Domain Specific Performance Evaluation of Sequential Pattern Mining Approaches

A. R. Naseer, Senior Member IEEE, and V. Malsoru, *Member, IAENG*

***Abstract*—The judicious selection of Sequential Pattern Mining (SPM) approaches becomes a challenge with the variety of datasets pertaining to various application domains. This paper presents the domain specific performance evaluation of the most commonly used SPM approaches on real-life datasets. The main objective of this study is to analyze the behavior of SPM algorithms on the real-life datasets which represent characteristics of truly real-life situations rather than the synthetic ones and to select the one which best suits the given application domain characteristics. Further, this study aims at building a recommendation system for judicious selection of algorithm(s) and minimum support thresholds for application domains considering the nature of the application(s) and characteristics of the involved Data Sets.**

***Index Terms*— Sequential Pattern Mining, Data mining, Frequent sequential pattern, Domain specific performance evaluation, Minimum support threshold, Real-life datasets**

I. Introduction

Sequential pattern mining approaches have been found to be applicable in a variety of domains like retail industry, healthcare, education, web usage mining, text mining, bioinformatics, telecommunications, intrusion detection, etc. In these domains, SPM techniques are employed to analyze the available enormous data to identify sequential patterns in order to implement efficient recommendation systems that can aid in detecting events of utmost interest, in making predictions based on previously observed patterns, and in taking strategic product decisions.

Most commonly used SPM approaches are GSP[2], SPADE[3], PrefixSpan[4], SPAM[5], CloSpan[6],ClaSP[7]. GSP is a generalized version of SPM approach [1] which incorporates time constraints, sliding time windows, and taxonomies in discovered sequential patterns. The performance bottleneck of this approach is that it requires several scans of the database to check the support of the candidates and the use of a breath-first search technique for the candidate generation, leading to high memory consumption. SPADE is an apriori-based SPM lgorithm that uses a *vertical id-list* database format, efficient lattice search techniques and simple joins to discover frequent sequences. The search space in SPADE is represented as a *lattice* structure and it uses the notion of equivalence classes to partition it. SPADE not only minimizes I/O costs by reducing database scans, but also minimizes computational costs by using efficient search schemes. PrefixSpan proposes a pattern-growth approach for mining frequent patterns without candidate generation. PrefixSpan recursively projects a sequence database into a set of smaller projected sequence databases and grows sequential patterns in each projected database by exploring only locally frequent fragments. SPAM is an apriori-based candidate generation and pruning approach that uses a vertical bitmap data structure representation of database which is similar to the id-list of SPADE. CloSpan discovers only frequent closed subsequences, i.e., those containing no super-sequences with user specified minsup, instead of mining the complete set of frequent subsequences. ClaSP algorithm mines frequent closed sequential patterns in temporal transaction data. It employs a vertical database format strategy inspired by the SPADE algorithm and uses a heuristic to prune non-closed sequences inspired by the CloSpan algorithm. A crucial performance bottleneck of vertical algorithms such as SPADE, SPAM, ClaSP is that they use a generate-candidate-and-test approach that can generate a large amount of infrequent candidates. To address this issue, a generic candidate pruning mechanism based on the item co-occurrences is proposed in [8] using a new structure named CMAP (Co-occurrence MAP) for storing co-occurrence information. This pruning mechanism is integrated into three state-of-the-art algorithms ClaSP, SPADE and SPAM and the resulting algorithms are renamed as CM-ClaSP, CM-SPADE and CM-SPAM.

All the existing and most commonly used sequential pattern mining approaches are based on the support model which uses single minimum support(minsup) threshold for the entire database with the implicit assumption that all items in the data are of the same nature and/or have similar frequencies in the data. This is often not observed in many of the real-life applications where some items may appear very frequently in the data, while others may rarely appear. This leads to the dilemma which is termed in the literature as rare item problem [9] wherein infrequent or rare items will not be extracted if the minsup threshold is set to a very high value. Further, setting the minsup threshold to a very low value to retrieve rare items may cause combinatorial explosion due to the association of frequent items with one another in all possible ways and many of these will be meaningless. It is evident that using a single minsup for the whole database is inadequate as it cannot capture the inherent natures and frequency differences of the items in the database. In order to tackle this problem, some adhoc and approximate approaches [10][11][12] were proposed without much progress. The approaches should be based on the fact that different rules may need to satisfy different minimum supports depending on the type of items involved. Hence, the judicious selection of SPM approaches and minsup

Manuscript received March 17, 2016; revised March 30, 2016.

A. R. Naseer is Principal & Professor at the Department of Computer Science & Engineering, Jyothishmathi Institute of Technology & Science(JITS Karimnagar) affiliated to Jawaharlal Nehru Technological University(JNTU), Hyderabad, Telangana State, India (corresponding author phone: +91 9052430745; e-mail: dr_arnaseer@hotmail.com).

V. Malsore is with the Department of Computer Science & Engineering, Jyothishmathi Institute of Technology & Science(JITS Karimnagar) affiliated to Jawaharlal Nehru Technological University(JNTU), Hyderabad, Telangana State, India(e-mail: malsoru@gmail.com).

values becomes a challenge with the variety of datasets pertaining to various application domains.

This paper presents the domain specific performance analysis of the most commonly used sequential pattern mining approaches on real-life datasets. In this work, we considered nine most commonly used SPM approaches and eight real-life datasets having varied characteristics and representing five different domains - web click stream, text from books, sign language utterances, protein sequences and Retail data from Supermarket.

The rest of the paper is organized as follows. Section II presents the motivation and proposed work. The performance evaluation of eight Sequential pattern mining approaches on nine real-life datasets are presented in results and discussion Section III, followed by concluding remarks in section IV.

II. Motivation and Proposed Work

Most of the Sequential Pattern Mining algorithms developed consider only the IBM Synthetic Benchmarks to compare their performance with other previously reported approaches without considering the real-life datasets. Moreover, there has been no performance evaluation reported using Real-life datasets drawn from various application domains in order to determine which particular approach best suits the given application domain based on performance measures such as total Execution time and maximum memory utilized for retrieving maximum number of frequent sequential patterns over a variation in minsup threshold values.

It is to be further noted that there are several application domains which require sequential patterns appearing infrequently but they are important to be considered which can be captured with lower thresholds i.e., with lower minsup. They are ample number of application domains which need sequential patterns which appear most frequently which can be retrieved from the datasets with higher thresholds, i.e. higher minsup. There are large number of application domains which require frequent sequence patterns within a certain allowable range of minsup thresholds. Moreover, shorter and longer sequences are also important to be considered in many of the application domains. In most of the cases, these sequences are missed out due to the selection of minsup threshold values normally in the middle of the threshold range.

For example, a pharmaceutical store requires information on infrequent sequential patterns, frequent sequential patterns and most frequent sequential patterns most of the time. This is a critical support system for health care unit which need to keep track of the medicines considering these sequential patterns and stock them accordingly so that patients need may be satisfied instantaneously. Whereas for a Retail Industry or supermarket, frequent sequential patterns and most frequent sequential patterns are the most important items to be considered to stock up in large numbers to satisfy the immediate needs of the customers. Hence, it is required to develop a framework for mining sequential patterns in accordance with various application domain needs and to provide certain guidelines for selecting suitable approach that suits the given application domain. In order to provide a general framework taking into account the above discussed issues, it is required to carry out domain specific performance analysis of the most commonly used SPM approaches with the main objective of analyzing the behavior of these algorithms on the real-life datasets which represent characteristics of truly real-life situations rather than the synthetic ones. In this work, we carried out performance evaluation of the nine popular SPM approaches - GSP, SPADE, SPAM, PrefixSpan, CloSpan, ClaSP, CM-SPADE, CM-SPAM, CM-ClaSp on eight real-life datasets having varied characteristics and representing five different domains - web click stream, text from books, sign language utterances, protein sequences and Retail data from Supermarket. Performance metrics used in this study are Total execution Time taken and Main Memory utilized by each of the algorithm in retrieving maximum number of frequent sequential patterns. The other parameter considered in this work is the rate of decay of frequent sequential patterns over a wider variation of minsup threshold.

The interesting aspect of this study is to determine the algorithm which best suits a particular scenario by considering the following three situations requiring:

a) Strict Time Constraints (with no Memory Constraint)
b) Strict Memory Constraints (with no Time Constraint)
c) Moderate Time & Memory Constraints

Depending on the nature and demands of various applications and scenarios, most suitable algorithm may be selected which best suits that particular situation. Moreover, the outcomes of this study can be used to provide certain guidelines in building efficient recommendation System for judicious selection of algorithm(s) and minsup values for a given application domain considering the nature of the applications and characteristics of the involved Data Sets.

Second interesting aspect of this study is to analyze the count and length of the frequent sequence patterns generated and the decay i.e., rate of decrease in the occurrence of these frequent sequential patterns over the variation in minsup threshold from low to high values. This study not only throws some light on the amount of frequent sequential patterns generated but also on the length of the sequential patterns identified with the variation of minsup from low to a high value. This would also give an insight into the kind of sequential patterns one would be interested to look for over a range of minsup threshold values.

Third interesting aspect is that this study can also be used to design and implement an improved & efficient algorithm taking into consideration the nature of the application domains, the characteristics of the datasets in the domain and behavior of the existing algorithms on these patterns/datasets.

III. Results and Discussion

In this section, we present the extensive experiments carried out to assess the performance of the GSP, SPADE, SPAM, PrefixSpan, CloSpan, ClaSP, CM-SPADE, CM-SPAM, CM-ClaSp algorithms. For this study, we have used the SPMF platform which is an open source data mining platform written in Java maintained by P. Fournier Viger [13]. All algorithms were implemented in Java. Experiments were performed on a system with Core i7 processor with 8 GB RAM running Windows 7. All memory measurements were done using the Java API. Experiments were carried out on eight real-life datasets having varied characteristics and representing five different types domains - web click stream,

ISBN: 978-988-19253-0-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

text from books, sign language utterances, protein sequences and Retail data from Supermarket. Those real-life datasets are Kosarak (Web Click-Stream domain), Leviathan (Book (Novel) domain), Sign (Sign Language Utterance Domain), FIFA(Web Click-Stream Domain), Snake(Protein Sequences Domain), Bible(Book Conversion Domain), MSNBC (Web Click Stream Domain) & Retail (Super market Domain).

A. Performance Evaluation – Total execution Time and Main Memory Utilized

In this section, we present the results on the total execution time taken and maximum memory consumed by the nine SPM algorithms on the eight data sets by varying the minimum support from low value to high value. This variation was done to analyze the behavior of these nine algorithms in finding out the sequential patterns over the minsup variations and their computational time and maximum Memory requirements in generating the required sequences for a given minsup threshold. Due to the paucity of space, we have included in this section the figures depicting Total Execution Time vs minsup plots, Max. Memory vs. minsup plots and Performance comparison tables for only six selected datasets - Kosarak, Leviathan, Sign, Bible, snake & Retail representative of six application domains. For the remaining datasets, we have presented only the performance results. Figures 1 to 12 show the graphs plotted by considering total execution time taken and maximum memory required by GSP, SPADE, SPAM, PrefixSpan, CloSpan, CM-SPADE, CM-SPAM, CM-ClaSP and ClaSP algorithms for generating sequential patterns from Kosarak, Leviathan, Sign, Bible, Snake and Retail datasets with respect to variations in minsup threshold.

Kosarak dataset (Web click-stream) : It is evident from the figs. 1 & 2 that GSP performance is extremely poor when both Total Execution Time and Main Memory usage are considered for minsup threshold value of 1. Since GSP performs poorly, it is compared with all other algorithms and the results are presented in the table I. SPADE takes least execution time and it is 78.69 (TFT) times faster than GSP. Further, CloSpan performs very well when only Max. memory requirement is considered. It requires 78.96%(PLM) less memory when compared to poor performing GSP algorithm.

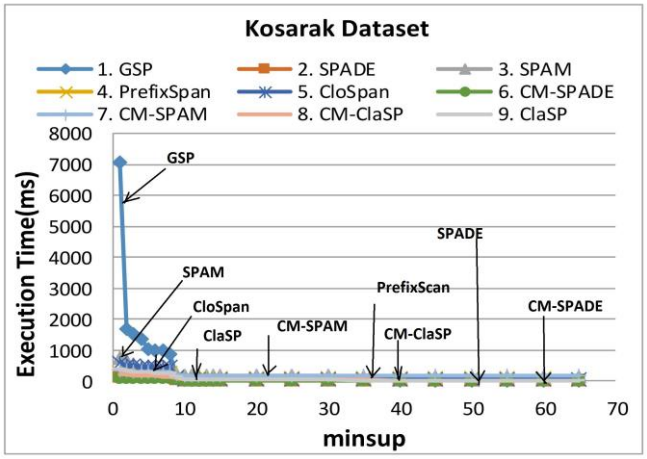

Fig 1. Kosarak Dataset-Total Execution Time vs minsup

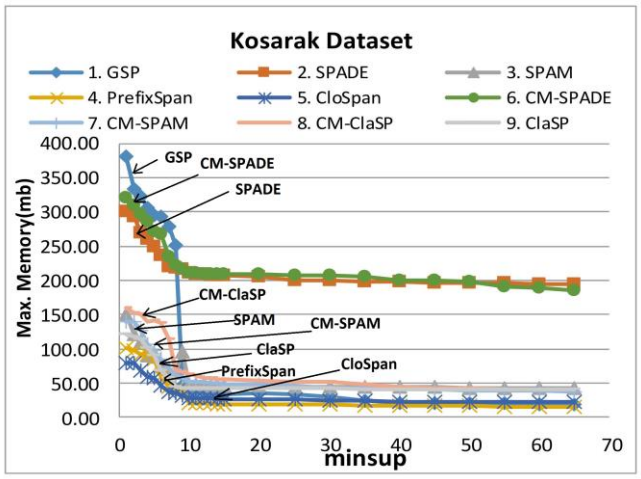

Fig 2. Kosarak Dataset- Max. Memory vs minsup

Table I: Kosarak : Comparison of Execution Time & Max Memory Usage
0

KOSARAK DATASET when minsup=1			
Execution Time		Max Memory Required	
Algorithm compared with GSP	TFT (Times faster than) GSP	Algorithm compared with GSP	PLM (%age of less memory) required when compared to GSP
CM-SPADE	78.69	CloSpan	78.96
SPADE	47.21	PrefixSpan	73.53
CM-ClaSP	33.72	ClaSP	68.22
PrefixSpan	26.23	CM-SPAM	63.09
CM-SPAM	21.46	SPAM	60.32
ClaSP	19.14	CM-ClaSP	57.93
CloSpan	12.00	SPADE	20.94
SPAM	9.70	CM-SPADE	15.76

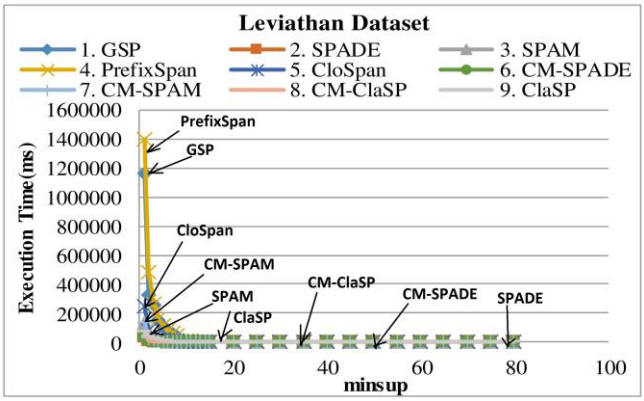

Fig 3 Leviathan Dataset-Execution Time vs minsup

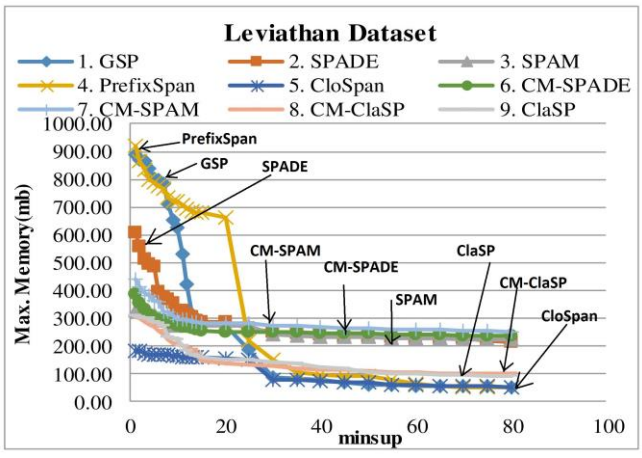

Fig 4. Leviathan Dataset- Max. Memory vs minsup

ISBN: 978-988-19253-0-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Leviathan dataset(Book - Novel): It is clear from figs. 3 & 4 that PrefixSpan performance is poor when both Total Execution Time and Main Memory usage are considered for minsup threshold value of 1. As seen in Table II, SPADE takes least execution time and it is 34.61 (TFT) times faster than PrefixSpan. Further, CloSpan performs very well when only Max. memory requirement is considered. It requires 80.19%(PLM) less memory when compared to poor performing PrefixSpan algorithm.

Table II-Leviathan :Comparison of Execution Time & Main Memory Usage

LEVIATHAN DATASET when minsup=1			
Execution Time		Max. Memory Required	
Algorithm compared with PrefixSpan	TFT (Times faster than) PrefixSpan	Algorithm compared with PrefixSpan	PLM (%age of less memory) required when compared to PrefixSpan
SPADE	34.61	CloSpan	80.19
CM-SPADE	34.58	CM-ClaSP	67.05
CM-ClaSP	27.45	ClaSP	66.37
ClaSP	19.71	SPAM	64.42
SPAM	10.28	CM-SPADE	58.35
CM-SPAM	10.20	CM-SPAM	52.43
CloSpan	5.75	SPADE	33.88
GSP	1.20	GSP	3.19

Sign dataset (Sign-Language Utterance): It is seen from figs. 5 & 6 that GSP performance is extremely poor when Total Execution Time is considered and ClaSP performs poorly as far as Max Memory usage is considered for minsup threshold value of 1. As evident in Table III, SPADE takes least execution time and it is 13.45 times faster than GSP.

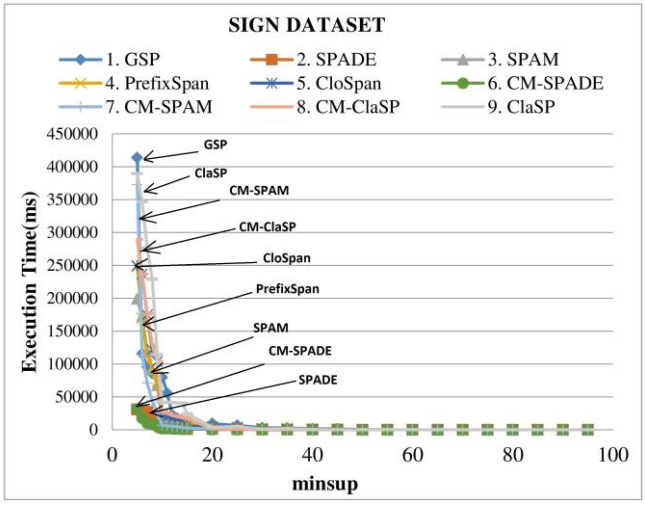

Fig 5. Sign Dataset-Total Execution Time vs minsup

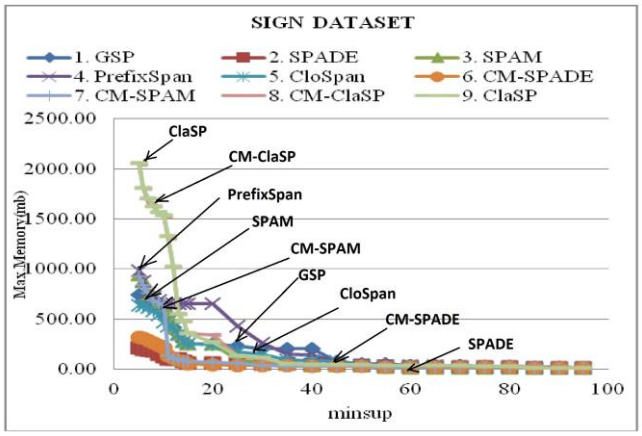

Fig 6. Sign Dataset- Max. Memory vs minsup

Further, SPADE performs very well when only Max. memory requirement is considered. It requires 89.77% less memory when compared to poor performing ClaSP algorithm.

Table III - Sign: Comparison of Execution Time & Main Memory Usage

SIGN Data Set when minsup=5			
Execution Time		Main Memory Usage	
Algorithm compared with GSP	TFT(Times faster than) GSP	Algorithm compared with ClaSP	PLM(%age of less memory) required when compared to ClaSP
SPADE	13.45	SPADE	89.77
CM-SPADE	13.24	CM-SPADE	84.43
SPAM	2.08	CloSpan	68.91
PrefixScan	1.67	GSP	64.05
CloSpan	1.66	CM-SPAM	54.83
CM-ClaSP	1.43	SPAM	54.35
CM-SPAM	1.11	PrefixSpan	52.41
ClaSP	1.06	CM-ClaSP	0.97

Snake dataset (Protein Sequences): It is interesting to note from figs. 7 & 8 that all algorithms fail for variation of minsup value below 40 due to insufficient memory. PrefixScan performance is extremely poor when both Total Execution Time and Main Memory usage are considered for minsup threshold value of 40. As seen in Table IV, CM-SPADE takes least execution time and it is 27.61 times faster than PrefixScan. Further, SPADE performs very well when only Max. memory requirement is considered. It requires 61.09% less memory when compared to poor performing PrefixScan algorithm.

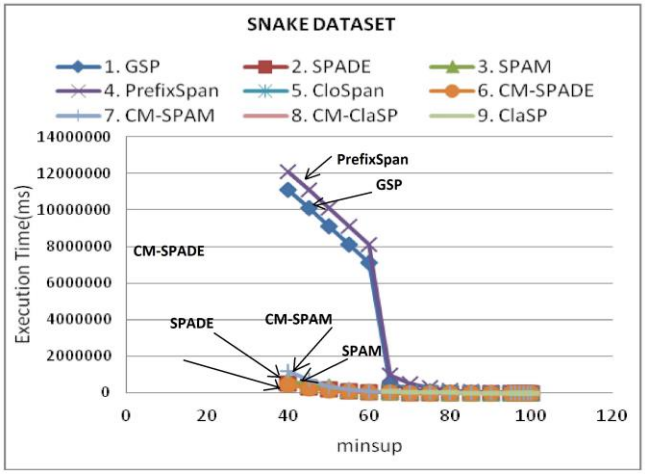

Figure 7. Snake Dataset-Total Execution Time vs minsup

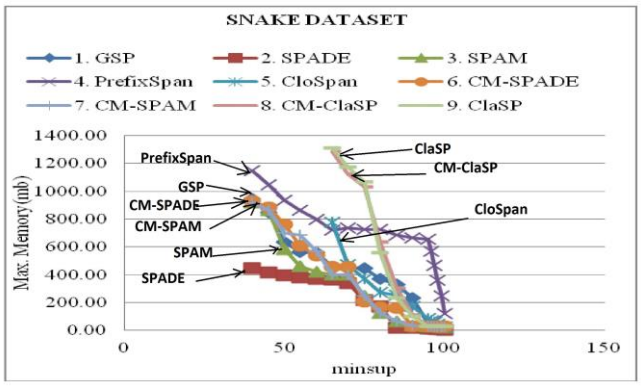

Figure 8. Snake Dataset- Max. Memory vs minsup

ISBN: 978-988-19253-0-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Table IV :Snake -Comparison of Execution Time & Main Memory Usage

SNAKE Data Set when minsup = 40			
Execution Time		Maim Memory Usage	
Algorithm compared with PrefixScan	Times faster than PrefixScan	Algorithm compared with PrefixScan	%age of less memory required when compared to PrefixScan
CM-SPADE	27.61	SPADE	61.09
SPADE	25.29	SPAM	19.95
SPAM	17.21	CM-SPAM	19.94
CM-SPAM	10.08	CM-SPADE	18.73
GSP	1.09	GSP	17.45
CM-ClaSP, CloSpan, ClaSP algorithms fail for variation of minsup from 40 to 60 due to insufficient memory			

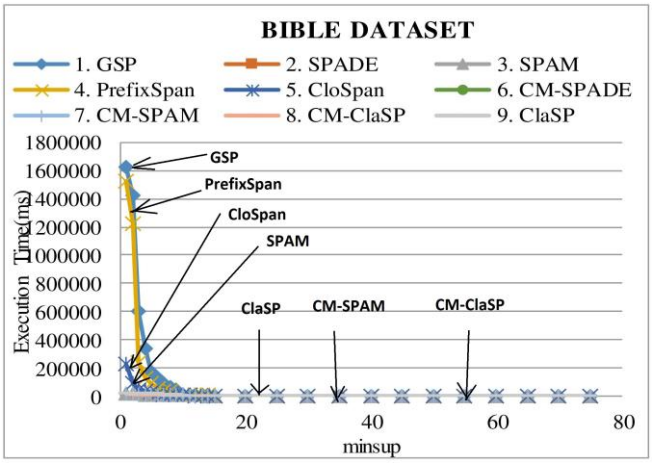

Fig 9. Bible Dataset-Total Execution Time vs minsup

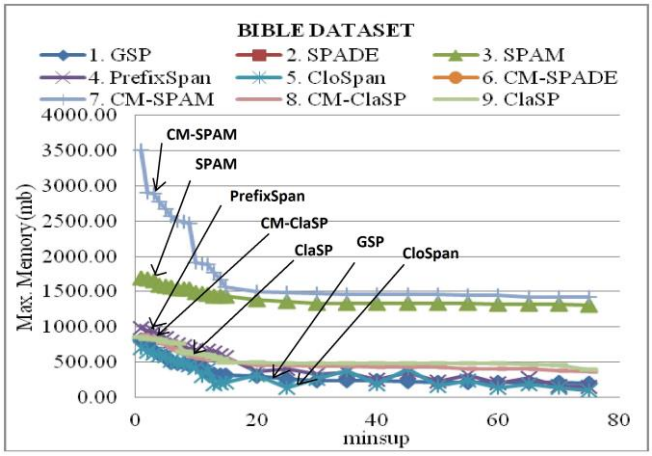

Fig 10. Bible Dataset- Max. Memory vs minsup

Table V-Bible:Comparison of Execution Time & Main Memory Usage

BIBLE Data Set when minsup =1			
Execution Time		Maim Memory Usage	
Algorithm compared with GSP	Times faster than GSP	Algorithm compared with CM-SPAM	%age of less memory required when compared to CM-SPAM
CM-ClaSP	129.25	CloSpan	80.01
CM-SPAM	103.72	GSP	76.71
ClaSP	84.49	ClaSP	75.68
SPAM	82.61	CM-ClaSP	74.83
CloSpan	7.03	PrefixSpan	71.98
PrefixSpan	1.07	SPAM	51.80
SPADE and CM-SPADE algorithms fail for all variations of minsup due to insufficient memory			

Bible (Book Conversion): It is interesting to note from figs. 9 & 10 that SPADE & CM-SPADE algorithms fail for entire variation of minsup due to insufficient memory. GSP performance is extremely poor when Total Execution. Time is considered and CM-SPAM performs poorly when Max Memory usage is considered for minsup threshold value of 1. As seen in the table V, CM-ClaSP takes least execution time and it is 129.25 times faster than GSP. Further, CloSpan performs extremely well when only Max. memory requirement is considered. It requires 80.01% less memory when compared to poor performing CM-SPAM algorithm.

FIFA dataset (Web click-stream): PrefixSpan performs poorly when Total Execution Time and Max Memory usage are considered for minsup threshold value of 8. CM-SPADE takes least execution time and it is 15.06 times faster than PrefixSpan. Further, CloSpan performs very well when only Max. memory requirement is considered. It requires 67.58% less memory when compared to poor performing PrefixScan algorithm.

MSNBC dataset (Web click-stream): It is interesting to note that ClaSP and CM-ClaSP algorithms fail for entire variation of minsup due to insufficient memory. GSP performance is extremely poor when Total Execution Time is considered and PrefixScan performs poorly when Max Memory usage is considered for minsup threshold value of 1. SPADE takes least execution time and it is 1503.03 times faster than GSP. Further, CM-SPADE performs extremely well when only Max. memory requirement is considered. It requires 98.96% less memory when compared to poor performing PrefixScan algorithm.

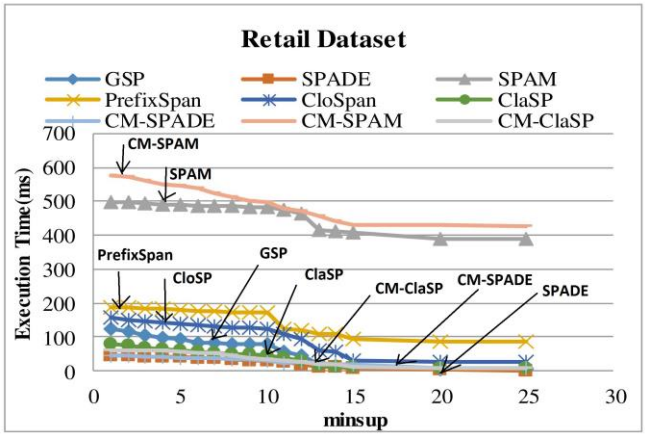

Figure 11. Retail Dataset -Total Execution Time vs minsup

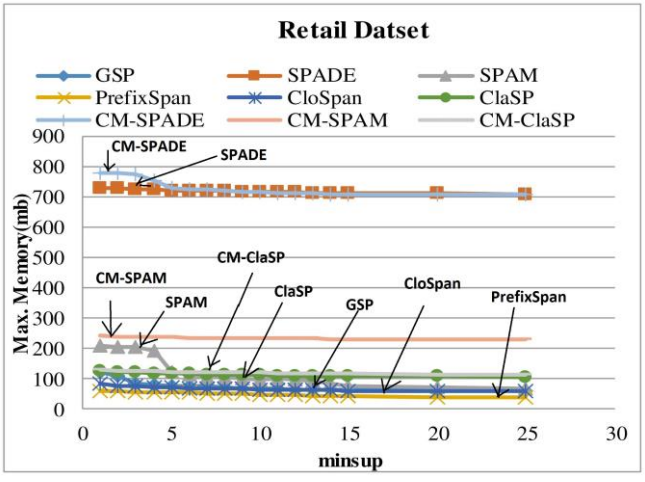

Fig 12. Retail Dataset- Max. Memory vs minsup

Retail dataset (Belgian Retail Store): It is evident from figs. 11 & 12 that CM-SPAM performance is poor when

ISBN: 978-988-19253-0-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

Total Execution Time is considered and CM-SPADE performs badly when Max Memory usage is considered for minsup threshold value of 1. As it is evident from table VI, SPADE takes least execution time and it is 12.54 times faster than CM-SPAM. Further, PrefixScan performs extremely well when only Max. memory requirement is considered. It requires 92.53% less memory when compared to poor performing CM-SPADE algorithm.

Table VI-Retail: Comparison of Execution Time & Main Memory Usage

RETAIL Data Set when minsup =1			
Execution Time		Maim Memory Usage	
Algorithm compared with CM-SPAM	TFT (Times faster than) CM-SPAM	Algorithm compared with CM-SPADE	PLM-%age of less memory required when compared to CM-SPADE
SPADE	12.54	PrefixScan	92.53
CM- SPADE	12.28	CloSpan	89.75
CM-ClaSP	9.31	GSP	84.92
ClaSP	7.4	ClaSP	84.38
GSP	4.65	CM-ClaSP	83.76
CloSpan	3.70	SPAM	73.60
PrefixScan	3.09	CM-SPAM	69.27
SPAM	1.16	SPADE	6.51

B. Frequent Sequential Pattern count and Sequence Length

In this section, we present the results of the study performed to assess the influence of variation in the minsup threshold on the Frequent Sequential Patterns(FSPs) retrieved from the eight datasets by the nine most popular data mining algorithms. This study not only throws some light on the amount of FSPs generated but also on the length of the sequential patterns identified with the variation of minsup from low to a high value. This would also give an insight into the kind of sequential patterns one would be interested to look for over a range of minsup threshold values.

Table VII - Frequent Sequence Length & corresponding Sequence Count

FSL	**Kosarak 373 FSP Count**	**FIFA 117738 FSP Count**	**Retail 363 FSP Count**	**FSL**	**MSNBC 45503 FSP Count**	**Snake 13133171 FSP Count**
1	56	132	54	1	17	20
2	134	1187	120	2	241	351
3	122	11397	133	3	2449	4601
4	52	12627	47	4	8701	26567
5	9	88189	9	5	11624	130632
6	-	4206	-	6	9296	261149
FSL	**Leviathan 167189 FSP Count**	**Bible 23308 FSP Count**	**Sign 944486 FSP Count**	7	5789	2180129
				8	3124	2572713
				9	1876	4517459
1	345	239	151	10	1307	2322552
2	4243	1708	8382	11	765	1022667
3	18738	3958	65058	12	282	30198
4	38480	4970	158724	13	30	7956
5	43922	4292	230061	14	2	31243
6	33194	3170	232494	15	-	18600
7	18173	2457	158071	16	-	4710
8	7433	1688	69388	17	-	1508
9	2222	717	18994	18	-	112
10	414	104	2942	19	-	4
11	25	5	216	**FSL – Frequent Sequence Length**		
12	-	-	5	**FSP- Frequent Sequential Pattern**		

The table VII shows the count of FSPs of different lengths retrieved from eight datasets by the SPM approaches when the minsup threshold value is at the least minimum. As seen in table VII, SPM approaches extracted 373 FSPs having upto 5-length patterns from Kosarak dataset when minsup=1, from Bible dataset 23308 FSPs having upto 11-length patterns when minsup=1, from MSNBC dataset 45503 FSPs having upto 14-length patterns when minsup=1, from Retail dataset 363 FSPs having upto 5-length patterns when minsup=1, from Leviathan dataset 167189 FSPs consisting upto 11-length patterns when minsup=1, from sign dataset when minsup=5, algorithms generate 944486 FSPs comprising upto 12-length patterns, from FIFA dataset 117738 FSPs having upto 6-length patterns when minsup=8, and from Snake dataset 13133171 FSPs comprising upto 19-length sequences when minsup=40.

Considering the Frequent Sequence Count decay over minsup variation as depicted in figure 13, it is to be noted that Kosarak, Leviathan, Bible, MSNBC, Retail datasets show almost similar trend of sudden(sharp) decrease in the occurrence of Frequent Sequential patterns when minsup is varied from 1 to 2.

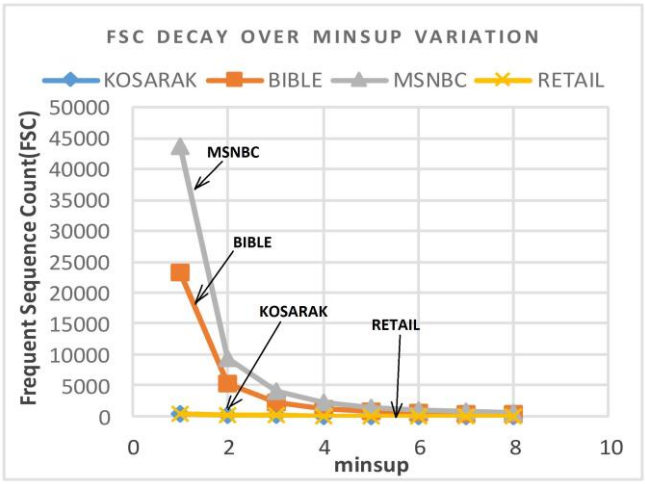

Fig 13. KBMR FSC Decay over minsup variation

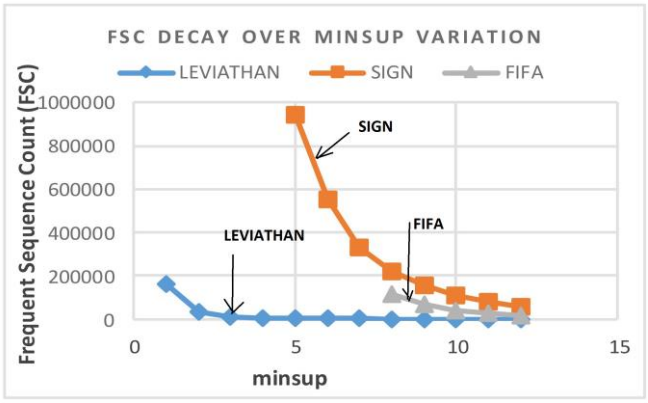

Fig 14. LSF FSC Decay over minsup variation

For instance, rate of decrease of Frequent Sequent Count is 67.56% in Kosarak, 79.64% in Leviathan, 77.29% in Bible, 78.67% in MSNBC and 68.32% in Retail datasets when minsup is changed to 2. This trend continues till further as it is evident from the figure when minsup is increased to 5, most of the higher length frequent sequences vanish. For instance, when minsup is increased to 5, 91.69% of the FSPs disappear in the case of Kosarak dataset, 97.71% in the case of Leviathan, 96.68% in the case of Bible, 96.80% in the case of MSNBC and 91.46% in the

ISBN: 978-988-19253-0-5
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

case of Retail dataset. For variation of minsup beyond 5, less than 10% or even much less (below 5%) of the FSPs exist.

As shown in figure 14, the datasets having larger length FSPs like Sign, FIFA and Snake, the rate of decrease of the FSPs is somewhat slower. For instance, 41.27% of the FSPs vanish in the case of Sign Dataset when minsup is varied from 5 to 6, 41.85% in the case of FIFA when minsup varied from 8 to 9, 55.19% in the case of Snake Dataset when minsup is changed from 40 to 45. Further, substantial amount of higher length patterns exist over a wide variation of misup threshold.

This study draws the conclusion that largest length FSPs appear only when minsup threshold is at its least minimum value for all datasets and remain live only for a very short variation of minsup. It is also observed that in some of the datasets with larger length FSPs, substantial amount of higher length patterns remain live over a wider variation of misup threshold.

C. Recommendation

The detailed analysis presented in subsections A & B of Section III provides sufficient guidelines to build a recommendation system for selection of appropriate algorithm and minsup for a given application domain. The recommended algorithms for the eight datasets based on the performance results are given in table VIII. It has been found that SPADE and CM-SPADE outperform all other algorithms in most of the Domains as far as the total execution time is concerned. It is interesting to note that for Bible Dataset from Book(Conversion) domain, CM-ClaSP is the preferred approach when least total execution time is considered. Further, when we consider only minimum memory usage, CloSpan fairs extremely well for datasets (Kosarak, FIFA, Bible, Leviathan) from domains Webclick Streams, Book(Conversion), and Book(Novel). For datasets from Sign language utterance and Protein Sequences, SPADE is the recommended approach and PrefixScan is the ultimate choice for Retail dataset from Super Market domain as far as the least memory usage is considered.

Table VIII: Recommended algorithms for various domains

Datasets	**Domain**	**Selected Algorithm based on**		
		least execution time	**Min memory utilized**	**Both Execution time & Memory**
Kosarak	Web Click-Stream	CM-SPADE	CloSpan	CM-ClaSP, PrefixSpan
Leviathan	Book (Novel)	SPADE	CloSpan	CM-SPADE, CM-ClaSP
Sign	Sign Language Utterance	SPADE	SPADE	SPADE, CM-SPADE
FIFA	Web Click-Stream	CM-SPADE	CloSpan	CM-SPADE, SPADE
Snake	Protein Sequences	CM-SPADE	SPADE	CM-SPADE, SPADE
Bible	Book(Conversion)	CM-ClaSP	CloSpan	CM-ClaSP, ClaSP
MSNBC	Web Click-Stream	SPADE	CM-SPADE	SPADE, CM-SPADE
Retail	Super Market	SPADE	PrefixScan	CM-ClaSP, ClaSP

This study also recommends to consider a range of minsup threshold values for retrieving longer sequences from the datasets so that interesting sequences can be captured within the specified window at the lower spectrum of minsup threshold variations instead of using a single minsup value.

IV. Conclusion

In this paper, we presented our experimental study to evaluate the performance of most popular nine sequential pattern mining algorithms on eight real data sets representing different application domains. In these experiments, we varied the minsup threshold from lower values to higher values in order to study the behavior of these algorithms in retrieving the sequential patterns from different datasets. Also this study was extended to assess the influence of variation in the minsup threshold on the amount of frequent sequential patterns generated and on the length of the sequential patterns identified. This work provided an insight into the kind of sequential patterns one would be interested to look for over a range of minsup threshold values.

As a future work, this study can be extended to cover all the Application domains and the findings of this detailed study can be used in building efficient recommendation System for judicious selection of algorithm(s) and minsup values for any given application domain considering the nature of the applications and characteristics of the involved Data Sets. Further, this study can also be used to design and implement an improved & efficient algorithm which takes into consideration the nature of the application domains, the characteristics of the datasets in the domains and behavior of the existing algorithms on these patterns/datasets.

References

[1] Agrawal, Rakesh, and Ramakrishnan Srikant. "Mining sequential patterns", Proceedings of the Eleventh International Conference on Data Engineering, Taipei, Taiwan, March 1995.

[2] R. Srikant and R. Agrawal, “Mining sequential patterns: Generalizations and performance improvements”, Proc. Of 5th International Conference on Extending Database Technology (EDBT’96), pp. 3-17, 1996.

[3] Zaki, Mohammed J. "SPADE: An efficient algorithm for mining frequent sequences", Machine learning 42.1-2 (2001): 31-60.

[4] Pei, Jian, et al. "Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth.", 17th International Conference on Data Engineering(ICDE), April 2001.

[5] Ayres, Jay, et al. "Sequential pattern mining using a bitmap representation", Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2002.

[6] X. Yan, J. Han, and R. Afshar, “Clospan: Mining Closed Sequential Patterns in Large Datasets,” Proc. Third SIAM Int’l Conf. Data Mining (SDM’03), pp. 166-177, May 2003.

[7] A. Gomariz, M. Campos, R. Marin, B. Goethals, “ClaSP : An Efficient Algorithm for Mining Frequent Closed Sequences”, J. Pei et al. (Eds.): PAKDD 2013, Part I, LNAI 7818, pp. 50–61, 2013.© Springer-Verlag Berlin Heidelberg 2013.

[8] P. Fournier-Viger, A. Gomariz, M. Campos, R. Thomas, “Fast Vertical Mining of Sequential Patterns using Co-Occurrence Information”, Advances in Knowledge Discovery and Data Mining, pp. 40-52, Springer International Publishing, 2014,

[9] Mannila, H. "Database methods for data mining.” *KDD-98* tutorial, 1998.

[10] Lee, W., Stolfo, S. J., and Mok K. W., 1998, ‘Mining audit data to built intrusion detection models.’ KDD-98.

[11] Han, J. and Fu, Y., 1995, ‘Discovery of multiple-level association rules from large databases.’ VLDB-95.

[12] B.Liu, W.Hsu and Y.Ma, “Mining association rules with multiple minimum supports”, Proceedings of the fifth ACM, SIGKDD Conference San diego, CA,USA, August 15-18,1999,P.341

[13] www.philippe-fournier-viger.com/spmf/index.php - SPMF open source data mining platform written in Java maintained by P. Fournier Viger.