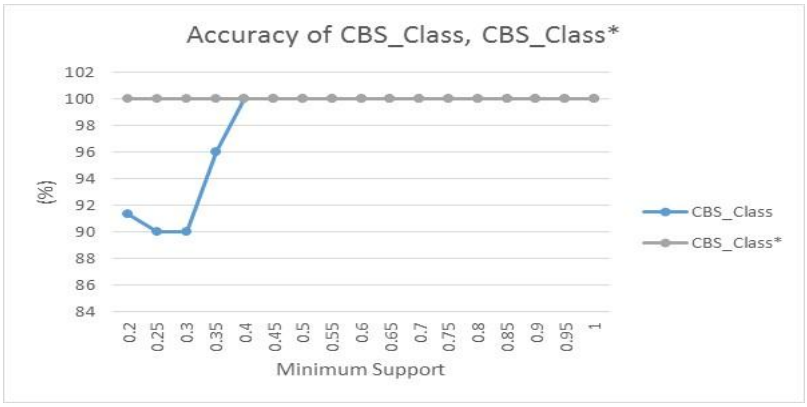

Figure 10. Comparison of number of sequences at different minimum support.

4. CONCLUSION

The experiment results proved that the proposed method CBS_CLASS* gives better results in classification execution time than CBS_CLASS significantly and also can maintain the accuracy level. Compared to CBS_CLASS, CBS_CLASS* can reduce the classification execution time by 89%. This is due to CBS_CLASS* only processes lesser number of sequential patterns that satisfy user's constraint so that classification execution time is reduced. Moreover, subsequence checking of test data to the list of CSP can increase the classification's accuracy level.

This result gives good expectation for further research to incorporate multiple constraints in sequential pattern mining and integrate it with classification process. We believe this approach can yield much better performance.

ACKNOWLEDGEMENTS

We would like to thank Prof. Sitohang and Dr. Saptawati for the guidance, assistance with this research and comments that greatly improved the manuscript.

REFERENCES

[1] X. Yan, "Design and Analysis of Parallel MapReduce based KNN-join Algorithm for Big Data Classification", *Indones. J. Electr. Eng. Comput. Sci.*, vol. 12, no. 11, pp. 7927–7934, 2014.
[2] N. Lesh, M. J. Zaki, M. Ogihara, *"Mining Features for Sequence Classification"*, in Proceedings of the fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, AC*M*, 1999, pp. 342–346.
[3] R. Y. Yasmin, G. A. P. Saptawati, B. Sitohang, "Survey on Sequential Pattern Mining", in *ICIBA*, 2013.
[4] D. Wu, J. Ren, "Sequence Clustering Algorithm Based on Weighed Sequential Pattern Similarity", *Indones. J. Electr. Eng. Comput. Sci.*, vol. 12, no. 7, pp. 5529–5536, 2014.
[5] R. Agrawal, R. Srikant, *"Mining Sequential Patterns"*, in Proceedings of the Eleventh International Conference on Data Engineering, 1995, pp. 3–14.
[6] R. Srikant, R. Agrawal, *"Mining Sequential Patterns : Generalizations and Performance Improvements"*, in 5th International Conference on Extending Database Technology (EDBT '96), 1996, pp. 1–17.
[7] M. J. Zaki, "SPADE : An Efficient Algorithm for Mining Frequent Sequences," *Mach. Learn.*, vol. 42, no. 1–2, pp. 31–60, 2001.
[8] J. Pei, J. Han, P. Helen, Behzad Mortazavi-asl, Q. Chen, U. Dayal, M.-C. Hsu, "PrefixSpan : Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth," 2001.
[9] J. Huang, C. Tseng, J. Ou, and M. Chen, "A General Model for Sequential Pattern Mining with a Progressive Database", in *IEEE Transactions on Knowledge and Data Engineering*, 2008, vol. 20, no. 9, pp. 1153–1167.
[10] R. Y. Yasmin, G. A. P. Saptawati, and B. Sitohang, "Survey on Sequential Pattern in Preprocessing Phase for Knowledge Data Discovery", in *ICIBA*, 2013.
[11] C. Lee, *"CBS : A New Classification Method by Using Sequential Patterns"*, in Proceedings of the 2005 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2005, pp. 596–600.
[12] R. Y. Yasmin, G. A. P. 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.
[13] J. Han, H. Cheng, D. Xin, X. Yan, "Frequent Pattern Mining: Current Status and Future Directions", *Data Min. Knowl. Discov.*, vol. 15, no. 1, pp. 55–86, Jan. 2007.
[14] 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, pp. 18–25.