Mining Frequent Sequential Rules with An Efficient Parallel Algorithm

Nesma Youssef[1,3], Hatem Abdulkader[3], and Amira Abdelwahab[2,3]
[1]Department of Information System, Sadat Academy for Management Science, Egypt
[2]Department of Information Systems, King Faisal University, Saudi Arabia
[3]Department of Information Systems, Menoufia University, Egypt

Abstract: *Sequential rule mining is one of the most common data mining techniques. It intends to find desired rules in large sequence databases. It can decide the essential information that helps acquire knowledge from large search spaces and select curiously rules from sequence databases. The key challenge is to avoid wasting time, which is particularly difficult in large sequence databases. This paper studies the mining rules from two representations of sequential patterns to have compact databases without affecting the final result. In addition, execute a parallel approach by utilizing multi core processor architecture for mining non-redundant sequential rules. Also, perform pruning techniques to enhance the efficiency of the generated rules. The evaluation of the proposed algorithm was accomplished by comparing it with another non-redundant sequential rule algorithm called Non-Redundant with Dynamic Bit Vector (NRD-DBV). Both algorithms were performed on four real datasets with different characteristics. Our experiments show the performance of the proposed algorithm in terms of execution time and computational cost. It achieves the highest efficiency, especially for large datasets and with low values of minimum support, as it takes approximately half the time consumed by the compared algorithm.*

Keyword: *Non-redundant rule, multi-core processors, dynamic bit vector, closed sequential patterns, sequential generator pattern.*

Received December 6, 2020; accepted April 28, 2021
https://doi.org/10.34028/iajit/19/1/13

1. Introduction

Pattern mining is a data mining technique that includes two subsections called itemset mining and Sequential Pattern Mining (SPM). The extraction of frequent sequences from large datasets is known as SPM. It has only one measure named minimum Support (minSup) that calculates existence items in sequence databases. It may be deceptive and insufficient to make a prediction.

Sequential Rule Mining (SRM) is proposed as a solution to the limitation of SPM. It takes into account another measure called; minimum confidence (minConf) that calculates the expectation of the following pattern. It aids in predicting situations by identifying relationships between sequential items occurrences. The mining process consists of two stages: first, mining frequent sequential patterns Fournier-Viger *et al*. [4]. Second, produce sequential rules which depend on the first phase. So, many researchers concentrate on enhancing the efficiency of mining sequential patterns Upadhyay *et al*. [26]. There are three concise representations for mining sequential patterns; Closed Sequential Patterns (CSP), Sequence Generator Patterns (SGP), and maximal sequential patterns. Most recent enhanced algorithms intend to discover non-redundant sequential rules based on only one representation. This method depends on data structure with many tasks implementing the same process. It suffers from consuming time and causes CPU idle, especially with low values of minSup. However, generating sequential rules based on compact information to reduce the time required for generating the rules as in closed sequential patterns requires expensive computations on cost and time. As a result, an enhanced algorithm for mining rules is required, particularly for large databases and with low values of minSup.

Utilizing two types of patterns concise (CSP and SGP) can resolve previous obstacles of producing many candidates. First, it can reduce the search space with the lowest values of minSup. Second, it can assist users analyze the sequences in large datasets efficiently.

Another aspect of improving the efficiency of sequential rules is using a parallel approach. One method to perform this approach is a multi-core processor. It can increase performance by applying many tasks simultaneously. So that large datasets can be analyzed in a reasonable time.

In this paper, we propose an enhanced algorithm for mining non-redundant sequential rules as follows:

- Performing sequential pattern mining based on blinding closed with sequential generator patterns in the same procedure. It aids in acquiring more information with a more compact database than mining the complete set of sequential patterns.

- Multi core processor architecture is performed on only two procedures of the proposed algorithm to achieve the highest possible efficiency.

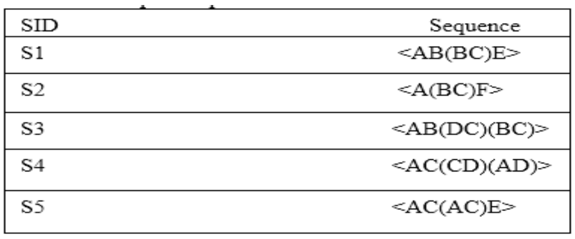
The rest of this paper is organized as follows: Section 2 introduces the definition of the problem and related work. Section 3 shows the data structure in a parallel approach. Section 4 represents the proposed algorithm. Section 5 discusses the experimental results. The conclusions are presented in section 6. Finally, recommendations and future works are summarized in section 7.

2. Background

2.1. Problem Definition

Suppose I={i_1, i_2,….., i_n} is a collection of n distinct items, and I_j are items or events where 1≤ j ≤ n. An itemset X is a set of unordered items that are represented with brackets. Such as (AC) indicates an itemset with two items ‘A’ and ‘C’. A sequence includes an ordered itemset denoted S = {I_1, I_2,…, I_m} where Ij is an itemset and1 ≤ j ≤ m. The number of events in a sequence named the sequence length. The frequent first sequence refers to F1-S; for example, the sample sequence database in Figure 1.

- *Definition* 1: (prefix and postfix), set two sequences s_y=<y_1, y2……y_n> and s_z=<z_1,z_2…..z_m>, (m≤n). The sequence s_z is a prefix of s_y if only z=y for (1≤ i ≤ m). We indicate s_z is a subsequence of s_y, and s_y refers to the super sequence of s_z as s_y⊂s_z.
- *Definition* 2: (substring of length),suppose s_y is a sequence and sub n, m(s_y), (n≤m) refer to the portion from locus n to locus m of s_y. Substring of length calculated as (m-n+1). For example, sub3,5 of (<AB(AC)D>) is <(AC)D>.
- *Definition* 3: (frequent sequence), a support measure is the number of included items in a database divided by |DB|. Frequent sequences are composed of sequence items that satisfy the minimum support threshold (minSup) and are referred to as sup (S_b) >=minSup. When given a minimum support threshold, the main challenge is to find the complete set of frequent subsequences.
- *Definition* 4: (frequently closed sequence), respect two frequent sequential patterns; s_y and s_z. Frequent Closed Sequential Patterns (FCSP) indicate that no s_z such that s_y⊂s_z ^ sup(s_y) =sub (s_z), s_y. FCSP is more compressed than mining the whole set of sequential patterns. It keeps information fully extracted without affecting the results. If the subsequence (sy) has the same support value of the super sequence (sz); (sy) will be ingested by (sz).
- *Definition* 5: (frequent generator patterns) if there is no subsequence s_y with the same support of the s_z; this called Frequent Sequential Generator Patterns (FSGP). FGSP is shorter than FCSP and indulgent to the noise produced in training data. It has more accuracy for the classification of the sequence.
- *Definition* 6: (combine FCSP with FGSP), based on definitions 4 and 5, it takes advantage of fetching additional information. The FCSP can't produce alone non-redundant sequential rules with higher accuracy.
- *Definition* 7: (sequential rule and frequent sequential rules) suppose r is a sequential rule in the form x→y, which states that if x happens, y will obey. Consider x, y are frequent sequences, and r has two measures (Sup,Conf).The support value is referred to as (minSup) that equals sup(X+Y), where the confidence value is indicated as (minConf) that regards the probability of the following patterns andequal to sup(X+Y)/sup(X). A rule that achieved a minSup threshold is named a frequent sequential rule. While; the rule that reached a min Conf value is a strong sequential rule.
- *Definition* 8: (redundant rule and non-redundant rule), if a rule can be deduced by another rule, it regards a redundant rule. For example, suppose two rules r_1:< B>→< (AC) CD> and r_2: <B>→<CD>. So, r2 is a redundant rule if it is included in r1 and has the same support as r1. Otherwise, a rule is non-redundant if x+y∈FCS and x∈k` such as k⊂k^sup (k`) =sup (k) referred to as a prefixed generator.



SID	Sequence
S1	<AB(BC)E>
S2	<A(BC)F>
S3	<AB(DC)(BC)>
S4	<AC(CD)(AD)>
S5	<AC(AC)E>

Figure 1. Sample sequence database.

2.2. Related Work

Sequential Pattern Mining is a significant phase for producing sequential rules. It is proposed by Ravikumar *et al.* [20] within the Apriori All technique. The purpose of this phase is to discover all sets of patterns from large search spaces. It includes two principal processes called s-extension and i-extension that are needed for growing the patterns.

The GSP algorithm appeared to solve the database's multiple scans problem that caused complex computations in cost and time. It is determined by the horizontal database and the downward closure property Titarenko *et al.* [25]. Many database searches, non-existent candidates, and holding candidates in memory are just some of the drawbacks. As a result, the depth-first search algorithms have been developed (vertical format). As an example, Spade Zaki [33], Prefix Span Mollenhauer *et al.* [15], Spam Wang and Cao [29], LapinSpam Alja’am *et al.* [1], CM-Spam, and CM-Spade Naseer and Malsoru [16]. The algorithms can increase the mining performance by searching the database once and calculating the pattern's support quickly. There are also various short representations of producing sequential patterns with higher precision and less time required than mining all sequential patterns Fournier-Viger, *et al*. [4]. Closed sequential patterns are one of these representations. It is compact with fully extracted information.

A sequential generator pattern is the other succinct representation. It's more tolerant regarding the noise in candidate patterns. Gen Miner Wu *et al*. [30], FSGP Mukhlash *et al*. [15], and VGEN Husák *et al*. [8]. Closed patterns are associated with generator patterns to generate a sequential rule that provides more detail than closed patterns alone Guyet [7]. However, still consumes time in building the candidate patterns.

Mining non-redundant sequential rules, such as CNR, Patel and Malviya [17] Spiliopoulou [21] is also more efficient than mining sequential rules. Several researchers have improved SRM algorithms by relying on the prefix tree to enhance their efficiency. They reducing search space as included in MNSR and IMSR pre-Tree. However still suffer from efficiency regarding the consuming time Pham *et al.* [19] and Van *et al*. [27] particularly for large databases. There were two directions for most researchers to improve sequential rule mining algorithms. The first direction is using the set of procedures that deletes un-candidate patterns from the beginning, thus enhancing the efficiency of the generated rules and improving memory usage. However, it was going through many steps, leading to an increase in the required time. The second direction is appending an additional constraint for producing an extension of the previous algorithm, as it is in the TRuleGrowth algorithm. It considers window size parameters to control the maximum number of the generated rules, but it proved limited efficiency with the large datasets Youssef *et al*. [32].

Numerous researchers fixed this problem by using a parallel technique to minimize the processor's execution time. Pspade Zhou *et al*. [34], PHUSP Zihayat *et al*. [35], and other distributed memory-based parallel algorithms have appeared. In addition, use a parallel approach in generating sequential pattern mining to reduce run time and memory consumption, but it has limitations by causing data corruption and overheating Huynh *et al*. [10].

The multicore processor has emerged as a result of enormous developments in modern processor engineering, and it has proven to be effective in improving performance. It allows several tasks completed at the same time Jamsheela and Gopalakrishna [11], Huynh *et al*. [9] Parallel algorithms have been used to solve several problems, including improving load balancing and achieving lock-free parallelism Le *et al*. [13]. Various researches have been proposed for improving association rule mining algorithms using a multitasking approach to avoid time-consuming, but they are still mining all patterns in the sequence database TAŞER *et al.* [24] Suresh Kumar and Thangamani [22], Kuriakose and Nedunchezhian [12].

Although these algorithms outperform serial algorithms efficiently, they require a lot of memory and go through multiple phases to construct the data structure. There is no previous research that concentrated on generating sequential rules and improving sequential pattern mining simultaneously. The mining process passes through many tasks executes the same processes that consume time and using a lot of memory. There is a constant needed to gain knowledge, predict patterns, and discover sequential rules in a reasonable time. Enhancing the efficiency of the sequential rule mining algorithms is one solution for this challenge.

3. Data Structure for Parallel Mining NRD-Rules

3.1. Architecture of Multi-Core Processor

Large datasets are searched in a short amount of time using a parallel mining method. A multi-core processor is one way to implement this strategy. On a single chip, it has several processor cores. It can improve productivity by doing several tasks at once.

Two or more autonomous cores are located in the same physical package in a multi-core processor Upadhyay *et al*. [26] Czarnul *et al*. [2]. As seen in Figure 2, each processor core has its memory and shares the main memory.

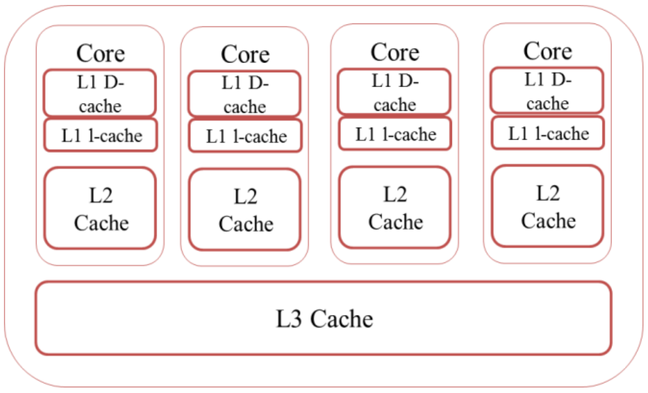


Figure 2. Quad-core processor diagram.

Each core has its L1, and L2 cache, with a single L3 cache for all cores. The highest resource use is accomplished by sharing more resources between cores. The data duplication is decreased through the shared cache. As a result, communications will be more efficient. We can implement it through the multiprocessing package in a python that performs multiple tasks simultaneously on a multi-core CPU. There are four modules included in Python for achieving parallelization. The first module is the threading module that suitable for I/O operations.

When a processor is waiting for the data from remote resources, it is in idle mode. It performs inefficiently for basic tasks. It increases the complexity degree of the program, and causes overhead during managing threads. Second, is utilizing the system method in the OS module. It allows the external command-line programs to run in a separate process. It causes overhead and makes it much more expensive than others. Third, the subprocess module that is depending on facilitates spawning processes. It connects them through signals and gathers the produced output with their fault message. It doesn't protect operations from the fault message. Fourth, the multiprocessing package can perform parallel execution through processes, pools, queues, and pipes. It provides both local and remote synchronization. It can effectively lock the global interpreter by using subprocess rather than threads.

The current study proposes apNRD-CloGen algorithm for parallel mining to enhance the execution time based on the multi-core processor. Utilizing the multiprocessing module with Pool class has the advantage of controls the execution of processes. It supports either synchronous or asynchronous parallel implementation. Figure 3 shows the applied method for the proposed parallel algorithm.

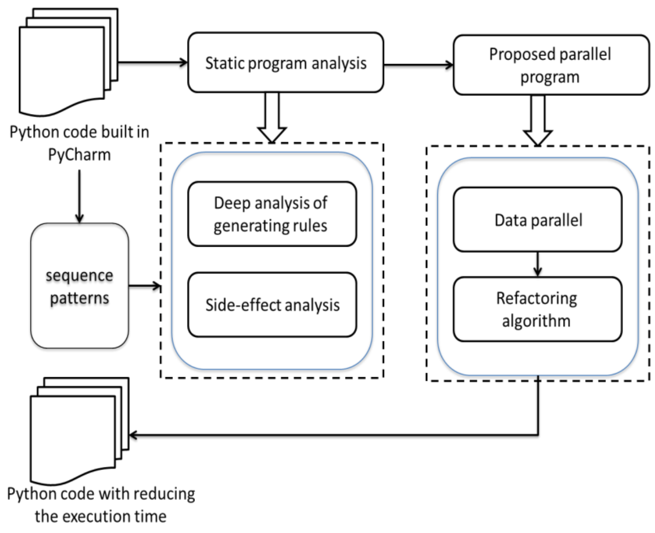

Static program analysis
Proposed parallel program
Python code built in PyCharm
Deep analysis of generating rules
Data parallel
sequence patterns
Side-effect analysis
Refactoring algorithm
Python code with reducing the execution time

Figure 3. Parallel approach for a proposed PNRD-CloGen algorithm.

3.2. Generators with Closed-DBV Patterns

3.2.1. Closed-DBV Patterns

Closed sequential pattern algorithms that rely on a vertical format are more efficient than those dependent on a horizontal data format Gan *et al*. [5]. It has a significant advantage in that it scans the database only once and speedily calculates the support count of sequences. Those formats store additional information. So, it demands a great deal of memory.

A bit vector solves this problem that represents itemsets as several transactions Xie and Tan [31]. However, it has an obstacle that increases the required memory and execution time because depending on a fixed size. The size depends on the appearance of each item that is expressed by '1' otherwise '0'. To overcome this problem, they proposed a Dynamic Bit Vector (DBV). After clearing the '0' bits at the beginning and end of the vector, it generates a bit vector. It improves efficiency for item sets with many '0' bits Tang *et al*. [23] Gokulapriya and Kumar [6] Consider the case where the sequence database has 16 transactions. It denotes the presence of two things, one with a '1' and the other with a '0'. As seen in Figure 4, the bit vector takes 16 bytes to store this operation, while DBV only considers the start of the index and the series from the first to the last '1'.The proposed algorithm uses the DBV structure to save sequences in vertical format. It facilitates the computations of sequence support. By combining the location of two items concerning the position of each item, it performs well in the ANDING operation. It provides a more accurate result. When performing the ANDING operation of item <C> to obtain<CB>, the result of DBV only is (111). When we consider the structure of DBV, terms of position, it is clear that it does not include the <CB>, and the correct result is (00000). The ANDING method should produce the precise outcome, which is (111) for <BC>. For example in Figure 1, item <B> in sequence database that existence in sequences 1, 2, and 3. The bit vector for <B> is (1,1,1,0,0) that the first appearance of item <B> in sequence 1 in the second and third position denoted as 2:{2,3}. Figure 5 represents the DBV structure of item<B>.

0	0	0	0	0	1	1	0	1	1	0	1	0	0	0	0

Figure 4. Example of bit vector.

Item	<B>		
Start bit	1		
Index	1	2	3
List of position	2:{2,3}	2:{2}	3:{3}

Figure 5. DBV structure for item <B>.

3.2.2. Generators Patterns

Sequential generator patterns are more compressed than closed sequential patterns. Numerous algorithms are proposed, such as the GenMiner algorithm. It has three-phases; compact /generate/ and filter Veroneze *et al*. [28]. The FEAT Mukhlash *et al*. [15] and Mollenhauer and Atzmueller [14] algorithm that relies on growth with forward and backward pruning techniques helps speed up the mining process. Mining Sequential Generator Patterns (MSGPs) Wu *et al*. [30] algorithm depends on sequential generator patterns feature and sequence extensions to discover all generator patterns Upadhyay *et al.* [26]. Generator patterns are the prioritized representation of the classes in classification and model selection.

3.2.3. Blend Sequential Closed with Generator Patterns

Using closed with sequential generator patterns can add more information than closed alone can't be provided. It can generate non-redundant rules with a high degree of Precision.

CSGM method is used to mine both patterns simultaneously, but it consumes much time to form databases Pham *et al*. [18]. So we propose the pNRD-CloGen algorithm to mine both type of sequential patterns and prove the higher performance in runtime.

4. The Proposed Algorithm

The proposed parallel NRD-CloGen (pNRD-CloGen) algorithm is described in this section. It performs in a parallel approach and depends on mining patterns from two concise representations. The first representation is frequent closed sequential patterns that use DBV data structure. The second representation is sequential generator patterns.

In the parallel mining approach, each branch in the prefix tree is distributed as a single task. Each node is dealt with as a separate task, and the total computations can perform in a parallel approach. Then, tasks are distributed within the available processor cores and independently handled to generate pNRD-CloGen rules.

The pNRD-CloGen algorithm consists of six steps:

1. Converting sequence dataset to DBV structure. Producing the first frequent sequence and assigning them in a prefix tree.
2. Utilizing a parallel approach for generated sequential patterns.
3. Pruning the prefix generator.
4. Sequence extension.
5. Generating pNRD-CloGen rules as shown in Figure 6.

The pseudo-code of the pNRD-CloGen algorithm is shown in Algorithm (1). The DB is scanned to create the list of items. Next, discovers all Frequent First Sequences (F1-S) that achieve the minSup threshold. The items in F1-S are set as child nodes in the prefix tree. Then, pNRD-CloGen creates new tasks for each node of the root. Each task implements the method CloGen-extension.

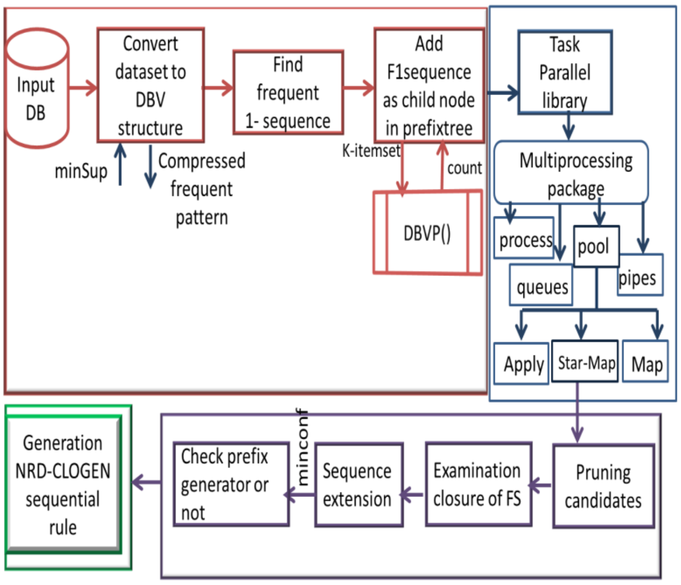

Input DB
Convert dataset to DBV structure
Find frequent 1- sequence
Add F1sequence as child node in prefixtree
minSup
Compressed frequent pattern
K-itemset
count
DBVP()
Task Parallel library
Multiprocessing package
process
pool
queues
pipes
Apply
Star-Map
Map
Generation NRD-CLOGEN sequential rule
Check prefix generator or not
minconf
Sequence extension
Examination closure of FS
Pruning candidates

Figure 6. Framework of a proposed pNRD-CloGen algorithm.

Multiple tasks run in parallel approach to producing frequent CloGen sequential patterns. Finally, generate NRD-SR that achieves the minConf threshold.

Algorithm 1: NRD-pCloGen algorithm

```
Input: SDB, minSup, minConf
Result: NRD-SR
1 Scan DB to create an ID list of items.
2 Calculate DB length& set the
3 minSup value
4 Find F1-S>= minSup
5 Establish DBV structure in a dictionary to have
6 the positions of each item
7 Add F1-S as child nodes, where node=null
8 For each node do
9     Create a new task, then call pCloGen-
10    extension.
11 Set minConf & generate NRD-SR
```

Consider the database in Figure 1, and set minSup equal to 0.5. The following procedure is executed to find the F1-S, as F1={a:5, b:3, c:5}. The procedure that extends the F1-S is shown in Algorithm (2), named CloGen pattern-Extension. This procedure sets the F1-S as child nodes in the prefix tree.

We apply the pruning technique in a parallel approach to eliminate candidates that cannot extend frequent patterns. For example, consider the DB in Figure 1. It is not required to extend prefix ‘B’, as there is an item ‘A’ which has (start position=1) that usually occurs before an item ‘B’ which has (start position=2). The prefix ‘A’ extension already has ‘B’ with the same support. So, ‘B’ will be absorbed. After that, the CloGen pattern extension procedure extends patterns by rendering sequence extension to produce new sequential patterns through a parallel approach.

Algorithm 2: CloGen pattern-Extension

```
Input:nodes in prefix tree and minSup
Output: set of frequent CloGen patterns
12    set listNode as child nodes
13    For each SP in listNode do
```

```
14        If SP is pruned then
15           set SP as SP node
16           set Spa as SP child node
17         If (sup (Spa)≥minSup)then
18           Add Spa as child node of SP
19        check and set the attribute of SP
20        (closed or prefixed generator)
```

Algorithm 3. pCloGen algorithm

```
Input: nodes, F1-S, MinSup
Output: all child nodes ready to generate rules
22   Set process equal to CPUs as pool
23   Set parameters of listNode as Childs
24   Get child node by star-map () in pool
```

The parallel approach is implemented in python by using the multiprocessing package. The package executes many tasks simultaneously depends on Central Processing Unit (CPU) cores. It utilizes a Pool object to perform many tasks and passes them to the Pool object. Starmap () function. This function can accept multiple arguments. So, the user can overcome multiple assumptions of one item having in the list, as seen in Algorithm (3).

After discovering all frequent CloGen patterns, the algorithm initializes to generate all NRD sequential rules, as shown in Algorithm (4). This algorithm produces all rules through a sub-tree. Each child node may be a prefix or closed. The prefix is pre and closed as post of the rule. It performs recursively in parallel until child nodes don't achieve the minConf value.

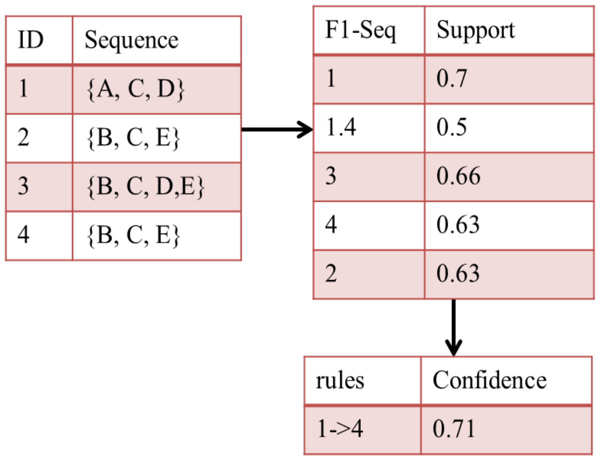
Algorithm 4: Generate pNRD-CloGen Rule

```
Input: root, minConf
Output: pNRD-CloGen Rule
25   Set prefix as a sequence of the root
26   Set sub-node as child node
27   For SP node in the root
28      Search prefix (SP node, rules, minConf)
29If a node is a prefix
30            Search closed (value of a node, nod,
31            rules, minConf)
32               For a child of a children node
33                  search prefix (child, rules,
34                  minConf)
35         If a sequence is closed
36            Conf= Sup of seq (Sup)/pre(Sup)
37               If Conf ≥ minConf
38                  Set rules = anterior → posterior
39                   Append the rule
40         Else stop producing rules
41         End if
42               End for
43          Call generate NRD-CloGen Rule
44   End for
```

5. Experimental Results

5.1. Experiments on a Sample Dataset

We assess the reliability of coding the proposed algorithm in python by using sample data. A sample data set includes 30 transactions with five consumer behavior items that include the purchase of various items. We performed it and have the result manually, then compared the results with the results of the python code. It has proved the validity of the code the emergence of the same results. Set the minSup=0.5 and minConf=0.5, Figure 7 shows the phases of implementing the algorithm. As shown in Figure 8, the tested code prints the number of transactions, objects, the F1-S count, and the CloGen patterns in the tree, generated sequential rules, and the spending time for each operation.

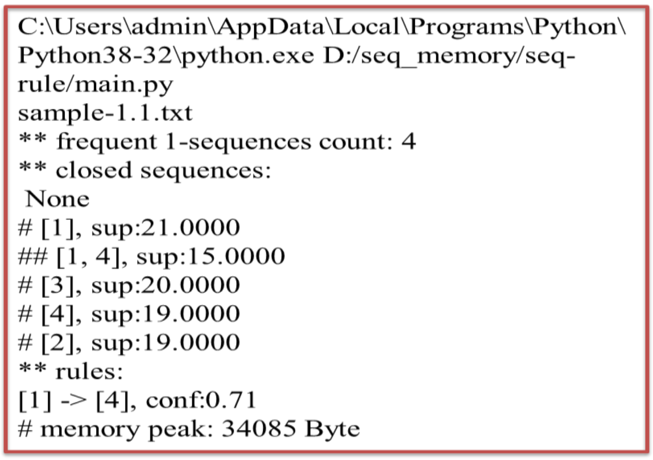


ID	Sequence
1	{A, C, D}
2	{B, C, E}
3	{B, C, D,E}
4	{B, C, E}

F1-Seq	Support
1	0.7
1.4	0.5
3	0.66
4	0.63
2	0.63

rules	Confidence
1->4	0.71

Figure 7. Sample execution of the NRD-DBV algorithm.



```
C:\Users\admin\AppData\Local\Programs\Python\
Python38-32\python.exe D:/seq_memory/seq-
rule/main.py
sample-1.1.txt
** frequent 1-sequences count: 4
** closed sequences:
 None
# [1], sup:21.0000
## [1, 4], sup:15.0000
# [3], sup:20.0000
# [4], sup:19.0000
# [2], sup:19.0000
** rules:
[1] -> [4], conf:0.71
# memory peak: 34085 Byte
```

Figure 8. Executed code of sample dataset.

5.2. Experiments on a Real Datasets

We evaluated the proposed algorithm by comparing it with the Non-Redundant with Dynamic Bit Vector (NRD-DBV) algorithm. The algorithms are run on a Windows 7 laptop with an Intel Core i5 2.33GHz processor and 6.58 GB of free RAM. They were implemented in python and run on Jet Brains Pycharm.

Four real datasets with different sizes and features are utilized to assess the performance of the algorithm. The first dataset is named BMSWebView1 (Gazelle), which contains 59601 sequences of click stream data included in an e-commerce website. The second data set called the Sign refers to, sign language utterance. It has 800 sequences transcript from videos and embedded 267 distinct items. The third dataset is named Korsarak, a large dataset containing 990,000 sequences of click-stream data from an online news portal. The fourth dataset, MSNBC, is huge which has over 1 million transactions. All used datasets are downloaded from SPMF Fournier-Viger *et al*. [3].

The experiments were applied to compare the runtime of the two algorithms for various minSup values. For this experiment, the minConf value was set to 0.5 % for all states. The parameters' values were calculated after a variety of preliminary tests to achieve the highest efficiency.

5.2.1. Runtime on a Gazelle Dataset

In the Gazelle dataset, items didn't iterate multiple times very often. So, we set a lower minSup to get the sequential rules. That means when setting minSup value as included in other experiments; we didn't have any sequential rules during the mining process. In general, the runtime increases with low minSup values that lead to an increased number of generated rules. Experience shows that the proposed pNRD-CloGen algorithm takes much less time than the NRD-DBV algorithm. When the minSup is set to low values, it has proven to be effective, as the time to generate sequential rules is roughly half that of the NRD-DBV, as shown in Figure 9.

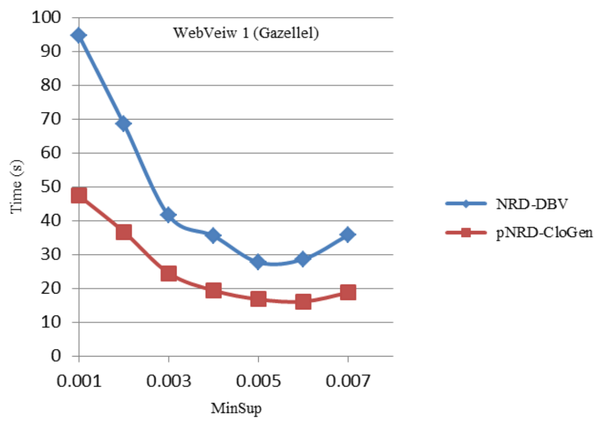


Figure 9. Runtime of Gazelle dataset.

5.2.2. Runtime on a Sign Dataset

The sign dataset has a variety of characteristics, including sparse, dense, having short and long sequences. It has an average sequence length of 51.997, while the Gazelle dataset has an average of 2.24.These features, related to its small size, allowed the algorithm to generate rules in a fraction of the time it took the Gazelle dataset. In both algorithms, runtime convergence occurs at low minSup values. When setting minSup with the lowest values, the pNRD-CloGen algorithm took less time to reach the highest performance.

As a result, it takes five times as long as the NRD-DBV algorithm. Figure 10, shows that when minSup is set to 0.1, pNRD-CloGen takes 105.6 (s) to complete, while NRD-DBV takes 514.3 (s).

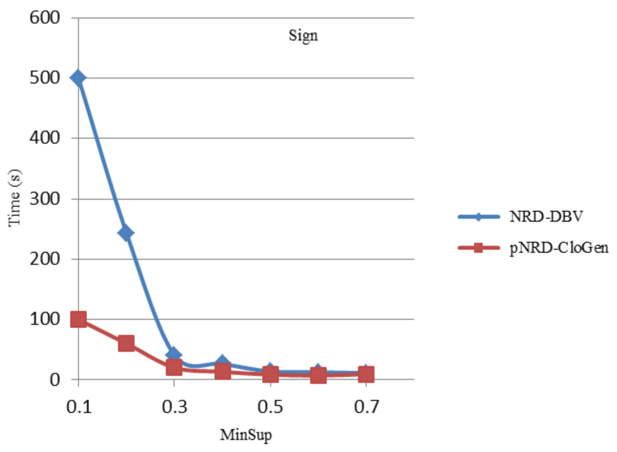


Figure 10. Runtime of sign dataset.

5.2.3. Runtime on a Korsarak Dataset

Unlike the Gazelle database, both algorithms didn't generate any rules with low values of min Sup because of the features of a korsarak dataset. Korsarak dataset has an item count of 41, 270 items and an average length equal to 8.1, and the Gazelle dataset has 497 items and 2.24 of average sequence length. The pNRD-CloGen achieved better performance with minimum values of minSup. The execution time required for generating the rules increases with the decreasing minSup values. This is because of increasing the computations for finding increased number of non-redundant sequential rules from a large search space area. The execution time between the two algorithms is approximated when setting high values to the minSup, but the pNRD-CloGen still consumes less time than the NRD-DBV algorithm, as shown in Figure 11.

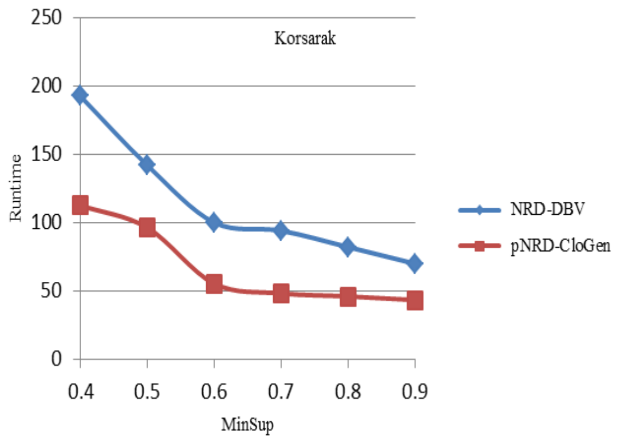


Figure 11. Runtime of korsarak dataset.

5.2.4. Runtime on a MSNBC Dataset

This dataset is a massive collection of click-stream data from the MSNBC website. It is a discrete sequence and very dense dataset. The proposed algorithm generated the sequential rules in the shortest amount of time. In terms of time consumption, the proposed algorithm, pNRD-CloGen, was achieved the most effective performance as shown in Figure 12.

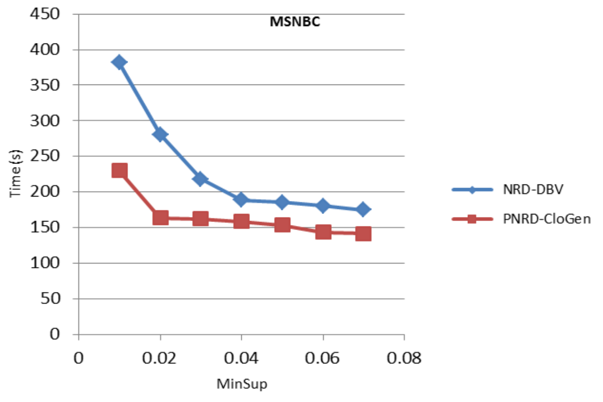


Figure 12. Runtime of MSNBC dataset.

In terms of memory usage, the NRD-DBV algorithm uses less memory than the pNRD-CloGen algorithm on all databases, as shown in Figures 13, 14, 15, and 16. Both algorithms, however, used the same method to store the sequences, with the pNRD-CloGen algorithm requiring more memory due to the parallel technique. It breaks down multiple tasks into smaller chunks and processes each one separately. As a result, more memory is needed to save the data.

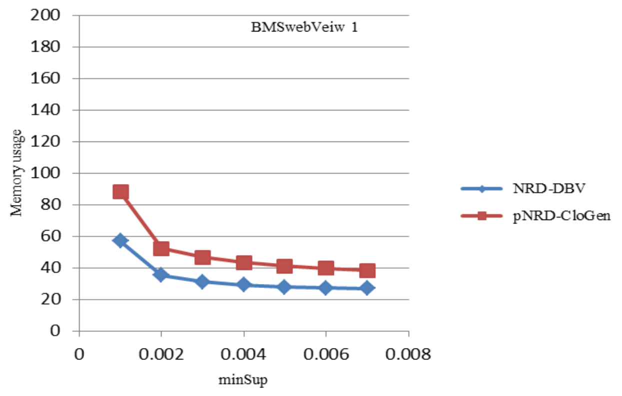


Figure 13. Memory usage for webview1 dataset.

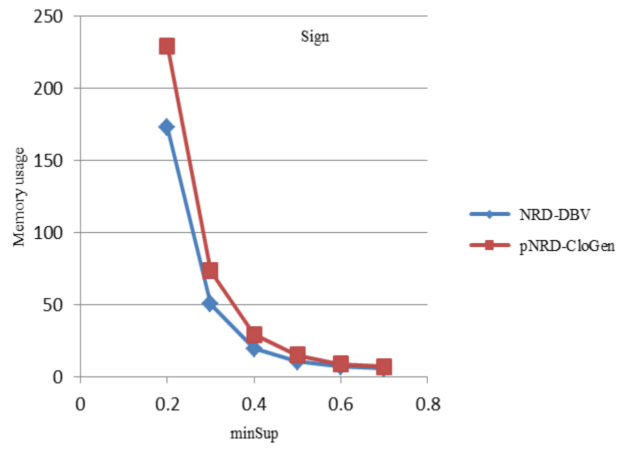


Figure 14. Memory usage for sign dataset.

Since there were fewer produced rules when the value of minSup was increased, memory consumption and runtime decreased. The proposed pNRD-CloGen algorithm has a significant advantage in that it makes a parallel approach only on two procedures. First, find F1-S and add them as child nodes in a prefix tree. Second, check all nodes whether, prefix or closed, to generate the sequential rules, so helps in minimizing the CPU idle time.

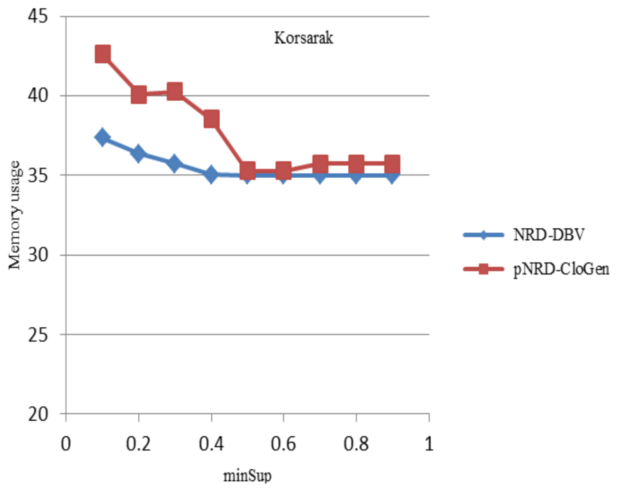


Figure 15. Memory usage for korsarak dataset.

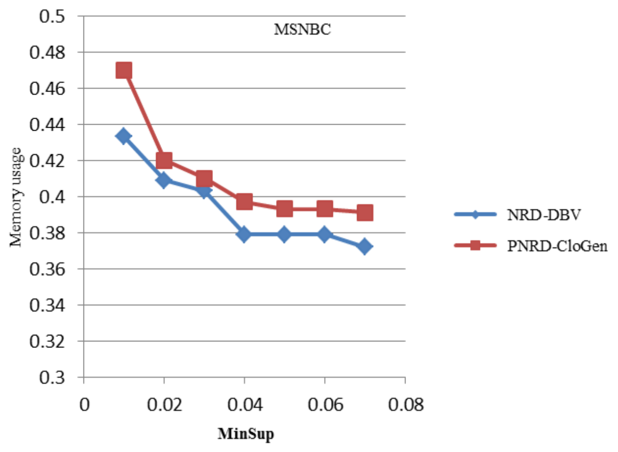


Figure 16. Memory usage for MSNBC dataset.

5.2.5. Measuring the Computational Cost

We measure the computational cost to evaluate the performance of the proposed algorithm. Producing non-redundant rules has a complexity of $O\ (n*c)$, where ‘n’ is the number of nodes and ‘c’ is the average number of child nodes. We must apply (n-1) operations for testing and generating sequential rules such as k<<n for each sequence. As a result, pNRD-CloGen has a complexity equal to $O \approx (n)$. It implies the algorithm's supreme running time is proportionate to the input size. It is the second-best state of big Onotation after the constant state, so the proposed algorithm proves its efficiency.

5.2.6. Measuring the Scalability

Scalability measures the system's capacity to raise or reduce its performance and cost in reply to changes in system processing requirements. We can analyze the scalability of a system by assessing how its performance diversifies as a function of the input size growth. We examined the scalability of the two compared algorithms on the MSNBC dataset for various data sizes and a fixed value of minSup equal to 0.03. The analysis shows the influence of generating rules on runtime. The results proved that the pNRD-CloGen algorithm achieved better scalability than the NRD-DBV algorithm, as shown in Figure 17.

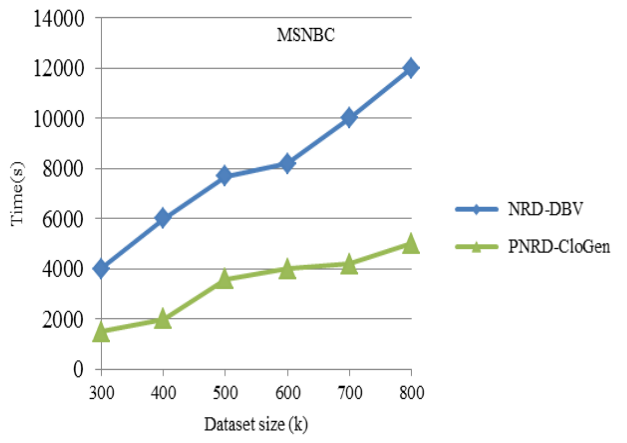


Figure 17. Scalability of compared algorithms for various dataset sizes with minSup=0.03.

6. Conclusions

Most algorithms mining sequential rules through only one representation like closed, generator, or maximal are time-consuming. The data structure goes through many steps that follow the same process. In this study, the proposed pNRD-CloGen algorithm is applied on four real datasets with different characteristics: sparse, dense, large, and very large datasets. We utilized a dynamic bit vector structure with a prefix tree to store frequent patterns. Also, pruning techniques are applied to remove uninteresting sequences early. In addition, a parallel approach is implemented on only two processes to acquire the highest efficiency. The first process of the parallel approach is the extension method. The second process is to test all nodes to generate sequential rules. The proposed algorithm proved its efficiency in saving time the approximately half time required from the compared algorithm, especially for large datasets and low values of minSup.

7. Recommendations and Future Works

The researcher deduced that not time-saving when implementing the parallel approach to the overall processes of the pNRD-CloGen algorithm. The best uses of the parallel technique are to test the child nodes that achieved the minSup and check each pattern, whether generator or closed pattern, to produce the non-redundant sequential rules.

There is no need to utilize DBV then DBV structure as applied in the NRD-DBV algorithm. It will take extra time. Using the DBV structure is sufficient and more accurate for generating valid rules with actual positions of items.

The use of processes instead of threads provides better memory utilization when memory goes down. The multiprocessing approach prevents data corruption and deadlocks that may appear with a multithreading approach.

The pNRD-DBV algorithm helps in detecting errors, intervening, and detecting bugs in much less time. It is necessary in areas that ordering sequences are needed, such as the therapeutic region (in the case of suffering from a fever). The order of symptoms is decreasing in coagulation levels, appearance a red color on the body. The fact that the quiet will cure dengue fever is tolerable. This timing structure is critical in predicting the right kind of treatment).

For future works, we suggest implementing a maximal patterns mininginstead of mining closed generator patterns. Also, try tostudyits impact on produced sequential rules in terms of the runtime, and the accuracy of rules. Additionally, increasing the number of cores, especially for large data bases and studying the performance. We hope to avoid the increase of the power consumption by increasing the clock frequency, which helps to eliminate overheating.

References

[1] Alja'am J., El Saddik A., and Sadka A., *Recent Trends in Computer Applications: Best Studies from the 2017 International Conference on Computer and Applications*, Springer, 2018.

[2] Czarnul P., Proficz J., and Drypczewski K., "Survey of Methodologies, Approaches, and Challenges in Parallel Programming Using High-Performance Computing Systems," *Scientific Programming*, vol. 2020, 2020.

[3] Fournier-Viger P., Lin J., Gomariz A., Gueniche T., Soltani A., Deng Z., and Lam H., "The SPMF Open-Source Data Mining Library Version 2," *in Proceedings of in Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, Riva del Garda, pp. 36-40, 2016.

[4] Fournier-Viger P., Lin C., Rage U., Koh Y., and Thomas R., "A Survey of Sequential Pattern Mining," *Data Science and Pattern Recognition*, vol. 1, no. 1, pp. 54-77, 2017.

[5] Gan W., Lin J., Fournier-Viger P., Chao H., and Yu P., "A Survey of Parallel Sequential Pattern Mining," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 3, pp. 1-34. 2019.

[6] Gokulapriya R. and Kumar G., "Research Aligned Analysis on Web Access Behavioral Pattern Mining for User Identification," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 6, pp. 2249-8958, 2019.

[7] Guyet T., "Enhancing Sequential Pattern Mining with Time and Reasoning," Doctoral

Dissertation, Université de Rennes 1, 2020.

[8] Husák M., Kašpar J., Bou-Harb E., and Čeleda P., "On the Sequential Pattern and Rule Mining in The Analysis of Cyber Security Alerts," in *Proceedings of the 12th International Conference on Availability, Reliability and Security*, New York, pp. 1-10, 2017.

[9] Huynh B., Vo B., and Snasel V., "An Efficient Method for Mining Frequent Sequential Patterns Using Multi-Core Processors," *Applied Intelligence*, vol. 46, no. 3, pp. 703-716, 2017.

[10] Huynh B., Vo B., and Snasel V., "An Efficient Parallel Method for Mining Frequent Closed Sequential Patterns," *IEEE Access*, vol. 5, pp. 17392-17402, 2017.

[11] Jamsheela O. and Gopalakrishna R., "Parallelization of Frequent Itemset Mining Methods with FP-tree: An Experiment with PrePost+Algorithm," *The International Arab Journal of Information Technology*, vol. 18, no. 2, pp. 208-213, 2021.

[12] Kuriakose S. and Nedunchezhian R., "Efficient Adaptive Frequent Pattern Mining Techniques for Market Analysis in Sequential and Parallel Systems," *The International Arab Journal of Information Technology.*, vol. 14, no. 2, pp. 175-185, 2017.

[13] Le B., Huynh U., and Dinh D., "A Pure Array Structure and Parallel Strategy for High-Utility Sequential Pattern Mining, " *Expert Systems with Applications*, vol. 104, pp. 107-120, 2018.

[14] Mollenhauer D. and Atzmueller M., "Sequential Exceptional Pattern Discovery Using Pattern-Growth: an Extensible Framework for Interpretable Machine Learning on Sequential Data," In XI-ML@ KI, 2020.

[15] Mukhlash, I., Mohammad I., and Astuti H., Sutikno S., "Performance Enhancement Of Cbs Algorithm Using Fsgp and Feat Algorithm," *Journal of Theoretical and Applied Information Technology*, vol. 67, no. 3, pp. 644-651, 2014.

[16] Naseera R. and Malsoru V., "Domain Specific Performance Evaluation of Sequential Pattern Mining Approaches," *in Proceedings of the World Congress on Engineering*, London, 2016.

[17] Patel P. and Malviya M., "A Review of Modern Sequential Rule Mining Techniques," *International Journal of Computer Applications*, vol. 88, no. 6, pp. 32-35, 2014.

[18] Pham T., Luo J., Hong T., and Vo B., "MSGPs: a Novel Algorithm for Mining Sequential Generator Patterns," *in Proceedings of the 4th International Conference on Computational Collective Intelligence: Technologies and Applications*, Ho Chi Minh City, pp. 393-401, 2012.

[19] Pham T., Luo, J., Hong T., and Vo B., "An Efficient Method for Mining Non-Redundant Sequential Rules Using Attributed Prefix-Trees," *Engineering Applications of Artificial Intelligence*, vol. 32, pp. 88-99, 2014.

[20] Ravikumar P., Likhitha P., Raj B., Kiran R., Watanobe Y., and Zettsu K., "Efficient Discovery of Periodic-Frequent Patterns in Columnar Temporal Databases," *Electronics*, vol. 10, no. 12, pp. 1-20, 2021.

[21] Spiliopoulou M., Managing Interesting Rules in Sequence Mining," *in Proceedings of the 3rd European Conference on Principles of Data Mining and Knowledge Discovery*, Prague, pp. 554-560, 1999.

[22] Suresh Kumar N. and Thangamani M., "Parallel Semi-Supervised Enhanced fuzzy Co-Clustering (PSEFC) and Rapid Association Rule Mining (RARM) Based Frequent Route Mining Algorithm for Travel Sequence Recommendation on Big Social Media," *Concurrency and Computation: Practice and Experience*, vol. 31, no. 14, pp. e4837, 2019.

[23] Tang K., Dai C., and Chen L., "An Efficient Mining Algorithm by Bit Vector Table for Frequent Closed Itemsets," *Journal of Software*, vol. 6, no. 11, pp. 2121-2128, 2011.

[24] Taşer P., Birant K., and Birant D., "Multitask-Based Association Rule Mining," *Turkish Journal of Electrical Engineering and Computer Sciences*, vol. 28, no. 2, pp. 933-955, 2020.

[25] Titarenko S., Titarenko V., Aivaliotis G., and Palczewski J., "Fast Implementation of Pattern Mining Algorithms with Time Stamp Uncertainties and Temporal Constraints," *Journal of Big Data*, vol. 6, no. 1, pp. 1-34, 2019.

[26] Upadhyay P., Pandey M., and Kohli N., "A Comprehensive Survey of Pattern Mining: Challenges and Opportunities," *International Journal of Computer Applications*, vol. 180, no. 24, pp. 32-39, 2018.

[27] Van T., Vo B., and Le B., "IMSR_PreTree: An Improved Algorithm for Mining Sequential Rules Based on the Prefix-Tree," *Vietnam Journal of Computer Science*, vol. 1, no. 2, pp. 97-105, 2014.

[28] Veroneze R., Corbi S., Da Silva B., De S-Rocha., C., Maurer-Morelli C., Orrico S., Cirelli J., Von Zuben F., and Scarel-Caminaga R., "Using Association Rule Mining to Jointly Detect Clinical Features and Differentially Expressed Genes Related to Chronic Inflammatory Diseases," *PloS one*, vol. 15, no. 10, pp. e0240269, 2020.

[29] Wang W. and Cao L., "VM-NSP: Vertical Negative Sequential Pattern Mining with Loose Negative Element Constraints," *ACM Transactions on Information Systems*, vol. 39, no. 2, pp. 1-27, 2021.

[30] Wu Y., Zhu C., Li Y., Guo L., and Wu X.,

"NetNCSP: Nonoverlapping Closed Sequential Pattern Mining," *Knowledge-Based Systems*, vol. 196, pp. 105812, 2020.

[31] Xie M. and Tan L., "An Efficient Algorithm for Frequent Pattern Mining over Uncertain Data Stream," *in Proceedings of 12th International Symposium on Computational Intelligence and Design*, Hangzhou, pp. 84-88, 2019.

[32] Youssef N., Abdulkader H., and Abdelwahab A., "Evaluating Non-Redundant Rules of Various Sequential Rule Mining Algorithms," *in Proceedings of International Conference on Advanced Intelligent Systems and Informatics*, Cairo, pp. 429-440, 2020,

[33] Zaki M., "SPADE: An Efficient Algorithm for Mining Frequent Sequences," *Machine Learning*, vol. 42, no. 1, pp. 31-60, 2001.

[34] Zhou S., Liu H., Chen B., Hou W., Ji X., Zhang Y., Chang W., and Xiao Y., "Status Set Sequential Pattern Mining Considering Time Windows and Periodic Analysis of Patterns," *Entropy*, vol. 23, no. 6, pp. 738, 2021.

[35] Zihayat M., Hut Z., An A., and Hut Y., "Distributed and Parallel High Utility Sequential Pattern Mining," *in Proceedings of IEEE International Conference on Big Data*, Washington, pp. 853-862, 2016.

Nesma Youssef has received her master's degree from Sadat Academy for management science, department of an information system, Cairo, Egypt, 2014. The equivalency score was obtained with computers and information at the Supreme Council of Universities. Currently, she is working as an assistant teacher in the information system department at Sadat Academy for management science, Cairo, Egypt. She has 10 years of teaching experience. She is a research Scholar in the Information System department at the Faculty of Computers and Information, Menoufia University, Egypt.

Hatem Abdulkadar has obtained his BS and M.SC., both in electrical engineering from the Alexandria University, faculty of Engineering, 1990 and 1995, respectively. He obtained his Ph.D. in electrical engineering also from faulty of engineering, Alexandria University, Egypt 2001. His area of interest is data security, web applications, artificial intelligence, and he is specialized neural networks. He is currently a professor in the information system department, faculty of computers and information, Menoufia University, Egypt.

Amira Abdelwahab has received BSc degree in computer science and information systems from Faculty of Computers and Information, Helwan University, Egypt in 2000, and Ph.D. in information systems from Chiba University, Japan in 2012. In 2013, she was a postdoctoral fellow in Chiba University, Japan. from 2012 till 2018, she has been an assistant professor in information systems department, Faculty of Computers and Information, Menoufia University, Egypt. Since 2018, she has been an assistant professor in information systems department, college of Computer science and information technology, King Faisal University. Her research interests include Software Engineering, Decision Support System, database Systems, Data Mining, Machine Learning, Recommendation Systems, Web intelligence, and Big Data analytics.