OPEN ACCESS
Freely Available Online

JOURNAL OF BIG DATA RESEARCH
ISSN NO: 2768-0207

Review Article
DOI: 10.14302/issn.2768-0207.jbr-21-3455

# Mining Frequent Sequential Patterns

Rajae KRIBII[1], Youssef FAKIR[1,*]

[1]Laboratory of Information Processing and Decision Support, University Sulan Moulay Slimane

**Abstract**

In recent times, the urge to collect data and analyze it has grown. Time stamping a data set is an important part of the analysis and data mining as it can give information that is more useful. Different mining techniques have been designed for mining time-series data, sequential patterns for example seeks relationships between occurrences of sequential events and finds if there exist any specific order of the occurrences. Many Algorithms has been proposed to study this data type based on the apriori approach. In this paper we compare two basic sequential algorithms which are General Sequential algorithm (GSP) and Sequential PAttern Discovery using Equivalence classes (SPADE). These two algorithms are based on the Apriori algorithms. Experimental results have shown that SPADE consumes less time than GSP algorithm.

**Corresponding author:** Youssef FAKIR, Laboratory of Information Processing and Decision Support, University Sulan Moulay Slimane, Email: info.dec07@yahoo.fr

**Keywords:** Data mining, Sequential Patterns, Apriori, SPADE, GSP.

**Received:** Jun 24, 2020 **Accepted:** Mar 10, 2021 **Published:** Mar 15, 2021

**Editor:** Hongwei Mo, Harbin Engineering University, Harbin 150001, China.

## Introduction

Data mining is a critical step in the field of data science. It involves sorting large data sets to identify patterns and build relationships to solve problems through data analysis. Data mining tools help companies predict future trends. Specifically, data mining benefits vary depending on the goal and the industry. Sales and marketing departments can mine customer data to improve lead conversion rates or to create one-to-one marketing campaigns. Data mining information on historical sales patterns and customer behaviors can be used to build prediction models for future sales, new products and services.

Data mining parameters include classification, sequence or path analysis, clustering and forecasting. Sequence or path parsing parameters look for patterns in which one event leads to another subsequent event. A sequence is an ordered list of element sets and is a common type of data structure in many databases.

A classification parameter searches for new models and may cause a change in the organization of the data. Classification algorithms predict variables based on other factors in the database. In data mining, association rules are created by analyzing the data to look for frequent "if / then" patterns, and then using the *support* and *trust* criteria to locate the most important relationships within data. The frequency with which items appear in the database is supported, while trust is the number of times the statements are accurate [1, 2, 3,4].

Sequence Mining was first introduced in 1995 by Argwal and Srikant [6] and is defined by the discovery a set of attributes, shared in time between a large numbers of objects of a given database. Sequential pattern is a sequence of item sets that frequently occurred in a specific order, all items in the same item sets are supposed to have the same transaction time value or within a time gap. Sequential pattern mining is trying to find the relationships between occurrences of sequential events, to find if there exist any specific order of the occurrences [6]. We can find the sequential patterns of specific individual items; also, we can find the sequential patterns cross-different items. Sequential pattern mining is widely used in analyzing of DNA sequence.

Various algorithms have been implemented to identify the frequent sequences from the sequence database. One of the approaches used to identify frequent sequences is the Apriori approach [7]. The approach is introduced in the context of the exploration of association rules. This property indicates that if a pattern is not common, no pattern that contains a pattern can be common. Two of the most effective algorithms using this approach are GSP and SPADE. The main difference between the two is that GSP uses a horizontal data format [8], while SPADE uses a vertical format [9]. These algorithms work well in a database of short frequent sequences. However, when exploiting databases composed of long, frequent sequences, e.g. stock values, DNA strings or machine monitoring data, their overall performance deteriorates by an order of magnitude [10].

This paper is organized as follows: in the second section, we describe the Generalized Sequential Models (GSP) algorithm, which use a horizontal format database. In the third section a Sequential Pattern Discovery using Equivalence classes (SPADE) which uses a vertical database is given. A case study is given in section four. The paper is ended by experimental results and conclusion.

## Gsp Algorithm

The Generalized Sequential Models (GSP) algorithm is an apriori-based algorithm applied to sequential models. It integrates with time constraints and relaxes the definition of transaction. For time constraints, maximum gap and minimal gap are defined to specify the gap between any two adjacent transactions in the sequence. If the distance is not in the range between maximum gap and minimal gap then this two cannot be taken as two consecutive transactions in a sequence. The transactions are then applied to generate multiple level sequential patterns to find all sequences whose support is greater than the user-defined minimum support [5].

GSP uses the Apriori property [1], that is to say, considering the minimal number of supports, if a sequence is not accepted; all its super sequence will be ignored. The features require several passes of the initial transaction data set, it uses the horizontal data format, at each pass, the entire candidate is generated by a self-join patterns found in the previous pass. In k-step, a

sequence pattern is accepted only if all its submodules (k-1) are accepted in (k-1) [8].

The *Pseudo code for GSP Algorithm is as follows:*

F1 = the set of frequent 1-sequence

K = 2,

do while Fk-1 != Null;

Generate candidate sets Ck (set of candidate k-sequences);

For all input sequences s in the database D

do

Increment count of all a in $C_k$ if s supports a

End do

$F_k$ = {a ∈ Ck such that its frequency exceeds the threshold}

k = k+1;

End do

Result = Set of all frequent sequences is the union of all $F_k$' s

**Spade Algorithm**

SPADE is an algorithm proposed to find frequent sequences using efficient lattice search techniques and simple joins. SPADE decomposes the original problem into smaller sub-problems using equivalence classes on frequent sequences so that each class can be solved independently. SPADE usually makes only three database scans, first one for frequent 1-sequences, second for frequent 2-sequences, and one more for generating all other frequent sequences. If the support of 2-sequences is available then only one scan is required [5].

SPADE outperforms GSP algorithm, by a factor of two, and by an order of magnitude with precomputed support of 2-sequences. It also has excellent scale up properties with respect to a number of parameters such as the number of input-sequences, the number of events per input-sequence, the event size, and the size of potential maximal frequent events and sequences [9].

The *Pseudo Code for SPADE Algorithm is as follows*

SPADE (D,min_ supp):

$F_1$={Frequent items or 1-sequences }

$F_2$={ Frequent 2 - sequences}

Ɛ={equivalence classes $[X]_{\theta 1}$}

for all [X]∈ Ɛ do EnumerateFrequentSeq([X])

EnumerateFrequentSeq(S) :

for all atoms $A_i \in S$ do

$T_i \neq \emptyset$

for all atoms $A_j \in S$ with j≥i do

$R=A_i \vee A_j$

if ( Prune(R)==false) then

$L(R)= L(A_i) \cap L(A_j)$

if σ(R)≥min_supp then

$T_i=T_i \cup \{R\}$; $F_{|R|} = F_{|R|} \cup \{R\}$

end

if (DepthFirstSearch) then EnumerateFrequentSeq(T_i )

end

if(BreadthFirstSearch) then

for all $T_i \neq \emptyset$ do EnumerateFrequentSeq($T_i$)

**Case Study**

To demonstrate an explanation of the algorithms GSP and SPADE an example is given (Table 1). For all the algorithms, we fix the minimum support to 2. Tab 1

The Candidate sets are denoted by C.

$C_k$ specifies candidate sets having K items.

The Candidate sets that satisfy minimum support belongs to $L_k$ .

K denotes number of items in sequence.

As a first step, the database needs to be grouped by *Client ID* and *Date*, and then we can visually notice the sequence flow. The column *Client ID* is denoted *SID*, which means the sequence identifier; *EID* is the element identifier in the sequence. Tab 2

*GSP Application*

$C_1$={ A,B,C,D,E,F,G,H} are the initial candidates for the first round. For each item, we should extract the number of occurrences or the support that would eliminate the less frequent items (Table 3).

The items that satisfies the minimum support 2 are $L_1$={A,B,D,F} $C_2$ can be obtained by joining $L_1 \times L_1$.

Tab 4

Table 1. Original database of transactions

| Client ID | Date | Items |
|---|---|---|
| 1 | 5/1/2015 | C, D |
| 1 | 16/1/2016 | A, B, C |
| 1 | 3/1/2017 | A, B, F |
| 1 | 3/1/2017 | A, C, D, F |
| 2 | 8/1/2017 | A, B, F |
| 2 | 10/1/2016 | E |
| 3 | 5/1/2016 | A, B, F |
| 4 | 5/1/2016 | D, H, G |
| 4 | 3/1/2017 | B, F |
| 4 | 8/1/2017 | A, G, H |

Table 2. Database after grouping the instances by sequence id and time

| SID | EID | ITEMS |
|---|---|---|
| 1 | 1,2,3,4 | <(CD)(ABC)(ABF)(ACDF)> |
| 2 | 1,2 | <(ABC)(E)> |
| 3 | 1 | <(ABF)> |
| 4 | 1,2,3 | <(DHG)(BF)(AGH)> |

Table 3. Number of occurrences of items in C1

| Items | N° d'occurrence |
|---|---|
| A | 4 |
| B | 4 |
| C | 1 |
| D | 2 |
| E | 1 |
| F | 4 |
| G | 1 |
| H | 1 |

Table 4. Number of occurrences of items in C2

| Items | N° d'occurrence |
|---|---|
| A →A | 1 |
| A→B | 1 |
| A→D | 1 |
| A→F | 1 |
| AB | 3 |
| AD | 1 |
| AF | 3 |
| B→A | 2 |
| B→B | 1 |
| B→D | 1 |
| B→F | 1 |
| BD | 1 |
| BF | 4 |
| D→A | 2 |
| D→B | 2 |
| D→D | 2 |
| D→F | 2 |
| DF | 1 |
| F→A | 2 |
| F→B | 1 |
| F→D | 1 |
| F→F | 1 |

$C_2$={A→A, A→B, A→D, A→F, AB, AD, AF,B→A, B→B, B→D, B→F, BD, BF, D→A, D→B, D→D, D→F, DF, F→A, F→B, F→D, F→F}

The items that satisfies the minimal support:

$L_2$={AB, AF, B→A, BF, D→A, D→B, D→F, F→A}

In this step, $C_3$ can be obtained by joining $L_2$ X $L_2$

Tab 5

*$C_3$={ABF, AB→A, AF→A,BF→A,D→B→A, D→F→A, D→BF,F→AF, D→FA, D→AB, F→AB, B→AB, B→AF}*

$L_3$={ABF, BF→A, D→B→A, D→F→A, D→BF}

In this step $C_4$ can be obtained by joining $L_3$ X $L_3$

Tab 6.

$L_4$={D→BF→A}

*Spade Application*

Most of the sequential pattern mining algorithms assume horizontal database layout. SPADE uses vertical database format. We can generate a new mapping model. <SID, EID> the first identifies the identifier of the sequence, and the second the identifier of the element location in the sequence. Tab 7

*Generating 2-sequences*

Then to generate the other sequence elements of size 2, we perform a join between the tables and compare the number of occurrences with the minimal support.

Joining A and B items makes two kind of elements, (AB) is the equality join which means they happen at the same time; thus, we'll notice the same *EID* for both items. A → B denotes actions that happen consecutively, that means A's *EID* has to be smaller than B's *EID*. Tab 8, 9.

The support for (AB) is 3, having two cases from the same sequence makes a redundancy problem in the model. For A → B, the support is 1. Tab 10

The support for (AD) is 1, it does not satisfy the minimum support. Tab 11

The support for (AF) is 3, it satisfies the minimum support, eve, if the number of occurrences is 4 in the database, there is two elements within the same sequence, so they will be considered as one. Tab 12

The number of occurrences is 0, it does not satisfy the minimum support. Tab 13

The number of occurrences is 4, it satisfies the minimum support. Tab 14

The number of occurrences is 1, it does not satisfy the minimum support. Tab 15

The number of occurrences is 2, both belongs to the same sequence, that makes the support 1, so it does not satisfy the min support. Tab 16

The support is 1, because the occurrences are from the same sequence. Therefore, it does not satisfy the min support. Tab 17

The support is 1. In addition, it does not satisfy the min support. Tab 18

The number of occurrences is 2, it satisfies the minimum support. Tab 19

The support is 1, it does not satisfy the minimum support. Tab 20

The support is 1, it does not satisfy the minimum support. Tab 21

The support is 1, it does not satisfy the minimum support. Tab 22

The support is 1, it does not satisfy the minimum support. Tab 23

The support is 2, it satisfies the minimum support. Tab 24

The support is 2, it satisfies the minimum support. Tab 25

The support is 2, it satisfies the minimum support. Tab 26

The support is 1, it does not satisfy the minimum support. Tab 27

The support is 1, it does not satisfy the minimum support. Tab 28

The support is 1, it does not satisfy the minimum support. Tab 29

The support is 2, it satisfies the minimum support. Tab 30

From equality join and temporal joins, the table showing count of occurrence of sequences of items.

$L_2$={(AB),(AF),(BF),B→A,D→A,D→F,D→B,F→A}

Generating 3-sequence: Tab 31

Table 5. Number of occurrences of items in C3

| **Items** | **N° Occurrence** |
|---|---|
| ABF | 3 |
| AB→A | 1 |
| AF →A | 1 |
| BF →A | 2 |
| D →B →A | 2 |
| D →F→ A | 2 |
| D →BF | 2 |
| F →AF | 1 |
| D →FA | 1 |
| D→AB | 1 |
| F→AB | 0 |
| B→AB | 1 |
| B→AF | 1 |

Table 6. Number of occurrences of items in C4

| Items | N° occurrence |
|---|---|
| D→BF→A | 2 |

Table 7. ID list for atoms

| A | | B | | D | | F | |
|---|---|---|---|---|---|---|---|
| SID | EID | SID | EID | SID | EID | SID | EID |
| 1 | 2 | 1 | 2 | 1 | 1 | 1 | 3 |
| 1 | 3 | 1 | 3 | 1 | 1 | 1 | 4 |
| 1 | 4 | 2 | 1 | 4 | 1 | 2 | 1 |
| 2 | 1 | 3 | 1 | | | 3 | 1 |
| 3 | 1 | 4 | 2 | | | 4 | 2 |
| 4 | 3 | | | | | | |

Table 8. ID list for (AB) Non-temporal join

| (AB) | | |
|---|---|---|
| SID | EID A | EID B |
| 1 | 2 | 2 |
| 1 | 3 | 3 |
| 2 | 1 | 1 |
| 3 | 1 | 1 |

Table 9. ID list for A ->B temporal join

| A→B | | |
|---|---|---|
| SID | EID A | EID B |
| 1 | 3 | 4 |

Table 10. ID list for AD non-temporal join

| (AD) | | |
|---|---|---|
| SID | EID A | EID D |
| 1 | 4 | 4 |

Table 11. ID list for AF non-temporal join

| (AF) | | |
|---|---|---|
| SID | EID A | EID F |
| 1 | 3 | 3 |
| 1 | 4 | 4 |
| 2 | 2 | 2 |
| 3 | 1 | 1 |

Table 12. ID list for BD non-temporal join

| (BD) | | |
|---|---|---|
| SID | EID B | EID D |
| - | - | - |

Table 13. ID list for BF non-temporal join

| (BF) | | |
|---|---|---|
| SID | EID B | EID F |
| 1 | 3 | 3 |
| 2 | 2 | 2 |
| 3 | 1 | 1 |
| 4 | 3 | 3 |

Table 14. ID list of DF non-temporal join

| (DF) | | |
|---|---|---|
| SID | EID D | EID F |
| 1 | 4 | 4 |

Table 15. ID list for A->A temporal join

| A→A | | |
|---|---|---|
| SID | EID A | EID A |
| 1 | 2 | 3 |
| 1 | 3 | 4 |

Table 16. ID list for A -> D temporal join

| A→D | | |
|---|---|---|
| SID | EID A | EID D |
| 1 | 2 | 4 |
| 1 | 3 | 4 |

Table 17. ID list for A->F temporal join

| A→F | | |
|---|---|---|
| SID | EID A | EID F |
| 1 | 2 | 3 |
| 1 | 2 | 4 |
| 1 | 3 | 4 |

Table 18. ID list for B->A temporal join

| B→A | | |
|---|---|---|
| SID | EID B | EID A |
| 1 | 2 | 3 |
| 4 | 3 | 4 |

Table 19. ID list for B->B temporal join

| B→B | | |
|---|---|---|
| SID | EID B | EID B |
| 1 | 2 | 2 |
| 1 | 3 | 3 |

Table 20. ID list for B->D temporal join

| B→D | | |
|---|---|---|
| SID | EID B | EID D |
| 1 | 2 | 4 |
| 1 | 3 | 4 |

Table 21. ID list for B->F temporal join

| B→F | | |
|---|---|---|
| SID | EID B | EID F |
| 1 | 3 | 4 |

Table 22. ID list for D->D temporal join

| D→D | | |
|---|---|---|
| SID | EID D | EID D |
| 1 | 1 | 4 |

Table 23. ID list for D->A temporal join

| D→A | | |
|---|---|---|
| SID | EID D | EID A |
| 1 | 1 | 3 |
| 1 | 1 | 4 |
| 4 | 1 | 4 |

Table 24. ID list for D->B temporal join

| D→B | | |
|---|---|---|
| SID | EID D | EID B |
| 1 | 1 | 2 |
| 1 | 1 | 3 |
| 4 | 1 | 3 |

Table 25. ID list for D->F temporal join

| D→F | | |
|---|---|---|
| SID | EID D | EID F |
| 1 | 1 | 3 |
| 1 | 1 | 4 |
| 4 | 1 | 3 |

Table 26. ID list for F->F temporal join

| F→F | | |
|---|---|---|
| SID | EID F | EID F |
| 1 | 3 | 4 |

Table 27. ID list for F->D temporal join

| F→D | | |
|---|---|---|
| SID | EID F | EID D |
| 1 | 3 | 4 |

Table 28. ID list for F->B temporal join

| F→B | | |
|---|---|---|
| SID | EID F | EID B |
| 1 | 3 | 4 |

Table 29. ID list for F->A temporal join

| F→A | | |
|---|---|---|
| SID | EID F | EID A |
| 1 | 3 | 4 |
| 4 | 3 | 4 |

Table 30. Number of occurrences of items

| Items | N° occurrence |
|---|---|
| AB | 3 |
| AD | 1 |
| AF | 3 |
| BD | 0 |
| BF | 4 |
| DF | 1 |
| A→A | 1 |
| A→B | 1 |
| A→D | 1 |
| A→F | 1 |
| B→A | 2 |
| B→B | 1 |
| B→D | 1 |
| B→F | 1 |
| D→D | 1 |
| D→A | 2 |
| D→B | 2 |
| D→F | 2 |
| F→F | 1 |
| F→D | 1 |
| F→B | 1 |
| F→A | 1 |

Table 31. ID list for ABF non-temporal join

| ABF | | | |
|---|---|---|---|
| SID | EID A | EID B | EID F |
| 1 | 3 | 3 | 3 |
| 2 | 2 | 2 | 2 |
| 3 | 1 | 1 | 1 |

The support is 3, it satisfies the minimum support. Tab 32

The support is 1, it does not satisfy the minimum support. Tab 33

The support is 1, it does not satisfy the minimum support. Tab 34

The support is 2, it satisfies the minimum support. Tab 35

The support is 2, it satisfies the minimum support. Tab 36

The support is 2, it satisfies the minimum support. Tab 37

The support is 2, it satisfies the minimum support. Tab 38

The support is 0, it does not satisfy the minimum support. Tab 39

The support is 2, it satisfies the minimum support. Tab 40

The support is 2, it satisfies the minimum support. Tab 41

The support is 0, it does not satisfy the minimum support. Tab 42

The support is 1, it does not satisfy the minimum support. Tab 43

The support is 1, it does not satisfy the minimum support. Tab 44

So, the results are: $L_3$={(ABF), (BF)→A, D→B→A, D→F→A, D→(BF)}

Generate 4-sequence:

The support is 1, it does not satisfy the minimum support. Tab 45

The support is 2, it satisfies the minimum support.

The result: $L_4$={D→(BF)→A}

**Experimental Results**

In this section, we have carried out some experiments in order to evaluate the performance of SPADE and GSP. The implementation of these algorithms has been carried out using JAVA as a coding language and a machine with an intel core i5 processor and 6GB of RAM. The experiment used the same dataset for both algorithms however; the input was varied according to a number of sequences each time. The results display the number of sequences, the CPU time, and the number of sequences found at a support of 30%. Tab 46. Fig 1.

**Conclusion**

This paper presented the concept explanation of two algorithms GSP and SPADE used for mining frequent sequential patterns and presented a case study of how the two algorithms work. The experiment results displayed a higher performance from the SPADE algorithm compared to GSP in consuming a minimal process time.

Table 32. ID list for AB->A non-temporal join

| AB→A | | | |
|---|---|---|---|
| SID | EID A | EID B | EID A |
| 1 | 2 | 2 | 3 |
| 1 | 3 | 3 | 4 |
| 2 | 2 | 2 | - |
| 3 | 1 | 1 | - |

Table 33. ID list for AF->A non-temporal join

| AF→A | | | |
|---|---|---|---|
| SID | EID A | EID F | EID A |
| 1 | 3 | 3 | 3 |
| 1 | 4 | 4 | - |
| 2 | 2 | 2 | - |
| 3 | 1 | 1 | - |

Table 34. ID list for BF->A non-temporal join

| BF→A | | | |
|---|---|---|---|
| SID | EID B | EID F | EID A |
| 1 | 3 | 3 | 4 |
| 2 | 2 | 2 | - |
| 3 | 1 | 1 | - |
| 4 | 3 | 3 | 4 |

Table 35. ID list for D->B->A non-temporal join

| D→B→A | | | |
|---|---|---|---|
| SID | EID D | EID B | EID A |
| 1 | 1 | 2 | 3 |
| 4 | 1 | 3 | 4 |

Table 36. ID list for D->F->A non-temporal join

| D→F→A | | | |
|---|---|---|---|
| SID | EID D | EID F | EID A |
| 1 | 1 | 3 | 4 |
| 4 | 1 | 3 | 4 |

Table 37. ID list for D->BF non-temporal join

| D→BF | | | |
|---|---|---|---|
| SID | EID D | EID B | EID F |
| 1 | 1 | 3 | 3 |
| 2 | - | 2 | 2 |
| 3 | - | 1 | 1 |
| 4 | 1 | 3 | 3 |

Table 38. ID list for F->AF non-temporal join

| F→AF | | | |
|---|---|---|---|
| SID | EID F | EID A | EID F |
| 1 | - | 3 | 3 |
| 1 | - | 4 | 4 |
| 2 | - | 2 | 2 |
| 3 | - | 1 | 1 |

Table 39. ID list for D->AF non-temporal join

| D→AF | | | |
|---|---|---|---|
| SID | EID D | EID A | EID F |
| 1 | 1 | 3 | 3 |
| 1 | 1 | 4 | 4 |
| 2 | - | 2 | 2 |
| 3 | - | 1 | 1 |

Table 40. ID list for D->AB non-temporal join

| D→AB | | | |
|---|---|---|---|
| SID | EID D | EID A | EID B |
| 1 | 1 | 3 | 3 |
| 1 | 1 | 4 | 4 |
| 2 | - | 2 | 2 |
| 3 | - | 1 | 1 |

Table 41. ID list for F->AB non-temporal join

| F→AB | | | |
|---|---|---|---|
| SID | EID F | EIDA | EID B |
| 1 | - | 2 | 2 |
| 1 | - | 3 | 3 |
| 2 | - | 2 | 2 |
| 3 | - | 1 | 1 |

Table 42. ID list for B->AB non-temporal join

| B→AB | | | |
|---|---|---|---|
| SID | EID B | EID A | EID B |
| 1 | - | 2 | 2 |
| 1 | 2 | 3 | 3 |
| 2 | - | 2 | 2 |
| 3 | - | 1 | 1 |

Table 43. ID list for B->AF non-temporal join

| B→AF | | | |
|---|---|---|---|
| SID | EID B | EID A | EID F |
| 1 | - | 2 | 2 |
| 1 | 2 | 3 | 3 |
| 2 | - | 2 | 2 |
| 3 | - | 1 | 1 |

Table 44. ID list for ABF->A temporal join

| ABF→A | | | | |
|---|---|---|---|---|
| SID | EID A | EID B | EID F | EID A |
| 1 | 3 | 3 | 3 | 4 |
| 2 | 2 | 2 | 2 | - |
| 3 | 1 | 1 | 1 | - |

Table 45. ID list for D->BF->A temporal join

| D→BF→A | | | | |
|---|---|---|---|---|
| SID | EID D | EID B | EID F | EID A |
| 1 | 1 | 3 | 3 | 4 |
| 2 | - | 2 | 2 | - |
| 3 | - | 1 | 1 | - |
| 4 | 1 | 3 | 3 | 4 |

Table 46. The following chart displays the chosen attributes of GSP and SPADE:

| | **GSP** | | **SPADE** | |
|---|---|---|---|---|
| Input Sequence Count | CPU time | Sequence Count | CPU time | Sequence Count |
| 5 | ~5 ms | 222 | ~2 ms | 222 |
| 10 | ~3 ms | 357 | ~1 ms | 357 |
| 100 | ~11 ms | 35 | ~ 4 ms | 35 |
| 500 | ~71 ms | 20 | ~2 ms | 20 |
| 1000 | ~80 ms | 20 | ~1 ms | 20 |
| 10000 | ~750 ms | 20 | ~16 ms | 20 |
| 100000 | ~6441 ms | 20 | ~ 658 ms | 20 |
| 500000 | ~31438 ms | 20 | ~425 ms | 20 |

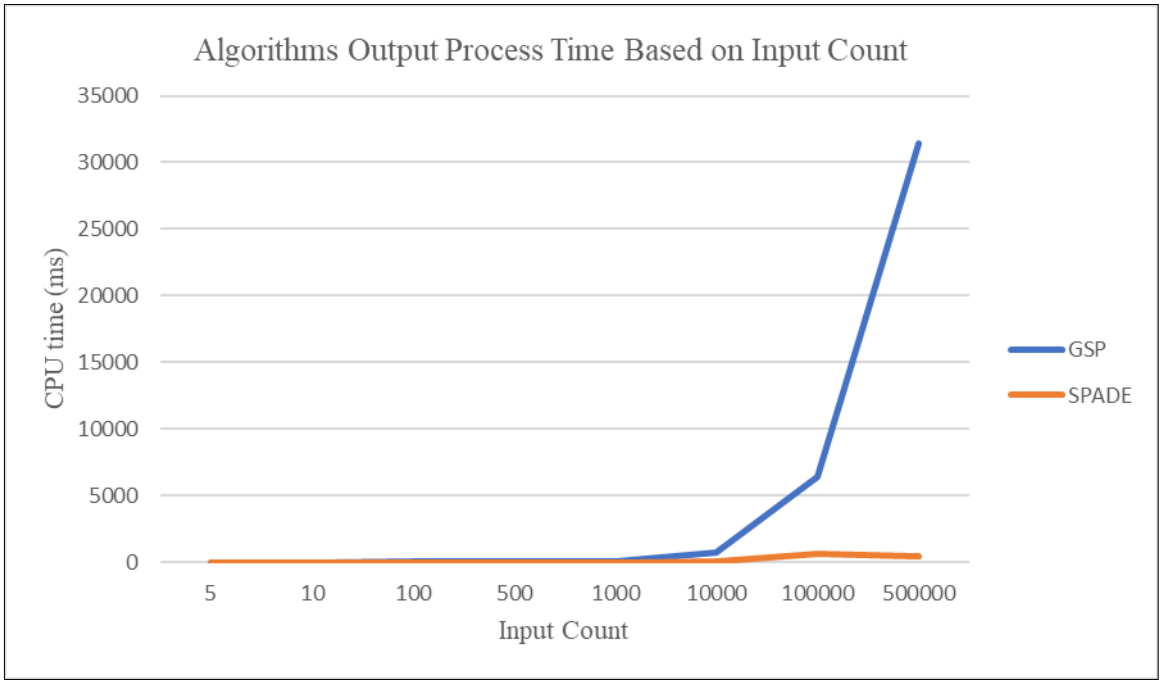


Figure 1. Comparison of SPADE and GSP in computing time

## References


1. Chakrabarti, S., Ester, M., Fayyad, U., Gehrke, J., Han, J., Morishita, S. & Wang, W. (2006). Data mining curriculum: A proposal (Version 1.0). *Intensive Working Group of ACM SIGKDD Curriculum Committee*, *140*.
2. Fayyad, U., Piatetsky-Shapiro, G., & Smyth, P. (1996). From data mining to knowledge discovery in databases. *AI magazine*, *17*(3), 37-37.
3. Bharati, M., & Ramageri, MData mining techniques and applications, Indian Journal of Computer Science and Engineering 1(4), . (2010).
4. Verma, M., & Mehta, D. (2014). Sequential Pattern Mining: A Comparison between GSP, SPADE and Prefix SPAN 1.
5. Zhao, Q., & Bhowmick, S. S. (2003). Sequential pattern mining: A survey. *ITechnical Report CAIS Nayang Technological University Singapore*, *1*(26), 135.
6. Agrawal, R., & Srikant, R. (1995, March). Mining sequential patterns. In *Proceedings of the eleventh international conference on data engineering* (pp. 3-14). IEEE.
7. Agrawal, R., & Srikant, R. (1994, September). Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB* (Vol. 1215, pp. 487-499).
8. Srikant, R., & Agrawal, R. (1996, March). Mining sequential patterns: Generalizations and performance improvements. In *International Conference on Extending Database Technology* (pp. 1-17). Springer, Berlin, Heidelberg.
9. Zaki, M. J. (2001). SPADE: An efficient algorithm for mining frequent sequences. *Machine learning*, *42* (1-2), 31-60.
10. Hornik, K., Grün, B., & Hahsler, M. (2005). arules-A computational environment for mining association rules and frequent item sets. *Journal of Statistical Software*, *14*(15), 1-25.
11. Hahsler, Michael (2005). "Introduction to arules – A computational environment for mining association rules and frequent item sets". Journal of Statistical Software.
12. Hahsler, M. (2015). A probabilistic comparison of commonly used interest measures for association rules. *United States. Southern Methodist University*.