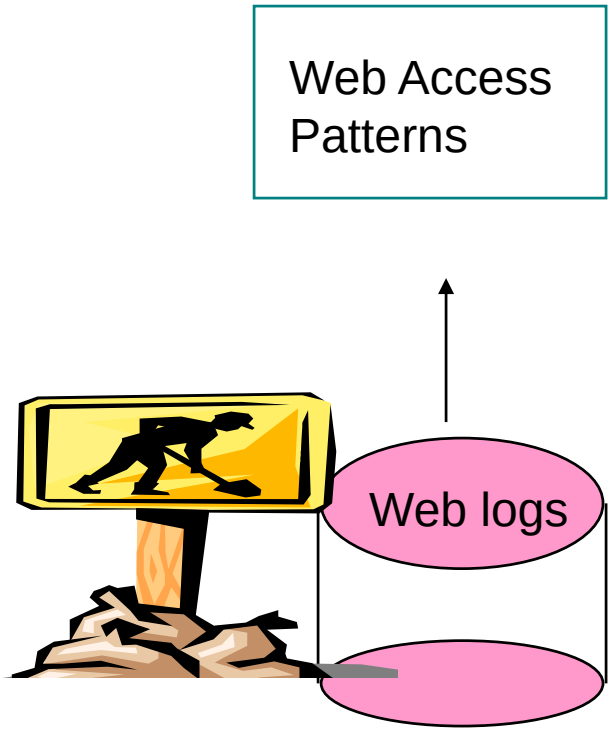
Mining Web Log Sequential Patterns with Position Coded Pre-Order Linked WAP-Tree



By

C.I. Ezeife
School of CS,
Univ. of Windsor,
Canada.
cezeife@uwindsor.ca

And

Yi Lu

OUTLINE

- INTRODUCTION/OVERVIEW
  - Sequential Pattern Mining Overview
  - Related Work
  - Problem Definition & Motivations
  - Contributions of Work
- Proposed PLWAP Algorithms Through An Example
- Performance Analysis
- CONCLUSIONS
  - Summary of this work's contributions
  - Recent Research leading up to this work
  - Current and Future Research

Introduction -
What is Sequential Pattern Mining?

- Data Mining allows data analysis for identifying regular and irregular patterns in large pools of data using some computer tools.
- Mining tools include Association Rules, Sequential Pattern, Decision tree classification, clustering, Neural networks and genetic algorithms.
- Traditional association rule mining is used to find frequent patterns of items(events) in non-sequenced dataset of items, **sequential pattern mining** is used to find frequent patterns of items (events) in sequenced dataset.

Introduction -
What is Sequential Pattern Mining?

TID	Sequence	Frequent Sequence
100	*abdac*	*abac*
200	*eaebcac*	*abcac*
300	*babfaec*	*babac*
400	*afbacfc*	*abacc*

Table 1: Example Sequential Database

- **Sequence*:*** *is a series of events(items). Repetition is allowed.* *abdac* is a sequence, two "*a*" occur in one sequence.
- **n-sequence***: A* sequence *with* n *events.* *abdac* is 5-sequence

Introduction -
What is Sequential Pattern Mining?

- **A Subsequence, S' of S exists when** (S'$\subseteq$S) and each event in S' equals one of the events in S, and the order of events in S' is same as in S. E.g., *bdc* is a subsequence of *abdac*.
- Given a sequence S = S_1S_2, any subsequence of S_1 can be the **prefix** sequence of S_2, while any subsequence of S_2 can be the **suffix** sequence of S_1. In sequence *abdac*, *ab* is a prefix of *dac*, while *dac* is a suffix of *ab*.
- **Support(**S):the number of sequences (records), which contain the subsequence S, divided by the total number of sequences in the database. The support of sequence *fa* is 50%, since it occurs in transaction 3 and 4

Introduction -
What is Sequential Pattern Mining?

- In sequential pattern mining, the higher the support of a sequence, the stronger the rules generated from this sequence. Thus, sequences with high support are considered useful. The ***minimum support*** and ***minimum confidence*** are used to identify these subsequences.
- All [sub]sequences with support higher than a specified minimum support are called ***frequent [sub]sequences***.
- Sequential pattern mining is used to find sequential pattern in web log database by computing all frequent sequences.

Introduction - Related Work

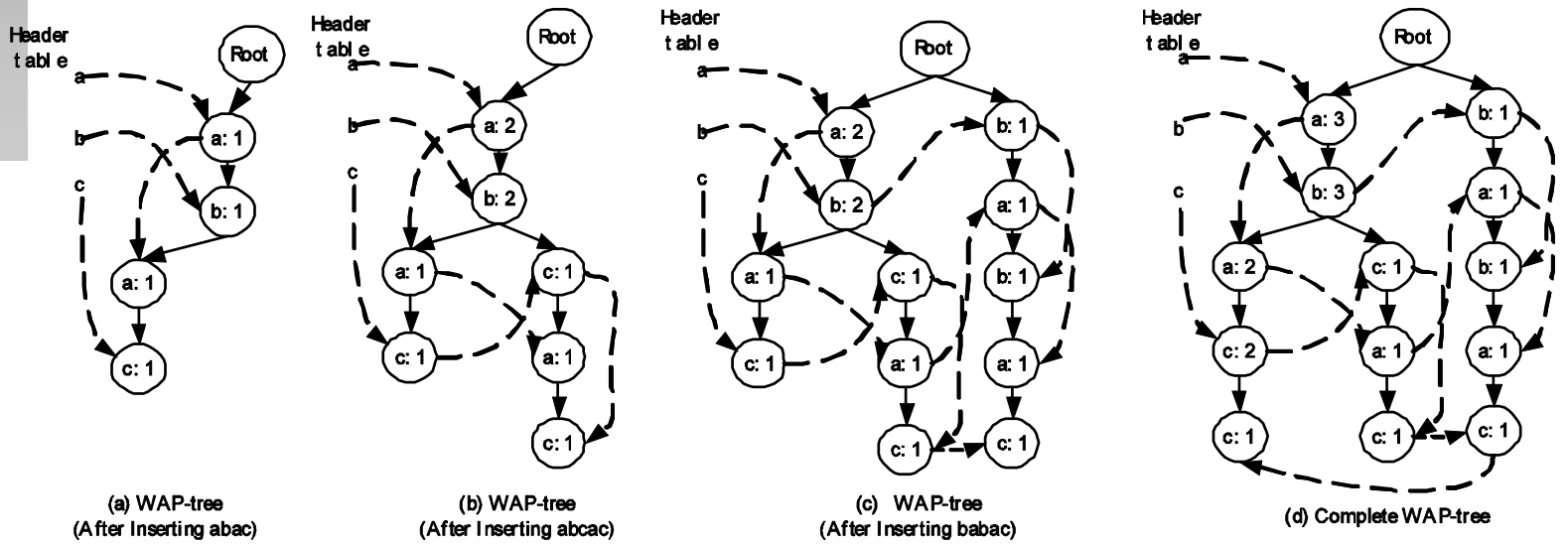
- GSP algorithm [SA96]: Uses (k-1)-sequence to generate the candidate k-sequence by joining itself. Then, scans db to compute (k)-sequence and the process goes on.
- Traversal Graph [NM00, NM01]: An Apriori-like method which adds the consideration of the web site's structure to reduce the candidate sequences for testing.
- G Sequence [Spi99]: A non-Apriori-like method. Introduces use of template and wild cards to narrow the goal of mining process
- WAP-tree [PHM+00]. : A compact prefix tree stores all database transaction sequences and mines the tree for patterns.

Introduction - Related Work

- Table 1: Example For Sequence Pattern Mining with minimum support of 75%

- From above, C1 ={a:4, b:4, c:4, d:1, e:2, f:2}
- F1 = ={a:4, b:4, c:4}
- Only frequent sequences (column 3 of WASD table 1) are inserted in WAP tree and maintained in the Header linkage nodes.
- Construction of the WAP tree would proceed as follows:

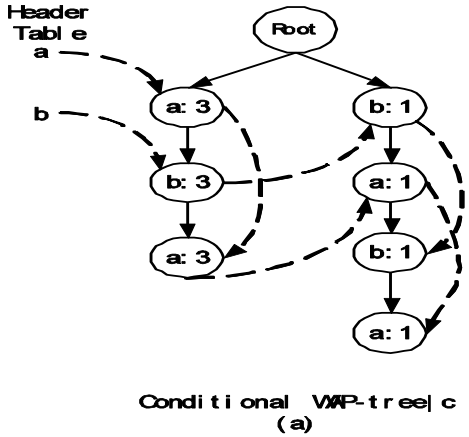
Introduction - Related Work



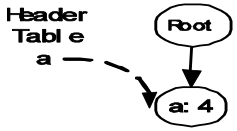
Introduction - Related Work

- Next, the WAP-tree algorithm mines the WAP tree recursively by first finding the frequent suffix pattern of length 1 (e.g., c, b, a), then, recursively, finding frequent prefix patterns of these of length 2, and so on until all patterns are found.
- It starts with the lowest header node “c” and constructs the conditional pattern base of “c” from the tree as:
  c: aba(2); ab(1); abca(1); ab(-1); baba(1); abac(1); aba(-1).
- Since minsupport is 3, only frequent events in the above (a, b) are used in constructing the intermediate WAP-tree|c shown

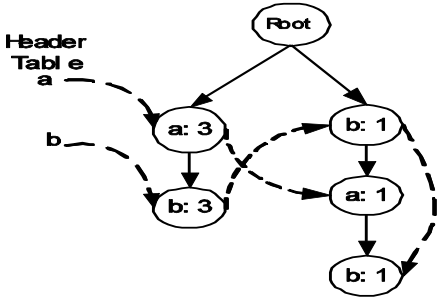
Introduction - Related Work



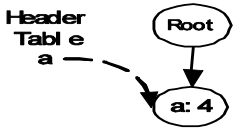
Conditional WAP-tree|c
(a)



Conditional WAP-tree|bc
(b)



Conditional WAP-tree|ac
(c)



Conditional WAP-tree|bac
(d)

Conditional WAP-tree|aac
(e)

Introduction – Problem Definition

- Given web access sequence database WASD and a minimum support threshold λ, the problem of web usage mining is to find all sequences which have support greater than λ.
- Motivation for Problem

  Existing methods like GSP are Apriori-like and need to scan huge DB several times. The WAP tree method scans DB only twice improving on Apriori-like methods, but still has to recursively reconstruct intermediate WAP trees during mining. There is need to eliminate the multi reconstruction of intermediate WAP trees for better response time.

Proposed PLWAP Algorithms(Formal)

- Main Contributions of This Work(1)An algorithm for assigning position codes to WAP tree nodes is introduced, frequent header nodes are linked pre-order fashion instead of “as inserted”, (2)an algm that mines patterns with PLWAP tree without reconstructing intermediate WAP trees using these codes and linkage is defined.
- Main steps in the Proposed PLWAP algorithm are:

  It builds the WAP tree with pre-order linkage and position codes.

  It mines the WAP tree using the following idea:

  It finds prefix sequence first, then recursively extends at the end of sequence. e.g. frequent sequence abcd can be obtained as: a->ab->abc->abcd

Proposed PLWAP Algorithms- Example

TID	Sequence	Frequent Sequence
100	*abdac*	*abac*
200	*eaebcac*	*abcac*
300	*babfaec*	*babac*
400	*afbacfc*	*abacc*

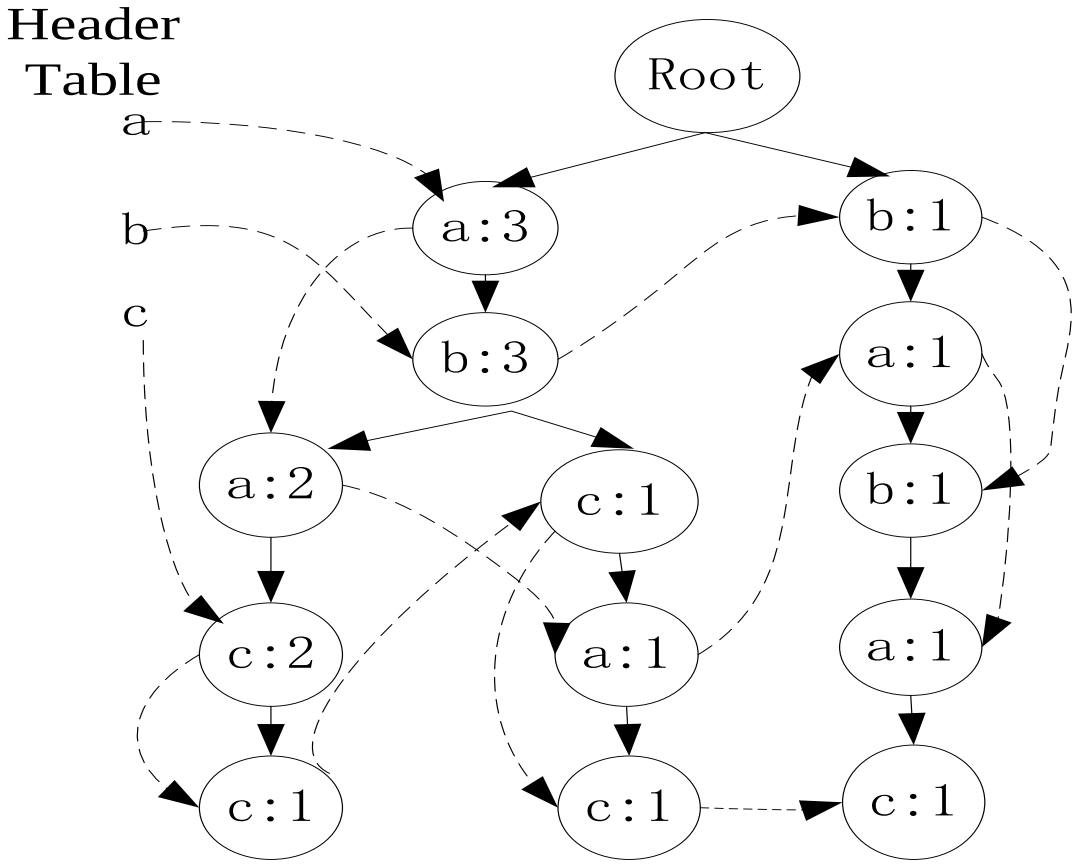
Table 2: Example For Sequence Pattern Mining with minimum support of 75%

From above, C1 ={a:4, b:4, c:4, d:1, e:2, f:4}

F1 = ={a:4, b:4, c:4}

Only frequent sequences are inserted in WAP tree and maintained in the Header linkage nodes.

Pre-Order Linked WAP-tree of Table 2



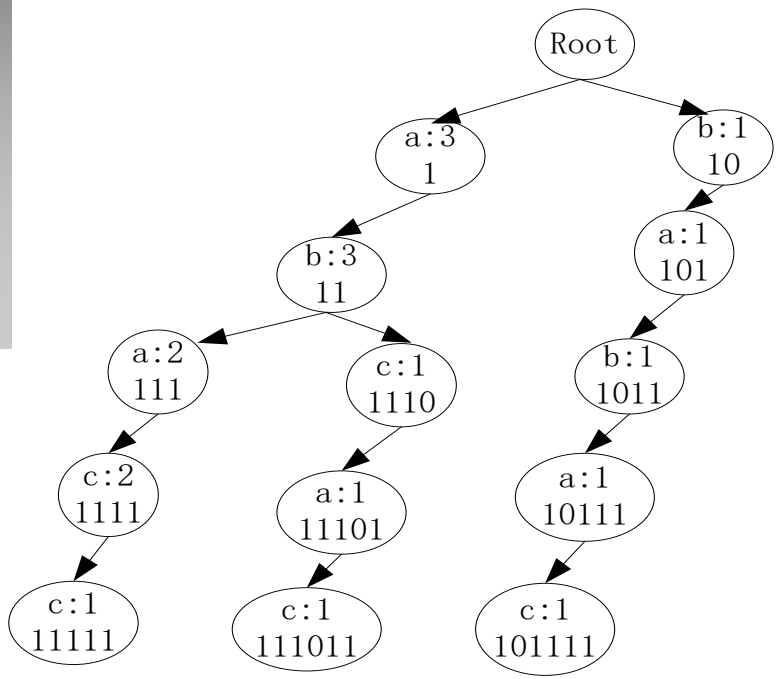
Pre-Order:

Visit Root

Visit Left Subtree

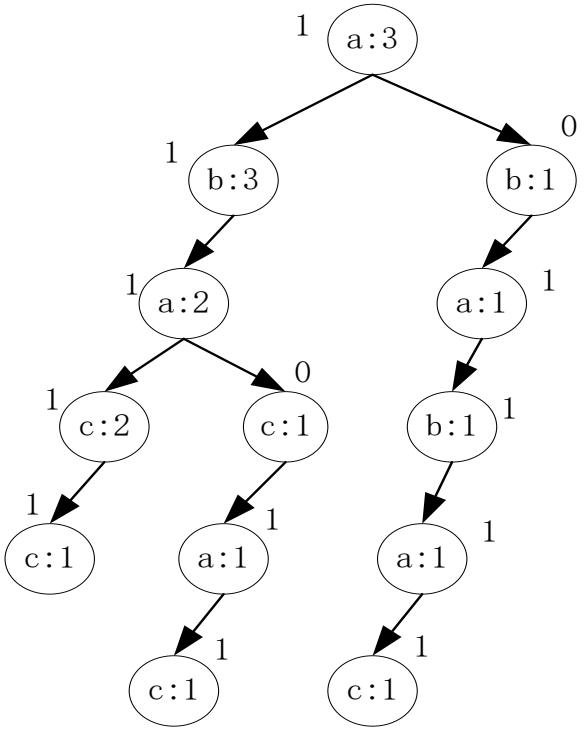
Visit Right Subtree

Binary Position Code for PLWAP-tree



(a)

Original tree



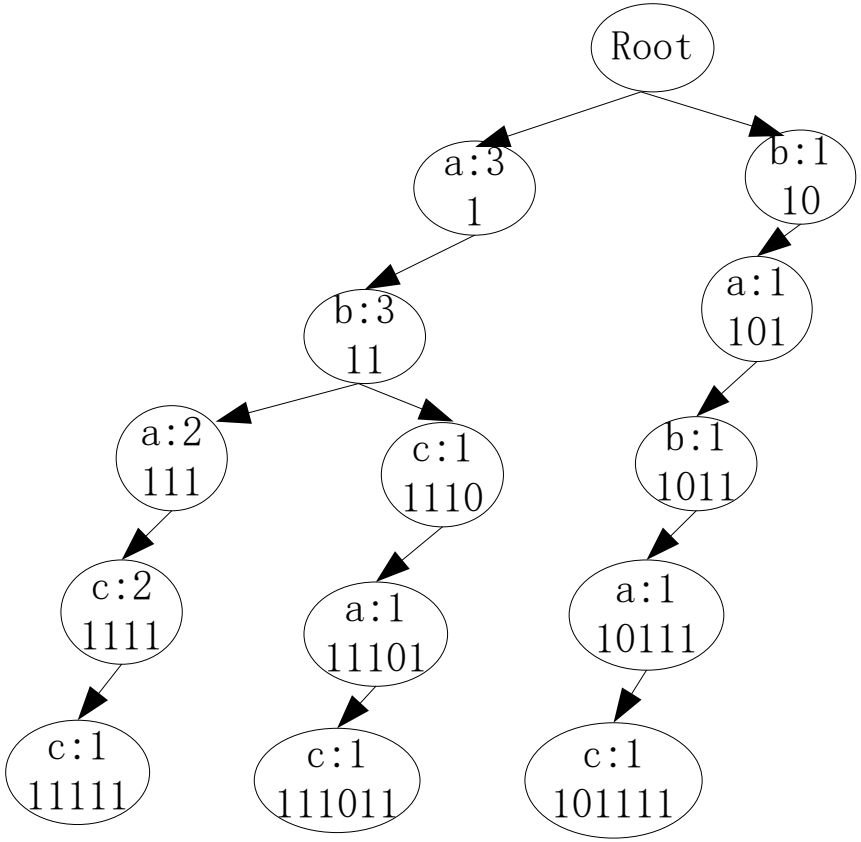
(b)

Binary tree of original tree

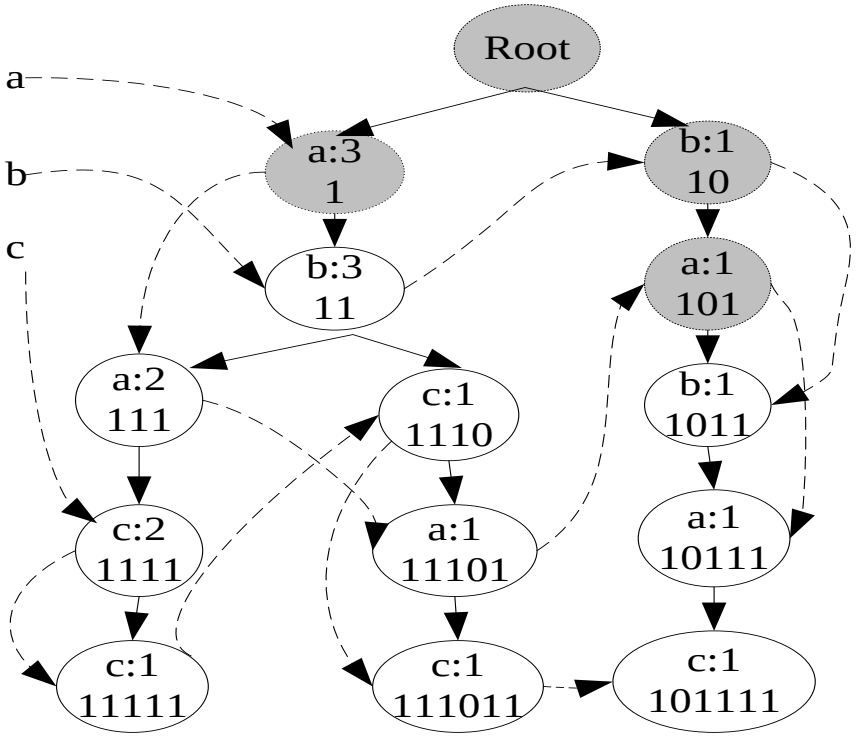
Binary Position Code for PLWAP-tree

A node a is ancestor of another node b if and only if the *position code* of a with “1” appended to its end, equals first ***X*** number of bits in position code of b.

***X*= |a .PositionCode| +1**



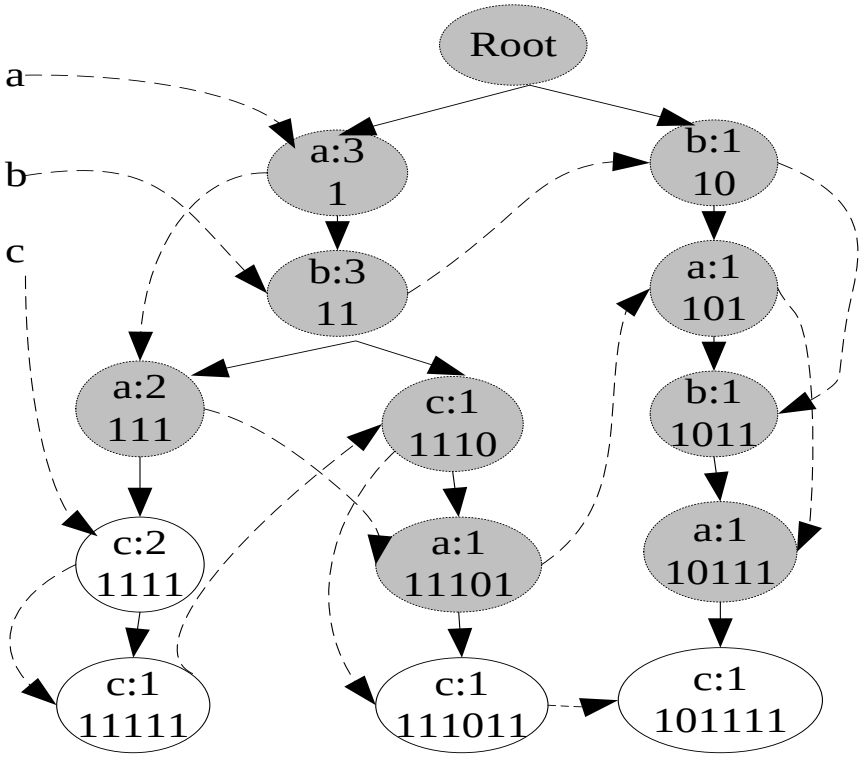
Mining of PLWAP-tree



a| suffix tree
{b:11, b:1011}

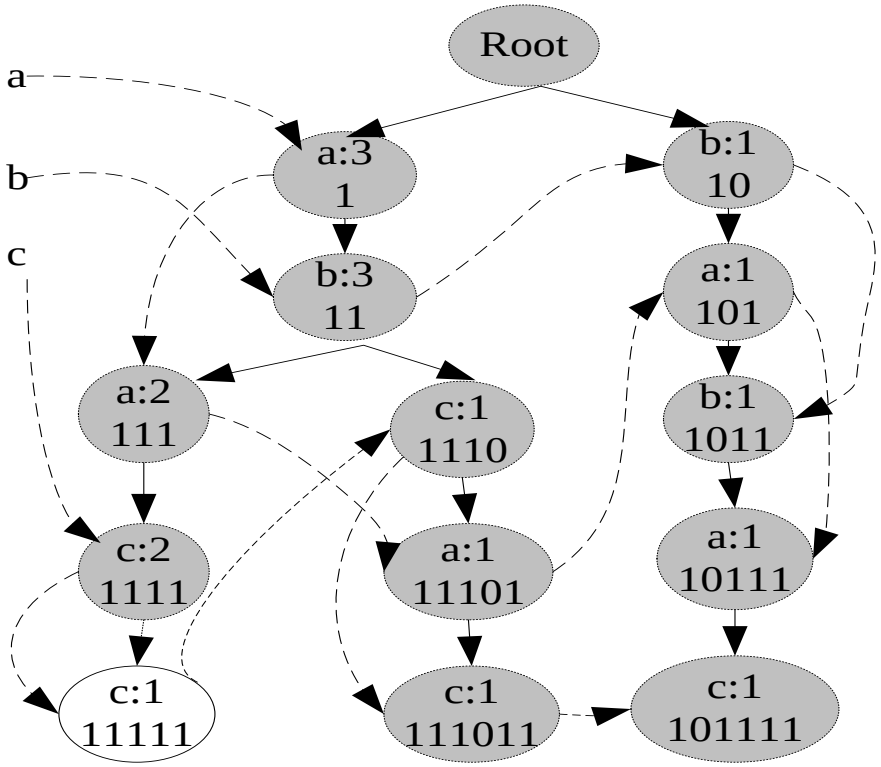
For event in the linkage header, find the first occurrence in every suffix trees of conditional PLWAP tree being mined. Sum the count of nodes in different suffix trees.

Mining of PLWAP-tree



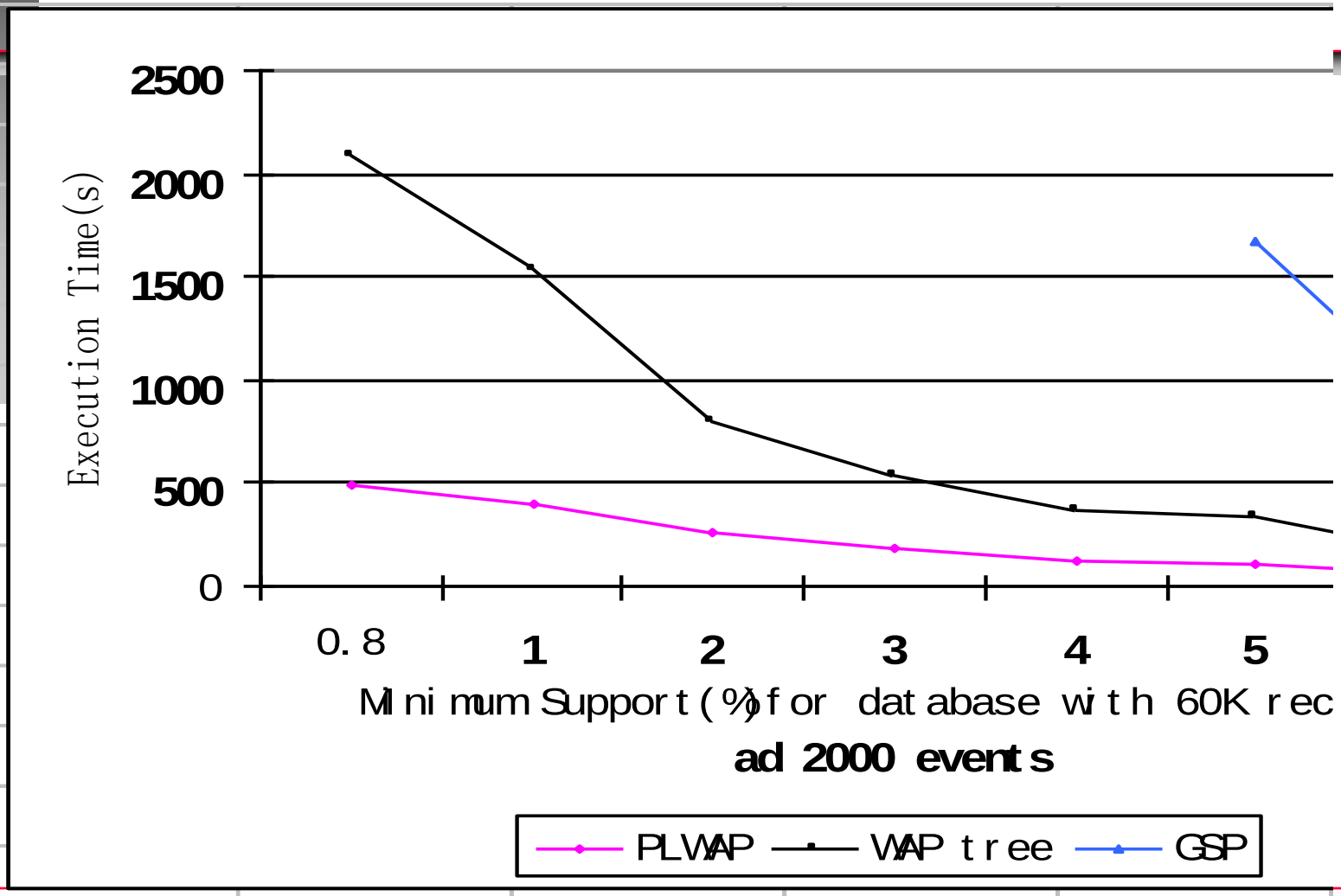
aa| suffix tree
{c:1111, c:111011, c:101111}

Mining of PLWAP-tree

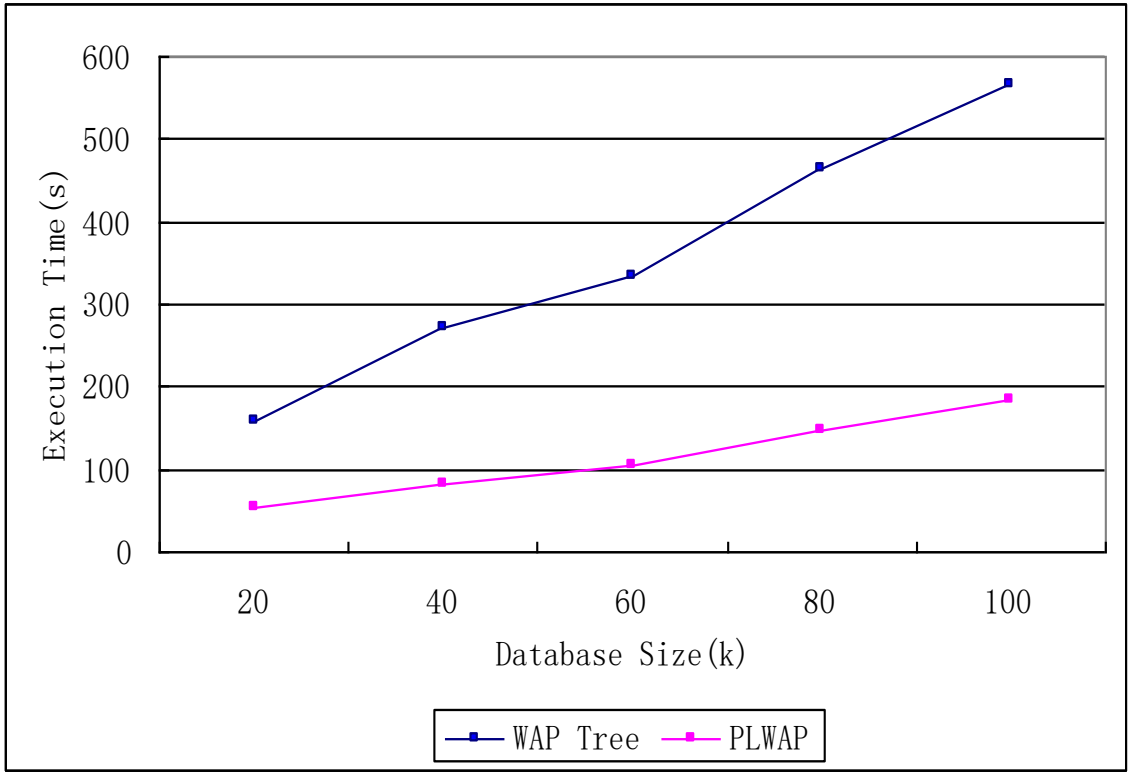


aac| suffix tree
{c:11111}

Performance Analysis



Performance Analysis



Conclusions –

- Two algorithms have been proposed in this paper- one algm for binary code assignment is used by the main algm (PLWAP)
- The main algorithm, PLWAP mines web log sequential data for frequent patterns using preorder linked WAP tree without reconstructing numerous intermediate WAP trees.
- Experiments have shown that the proposed PLWAP algorithm performs better than existing sequential pattern mining algorithms and in particular with the frequent sequence becoming longer or original database becoming larger.
- Future work may consider applying PLWAP mining techniques to distributed mining as well as to incremental mining of web logs sequential patterns.