Pattern Mining

What I have learned.

Jakub Peschel

May 10, 2019

Outline of Presentation

Part I

Introduction

Outline for Introduction

About Me

DISA
Laboratory of Data Intensive Systems and Applications

Previous experience:

- Bc. and Mgr of Artificial Intelligence at FI MUNI
- Focus on learning methods
- Bigger focus on Deep Learning
- Interested in Reinforcement Learning

PhD. studies:

- Prof. Zezula
- Focus on pattern mining
- Community searching

Pattern

Pattern is a discernible regularity in the world or man-made design.

Pattern can be:

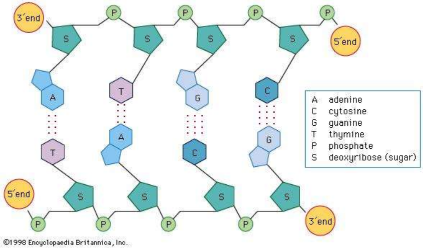
- co-occurrence of items (Shopping basket analysis)

Pattern

Pattern is a discernible regularity in the world or man-made design.

Pattern can be:

- co-occurrence of items (Shopping basket analysis)
- repetitions of items (Genes in DNA)

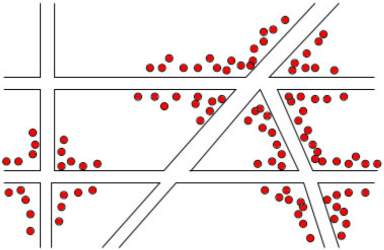


Pattern

Pattern is a discernible regularity in the world or man-made design.

Pattern can be:

- co-occurrence of items (Shopping basket analysis)
- repetitions of items (Genes in DNA)
- locality of items (Location of ill people)



Association Rule

Association rules are set of implications, which tells us with some probability that some item will occur together with some other.

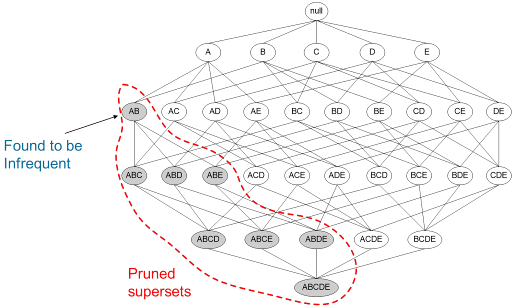
Example

$$\{\text{bread}\} \implies \{\text{butter}\}$$

Apriori Principle

Apriori Principle (downward closure property)

Occurrence of pattern $A \cup B$ sets lower bound for occurrence of pattern A and pattern B.



Closed and Maximal Patterns

Closed Pattern

Closed pattern is such pattern that every subpattern has at least same support.

- Such pattern is good for reduction of needed space

Maximal Pattern

Maximal pattern is such pattern such that there is not longer pattern which is frequent.

Basic Metrics

We need to evaluate association rules

- to assign importance
- to estimate validity

Available metrics:

- Support/Confidence
- Lift
- Null-Invariant Measures

Support/Confidence

Definition:

- *Support*: Indicator, how often the itemset appears in dataset.
  - Used for frequent pattern.
- *Confidence*: Indicator how often the rule has been find true.
  - Used for association rule.

- Most important concept in frequent pattern mining.
- Used for computing lot of others.

Lift

$$LIFT(X \implies Y) = \frac{supp(X \cup Y)}{supp(X) \times supp(Y)}$$

- Ratio of the observed support to expected value if X and Y were independent.
- Lift > 1 means that rule can be potentially useful for prediction.

Null-Invariant Measures

- *Null-transaction*: Transaction which doesn't contain our subpattern.
- Aren't affected by # of null transactions

Measures:

- $Jaccard(X \implies Y) = \frac{supp(X \cup Y)}{supp(X) + supp(Y) - supp(X \cup Y)}$
- $Cosine(X \implies Y) = \frac{supp(X \cup Y)}{\sqrt{supp(X) \times supp(Y)}}$
- $Kulczynski(X \implies Y) = \frac{1}{2}\left(\frac{supp(X \cup Y)}{supp(X)} + \frac{supp(X \cup Y)}{supp(Y)}\right)$

Basic Problems

Example

Let number of possible objects be 100. How many possible patterns are there which can occur in data?

Basic Problems

Example

Let number of possible objects be 100. How many possible patterns are there which can occur in data?

There is $X = \sum_{i=1}^{n} \binom{n}{i}$ possible combinations which is approx. one quintillion (10^{30}) possible patterns.

Basic Problems

Example

Let number of possible objects be 100. How many possible patterns are there which can occur in data?

There is $X = \sum_{i=1}^{n} \binom{n}{i}$ possible combinations which is approx. one quintillion (10^{30}) possible patterns.

Problematic operations:

- counting occurrence
- generating potential candidates

Part II

Frequent Item Analysis Methods

Outline for Frequent Item Analysis Methods

Basic concepts

Join Based Methods
- Apriori

Tree Based Methods
- Tree Projection

Pattern Growth Methods
- FP-Growth

Vertical Methods
- Eclat

Basic concepts

- Database
- Transaction
- Pattern
- Min_support

t1	{A, C, D, E}
t2	{A, B, D, E}
t3	{C, D}
t4	{A, C, E}
t5	{A, E}
t6	{A, B}
t7	{A, C, D, E}
t8	{A, B, C, E}

min_support = 4

Basic concepts

- Database
- Transaction
- Pattern
- Min_support

t1	{A, C, D, E}
t2	{A, B, D, E}
t3	{C, D}
t4	{A, C, E}
t5	{A, E}
t6	{A, B}
t7	{A, C, D, E}
t8	{A, B, C, E}

min_support = 4

Basic concepts

- Database
- Transaction
- Pattern
- Min_support

t1	{A, C, D, E}
t2	{A, B, D, E}
t3	{C, D}
t4	{A, C, E}
t5	{A, E}
t6	{A, B}
t7	{A, C, D, E}
t8	{A, B, C, E}

min_support = 4

Basic concepts

- Database
- Transaction
- Pattern
- Min_support

t1	{A, C, D, E}
t2	{A, B, D, E}
t3	{C, D}
t4	{A, C, E}
t5	{A, E}
t6	{A, B}
t7	{A, C, D, E}
t8	{A, B, C, E}

min_support = 4

Apriori

- Based on Apriori principle.
- Breath first search.
- Sometimes called Level-wise search

- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning

Apriori

t1: {A,C,D,E}
t2: {A,B,D,E}
t3: {C,D}
t4: {A,C,E}
t5: {A,E}
t6: {A,B}
t7: {A,C,D,E}
t8: {A,B,C,E}

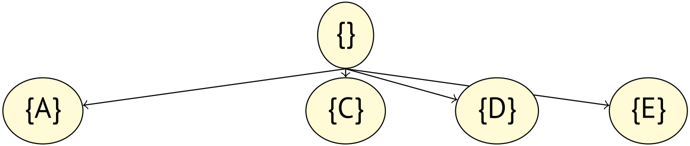
{}

- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning

Apriori

t1: {A,C,D,E}
t5: {A,E}
t2: {A,B,D,E}
t6: {A,B}
t3: {C,D}
t7: {A,C,D,E}
t4: {A,C,E}
t8: {A,B,C,E}

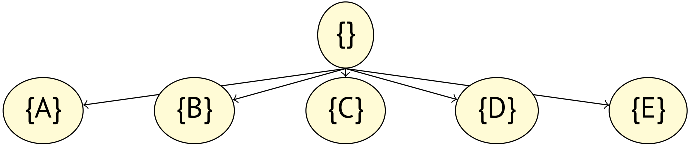
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

- t1: {A,C,D,E}
- t2: {A,B,D,E}
- t3: {C,D}
- t4: {A,C,E}
- t5: {A,E}
- t6: {A,B}
- t7: {A,C,D,E}
- t8: {A,B,C,E}

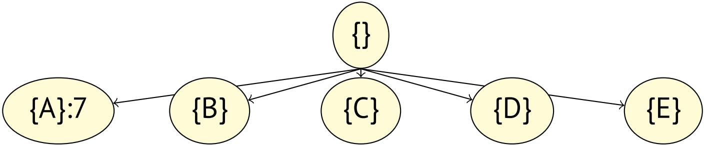
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

t1: {A,C,D,E}
t2: {A,B,D,E}
t3: {C,D}
t4: {A,C,E}
t5: {A,E}
t6: {A,B}
t7: {A,C,D,E}
t8: {A,B,C,E}

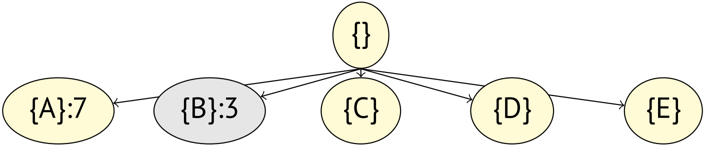
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

t1: {A,C,D,E}
t5: {A,E}
t2: {A,B,D,E}
t6: {A,B}
t3: {C,D}
t7: {A,C,D,E}
t4: {A,C,E}
t8: {A,B,C,E}

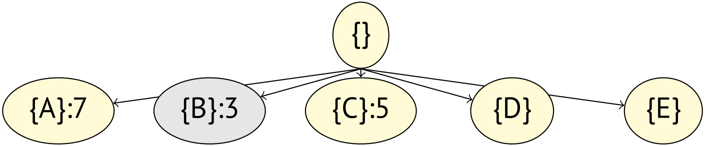
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

t1: {A,C,D,E}
t5: {A,E}
t2: {A,B,D,E}
t6: {A,B}
t3: {C,D}
t7: {A,C,D,E}
t4: {A,C,E}
t8: {A,B,C,E}

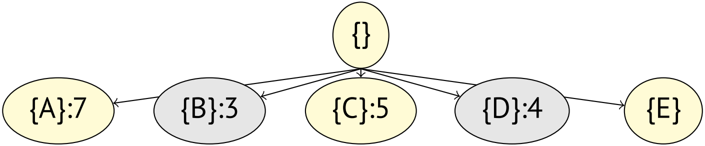
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

t1: {A,C,D,E}
t2: {A,B,D,E}
t3: {C,D}
t4: {A,C,E}
t5: {A,E}
t6: {A,B}
t7: {A,C,D,E}
t8: {A,B,C,E}

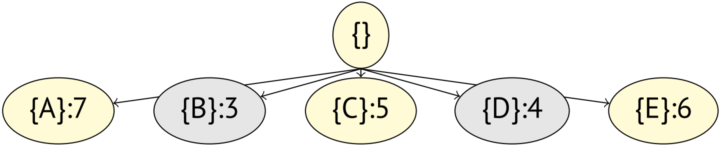
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

t1: {A,C,D,E}
t2: {A,B,D,E}
t3: {C,D}
t4: {A,C,E}
t5: {A,E}
t6: {A,B}
t7: {A,C,D,E}
t8: {A,B,C,E}

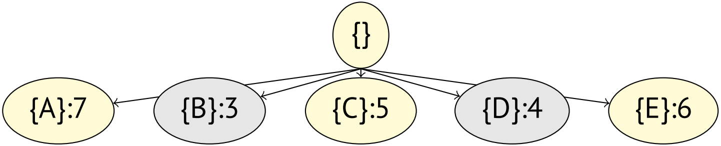
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

t1: {A,C,D,E}
t5: {A,E}
t2: {A,B,D,E}
t6: {A,B}
t3: {C,D}
t7: {A,C,D,E}
t4: {A,C,E}
t8: {A,B,C,E}

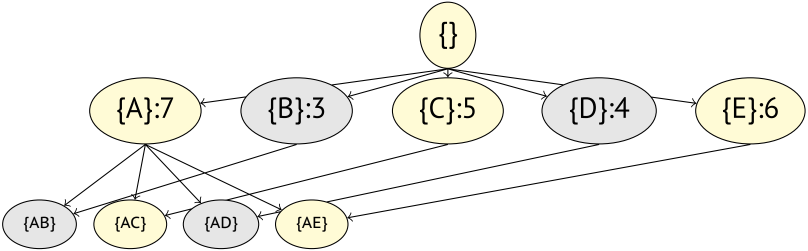
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

- t1: {A,C,D,E}
- t2: {A,B,D,E}
- t3: {C,D}
- t4: {A,C,E}
- t5: {A,E}
- t6: {A,B}
- t7: {A,C,D,E}
- t8: {A,B,C,E}

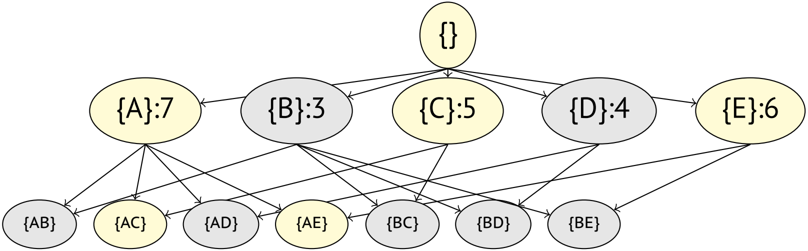
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}

- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

t1: {A,C,D,E}

t2: {A,B,D,E}

t3: {C,D}

t4: {A,C,E}

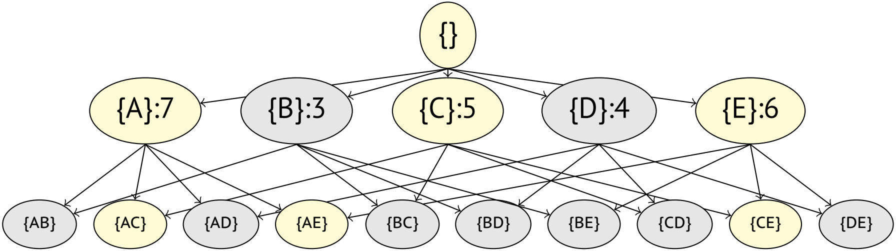
t5: {A,E}

t6: {A,B}

t7: {A,C,D,E}

t8: {A,B,C,E}

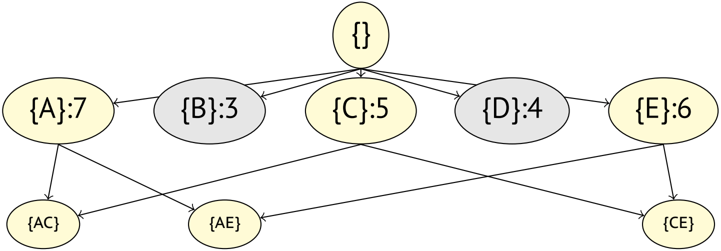
- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Apriori

t1: {A,C,D,E}
t2: {A,B,D,E}
t3: {C,D}
t4: {A,C,E}
t5: {A,E}
t6: {A,B}
t7: {A,C,D,E}
t8: {A,B,C,E}

- Consist of three stages:
  - Candidate generation
  - Support counting
  - Prunning



Tree Projection

- Defined ordering of items.
- Depth first search of space.
- Prefix or suffix version.
- Reduction of database each step.

Tree Projection

ordering: $A < B < C < D < E$

Tree Projection

ordering: $A < B < C < D < E$

t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}

Tree Projection

ordering: $A < B < C < D < E$

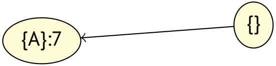
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}

{}

Tree Projection

ordering: $A < B < C < D < E$

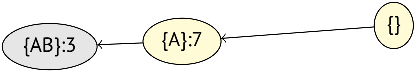
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

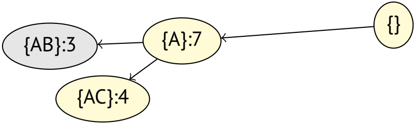
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

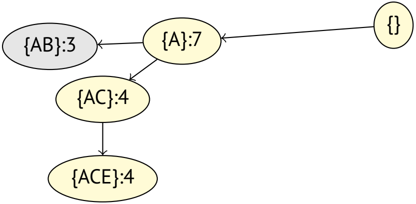
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

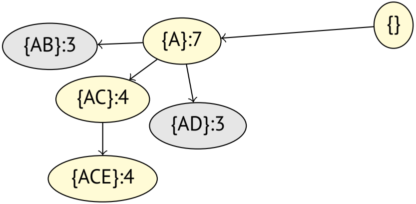
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

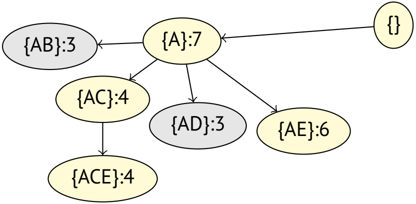
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

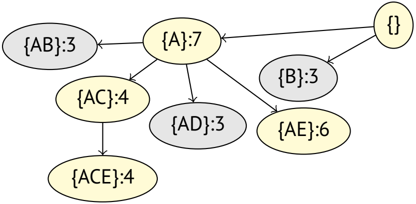
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

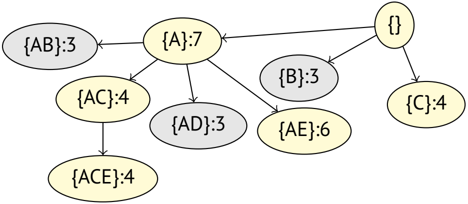
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

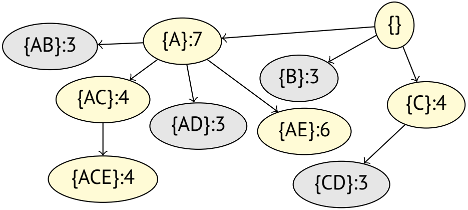
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

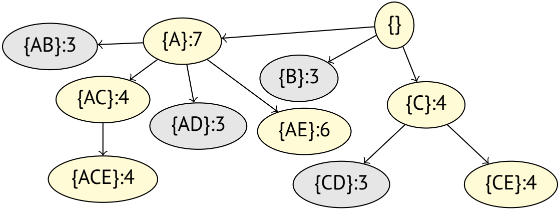
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

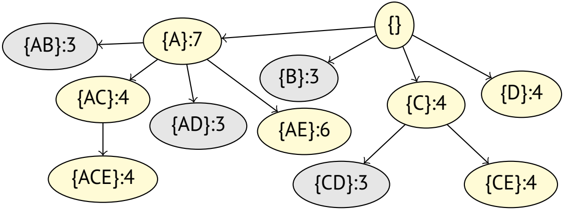
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

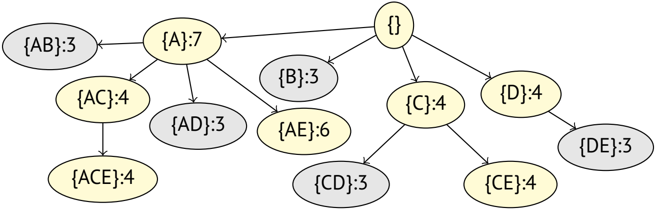
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

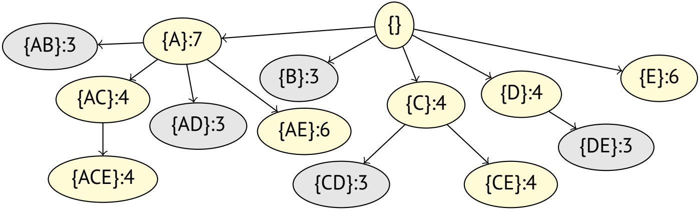
t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}



Tree Projection

ordering: $A < B < C < D < E$

t1: {A,C,D,E}	t5: {A,E}
t2: {A,B,D,E}	t6: {A,B}
t3: {C,D}	t7: {A,C,D,E}
t4: {A,C,E}	t8: {A,B,C,E}

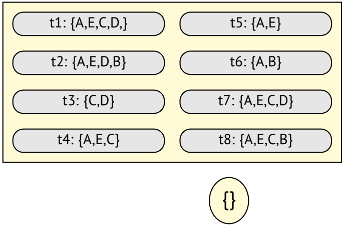


FP-Growth

t1: {A,E,C,D,}	t5: {A,E}
t2: {A,E,D,B}	t6: {A,B}
t3: {C,D}	t7: {A,E,C,D}
t4: {A,E,C}	t8: {A,E,C,B}

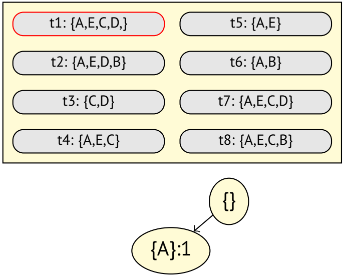
- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item

FP-Growth



- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item

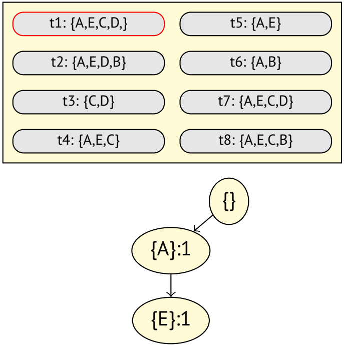
FP-Growth



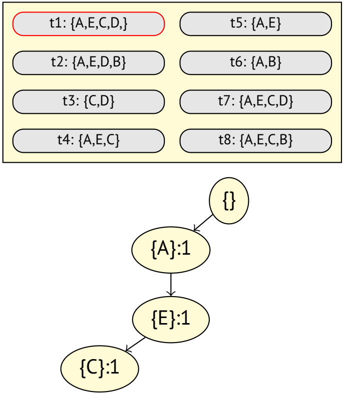
- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item

FP-Growth

- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item



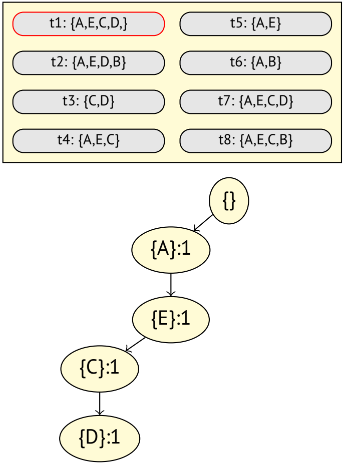
FP-Growth



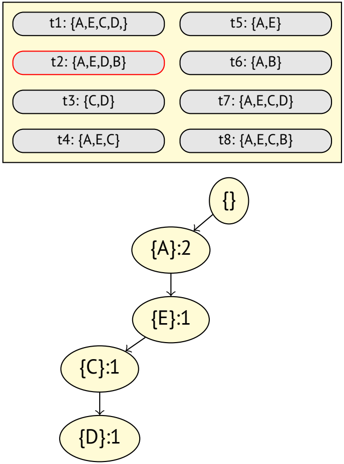
- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item

FP-Growth

- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item

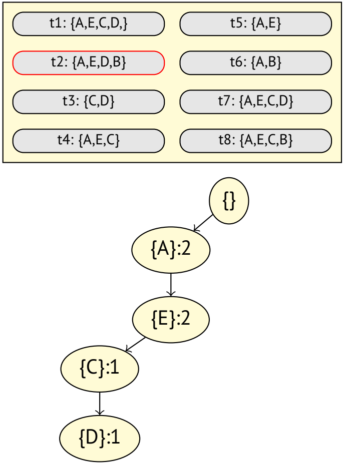


FP-Growth



- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item

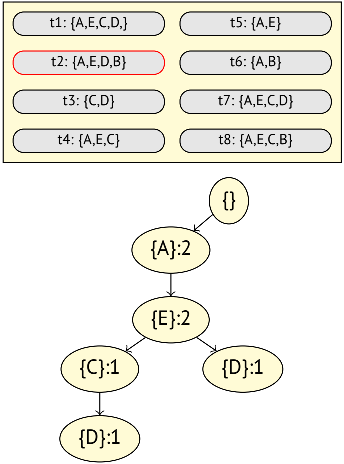
FP-Growth



- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item

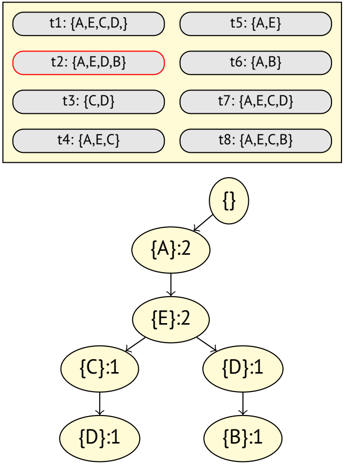
FP-Growth

- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item



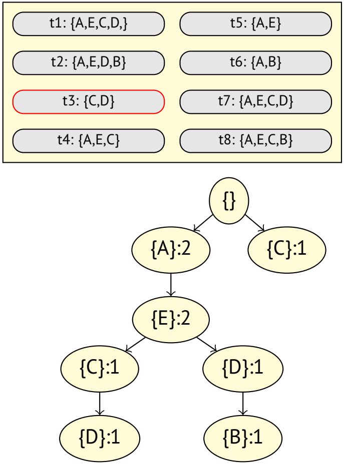
FP-Growth

- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item



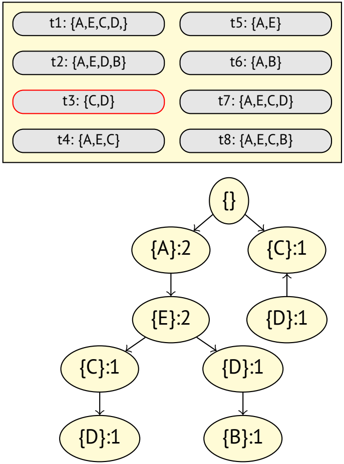
FP-Growth

- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item



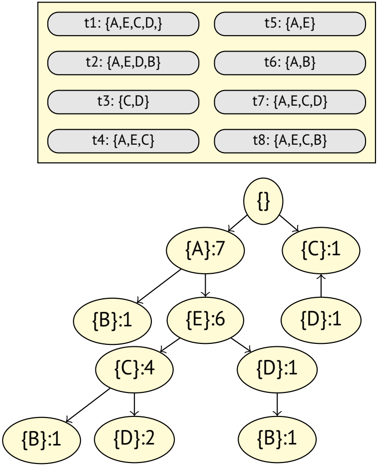
FP-Growth

- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item



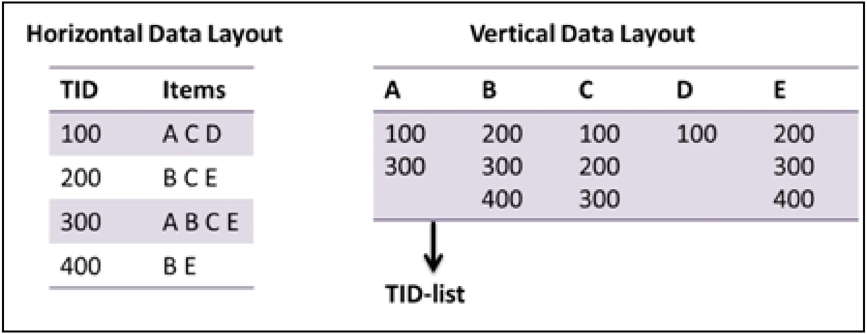
FP-Growth

- Using FP-tree
- Saving memory by minimizing tree
- Building conditional FP-trees from least frequent item



Eclat

- Transform database that each item is basket of transactions.
- Support counting is simplified to counting elements of basket.
- Generation by intersecting baskets.



Horizontal Data Layout

TID	Items
100	A C D
200	B C E
300	A B C E
400	B E

Vertical Data Layout

A	B	C	D	E
100	200	100	100	200
300	300	200		300
	400	300		400

Part III

Graph Mining

Outline for Graph Mining

Graph Communities

Approaches

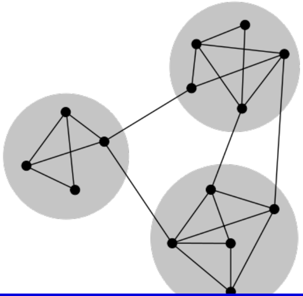
- Betweenness
- Bipartite Graphs Methods
- Graph Partitioning
- Frequent Itemset Mining

Graph Communities

Social network is representation of complex data.

Example

- Social interaction between people
- Protein interactions
- Web

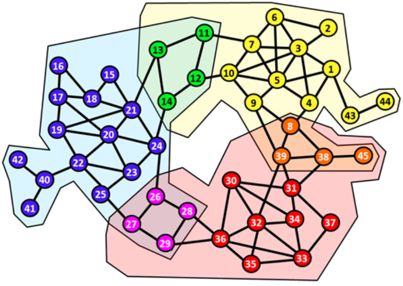


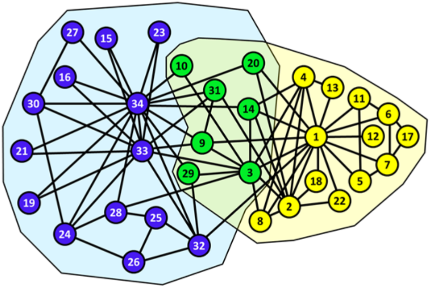
Communities

Groups of items which are internally densely connected.

Graph Communities

- Grouping nodes together.
- Communities are often overlapping.
- Standard methods are not suitable.



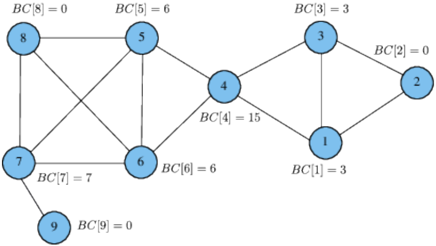


Problem definition

Graph pattern mining

Given a function $f(g)$ and threshold θ, find all subgraphs g such that $f(g) \geq \theta$

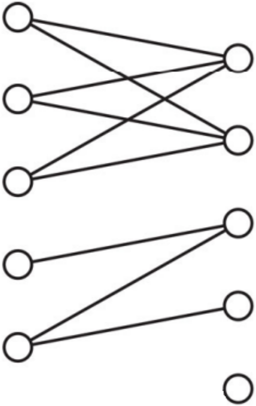
Betweenness



Betweenness shows which nodes are probably borders of community.

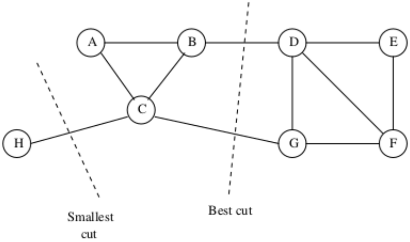
Bipartite Graphs Methods

- Clique searching is NP-complete for normal graphs
- Much easier for bipartite graphs
- Algorithms for frequent patterns are applicable
- Good transformation necessary



Graph Partitioning

- Another approach
- Minimal cut is not usable
- Normalized cut is used.



Normalized cut

$$\frac{Cut(S,T)}{Vol(S)} + \frac{Cut(S,T)}{Vol(T)}$$

S and T: sets of nodes (clusters)
$Cut(S,T)$: # of edges between S and T
$Vol(S)$: # of edges with at least one end in S

Frequent Itemset Mining

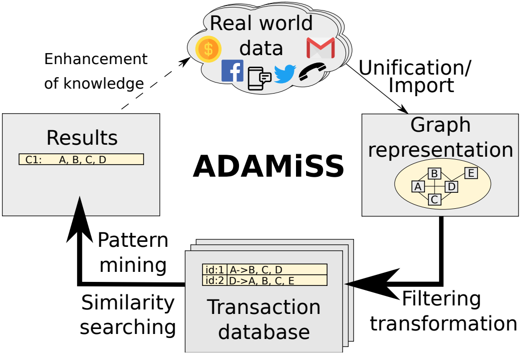


Figure: Model of ADAMiSS

Part IV

Sequence mining

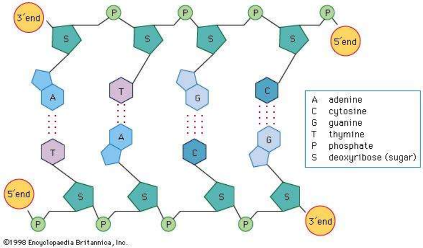
Outline for Sequence mining

Basic Concepts

How to mine sequences?

Basic Concepts

- Sequence data
  - DNA sequences
  - Choreography
  - Video



Problem definition

- Sequence: ordered list of elements.
- Element: item or set of items.
- Sequence database: set of sequences.
- Periodic pattern
  - Sublist of elements that shows periodically.
- Significant pattern
  - Sublist of elements which has high enough support.
- Approximate pattern
  - Sublist of elements which approximate elements occurred in sets well.

Basic Concepts- Sequence data

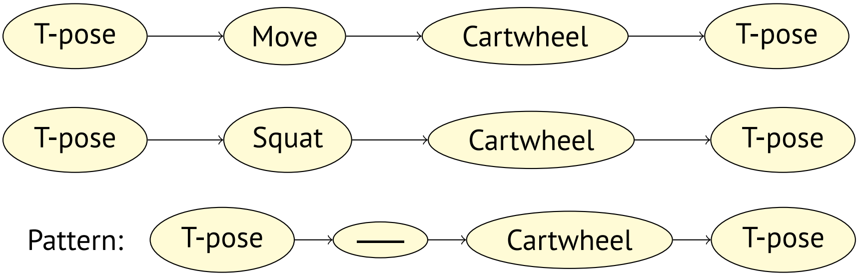
- Sequence of items

Basic Concepts- Sequence data

- Sequence of items
  - Can consist of baskets (ordering doesn't matter)
  - Items can repeat
  - Ordering in sequence matter

Basic Concepts- Sequence data

- Sequence of items
  - Can consist of baskets (ordering doesn't matter)
  - Items can repeat
  - Ordering in sequence matter



- In pattern there can be holes

How to mine sequences?

- Algorithm GSP:
  - similar approach as Apriori
- Alorithm SPADE:
  - based on Eclat
- Algorithm PrefixSpan:
  - Depth first search
  - We are growing prefix of sequence
  - In each step we create projected database
  - Used because of scalability

Thank you for your attention!

Part V

Appendix

Outline for Appendix

Betweenness

Prefixspan

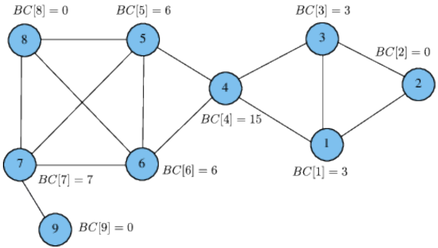
Betweenness

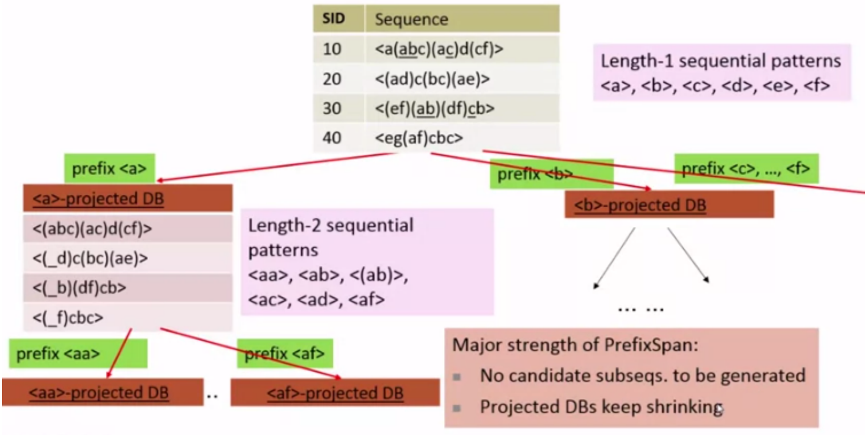
Formula

$$BC(n) = \sum_{s \neq n \neq t} \frac{\sigma_{st}(n)}{\sigma_{st}}$$

σ_{st}: # of shortest paths between node s and t

$\sigma_{st}(n)$: # of shortest paths between node s and t containing n




SID | Sequence
10 | <a(abc)(ac)d(cf)>
20 | <(ad)c(bc)(ae)>
30 | <(ef)(ab)(df)cb>
40 | <eg(af)cbc>
Length-1 sequential patterns
<a>, <b>, <c>, <d>, <e>, <f>
prefix <a>
prefix <b>
prefix <c>, ..., <f>
<a>-projected DB
<(abc)(ac)d(cf)>
<(_d)c(bc)(ae)>
<(_b)(df)cb>
<(_f)cbc>
<b>-projected DB
Length-2 sequential patterns
<aa>, <ab>, <(ab)>, <ac>, <ad>, <af>
… …
prefix <aa>
prefix <af>
<aa>-projected DB
..
<af>-projected DB
Major strength of PrefixSpan:
No candidate subseqs. to be generated
Projected DBs keep shrinking