Mining Maximal Sequential Patterns without Candidate Maintenance

Artificial Intelligence
Seminar
10 March 2016

Syméon Malengreau
Ayad Aldayeh

Table of contents

Authors

Vincent S. Tseng

Philippe Fournier-Viger

Cheng-Wei Wu

Authors : Vincent S. Tseng

Ph.D. in Computer Science

Professor, Dept. Computer Science, National Chiao Tung University, Taiwan

H-index : 32

Citations : 3308

Authors : Philippe Fournier-Viger

Ph.D. in Computer Science

Associate Professor, Harbin Institute of Technology, Shenzhen Graduate School

He created the SPMF library

H-index : 17

Citations : 712

Authors : Cheng-Wei Wu

H-index : 12

Citations : 518

About the article

We can trust the author, but

- We found errors in some examples,
- Some part of the articles were mathematicaly not clear enough, and some definition were simply wrong,
- In the end, the article was a complete mess, we cannot understand who accepted to publish it !

After some search else where we made it possible to understand the content of the articles. But based only on it, it wouldn't have be possible.

It was published in a book : *Advanced Data Mining and Applications*

This particular article has (when the slides where made) 10 citations

Concepts

- **Sequence Database**
- **Sequential pattern**
- **Closed sequential pattern**
- **Maximal sequential pattern**

Sequence Database

A sequence database consist of :

A set of items

{1, 2, 3, 4, …, N}

Itemset (set of item, distinct and unordered)

{1, 2, 3, 5} or {4, 5} or {3, 7} or ...

Sequence (set of itemsets)

<{1,2}, {3}, {5}> or <{4}, {6}> or ...

The sequence database is a set of sequences

What do theses concepts represents ?

Sequence Database : Illustration

Let's take as an example a book

Set of item → The words

{He, nice, the, is, a, guy, sun, shine, …}

Itemset → A sentence (where words are distinct and unordered)

{He, a, nice, guy, is}

{The, sun, shine, in, the, sky}

Sequence → A chapter of the book

Sequence Database → The book

Sequential pattern

Synonyms are *sub-sequence* **or** *frequent* ***sequence***

It is a sequence of item that appears a certain number of time, that number is the *minimum support threshold* (or *minsup*)

Sequence database

<{1,2},{3},{4},{6}>

<{2},{5},{6}>

<{1,3},{5},{6}>

With minsup = 2, some examples of sequential pattern

{5},{6}

{1}

{3},{6}

…

Closed sequential pattern

A closed sequential pattern is a sequential pattern not included in another closed pattern having the same frequency.

<{1},{1 2 3},{1 3},{4},{3 6}>

<{1 4},{3},{2 3},{1 5}>

<{5 6},{1 2},{4 6},{3},{2}>

<{5},{7},{1 6},{3},{2},{3}>

With support 2 (or 2/4 entry → 50 %), here are some closed sequential pattern

{1}, {3}	100 % (4/4)
{1}, {3}, {2}	75 % (3/4)
{5}, {1}, {3}, {2}	50% (2/4)
{5}	75 % (3/4)

And this one is NOT

{1} 100 % (4/4)

Maximal sequential pattern

The same as the closed sequential pattern, but if one sequence is in another one, it is not maximal.

Interesting property :

You can derive every closed sequential patterns from the maximal sequential patterns

Question 1 : Closed and Maximal pattern

Considering the database

<{a,b},{c},{f,g},{g},{e}>

<{a,d},{c},{b},{a,b,e,f}>

<{a},{b},{f,g},{e}>

<{b},{f,g}>

1. Which one of these is not a closed sequential pattern ? Why ?

   → <{b},{f}>

   → <{b}>

   → <{a,b}>

   → <{a},{b},{e}>

2. Which one of these is a maximal sequential pattern ? Why ?

   → <{a},{e}>

   → <{b}, {b}>

   → <{b}, {f}, {e}>

   → <{a}, {f}>

MaxSP Algorithm

Find the maximal sequential pattern

It is build uppon the PrefixSpan Algorithm

Why the need for a new algorithm ?

→ Less memory usage

→ Faster to find sequential pattern

PrefixSpan : Start

First let's explain the PrefixSpan Algorithm

It's the most efficient pattern mining algorithm

We start with a sequence database

<{1},{3},{4},{6},{2},{3}>

<{4},{3},{2},{1}>

<{5},{1},{4},{3},{2}>

<{5},{7},{1},{3},{2},{3}>

PrefixSpan : Pattern-growth

It works by pattern-growth, which does not generate any candidates (saving memory)

1. Scan : *Calculate support for each item and existing itemset*
2. Output : *Output item that have enough support*
3. Projection : *Recursively project the database with every item that have enough support*

PrefixSpan : Scan

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>
<{4},{3},{2},{1}>
<{5},{1},{4},{3},{2}>
<{5},{7},{1},{3},{2},{3}>

Item	Support
1	
2	
3	
4	
5	
6	
7	

PrefixSpan : Scan

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>
<{4},{3},{2},{1}>
<{5},{1},{4},{3},{2}>
<{5},{7},{1},{3},{2},{3}>

Item	Support
1	100% (4)
2	
3	
4	
5	
6	
7	

PrefixSpan : Scan

MinSup 75 % (3)

<{ },{3},{4},{6},{2},{3}>
<{4},{3},{2},{ }>
<{5},{ },{4},{3},{2}>
<{5},{7},{ },{3},{2},{3}>

Item	Support
1	100% (4)
2	100% (4)
3	
4	
5	
6	
7	

PrefixSpan : Scan

MinSup 75 % (3)

<{ },{3},{4},{6},{ },{3}>
<{4},{3},{ },{ }>
<{5},{ },{4},{3},{ }>
<{5},{7},{ },{3},{ },{3}>

Item	Support
1	100% (4)
2	100% (4)
3	100% (4)
4	
5	
6	
7	

PrefixSpan : Scan

MinSup 75 % (3)

<{ },{ },{4},{6},{ },{ }>
<{4},{ },{ },{ }>
<{5},{ },{4},{ },{ }>
<{5},{7},{ },{ },{ },{ }>

Item	Support
1	100% (4)
2	100% (4)
3	100% (4)
4	75% (3)
5	
6	
7	

PrefixSpan : Scan

MinSup 75 % (3)

<{ },{ },{ },{6},{ },{ }>
<{ },{ },{ },{ }>
<{5},{ },{ },{ },{ }>
<{5},{7},{ },{ },{ },{ }>

Item	Support
1	100% (4)
2	100% (4)
3	100% (4)
4	75% (3)
5	50% (2)
6	
7	

PrefixSpan : Scan

MinSup 75 % (3)

<{ },{ },{ },{6},{ },{ }>
<{ },{ },{ },{ }>
<{ },{ },{ },{ },{ }>
<{ },{7},{ },{ },{ },{ }>

Item	Support
1	100% (4)
2	100% (4)
3	100% (4)
4	75% (3)
5	50% (2)
6	25% (1)
7	

PrefixSpan : Scan

MinSup 75 % (3)

<{ },{ },{ },{ },{ },{ }>
<{ },{ },{ },{ }>
<{ },{ },{ },{ },{ }>
<{ },{7},{ },{ },{ },{ }>

Item	Support
1	100% (4)
2	100% (4)
3	100% (4)
4	75% (3)
5	50% (2)
6	25% (1)
7	25% (1)

PrefixSpan : Scan

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>
<{4},{3},{2},{1}>
<{5},{1},{4},{3},{2}>
<{5},{7},{1},{3},{2},{3}>

Item	Support
1	100% (4)
2	100% (4)
3	100% (4)
4	75% (3)
5	50% (2)
6	25% (1)
7	25% (1)

PrefixSpan : Scan

We take each item with the support ≥ minsup, and output it as a sequence with one item.

Here the output :

<{1}>

<{2}>

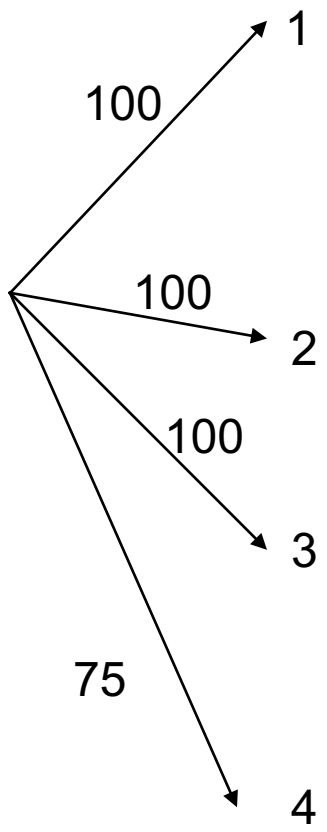
<{3}>

<{4}>

PrefixSpan : Scan

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>
<{4},{3},{2},{1}>
<{5},{1},{4},{3},{2}>
<{5},{7},{1},{3},{2},{3}>



New Concept : Projection

We need to define a new concept → *Projection*

If we project a sequence <{1},{2},{3}> by a prefix <{1}>, we take the part of the sequence that follow the prefix. Here <{2},{3}>

Some examples :

<{1},{2},{1},{3}> by <{1}> → <{2},{1},{3}>

<{3},{4},{5}> by <{3},{4}> → <{5}>

<{1}, {3, 4}, {5}, {6}> by <{3}> → <{5},{6}>

<{2}, {3}, {4}, {5}, {6}> by <{3}, {5}> → <{6}>

→ Projecting a database, means to project every sequence

PrefixSpan : Projection

First we will output the result we found with enough support.

Then we will recursively project the database with every of those items.

Lets take the result we have found so far to make it clearer.

We keep *minsup* of 75 % (3)

PrefixSpan : Projection

1. Scan Database

<{1},{3},{4},{6},{2},{3}>
<{4},{3},{2},{1}>
<{5},{1},{4},{3},{2}>
<{5},{7},{1},{3},{2},{3}>

Item	Support
1	100% (4)
2	100% (4)
3	100% (4)
4	75% (3)
5	50% (2)
6	25% (1)
7	25% (1)

PrefixSpan : Projection

2. Output first item

<{1}> → Support : 100 % (4)

3. Project first item (1)

<{1},{3},{4},{6},{2},{3}>

<{4},{3},{2},{1}>

<{5},{1},{4},{3},{2}>

<{5},{7},{1},{3},{2},{3}>

Item	Support
1	100% (4)
2	100% (4)
3	100% (4)
4	75% (3)
5	50% (2)
6	25% (1)
7	25% (1)

PrefixSpan : Projection

1. Scan Again

<{3},{4},{6},{2},{3}>

< >

<{4},{3},{2}>

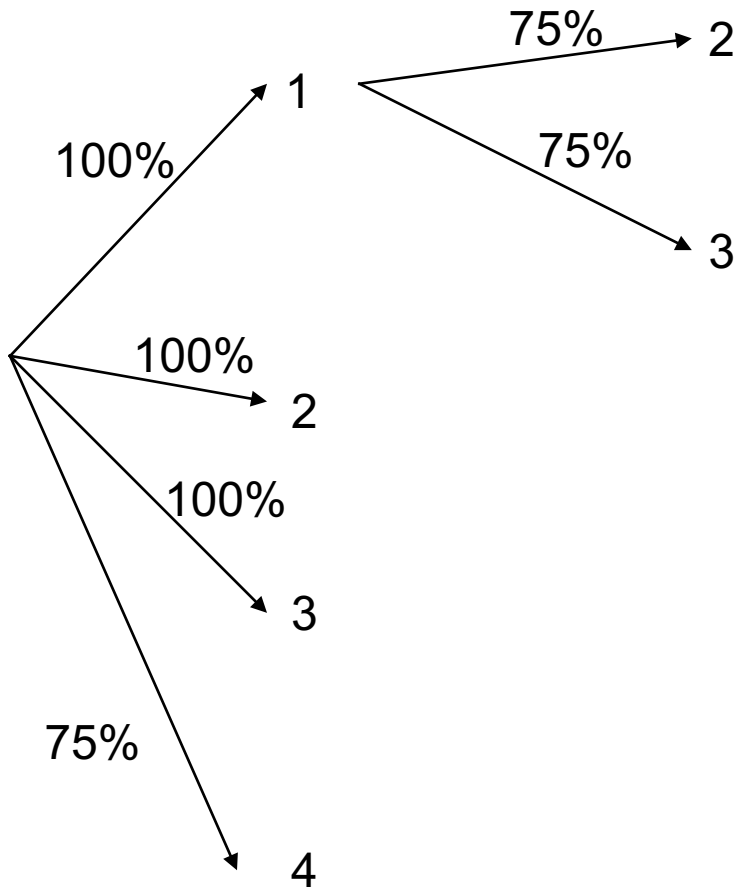
<{3},{2},{3}>

Item	Support
1	0% (0)
2	75% (3)
3	75% (3)
4	50% (2)
5	0% (0)
6	25% (1)
7	0% (0)

PrefixSpan : Projection

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>
<{4},{3},{2},{1}>
<{5},{1},{4},{3},{2}>
<{5},{7},{1},{3},{2},{3}>



PrefixSpan : Projection

2. Output the sequence

<{1},{2}> → Support : 75 % (3)

3. Project first item (2)

<{3},{4},{6},{2},{3}>

< >

<{4},{3},{2}>

<{3},{2},{3}>

Item	Support
1	0% (0)
2	75% (3)
3	75% (3)
4	50% (2)
5	0% (0)
6	25% (1)
7	0% (0)

PrefixSpan : Projection

1. Scan again

<{3}>

< >

<>

<{3}>

Item	Support
1	0% (0)
2	0% (0)
3	50% (2)
4	0% (0)
5	0 % (0)
6	0% (0)
7	0 % (0)

PrefixSpan : Projection

Operation are over

We continue with other items

Item	Support
1	0% (0)
2	0% (0)
3	50% (2)
4	0% (0)
5	0 % (0)
6	0% (0)
7	0 % (0)

PrefixSpan : Projection

2. Output the sequence

<{1},{3}> → Support : 75 % (3)

3. Project second item (3)

<{3},{4},{6},{2},{3}>

< >

<{4},{3},{2}>

<{3},{2},{3}>

Item	Support
1	0% (0)
2	75% (3)
3	75% (3)
4	50% (2)
5	0% (0)
6	25% (1)
7	0% (0)

PrefixSpan : Projection

1. Scan again

<{4},{6},{2},{3}>

< >

<{2}>

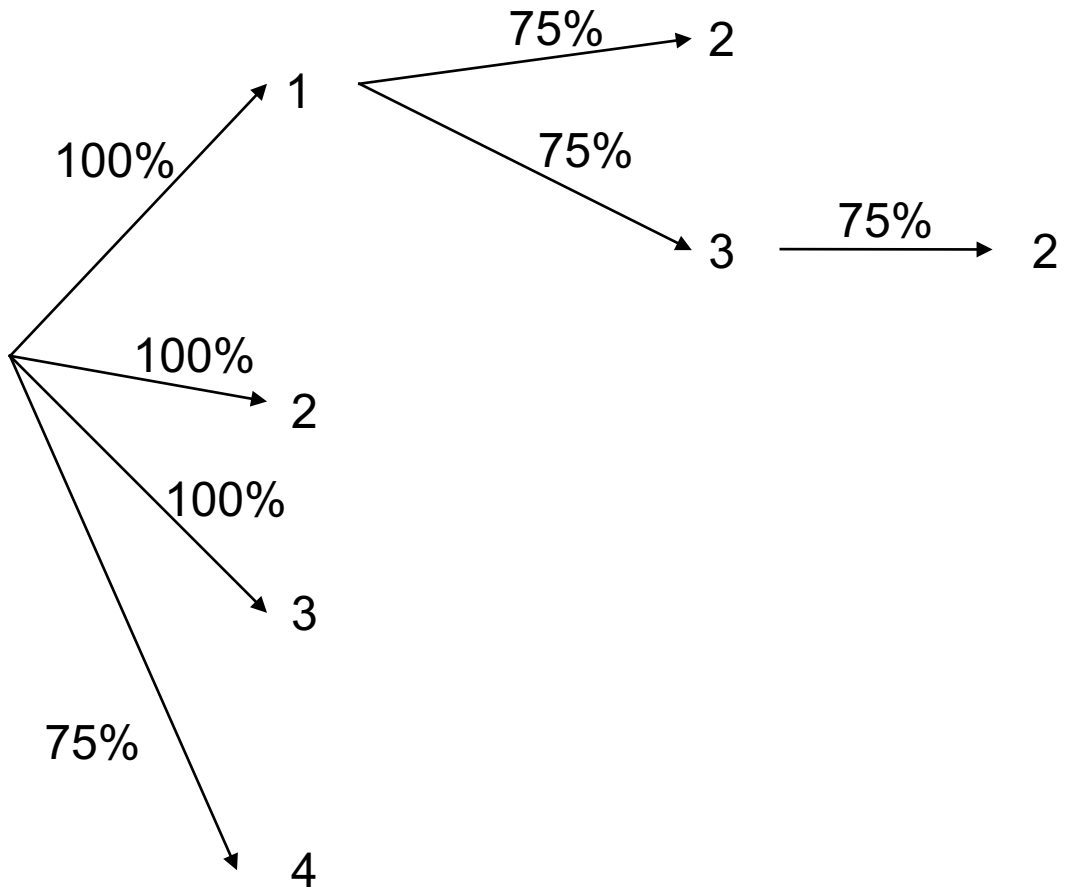
<{2},{3}>

Item	Support
1	0% (0)
2	75% (3)
3	50% (2)
4	25% (1)
5	0% (0)
6	25% (1)
7	0% (0)

PrefixSpan : Projection

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>
<{4},{3},{2},{1}>
<{5},{1},{4},{3},{2}>
<{5},{7},{1},{3},{2},{3}>



PrefixSpan : Projection

2. Output the sequence

<{1}, {3}, {2}> → Support : 75 % (3)

3. Project first item (2)

<{4},{6},{2},{3}>

< >

<{2}>

<{2},{3}>

Item	Support
1	0% (0)
2	75% (3)
3	50% (2)
4	25% (1)
5	0% (0)
6	25% (1)
7	0% (0)

PrefixSpan : Projection

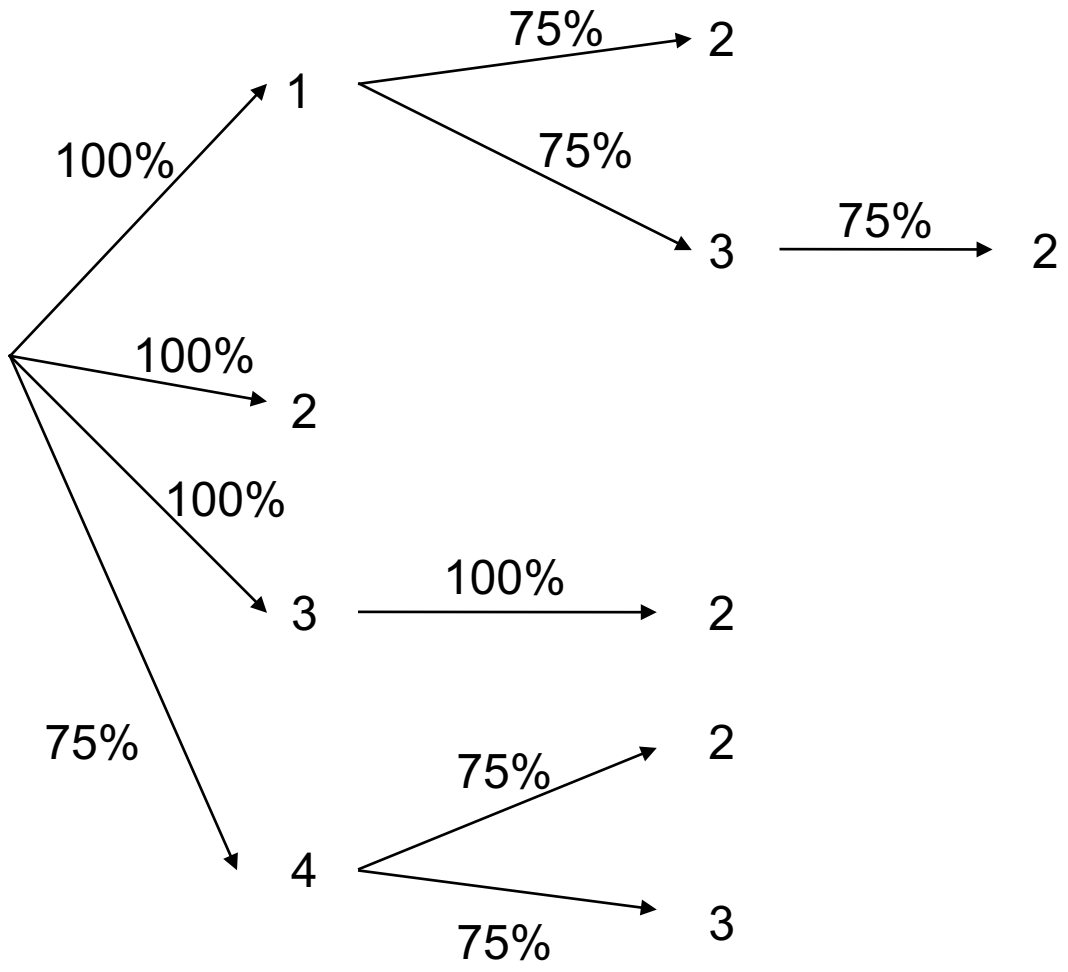
Doing all recursive projection would take time

We will skip to the final result, as the process repeats itself

PrefixSpan : Projection

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>
<{4},{3},{2},{1}>
<{5},{1},{4},{3},{2}>
<{5},{7},{1},{3},{2},{3}>



PrefixSpan : Projection

At the end of the execution we will get the following result.

Pattern	Support (≥ 75%)	Pattern	Support (≥ 75%)
<{1}>	100 %	<{3}>	100 %
<{1},{2}>	75 %	<{3},{2}>	100 %
<{1},{3}>	75%	<{4}>	75 %
<{1},{3},{2}>	75 %	<{4},{2}>	75 %
<{2}>	100 %	<{4},{3}>	75 %

Question 2 : Projection

Considering the database

<{a,b},{c},{f,g},{g},{e}>

<{a,d},{c},{b},{a,b,e,f}>

<{a},{b},{f,g},{e}>

<{b},{f,g}>

1. What is the result of the projection of <{b},{f}> on the database ?
2. In previous sequence, which are not closed and which are maximal ?

Pattern	Support (≥ 75%)	Pattern	Support (≥ 75%)
<{1}>	100 %	<{3}>	100 %
<{1},{2}>	75 %	<{3},{2}>	100 %
<{1},{3}>	75%	<{4}>	75 %
<{1},{3},{2}>	75 %	<{4},{2}>	75 %
<{2}>	100 %	<{4},{3}>	75 %

PrefixSpan : Projection

Here are the closed.

Pattern	Support (≥ 75%)	Pattern	Support (≥ 75%)
<{1}>	100 %	<{3}>	100 %
<{1},{2}>	75 %	<{3},{2}>	100 %
<{1},{3}>	75%	<{4}>	75 %
<{1},{3},{2}>	75 %	<{4},{2}>	75 %
<{2}>	100 %	<{4},{3}>	75 %

PrefixSpan : Projection

Here are the closed and maximal.

Pattern	Support (≥ 75%)	Pattern	Support (≥ 75%)
<{1}>	100 %	<{3}>	100 %
<{1},{2}>	75 %	<{3},{2}>	100 %
<{1},{3}>	75%	<{4}>	75 %
<{1},{3},{2}>	75 %	<{4},{2}>	75 %
<{2}>	100 %	<{4},{3}>	75 %

MaxSp : Basic Idea

MaxSP extends the PrefixSpan

A naïve approach would be to keep all sequence in memory and to check every time a new sequence arrives if it is maximal.

That is CloSpan

→ Inefficient

→ Memory consuming

MaxSp : Basic Idea

The question is

How to know if a pattern P is maximal, without maintening pattern in memory ?

The solution : Can P be extended by appending items ?

→ YES ? It isn't a maximal sequential pattern

Two check :

1. Maximal backward extension check
2. Maximal forward extension check

MaxSp : Maximal forward extension check

With the Maximal forward extension, we search if we can extends a pattern with upcoming items

→ Concretly it is what PrefixSpan already do

MaxSp : Maximal backward extension check

With maximal backward extension we check if we can extends the pattern with item that we might have passed.

If so we stop looking further because we will find the pattern in another branch.

Let's look at an example.

MaxSp : Maximal backward extension check

To find the maximal backward extension of a prefix P in a sequence S, there is a few steps

1. Find the last in last appearance of all items
2. Find the Maximum period
3. If an item support is ≥ than the minsup, there is a backward extension

Lets demonstrate, step by step

MaxSp : Maximal backward extension check

1. Find the last in last appearance of all item.

→ Just find the prefix in reverse order (starting from the end of the Sequence)

P = ABC

S = ABDBCAB

Current

Found Item

Discarded

MaxSp : Maximal backward extension check

1. Find the last in last appearance of all item.

→ Just find the prefix in reverse order (starting from the end of the Sequence)

P = ABC

S = ABDBCAB

Current

Found Item

Discarded

MaxSp : Maximal backward extension check

1. Find the last in last appearance of all item.

→ Just find the prefix in reverse order (starting from the end of the Sequence)

P = ABC

S = ABDBCAB

Current

Found Item

Discarded

MaxSp : Maximal backward extension check

1. Find the last in last appearance of all item.

→ Just find the prefix in reverse order (starting from the end of the Sequence)

P = ABC

S = ABDBCAB

Current

Found Item

Discarded

MaxSp : Maximal backward extension check

1. Find the last in last appearance of all item.

→ Just find the prefix in reverse order (starting from the end of the Sequence)

P = ABC

S = ABDBCAB

Current

Found Item

Discarded

MaxSp : Maximal backward extension check

1. Find the last in last appearance of all item.

→ Just find the prefix in reverse order (starting from the end of the Sequence)

P = ABC

S = ABDBCAB

Current

Found Item

Discarded

MaxSp : Maximal backward extension check

2. Find the maximum period of the prefix in the sequence.

→ There is as much maximum period as there is items in the prefix (so here 3)

→ It is the space between the last-in-last appearance of item i and the first occurrence of the prefix before the item i

1st Maximum Period

P = ABC

S = [][ABDBCAB]

MP = (none)

2nd Maximum Period

P = ABC

S = [A]BD[BCAB]

MP = BD

3rd Maximum Period

P = ABC

S = [AB]DB[CAB]

MP = DB

MaxSp : Maximal backward extension check

We do that with all sequence in the database

3. We count support for all item in every maximum period.

If for a same i-th maximum period, one item support $\geq$ minsup $\rightarrow$ There is a Maximal backward extension, the element is not maximal.

Question 3 : Extension

With sequence Database and minsup = 50 %

ABCDBC

CDBDCA

ACBDB

1. Is there any Maximal Forward Extension ?

P = DB

2. Is there any Maximal Backward Extension ?

P = AD

MaxSp Algorithm

Concretly, how does one implement MaxSP based on PrefixSpan ?

The same as PrefixSpan

BUT

1. Before adding a prefix, check that there is no maximal backward extension
2. Add a pattern at the end of the recursive call (at a leaf on the DFS)

MaxSp Algorithm : Steps

Here are the different steps :

1. Scan
2. Project
3. Is there any Maximal Forward Extension in non modified DB (PrefixSpan) ?

   Yes, not maximal
4. Else Is there any Maximal Backward Extension in non modified DB ?

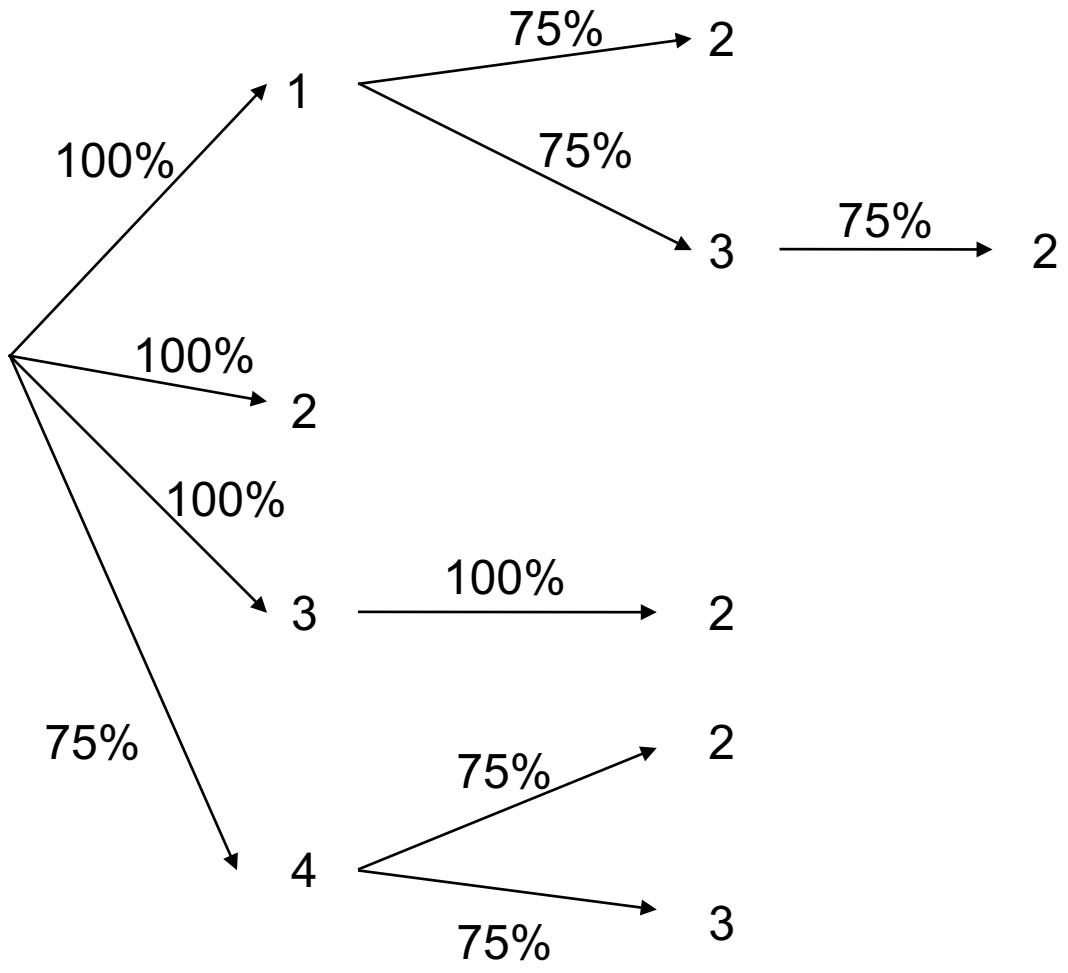
   Yes, not maximal
5. If previous questions are No ?

   Output the sequence

MaxSP: Lets Start From the end

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>
<{4},{3},{2},{1}>
<{5},{1},{4},{3},{2}>
<{5},{7},{1},{3},{2},{3}>



MaxSP: Lets Start From the end

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>

<{4},{3},{2},{1}>

<{5},{1},{4},{3},{2}>

<{5},{7},{1},{3},{2},{3}>

Here are all possibilities without Maximal Forward Extension

<{1},{2}>

<{1},{3},{2}>

<{2}>

<{3},{2}>

<{4},{2}>

<{4},{3}>

To know if we must add them as maximal, you just check for each of the if there is a Maximal Backward Extension

MaxSP: Lets Start From the end

MinSup 75 % (3)

<{1},{3},{4},{6},{2},{3}>

<{4},{3},{2},{1}>

<{5},{1},{4},{3},{2}>

<{5},{7},{1},{3},{2},{3}>

Lets show with *<{1}, {2}>*

MaxSP: Lets Start From the end

First Maximal Period: MinSup 75 % (3) With <{1}, {2}>

<{1},{3},{4},{6},{2},{3}>

<{4},{3},{2},{1}> No item has support ≥ MinSup

<{5},{1},{4},{3},{2}>

<{5},{7},{1},{3},{2},{3}>

Second Maximal Period: MinSup 75 % (3) With <{1}, {2}>

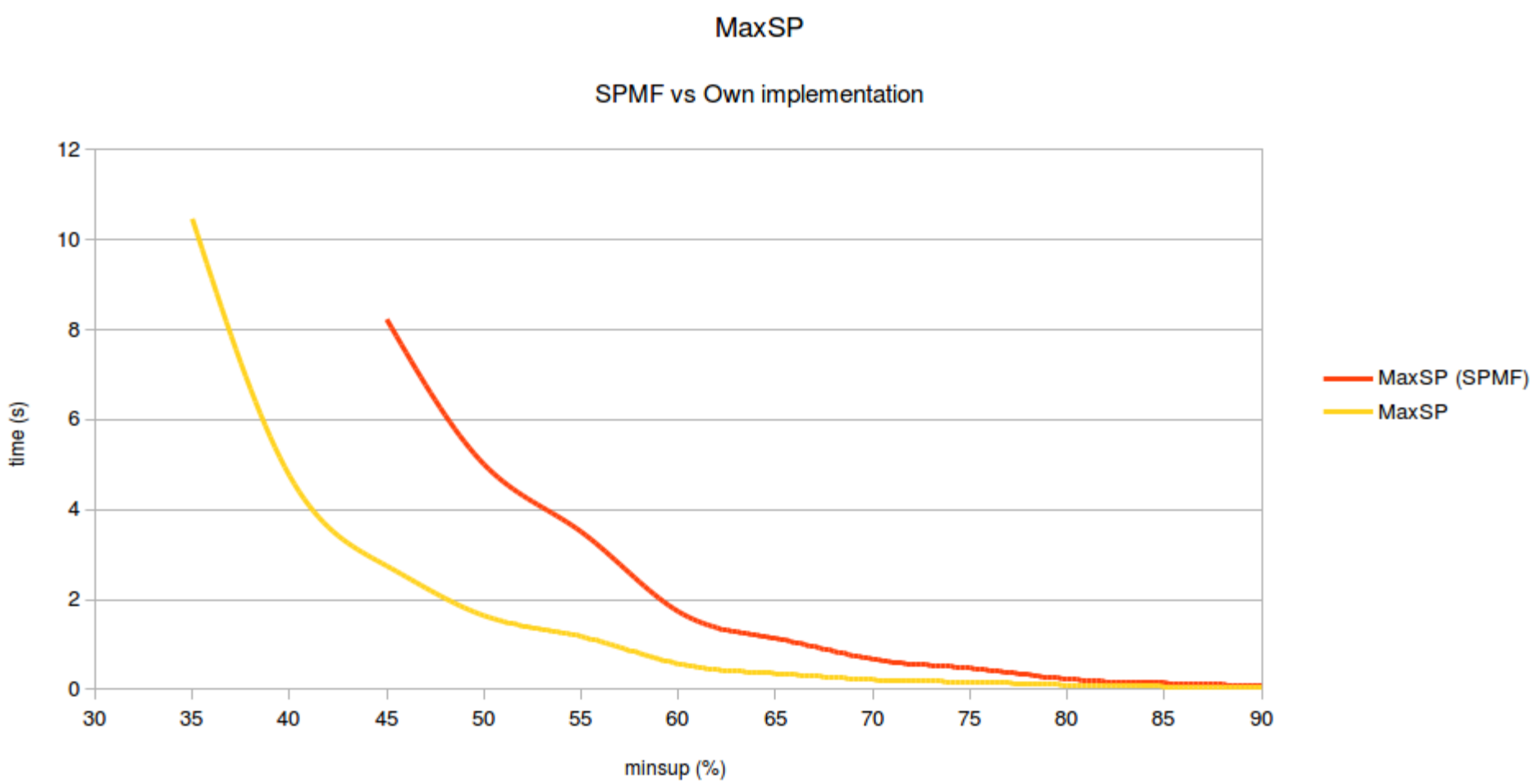
<{1},{3},{4},{6},{2},{3}>

<{4},{3},{2},{1}> Item {3} has support of 75%

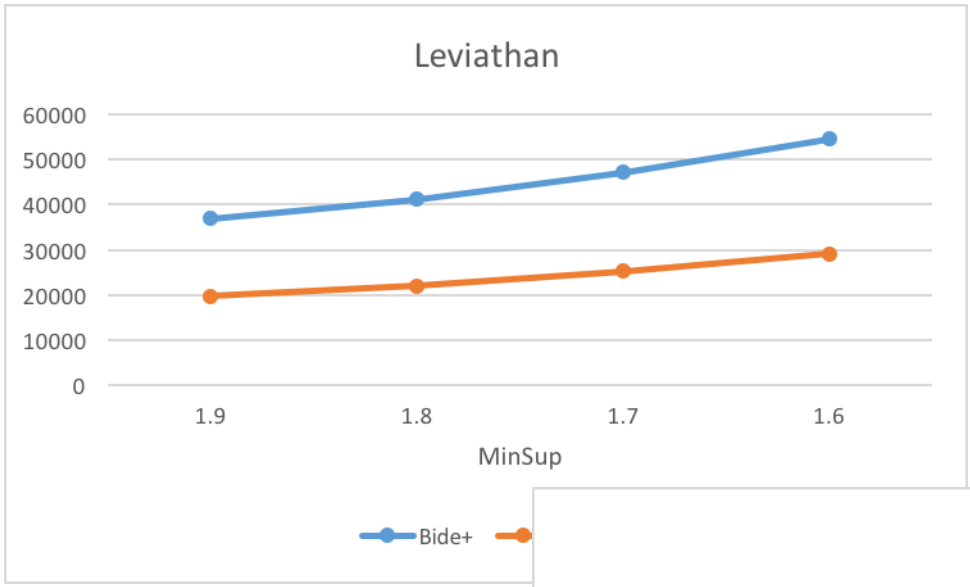
<{5},{1},{4},{3},{2}> there is a MBE, so <{1},{2}>

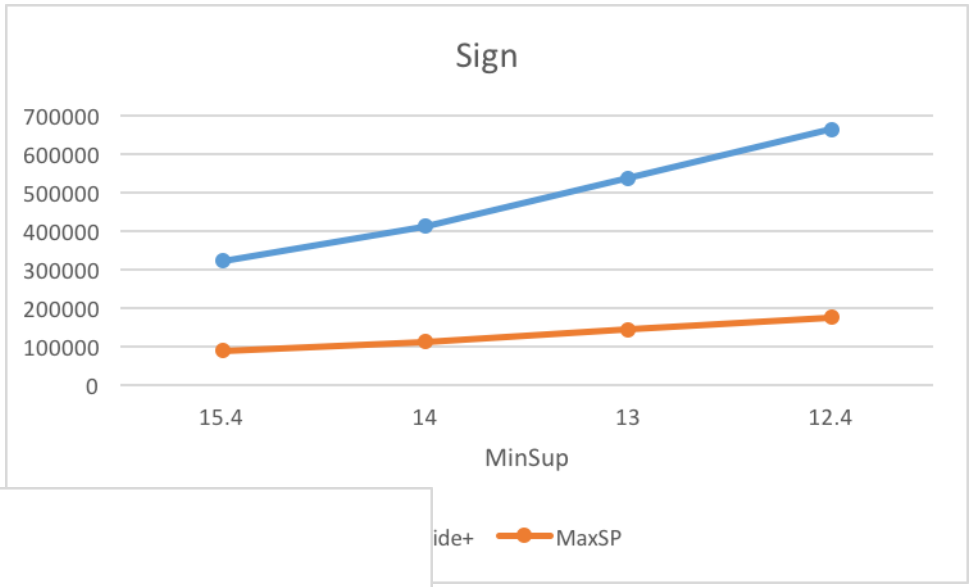
<{5},{7},{1},{3},{2},{3}> is not maximal sequential pattern

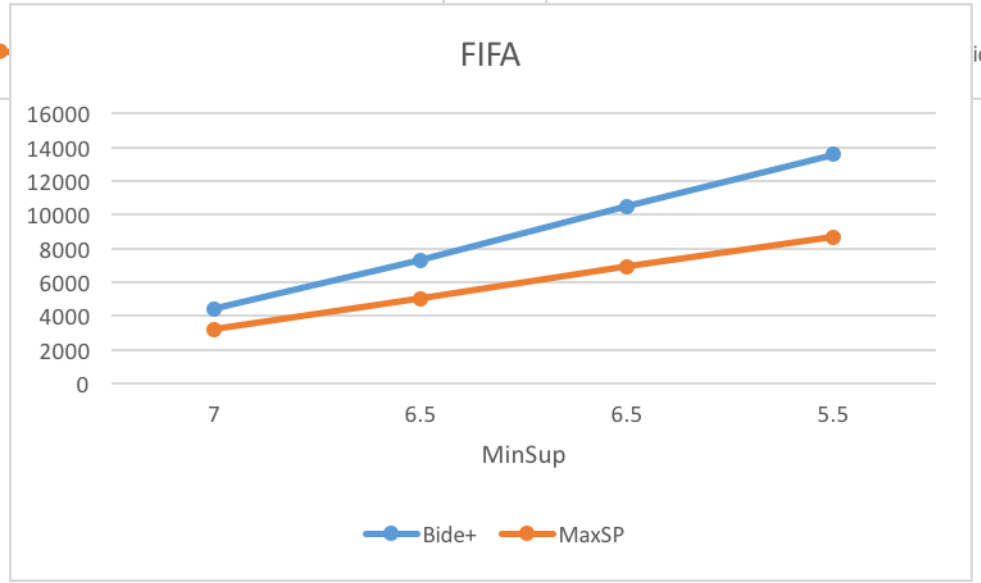
Measures : Own MaxSP vs SPMF MaxSP



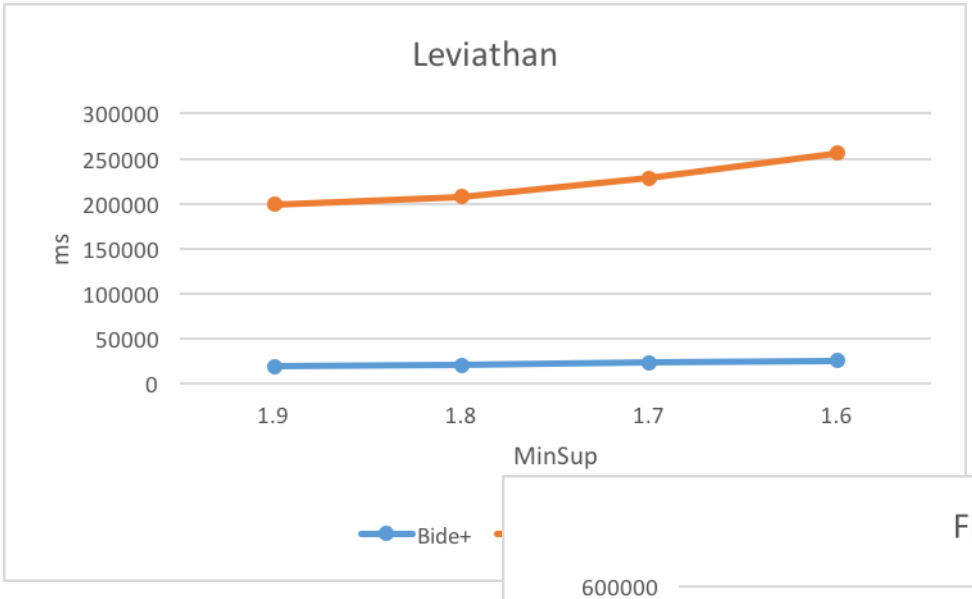
Measures : MaxSP vs Bide+ (Frequent sequences count)

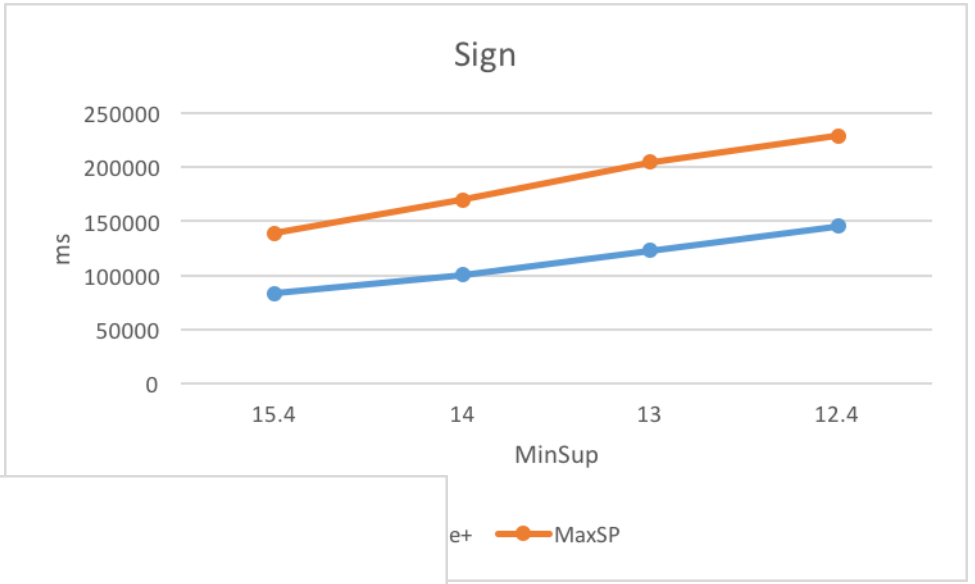


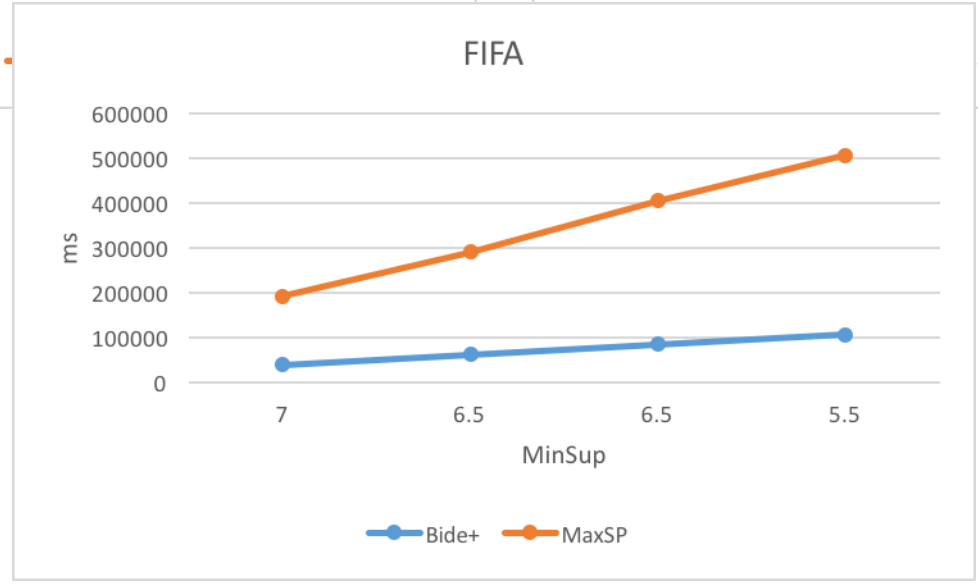




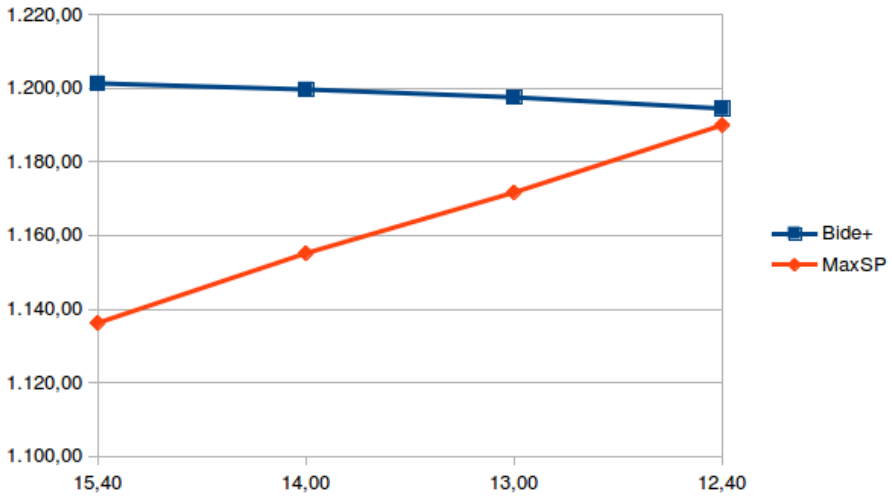
Measures : MaxSP vs Bide+ (Time)



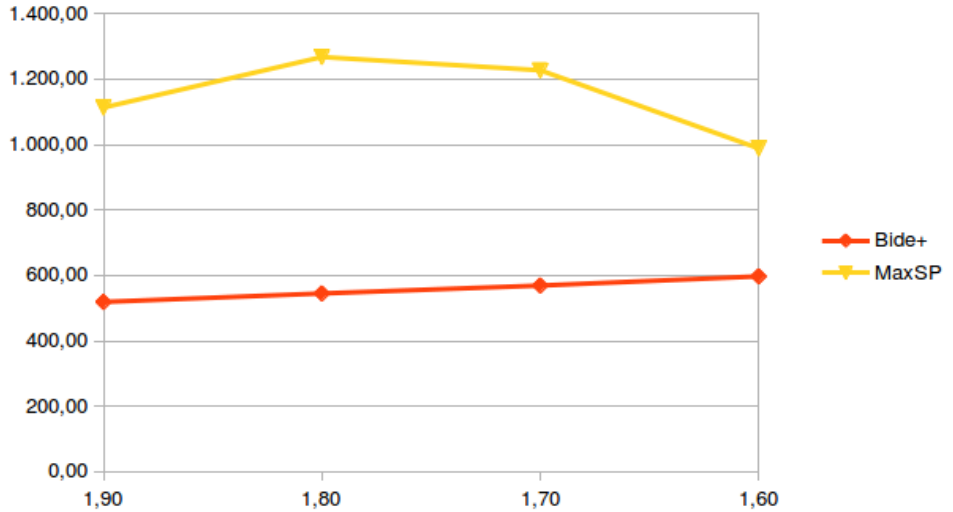




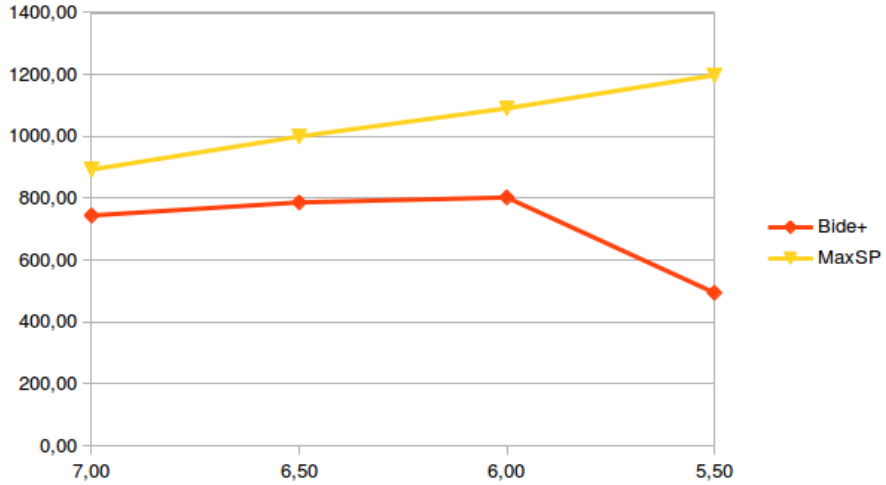
Measures : MaxSP vs Bide+ (Memory)



Sign



Leviathan



FIFA

Last Considerations

We did not considered examples with itemsets (as in the articles), it is not really different, you can apply the same rules

We implemented the Algorithm, the presentation and algorithm are available on our git repository. The algorithm can be optimized as stated in the paper

The End

Thanks for listening !