Extracting Multi-item Sequential Patterns by Wap-tree Based Approach

Kezban Dilek Onal and Pinar Karagoz
Department of Computer Engineering, Middle East Technical University, Ankara, Turkey

Keywords: WAP-Tree (Web Access Pattern Tree), Sequential Pattern Mining, FOF (First Occurrence Forest), Sibling Principle, Web Usage Mining.

Abstract: Sequential pattern mining constitutes a basis for solution of problems in web mining, especially in web usage mining. Research on sequence mining continues seeking faster algorithms. WAP-Tree based algorithms that emerged from the web usage mining literature have shown a remarkable performance on single-item sequence databases. In this study, we investigate the application of WAP-Tree based mining to multi-item sequential pattern mining and we present MULTI-WAP-Tree, which extends WAP-Tree for multi-item sequence databases. In addition, we propose a new algorithm MULTI-FOF-SP (MULTI-FOF-Sibling Principle) that extracts patterns on MULTI-WAP-Tree. MULTI-FOF-SP is based on the previous WAP-Tree based algorithm FOF (First Occurrence Forest) and an early pruning strategy called "Sibling Principle" from the literature. Experimental results reveal that MULTI-FOF-SP finds patterns faster than PrefixSpan on dense multi-item sequence databases with small alphabets.

1 INTRODUCTION

Sequential pattern mining is one of the major tasks in data mining and it constitutes a basis for solution of pattern discovery problems in various domains (Mooney and Roddick, 2013). Use of sequential pattern mining in web usage mining problems helps discovering user navigation patterns on web sites that can guide processes like recommendation and web site design.

In some applications of sequence mining like web usage mining and bioinformatics, the sequences have fixed transaction size of 1. This specific case of sequence mining is referred as *Single-Item Sequential Pattern Mining*. In accordance with this naming, the general form of the problem is referred as *Multi-Item Sequential Pattern Mining*.

WAP-Tree (Web Access Pattern Tree) based algorithms have shown remarkable execution time performance on single-item sequential pattern mining (Mabroukeh and Ezeife, 2010). WAP-Tree is a compact data structure for representing single-item sequence databases (Pei et al., 2000). There is a considerable number of WAP-Tree based algorithms in the literature. Among the previous WAP-Tree based algorithms, PLWAP (Ezeife and Lu, 2005) is reported to outperform well known general sequential pattern mining algorithms PrefixSpan and LAPIN on single-item sequence databases (Mabroukeh and Ezeife, 2010) for single item patterns.

Inspired by the success of WAP-Tree, we designed a new data structure MULTI-WAP-Tree which extends WAP-Tree for representing multi-item/general sequence databases. Secondly, we propose a new sequential pattern mining algorithm MULTI-FOF-SP based on MULTI-WAP-Tree inspired by the WAP-Tree based algorithm FOF (First Occurrence Forest) (Peterson and Tang, 2008). MULTI-FOF-SP, integrates FOF approach and the early prunning idea "Sibling Principle" from the previous studies (Masseglia et al., 2000) and (Song et al., 2005).

We have analyzed the performance of the proposed method on several test cases. The results show that MULTI-WAP-Tree and the associated mining algorithm on this structure presents successful results, especially for dense multi-item sequence databases with small alphabets.

The rest of the paper is composed of four sections. In Section 2, we provide background information on WAP-Tree and the FOF algorithm. We present our contributions, MULTI-WAP-Tree and MULTI-FOF-SP, in Section 3 and Section 4, respectively. Finally, we present experiment results in Section 5 and conclude in Section 6.

Onal K. and Karagoz P.
Extracting Multi-item Sequential Patterns by Wap-tree Based Approach.
DOI: 10.5220/0004788102150222
In *Proceedings of the 10th International Conference on Web Information Systems and Technologies* (WEBIST-2014), pages 215-222
ISBN: 978-989-758-024-6

2 RELATED WORK AND BACKGROUND

Sequential pattern mining is the extraction of the sequences that occur at least as frequently as the minimum support *minSupport* in a sequence database *D* (Agrawal and Srikant, 1995). The sequential pattern mining algorithms in the literature can be reviewed under four categories (Mabroukeh and Ezeife, 2010), namely Apriori based, vertical projection, pattern growth and early prunning.

The WAP-Tree based algorithms follow the pattern growth approach on single-item sequential pattern mining. Since our study is focused on WAP-Tree based mining, we first review the WAP-Tree data structure in the next subsection. Secondly, we summarize the pattern growth approach in Subsection 2.2. Finally, we review the FOF (First Occurrence Forest) algorithm in comparison with the other WAP-Tree based algorithms in Subsection 2.3.

2.1 WAP-Tree

The WAP-Tree (Web Access Pattern Tree) is a tree data structure that is designed to represent a single-item sequence database. This data structure was first introduced together with the WAP-Mine algorithm (Pei et al., 2000). WAP-Mine algorithm converts the sequence database into a WAP-Tree with an initial database scan and performs mining on the WAP-Tree to find frequent patterns.

Table 1: Sample Single-item Sequence Database.

Sequence Id	Sequence
1	aba
2	adcdb
3	beae
4	ac

Figure 1 illustrates the WAP-Tree for the sequence database given in Table 1 under minimum support 0.5. Each node of the WAP-Tree comprises two fields: Item and Count. Count field of a node *n* stores the count of the sequences starting with the prefix obtained by following the path from root to *n*. The two children of *R*, *(a:3)* and *(b:1)* in Figure 1 indicate that there are 3 sequences starting with *a* and a single sequence starting with *b*. WAP-Tree provides a compact representation since shared prefixes can be encoded on the nodes.

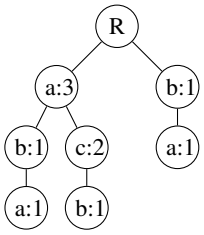


Figure 1: WAP-Tree For The Sequence Database in Table 1 Under Support Threshold 0.5.

2.2 Pattern Growth Approach

The pattern growth approach follows the divide and conquer paradigm (Han et al., 2005). The original problem of finding the frequent sequences is recursively broken down into smaller problems by shrinking the database into projected databases. The lexicographic search space is traversed depth-first and whenever a pattern is found to be frequent, the database is projected by the pattern. The mining process is recursed on the projected database.

The realization of the pattern growth approach depends on the pattern growing direction, how the projected databases are represented and located during mining (Han et al., 2005). There are two alternative directions for growing patterns: namely suffix growing and prefix growing. The patterns are grown by appending symbols in prefix growing whereas they are grown by prepending symbols in suffix growing.

$D\|_{\epsilon}$
aba
adcdb
beae
ac
(a)

$D\|_{a}$
ba
dcdb
e
c
(b)

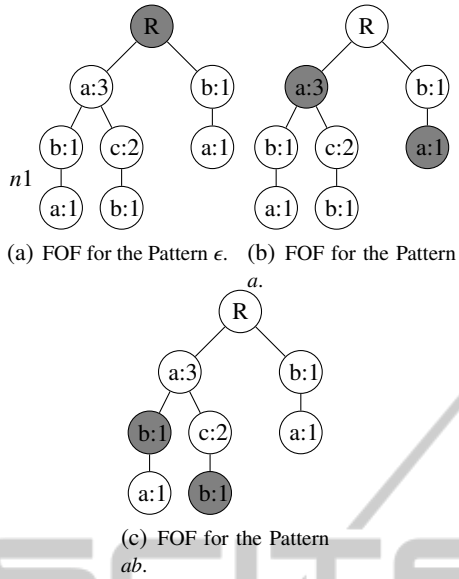
$D\|_{ab}$
a
ϵ
ϵ
ϵ
(c)

Figure 2: Projected Databases for Different Patterns.

Three sample projected databases for prefix-growing are given in horizontal database representation in Figure 2. From left to right, the tables in Figure 2 are the projected databases of the patterns ϵ,*b*,*ba* in the sequence database D given in Table 1. The projections of the sequences can be traced by following the rows of the tables at the same level. For example, the *a-projection* of the sequence *aba* is *ba*. The *b-projection* of the sequence *ba* is *a* and it is equal to the *ab-projection* of *aba*.

2.3 FOF (First Occurrence Forest)

FOF is a prefix growing WAP-Tree based algorithm. The FOF algorithm represents the sequence database as a WAP-Tree and it adopts the recursive projected



(a) FOF for the Pattern ϵ. (b) FOF for the Pattern a.

(c) FOF for the Pattern ab.

Figure 3: FOFs for Prefix Growing On WAP-Tree.

database mining approach of the pattern growth algorithms. The FOF algorithm represents the projected databases as First Occurrence Forests (FOFs). First Occurrence Forest is a list of WAP-Tree nodes which root a forest of WAP-Trees. The WAP-Trees rooted by the FOF nodes of a pattern encode the projections of the sequences by the pattern. FOF representation enables easy support counting for a pattern by summing the count values of the FOF nodes.

The mining process of the FOF algorithm is based on searching the first level occurrences of symbols recursively. At each pattern growing step by a symbol, the first level occurrences of the symbol in the forest of WAP-Trees defined by the FOF of the original pattern are assigned as the FOF nodes for the grown pattern. The mining process is recursed on the FOF nodes for the grown pattern.

The mining process of the FOF algorithm on the WAP-Tree in Figure 1 is partially illustrated in Figure 3. FOF of three different patterns are presented in Figure 3. The shaded nodes in the WAP-Trees indicate the FOF nodes. Initially, the FOF for the pattern ϵ is $\{R:4\}$. Secondly, the first level occurrences of the symbol a under the node R:4 are found as the FOF for the pattern a. The node n_1 is not included in the FOF of the symbol a since it is not a first level occurrence. In the next level of recursion, the FOF algorithm mines the FOF for a. The FOF for ab is the set of first level b occurrences in the sub-trees rooted by the shaded nodes for pattern a. FOF of ab is found by searching the first level b occurrences in the sub-trees rooted by the FOF nodes for the pattern a.

The projected database representation as a list of nodes, i.e. FOF, was adapted by previous WAP-Tree based algorithms except WAP-Mine. All WAP-Tree based algorithms in the literature follow pattern-growth approach. However, they differ in the pattern growing direction. WAP-Mine does suffix growing and represents projected databases as WAP-Trees. All of the latter algorithms, namely PLWAP (Ezeife and Lu, 2005), FLWAP (Tang et al., 2006), FOF (Peterson and Tang, 2008) and BLWAP (Liu and Liu, 2010), do prefix growing.

Prefix growing WAP-Tree based algorithms differ in their approach for locating the first level occurrences of a symbol in a WAP-Tree. The FOF algorithm locates first level occurrences of an item by simple depth-first search. On the contrary, PLWAP, BLWAP and FLWAP leverage additional data structures *links* and *header table* besides the WAP-Tree. The links connect all occurrences of an item in a WAP-Tree and the links can be followed starting from the header table. The first level occurences of an item in a WAP-Tree can be easily found by following the links of an item and filtering the occurrences which are found at the first level. The FOF algorithm is reported to outperform PLWAP and FLWAP in (Peterson and Tang, 2008) although. FOF uses less memory since it mines WAP-Tree without additional structures. Besides, although the links provide direct access to the occurrences, filtering the first level occurrences brings an extra cost to PLWAP and FLWAP.

3 MULTI-WAP-TREE

MULTI-WAP-Tree is an extended WAP-Tree which can represent both single and multi-item sequence databases. MULTI-WAP-Tree is identical to the WAP-Tree in that it contains only frequent items in its nodes and each sequence in the sequence database is encoded in MULTI-WAP-Tree on a path from the root to a node of the tree.

MULTI-WAP-Tree differs from the WAP-Tree in two points:

- MULTI-WAP-Tree has two types of edges between nodes : *S-Edge* and *I-Edge* whereas WAP-Tree has a single edge type.
- MULTI-WAP-Tree nodes keep a pointer to their parent nodes in addition to the fields of WAP-Tree.

MULTI-WAP-Tree is able to represent a multi-item sequence database owing to two different edge types between nodes. A multi-item sequence is composed of a series of item-sets. WAP-Tree is not able to represent multi-item databases since it cannot express boundaries between item-sets in multi-item se-

quences. S-Edges of MULTI-WAP-Tree can encode item set boundaries. An S-Edge from a node to its child indicates the separator between two item sets: item set ending with the parent and item set starting with the child. On the contrary, the nodes connected with I-Edges are always in the same item set.

Figure 4(d) shows the MULTI-WAP-Tree for the sample multi-item database in Table 2 under the support threshold 0.5. The dashed edges in the figure are I-Edges, whereas the plain edges are S-Edges. The prefix represented by the gray coloured node is *(ab)*. Although both of the edges originating from this node point to *c* labeled nodes, they yield different prefixes since their edge types are different. The child of this node connected with an S-Edge represents the prefix *(ab)(c)*, whereas the other child connected with the I-Edge represents the prefix *(abc)*.

Table 2: Sample Multi-item Sequence Database.

Sequence Id	Sequence
1	(ab)(c)
2	(a)(b)(c)
3	(abc)(c)

The second extension to the WAP-Tree, "pointer to parent node" in MULTI-WAP-Tree nodes, enables tracking item sets upwards in the tree in the mining phase. This additional field does not contribute to the database representation but it is an important construct for mining MULTI-WAP-Tree.

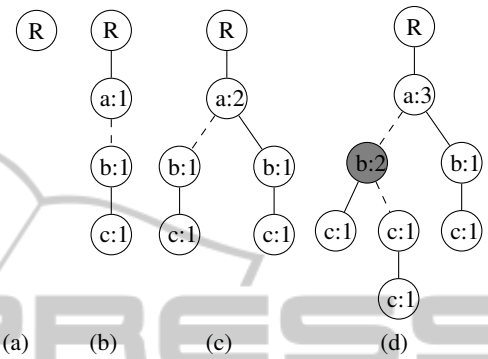
4 MULTI-FOF-SP

MULTI-FOF-SP algorithm is a multi-item sequential pattern mining algorithm based on the representation MULTI-WAP-Tree. MULTI-FOF-SP algorithm has three basic steps, as in all WAP-Tree based algorithms:

1. Scan database to find frequent items.
2. Scan database and build MULTI-WAP-Tree.
3. Mine frequent patterns from MULTI-WAP-Tree.

4.1 Building MULTI-WAP-Tree

For MULTI-WAP-Tree construction, after eliminating the infrequent items, each sequence is inserted into the tree starting from the root, updating counts of shared prefix nodes and inserting new nodes for the unshared suffix part, as in WAP-Tree construction. While constructing a WAP-Tree, checking only equality of item fields of nodes is sufficient. However, in the MULTI-WAP-Tree case, checking the equality of edge types is also required. To illustrate, Figure 4 depicts the MULTI-WAP-Tree database construction algorithm on the mini database in Table 2. When inserting the second sequence *(a)(b)(c)*, although the *a* node already has a *b* child, since this child is connected with an I-Edge, a new *b* node is added with an S-Edge.



(a) (b) (c) (d)

Figure 4: Building Steps of the MULTI-WAP-Tree for the Database in Table 2. Left to right: MULTI-WAP-Tree after sequences *(ab)(c)*, *(a)(b)(c)*, *(abc)(c)* are inserted successively.

4.2 Mining MULTI-WAP-Tree

MULTI-FOF-SP follows prefix growing and FOF approach for extracting multi-item sequential patterns on MULTI-WAP-Tree. MULTI-FOF-SP represents projected databases in FOF form and finds first level occurrences of an item by simple depth first search on the MULTI-WAP-Tree, as in FOF algorithm. However, MULTI-FOF-SP algorithm differs from FOF in two points which are explained in detail in the following two subsections.

4.2.1 S-Occurrence vs. I-Occurrence

Existence of two different edge types S-Edge and I-Edge in MULTI-WAP-Tree necessitates distinction between S-Occurrences and I-Occurrences. S-Occurrences are the first level occurrences that yield a sequence extension, whereas I-Occurrences yield an item-set extension.

MULTI-FOF-SP algorithm treats S-Occurrences and I-Occurrences as occurrences of different symbols and searches for them separately. Whenever an occurrence is located, it is subjected to a test for its occurrence type. If an occurrence is found in the subtree rooted by *startNode*, the algorithm decides on the type of occurrence based on the three rules below:

1. A node is an S-Occurrence if there exists at least one S-Edge on the path from the *startNode* to this node.

2. A node is an I-Occurrence if there are only I-Edges on the path from the *startNode* to this node.
3. A node is an I-Occurrence if the last itemset of the grown pattern can be found by following the ancestor nodes of the node before an S-Edge is encountered.

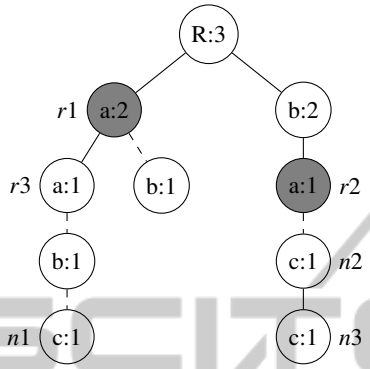


Figure 5: Find First Occurrences Illustration.

It is crucial to note that a node can be both an S-Occurrence and I-Occurrence at the same time. To illustrate the rules above, consider the MULTI-WAP-Tree in Figure 5. The gray shaded nodes *r1* and *r2* represent the FOF for the pattern *(a)*. There exists three *c* nodes, namely *n1*, *n2* and *n3*, under this FOF. There are two S-Occurrences {*n1,n3*} of *c* and two I-Occurrences {*n1, n2*} of *c* under the FOF for the pattern *(a)*. I-Occurrences contribute to support count of *(ac)*, whereas S-Occurrences contribute to that of *(a)(c)*.

n2 is an I-Occurrence according to Rule 1 since there exists no S-Edges between *n2* and its ancestor *r2*. On the contrary, *n3* is an S-Occurrence since there exists an S-Edge on the route from *n3* to *r2*. Finally, *n1* is both an S-Occurrence and I-Occurrence since it matches both Rule 1 and Rule 3. *n1* is an S-Occurrence because there exists an S-Edge between *n1* and *r1*. In addition, *n1* is an I-Occurrence since the last item-set of the grown pattern, namely (ac), can be obtained by backtracking with only I-Edges from *n1* to the ancestor node *r3*. The bactracking operation is performed using the pointers to the parent nodes which were mentioned as a difference from the WAP-Tree previously.

4.2.2 Sibling Principle

Sibling principle is an early pruning idea which was used in the previous studies (Song et al., 2005) and (Masseglia et al., 2000). It is an expression of the Apriori principle (Agrawal and Srikant, 1995) on the lexicographic search tree. According to the Apriori principle, the sequences can be judged as infrequent if any of its sub-sequences is known to be infrequent. This principle can be modeled in lexicographic tree considering the siblings of a frequent pattern. Sibling principle requires checking sibling nodes of a node in the lexicographic tree and imposes constraints on the set of grown patterns. If a sibling node *s* of a node *n* is not frequent, a sequence which is a super-sequence of both *s* and *n* is pruned.

Figure 6 shows the subspace of the lexicographic search space processed by the MULTI-FOF-SP algorithm during mining the database in Table 2. The dashed edges in the tree indicate item-set extensions whereas the normal edges correspond to sequence extensions. The ✓ sign indicates the sequence is frequent. All the patterns represented by the nodes except the underlined nodes are subjected to support counting by the algorithm MULTI-FOF-SP during mining the WAP-Tree given in Figure 4(d). The underlined nodes encode the sequences that are pruned owing to the sibling principle by MULTI-FOF-SP algorithm. For instance, the leftmost *c* node in the tree which represents the pattern *(abc)* can be early pruned by the sibling principle since the sibling pattern *(ac)* is found to be infrequent.

The numbers on the nodes indicate the order of traverse by MULTI-FOF-SP during mining. It is crucial to note that, the search space is traversed with a hybrid traversal strategy in order to apply the sibling principle. MULTI-FOF-SP combines the depth-first traversal of pattern growth approach and the breadth first approach imposed by the Apriori principle.

5 EXPERIMENTS

In this section, we present the experiments we have conducted in order to evaluate performance of the algorithm MULTI-FOF-SP. We compared execution time and memory usage performance of MULTI-FOF-SP on multi-item sequence databases with the algorithms PrefixSpan[1] and LAPIN-LCI[2].

We generated several synthetic sequence databases using IBM Quest Data Generator (Agrawal et al., 1993). We downloaded the IBM Quest Data Generator executable in Illimine Software Package Version 1.1.0 from the web site [3]. In order to obtain

[1]We downloaded PrefixSpan executable in Illimine Software Package Version 1.1.0 from web site of Illimine Project http://illimine.cs.uiuc.edu/download/.

[2]We downloaded LAPIN-LCI executable from http://www.tkl.iis.u-tokyo.ac.jp/ yangzl/soft/LAPIN/index.htm

[3]http://illimine.cs.uiuc.edu/download/

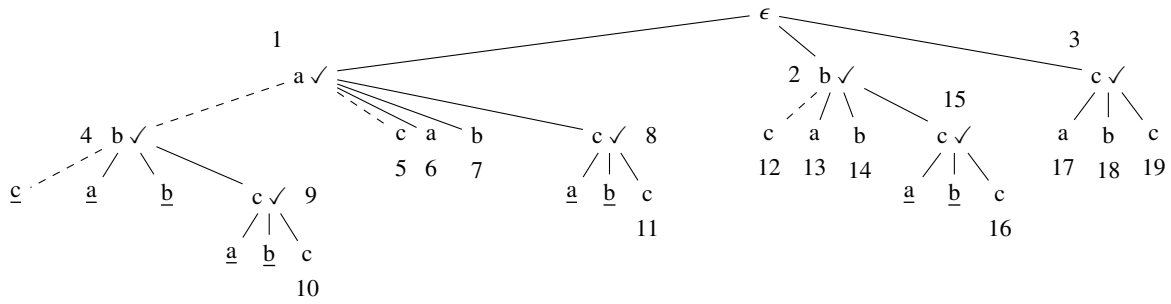


Figure 6: The Subspace of the Lexicographic Search Space Processed by the MULTI-FOF-SP Algorithm for Mining the Database in Table. 2

a comprehensive data set, we determined a set of values for each of the parameters C (Average length of sequences) ={*25*}, T (average size of itemsets) = {*3,7*}, D (number of sequences) = {*200K,800K*} and N (size of database alphabet) = {*10,500*}. We generated databases with combination of values from these sets.

Each sequence database in the experiment set is named in accordance with the parameter values. For instance C25T3S25I3N10D200k specifies a database generated with parameters C=25, T=3, S=25, I=3, N=10 and D=200K. The support values for the experiment sets are chosen such that comparative performance results can be obtained in a reasonable time.

We performed experiments on a personal computer with Intel(R) Core(TM) i7 860 @2.80Ghz 2.80 Ghz CPU, 8 GB installed memory and Windows 7 Professional 64-bit operating system. For each experiment case identified by the 4-tuple (Algorithm, Sequence Database, MinSupport), we report the execution time and memory consumption of algorithm on the sequence database. We repeated each test case run at least three times and reported the average value for all the algorithms. Due to space limitation, we present results of only a part of the conducted experiments.

As the first set of experiments, we present results on two different sequence databases of alphabet size N=500 in Figure 7 and Figure 8. In these experiments with large alphabet, PrefixSpan is faster then MULTI-FOF-SP in all of the cases. Although, LAPIN-LCI is faster than MULTI-FOF-SP in some cases, there are also cases in which LAPIN ends unexpectedly consuming more than 2 GBs of memory. MULTI-FOF-SP can prune the search space owing to sibling principle yet it may not be sufficient when the alphabet is large.

As the second set of experiments, we present results on three different sequence databases of alphabet size N=10 in Figure 9 and Figure 10.

MULTI-FOF-SP outperforms PrefixSpan in terms of execution time on all sequence databases with alphabet size N=10. In most of the cases, MULTI-

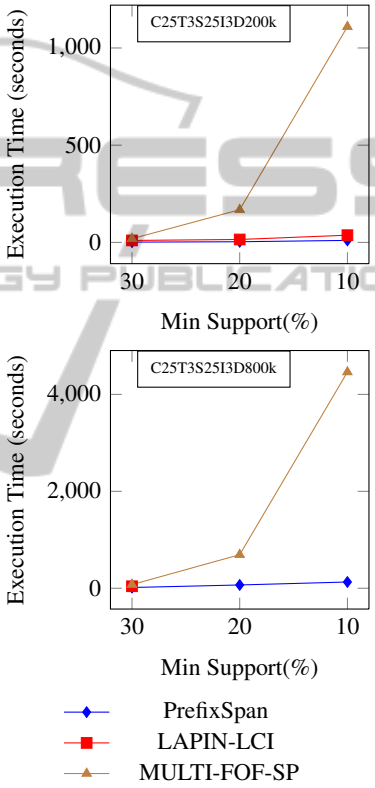


Figure 7: Execution Times of Algorithms on Databases with N=500.

FOF-SP is faster than LAPIN-LCI yet there are cases in which LAPIN-LCI is slightly faster. However, there are also cases LAPIN-LCI could not complete execution since it ran out of memory. One such case is given in Figure 10 for the database C25T7S25I7N10D800k when the minimum support value is decreased to 99.3.

MULTI-FOF-SP ranks the second on databases with N=10 in terms of memory consumption. The algorithm stores many FOFs in the memory in order to apply sibling principle. However, the memory consumption is moderate because this cost is compen-

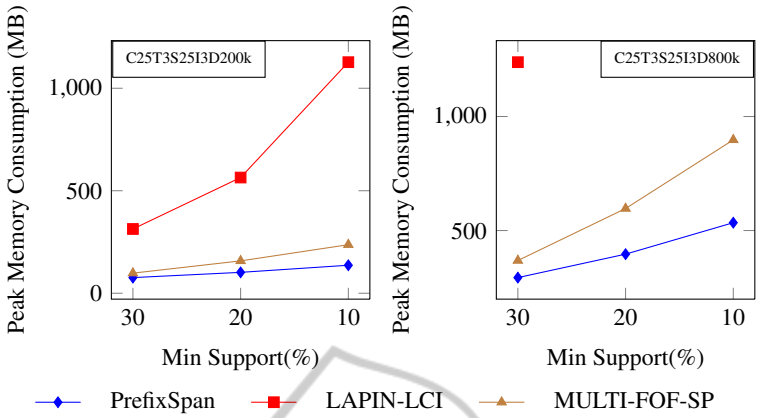


Figure 8: Memory Consumption of Algorithms on Databases with N=500.

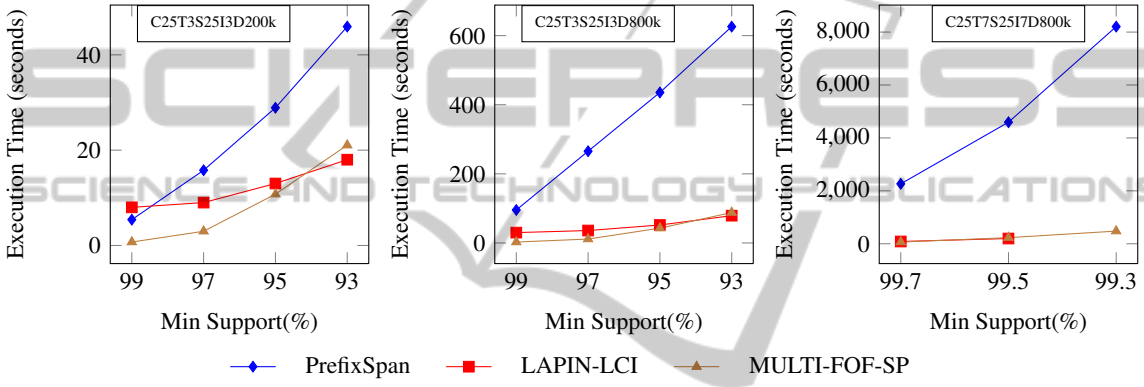


Figure 9: Execution Times of Algorithms on Databases with N=10.

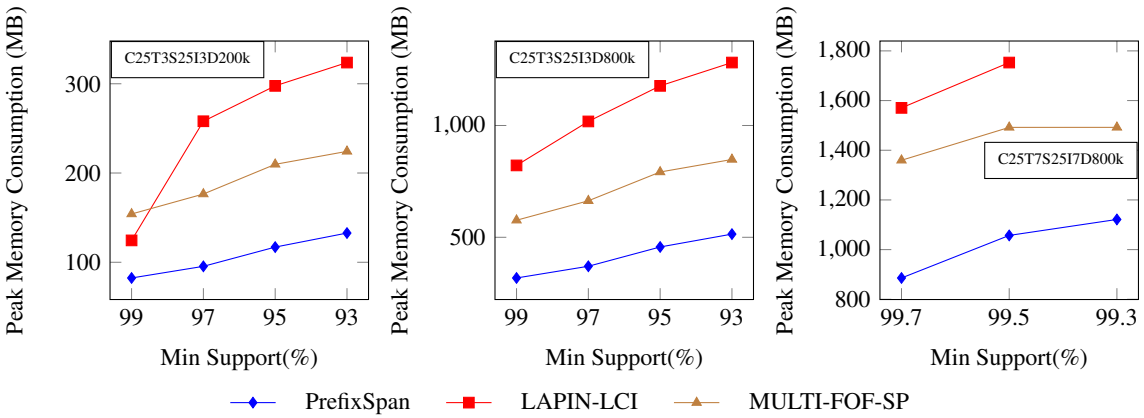


Figure 10: Memory Consumption Of Algorithms on Databases with N=10.

sated by the compression provided by the MULTI-WAP-Tree data structure.

MULTI-FOF-SP is faster on the databases with smaller alphabets owing to the compression provided by the MULTI-WAP-Tree. For instance, MULTI-WAP-Tree has at most 20 child nodes of the root regardless of the number of sequences in the database when the alphabet size is 10. Moreover, the width of the WAP-Tree is never larger than the number of sequences. However, it is crucial to note that it is more efficient to mine the projected database in horizontal representation than mining the tree structure. Scanning the tree requires tracking the edges between the nodes. Besides, WAP-Tree node occupies more

space than the space occupied by an item in horizontol representation. Consequently, if the FOF approach were applied on the MULTI-WAP-Tree without the sibling principle, both memory and time requirements of FOF approach would be higher in cases of large alphabets. A high degree of compression by the MULTI-WAP-Tree is required to outperform PrefixSpan and LAPIN in terms of both execution time and memory. We observed that this requirement cannot be met in case of large alphabets.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a new data structure MULTI-WAP-Tree and a new algorithm MULTI-FOF-SP for extracting multi-item sequence patterns. MULTI-WAP-Tree is the first tree structure for representing general sequence databases. MULTI-FOF-SP employs the early pruning idea Sibling Principle.

We have experimented on several test cases to compare MULTI-FOF-SP with previous multi-item sequence mining algorithms, PrefixSpan and LAPIN-LCI. Experiments revealed that MULTI-FOF-SP outperforms PrefixSpan and has a performance close to LAPIN-LCI in terms of execution time on dense multi-item databases with small alphabets. In addition, it has a better performance than LAPIN-LCI in terms of memory usage for these databases.

In this work, we devised a MULTI-WAP-Tree based algorithm that uses sibling principle and obtained good results. As a continuation of this line, other existing tree based algorithms can be investigated for multi-item sequence mining using the MULTI-WAP-Tree data structure.

REFERENCES

Agrawal, R., Imelinski, T., and Swami, A. (1993). Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD Conference on Management of Data*, pages 207–216. ACM.

Agrawal, R. and Srikant, R. (1995). Mining sequential patterns. In *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'95)*, pages 3–14. IEEE.

Ezeife, C. and Lu, Y. (2005). Mining web log sequential patterns with position coded pre-order linked wap-tree. *Data Mining and Knowledge Discovery*, 10(1):5–38.

Han, J., Pei, J., and Yan, X. (2005). Sequential pattern mining by pattern-growth: Principles and extensions*. In Chu, W. and Lin, T., editors, *Foundations and Advances in Data Mining*, volume 180 of *Studies in Fuzziness and Soft Computing*, pages 183–220. Springer Berlin Heidelberg.

Liu, L. and Liu, J. (2010). Mining web log sequential patterns with layer coded breadth-first linked wap-tree. In *International Conference of Information Science and Management Engineering (ISME'2010)*, volume 1, pages 28–31. IEEE.

Mabroukeh, N. and Ezeife, C. (2010). A taxonomy of sequential pattern mining algorithms. *ACM Computing Surveys (CSUR)*, 43(1):3.

Masseglia, F., Poncelet, P., and Cicchetti, R. (2000). An efficient algorithm for web usage mining. *Networking and Information Systems Journal*, 2(5/6):571–604.

Mooney, C. H. and Roddick, J. F. (2013). Sequential pattern mining – approaches and algorithms. *ACM Comput. Surv.*, 45(2):19:1–19:39.

Pei, J., Han, J., Mortazavi-Asl, B., and Zhu, H. (2000). Mining access patterns efficiently from web logs. *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pages 396–407.

Peterson, E. and Tang, P. (2008). Mining frequent sequential patterns with first-occurrence forests. In *Proceedings of the 46th Annual Southeast Regional Conference (ACMSE)*, pages 34–39. ACM.

Song, S., Hu, H., and Jin, S. (2005). Hvsm: A new sequential pattern mining algorithm using bitmap representation. In Li, X., Wang, S., and Dong, Z., editors, *Advanced Data Mining and Applications*, volume 3584 of *Lecture Notes in Computer Science*, pages 455–463. Springer Berlin Heidelberg.

Tang, P., Turkia, M., and Gallivan, K. (2006). Mining web access patterns with first-occurrence linked wap-trees. In *Proceedings of the 16th International Conference on Software Engineering and Data Engineering (SEDE'07)*, pages 247–252. Citeseer.