BITS Pilani

BITS Pilani
Hyderabad Campus

Prof.Aruna Malapati
Department of CSIS

Sequential Association Rule Mining

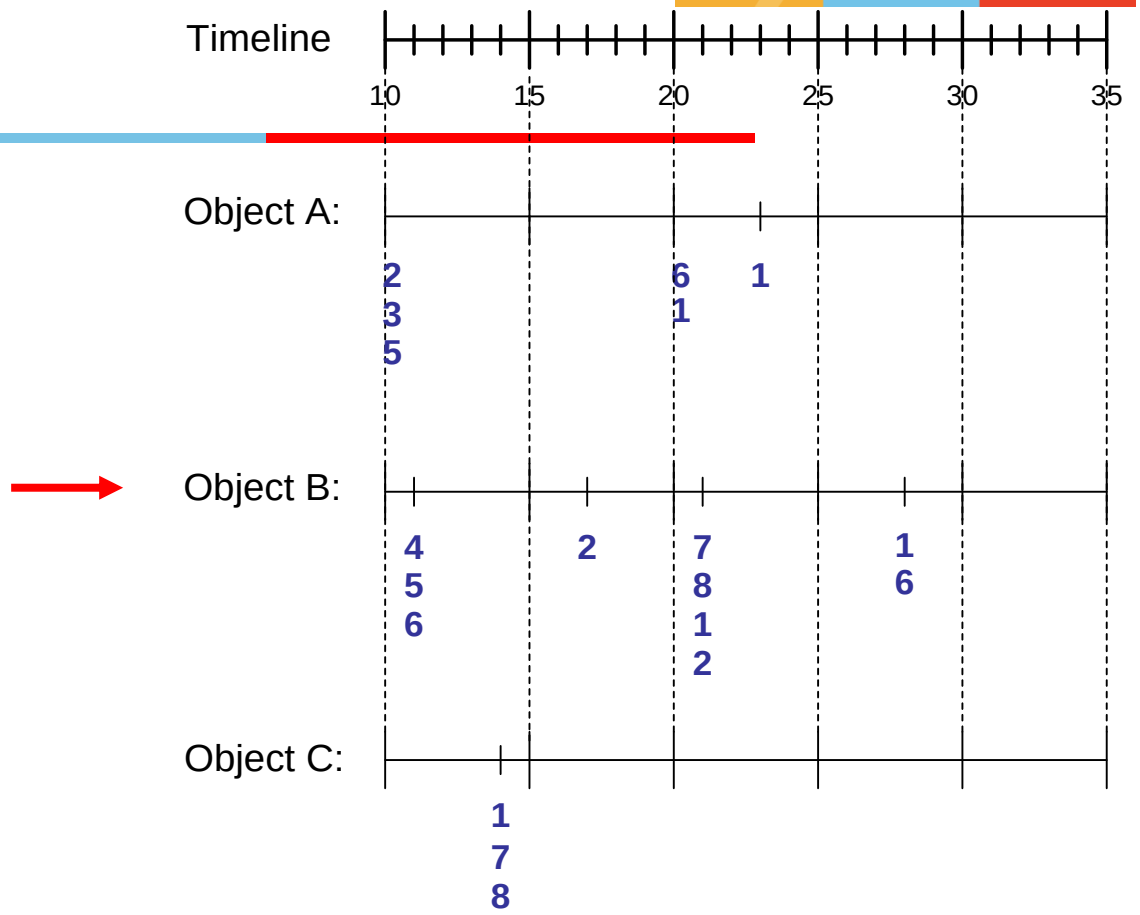
Today's Learning objective

- Generate Sequential association rules when the items are grouped as per the purchase over time.
- Implement GSP and Spade algorithms.
- Apply the window constraint based on the problem.

Sequence Data

Sequence Database:

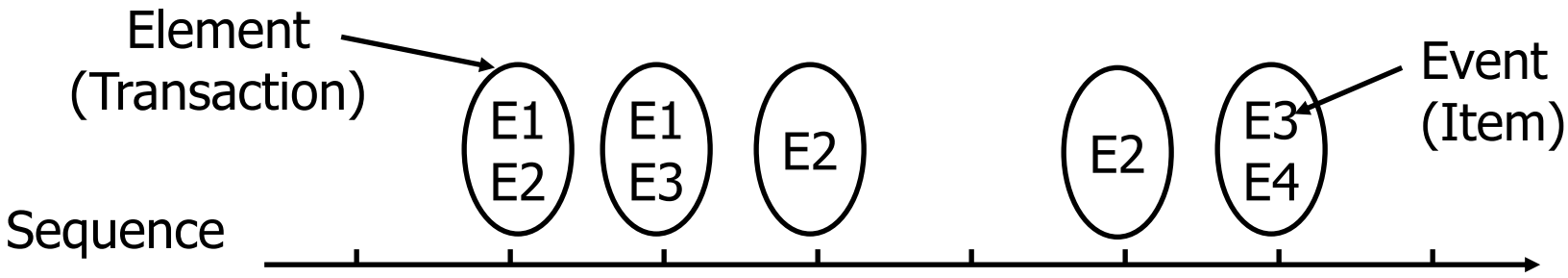
Object	Timestamp	Events
A	10	2, 3, 5
A	20	6, 1
A	23	1
B	11	4, 5, 6
B	17	2
B	21	7, 8, 1, 2
B	28	1, 6
C	14	1, 8, 7



- A sequence is an ordered list of elements.
- A sequence can be denoted as S = $<e_1e_2e_3\ldots e_n>$ where each element e_j is a collection of one or more events i.e $e_j=\{i_1,i_2,i_3,\ldots i_k\}$

Examples of Sequence Data

Sequence Database	Sequence	Element (Transaction)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, CDs, etc
Web Data	Browsing activity of a particular Web visitor	A collection of files viewed by a Web visitor after a single mouse click	Home page, index page, contact info, etc
Event data	History of events generated by a given sensor	Events triggered by a sensor at time t	Types of alarms generated by sensors
Genome sequences	DNA sequence of a particular species	An element of the DNA sequence	Bases A,T,G,C



Formal Definition of a Subsequence

- A sequence $\langle\ a_1\ a_2\ \ldots\ a_n\ \rangle$ is contained in another sequence $\langle\ b_1\ b_2\ \ldots\ b_m\ \rangle$ ($m \geq n$) if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i2}$, …, $a_n \subseteq b_{in}$

Data sequence	Subsequence	Contained?
⟨ (2,4) (3,5,6) } ⟩	⟨ 2 (3,5) ⟩	Yes
⟨ (1,2) (3,4) ⟩	⟨ 1 2 ⟩	No
⟨ (2,4) (2,4) (2,5) ⟩	⟨ 2 4 ⟩	Yes

- Support of a subsequence *w* is the fraction of data sequences that contain *w*
- A *sequential pattern* is a frequent subsequence (i.e., a subsequence whose support is ≥ *minsup*)

Formal Definition of a Sequence

- Example

$$s = < \{A,B\}, \{B,E,F\}, \{A\}, \{E,F,H\} >$$

  - Length of s: $|s| = 4$ elements
  - s is a 9-sequence
  - Times associated to elements:
    - {A,B} → time=0
    - {B,E,F} → time = 120
    - {A} → time = 130
    - {E,F,H} → time = 200

Sequences without Explicit Time Info

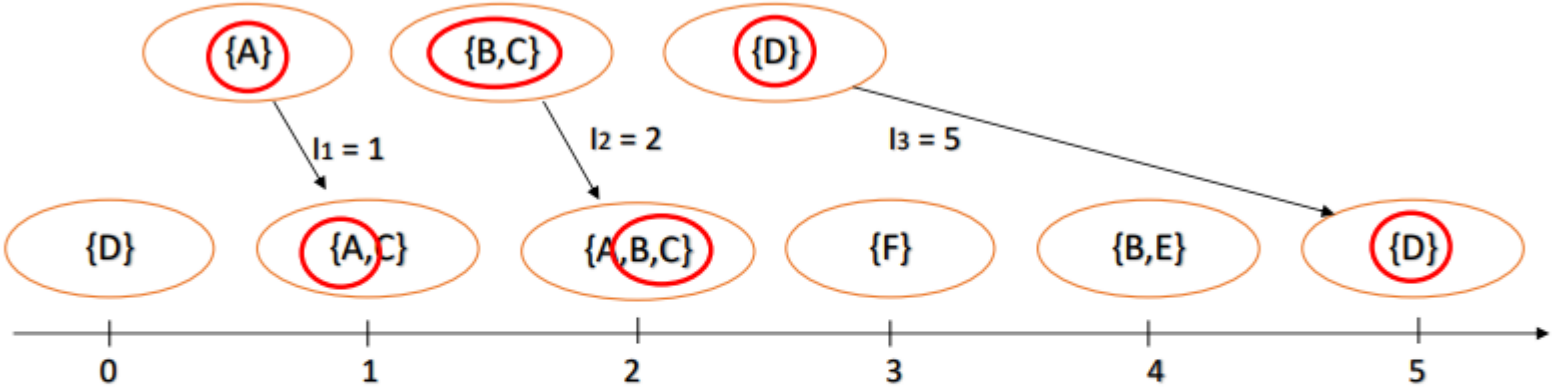
- Default: time of element = position in the sequence
- Example

  S = < {A,C}, {E}, {A,F}, {E,G,H} >

  - Default times associated to elements:
    - {A,C} → time=0
    - {E} → time = 1
    - {A,F} → time = 2
    - {E,G,H} → time = 3

Formal Definition of a Subsequence

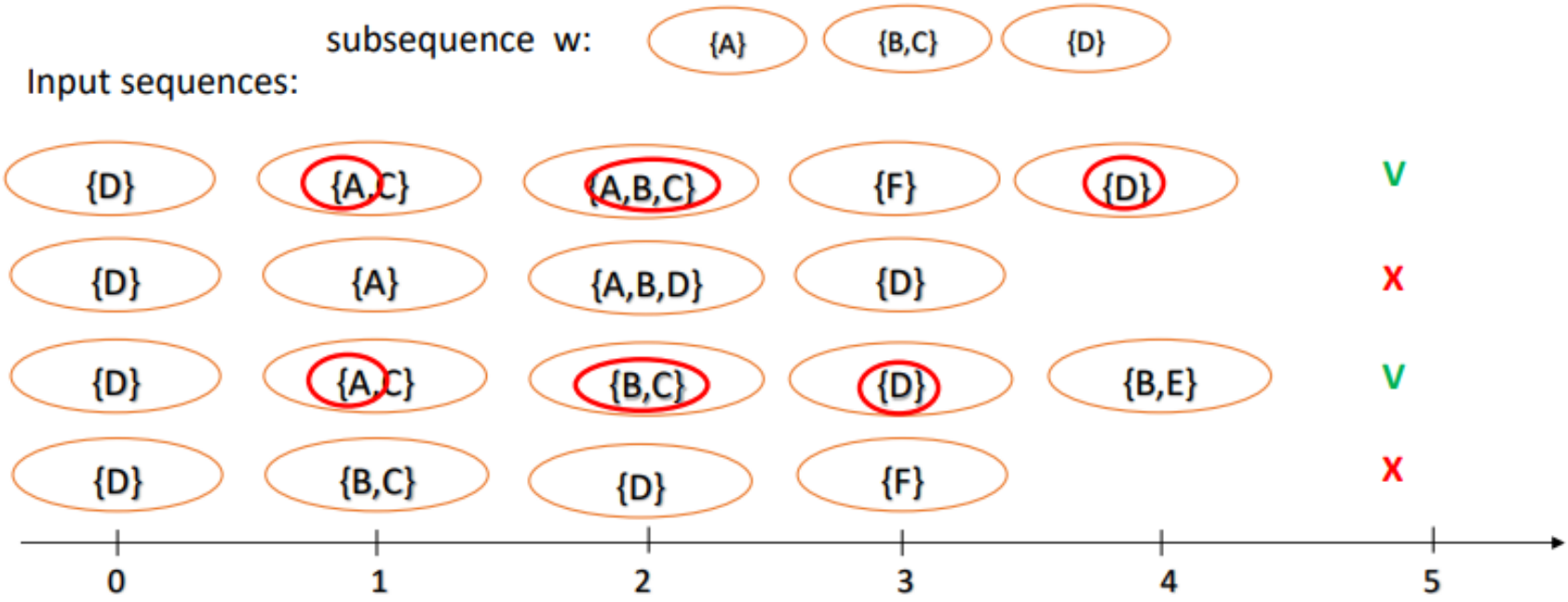
- A sequence $\langle a_1\, a_2 \ldots a_n \rangle$ is contained in another sequence $\langle b_1\, b_2 \ldots b_m \rangle$ $(m \geq n)$ if there exist integers $i_1 < i_2 < \ldots < i_n$ such that $a_1 \subseteq b_{i1}$, $a_2 \subseteq b_{i1}$, …, $a_n \subseteq b_{in}$



Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	
< {1,2} {3,4} >	< {1} {2} >	
< {2,4} {2,4} {2,5} >	< {2} {4} >	

Formal Definition of Sequential Pattern

- The **support** of a subsequence *w* is the fraction of data sequences that contain w



- A **sequential pattern**
  - is a **frequent** subsequence
  - i.e., a subsequence whose support is ≥ ***minsup***

support of w: 2/4 = 0.50 (50%)

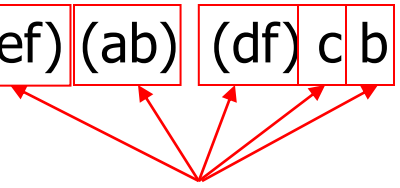
Sequential Pattern Mining: Definition

- Given:
  - a database of sequences
  - a user-specified minimum support threshold, *minsup*
- Task:
  - Find all subsequences with support ≥ *minsup*

Sequential pattern and Sequential pattern Mining

- Sequential pattern: Given a set of sequences from database, find the complete set of frequent subsequences (i.e., satisfy the min_sup threshold)

A *sequence* : <(ef)(ab)(df)cb>



A *sequence database*

SID	sequence
10	<a(abc)(ac)d(cf)>
20	<(ad)c(bc)(ae)>
30	<(ef)(ab)(df)cb>
40	<eg(af)cbc>

An element may contain a set of items. Items within an element are unordered and we list them alphabetically.

<a(bc)dc> is a *subsequence* of <a(abc)(ac)d(cf)>

Given *support threshold* *min_sup* =2, <(ab)c> is a *sequential pattern*

Extracting Sequential Patterns

- Given *n* events: $i_1, i_2, i_3, \ldots, i_n$
- Candidate 1-subsequences:
  - $\langle i_1 \rangle, \langle i_2 \rangle, \langle i_3 \rangle, \ldots, \langle i_n \rangle$
- Candidate 2-subsequences:
  - $\langle (i_1, i_2) \rangle, \langle (i_1, i_3) \rangle, \ldots, \langle i_1\ i_1 \rangle, \langle i_1\ i_2 \rangle, \ldots, \langle i_{n-1}\ i_n \rangle$
- Candidate 3-subsequences:
  - $\langle (i_1, i_2, i_3) \rangle, \langle (i_1, i_2, i_4) \rangle, \ldots, \langle (i_1, i_2)\ \{i_1\} \rangle, \langle (i_1, i_2)\ i_2 \rangle, \ldots,$
  - $\langle i_1\ (i_1, i_2) \rangle, \langle i_1\ (i_1, i_3) \rangle, \ldots, \langle i_1\ i_1\ i_1 \rangle, \langle i_1\ i_1\ i_2 \rangle, \ldots$

APRIORI-like Algorithm

- Make the first pass over the sequence database to yield all the 1-element frequent sequences
- Repeat until no new frequent sequences are found
  - **Candidate Generation**:
    - Merge pairs of frequent subsequences found in the $(k-1)^{th}$ pass to generate candidate sequences that contain k items
  - **Candidate Pruning**:
    - Prune candidate k-sequences that contain infrequent (k-1)-subsequences
  - **Support Counting**:
    - Make a new pass over the sequence database to find the support for these candidate sequences
    - Eliminate candidate k-sequences whose actual support is less than *minsup*

Candidate Generation

- Base case (k=2):
  - Merging two frequent 1-sequences $<i_1>$ and $<i_2>$ will produce five candidate 2-sequences:
  - $<i_1 i_1>$, $<i_1 i_2>$, $<i_2, i_1>$, $<i_2\ i_2>$, $(<i_1\ i_2>)$
- General case (k>2):
  - A frequent (k-1) sequence s_1 is merged with another frequent (k-1) sequence s_2 to produce a candidate k-sequence if the subsequence obtained by removing the first event in s_1 is the same as the subsequence obtained by removing the last event in s_2
  - The resulting candidate after merging is given by the sequence s_1 extended with the last event of s_2. The last event will be decided as follows:
    - If the last two events in s_2 belong to the same element, then the last event in s_2 becomes part of the last element in s_1
    - Otherwise, the last event in s_2 becomes a separate element appended to the end of s_1

Candidate Generation

innovate achieve lead

First Seq (S1)	Second Seq(s2)	Candidate Seq
1 2 3	2 3 4	1 2 3 4
(1 2) 3	2 3 4	(1 2) 3 4
1 (2 3)	2 3 4	1 (2 3) 4
1 2 3	(2 3) 4	1 2 3 4
1 2 3	2 (3 4)	1 2 (3 4)
(1 2) 3	(2 3) 4	(1 2) 3 4
(1 2) 3	2 (3 4)	(1 2)(3 4)
1 (2 3)	(2 3) 4	1 (2 3) 4
1 (2 3)	**2 (3 4)**	**1 (2 3) (3 4)** ✗

Candidate Generation Examples

- Merging the sequences
  w_1=<1 (2 3) 4> and w_2 =<(2 3) (4 5)>
- will produce the candidate sequence < 1 (2 3) (4 5)> because the last two events in w_2 (4 and 5) belong to the same element

- Merging the sequences
  w_1=<1 (2 3) 4> and w_2 =<(2 3) 4 5>
- will produce the candidate sequence < 1 (2 3) 4 5> because the last two events in w_2 (4 and 5) do not belong to the same element

- Finally, the sequences <1 2 3> and <1 (2,5)> don't have to be merged (Why?)
- Because removing the first event from the first sequence doesn't give the same subsequence as removing the last event from the second sequence.

Example

innovate achieve lead

Frequent 3-sequences

< 1 2 3 >	< 1 2 3 >
	< 1 (2 5)>
	< 1 5 3 >
	< 2 3 4>
	< (2 5) 3 >
	< 3 4 5 >
	< 5 (3 4)>

Example (contd..)

innovate achieve lead

Frequent 3-sequences

< 1 (2 5) >	< 1 2 3 > ✗
	< 1 (2 5)> ✗
	< 1 5 3 > ✗
	< 2 3 4> ✗
	< (2 5) 3 > ✓
	< 3 4 5 > ✗
	< 5 (3 4)> ✗

Example (contd..)

innovate achieve lead

Frequent 3-sequences

< 1 5 3 >	< 1 2 3 >
	< 1 (2 5)>
	< 1 5 3 >
	< 2 3 4>
	< (2 5) 3 >
	< 3 4 5 >
	< 5 (3 4)>

Example (contd..)

innovate achieve lead

Frequent 3-sequences

< 1 2 3 4 >	< 1 2 3 >
	< 1 (2 5)>
	< 1 5 3 >
	< 2 3 4>
	< (2 5) 3 >
	< 3 4 5 >
	< 5 (3 4)>

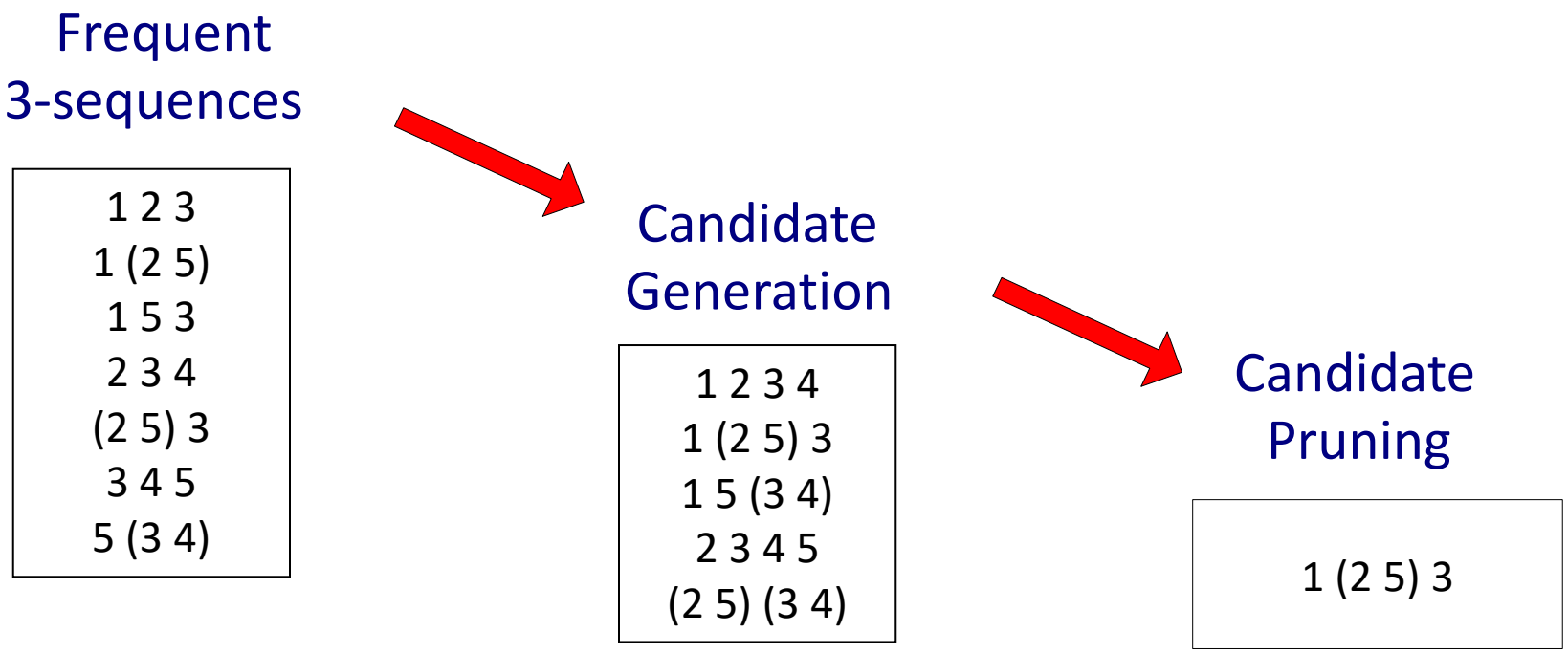
Example (contd..)

Frequent 3-sequences

< (2 5) 3 >	< 1 2 3 >
	< 1 (2 5)>
	< 1 5 3 >
	< 2 3 4>
	< (2 5) 3 >
	< 3 4 5 >
	< 5 (3 4)>

And so on for the rest of the 3 itemsets

Example (contd..)



Sequential Pattern Mining Algorithms

- Concept introduction and an initial Apriori-like algorithm
  - Agrawal & Srikant. Mining sequential patterns, ICDE'95
- Apriori-based method: GSP (Generalized Sequential Patterns: Srikant & Agrawal @ EDBT'96)
- Pattern-growth methods: FreeSpan & PrefixSpan (Han et al.@KDD'00; Pei, et al.@ICDE'01)
- Vertical format-based mining: SPADE (Zaki@Machine Leanining'00)
- Constraint-based sequential pattern mining (SPIRIT: Garofalakis, Rastogi, Shim@VLDB'99; Pei, Han, Wang @ CIKM'02)
- Mining closed sequential patterns: CloSpan (Yan, Han & Afshar @SDM'03)

The Apriori Property of Sequential Patterns

innovate achieve lead

- A basic property: Apriori (Agrawal & Sirkant'94)
  - If a sequence S is not frequent
  - Then none of the super-sequences of S is frequent
  - E.g, <hb> is infrequent → so do <hab> and <(ah)b>

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Given *support threshold* *min_sup* =2

GSP—Generalized Sequential Pattern Mining

- GSP (Generalized Sequential Pattern) mining algorithm
  - proposed by Agrawal and Srikant, EDBT'96
- Outline of the method
  - Initially, every item in DB is a candidate of length-1
  - for each level (i.e., sequences of length-k) do
    - scan database to collect support count for each candidate sequence
    - generate candidate length-(k+1) sequences from length-k frequent sequences using Apriori
  - repeat until no frequent sequence or no candidate can be found
- Major strength: Candidate pruning by Apriori

Finding Length-1 Sequential Patterns

- Examine GSP using an example
- Initial candidates: all singleton sequences
  - <a>, <b>, <c>, <d>, <e>, <f>, <g>, <h>
- Scan database once, count support for candidates

min_sup =2

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Cand	Sup
<a>	**3**
<b>	**5**
<c>	**4**
<d>	**3**
<e>	**3**
<f>	**2**
<g>	1
<h>	1

GSP: Generating Length-2 Candidates

51 length-2 Candidates

	<a>	<b>	<c>	<d>	<e>	<f>
<a>	<aa>	<ab>	<ac>	<ad>	<ae>	<af>
<b>	<ba>	<bb>	<bc>	<bd>	<be>	<bf>
<c>	<ca>	<cb>	<cc>	<cd>	<ce>	<cf>
<d>	<da>	<db>	<dc>	<dd>	<de>	<df>
<e>	<ea>	<eb>	<ec>	<ed>	<ee>	<ef>
<f>	<fa>	<fb>	<fc>	<fd>	<fe>	<ff>

	<a>	<b>	<c>	<d>	<e>	<f>
<a>		<(ab)>	<(ac)>	<(ad)>	<(ae)>	<(af)>
<b>			<(bc)>	<(bd)>	<(be)>	<(bf)>
<c>				<(cd)>	<(ce)>	<(cf)>
<d>					<(de)>	<(df)>
<e>						<(ef)>
<f>						

Without Apriori property, 8*8+8*7/2=92 candidates

Apriori prunes 44.57% candidates

The GSP Mining Process

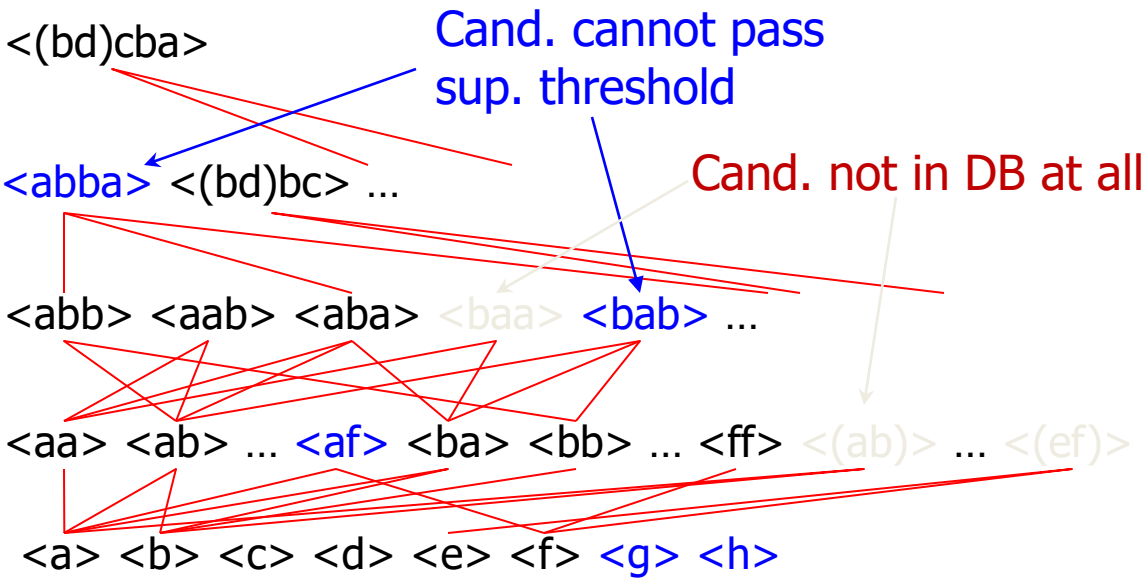
5th scan: 1 cand. 1 length-5 seq. pat.

4th scan: 8 cand. 6 length-4 seq. pat.

3rd scan: 46 cand. 19 length-3 seq. pat. 20 cand. not in DB at all

2nd scan: 51 cand. 19 length-2 seq. pat. 10 cand. not in DB at all

1st scan: 8 cand. 6 length-1 seq. pat.



min_sup =2

Seq. ID	Sequence
10	<(bd)cb(ac)>
20	<(bf)(ce)b(fg)>
30	<(ah)(bf)abf>
40	<(be)(ce)d>
50	<a(bd)bcb(ade)>

Transaction Date	Customer ID	Items Purchased
1	01	A
3	01	B
7	01	FG
9	01	C
10	01	D
1	02	B
4	02	G
6	02	D
1	03	B
5	03	F
8	03	G
9	03	AB
2	04	F
6	04	AB
8	04	C
10	04	D
3	05	A
4	05	BC
7	05	G
9	05	F
10	05	DE

Item	Support
A	4
B	5
C	3
D	4
~~E~~	~~1~~
F	4
G	4

	A	B	C	D	F	G
A	AA	AB	AC	AD	AF	AG
B	BA	BB	BC	BD	BF	BG
C	CA	CB	CC	CD	CF	CG
D	DA	DB	DC	DD	DF	DG
F	FA	FB	FC	FD	FF	FG
G	GA	GB	GC	GD	GF	GG

	A	B	C	D	F	G
A		(AB)	(AC)	(AD)	(AF)	(AG)
B			(BC)	(BD)	(BF)	(BG)
C				(CD)	(CF)	(CG)
D					(DF)	(DG)
F						(FG)
G						

achieve lead

~~AA~~	~~0~~	AB	2	AC	3	AD	3	AF	2	AG	2
~~BA~~	~~1~~	~~BB~~	~~1~~	BC	2	BD	4	BF	3	BG	4
~~CA~~	~~0~~	~~CB~~	~~0~~	~~CC~~	~~0~~	CD	3	~~CF~~	~~1~~	~~CG~~	~~1~~
~~DA~~	~~0~~	~~DB~~	~~0~~	~~DC~~	~~0~~	~~DD~~	~~0~~	~~DF~~	~~0~~	~~DG~~	~~0~~
FA	2	FB	2	FC	2	FD	3	~~FF~~	~~0~~	~~FG~~	~~1~~
~~GA~~	~~1~~	~~GB~~	~~1~~	~~GC~~	~~1~~	GD	3	~~GF~~	~~1~~	~~GG~~	~~0~~

AB	AC	AD	AF	AG	BC	BD	BF	BG	CD	FA	FB	FC	FD	GD	(AB)

AB	AC	AD	AF	AG	BC	BD	BF	BG	CD	FA	FB	FC	FD	GD	(AB)

2-seq. (1)	2-seq. -1st	2-seq. (2)	2-seq. -Last	3-seq after join	3-seq. after prune	Support Count	3-seq. Supported
AB	B	BC	B	ABC	ABC	1	
AB	B	BD	B	ABD	ABD	2	ABD
AB	B	BF	B	ABF	ABF	2	ABF
AB	B	BG	B	ABG	ABG	2	ABG
AB	B	(AB)	B	A(AB)			
AC	C	CD	C	ACD	ACD	3	ACD
AF	F	FA	F	AFA			
AF	F	FB	F	AFB	AFB	0	
AF	F	FC	F	AFC	AFC	1	
AF	F	FD	F	AFD	AFD	2	AFD
AG	G	GD	G	AGD	AGD	2	AGD
BC	C	CD	C	BCD	BCD	2	BCD
BF	F	FA	F	BFA			
BF	F	FB	F	BFB			
BF	F	FC	F	BFC	BFC	1	
BF	F	FD	F	BFD	BFD	2	BFD
BG	G	GD	G	BGD	BGD	3	BGD
FA	A	AB	A	FAB	FAB	0	
FA	A	AC	A	FAC	FAC	1	
FA	A	AD	A	FAD	FAD	1	
FA	A	AF	A	FAF			
FA	A	AG	A	FAG			
FA	A	(AB)	A	F(AB)	F(AB)	2	F(AB)
FB	B	BC	B	FBC	FBC	1	
FB	B	BD	B	FBD	FBD	1	
FB	B	BF	B	FBF			
FB	B	BG	B	FBG			
FC	C	CD	C	FCD	FCD	2	FCD
(AB)	B	BC	B	(AB)C	(AB)C	1	
(AB)	B	BD	B	(AB)D	(AB)D	1	
(AB)	B	BF	B	(AB)F	(AB)F	0	
(AB)	B	BG	B	(AB)G	(AB)G	0	
(AB)	A	AB	A	(AB)B			

3-seq. (1)	3-seq. -1st	3-seq. (2)	3-seq. -Last	4-seq. after join	4-seq. after prune	Support Count	4-seq. Supported
ABF	BF	BFD	BF	ABFD	ABFD	2	ABFD
ABG	BG	BGD	BG	ABGD	ABGD	2	ABGD

Sequences

1-Item	2-Items	3-Items	4-Items
A	AB	ABD	ABFD
B	AC	ABF	ABGD
C	AD	ABG	
D	AF	ACD	
F	AG	AFD	
G	BC	AGD	
	BD	BCD	
	BF	BFD	
	BG	BGD	
	CD	F(AB)	
	FA	FCD	
	FB		
	FC		
	FD		
	GD		
	(AB)		

Candidate Generate-and-test: Drawbacks

- A huge set of candidate sequences generated
  - Especially 2-item candidate sequence
- Multiple Scans of database needed
  - The length of each candidate grows by one at each database scan
- Inefficient for mining long sequential patterns
  - A long pattern grow up from short patterns
  - The number of short patterns is exponential to the length of mined patterns

The SPADE Algorithm

- SPADE (Sequential PAttern Discovery using Equivalent Class) developed by Zaki 2001
- A vertical format sequential pattern mining method
- A sequence database is mapped to a large set of
  - Item: <SID, EID>
- Sequential pattern mining is performed by
  - growing the subsequences (patterns) one item at a time by Apriori candidate generation

The SPADE Algorithm

SID	Sequence
1	<a(abc)(ac)d(cf)>
2	<(ad)c(bc)(ae)>
3	<(ef)(ab)(df)cb>
4	<eg(af)cbc>

Min Support=2

SID	EID	Items
1	1	a
1	2	abc
1	3	ac
1	4	d
1	5	cf
2	1	ad
2	2	c
2	3	bc
2	4	ae
3	1	ef
3	2	ab
3	3	df
3	4	c
3	5	b
4	1	e
4	2	g
4	3	af
4	4	c
4	5	b
4	6	c

a		b		...
SID	EID	SID	EID	...
1	1	1	2	
1	2	2	3	
1	3	3	2	
2	1	3	5	
2	4	4	5	
3	2			
4	3			

ab			ba			...
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)	...
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

aba				...
SID	EID (a)	EID(b)	EID(a)	...
1	1	2	3	
2	1	3	4	

Timing Constraints

Buyer A: < {TV} … {DVD Player} >

Buyer B: < {TV} … {DVD Player} >

…

The sequential pattern of interest is

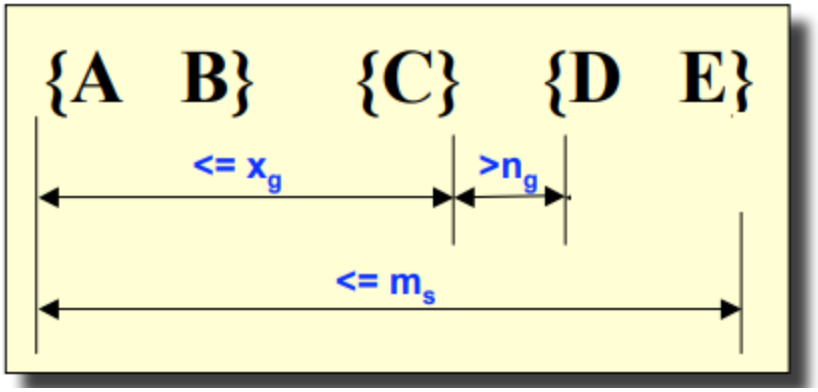
<{TV}{DVD Player}>

which suggests that people who buy TV will also soon buy DVD player.

A person who bought a TV ten years earlier should not be considered as supporting the pattern because the time gap between the purchases is too long.

Timing Constraints

- We consider two kinds of constraints:
- Max-span constraint (m_s): maximum allowed time between the first element and the last element in the sequence
- Max-gap constraint (x_g): maximum length of a gap between two consecutive element

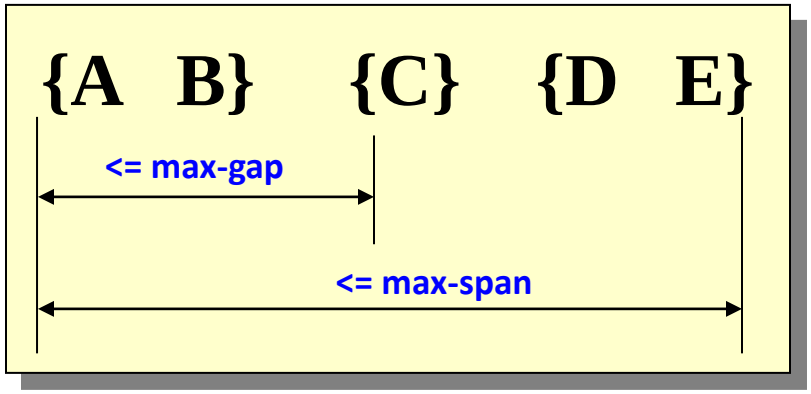


x_g: max-gap

n_g: min-gap

m_s: maximum span

Timing Constraints



max-gap = 2, max-span= 4

Data sequence	Subsequence	Contained?
<{2,4} {3,5,6} {4,7} {4,5} {8}>	< {6} {5} >	Yes
<{1} {2} {3} {4} {5}>	< {1} {4} >	No (max-gap =3)
<{1} {2,3} {3,4} {4,5}>	< {2} {3} {5} >	Yes
<{1,2} {3} {2,3} {3,4} {2,4} {4,5}>	< {1,2} {5} >	No(maximum span and max-gap=5)

Mining Sequential Patterns with Timing Constraints

Approach 1:

- Mine sequential patterns without timing constraints
- Postprocess the discovered patterns

Approach 2:

- Modify algorithm to directly prune candidates that violate timing constraints
- Question:
  - Does APRIORI principle still hold?

Apriori Principle for Sequence Data

innovate achieve lead

Object	Timestamp	Events
A	1	1,2,4
A	2	2,3
A	3	5
B	1	1,2
B	2	2,3,4
C	1	1, 2
C	2	2,3,4
C	3	2,4,5
D	1	2
D	2	3, 4
D	3	4, 5
E	1	1, 3
E	2	2, 4, 5

Suppose:

x_g = 1 (max-gap)

m_s = 5 (maximum span)

minsup = 60%

<{2} {5}> support = 40%

but

<{2} {3} {5}> support = 60%

Problem exists because of max-gap constraint!

Contiguous Subsequences

- s is a contiguous subsequence of

  $w = \langle e_1 \rangle \langle e_2 \rangle \ldots \langle e_k \rangle$

  if any of the following conditions hold:

  1. s is obtained from w by deleting an item from either e_1 or e_k
  2. s is obtained from w by deleting an item from any element e_i that contains more than 2 items
  3. s is a contiguous subsequence of s' and s' is a contiguous subsequence of w (recursive definition)

Data Sequence, s	Sequential Pattern, t	Is t a contiguous subsequence of s?
<{1} {2,3}>	< {1} {2} >	Yes
<{1,2} {2} {3} >	< {1} {2} >	Yes
<{3,4} {1,2} {2,3} {4} >	< {1} {2} >	Yes
<{1} {3} {2} >	< {1} {2} >	No
<{1,2} {1} {3} {2} >	< {1} {2} >	No

Modified Candidate Pruning Step

- Without maxgap constraint: – A candidate k-sequence is pruned if at least one of its (k-1)-subsequences is infrequent.
- With maxgap constraint: – A candidate k-sequence is pruned if at least one of its contiguous (k-1)-subsequences is infrequent.

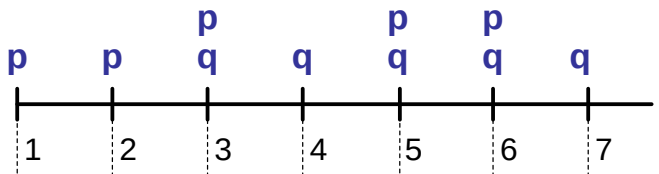
Modified Sequential Apriori for timing constraints

- Modified Apriori principle: If a k-sequence is frequent, then all of its contiguous k-1-subsequences are frequent
- Modified algorithm: Candidate generation step remains the same: we merge two frequent k-1 sequences that have the same middle part (excluding first and last event)
- In Candidate pruning, we only need to verify contiguous k-1-sequences
- - e.g. Given 5-sequence: 1(23)45 we need to verify 1245,1345, and need not to verify 1(23)5
- In support counting need to check that maxspan constraint is not

Support of a sequential pattern

- Support of a sequential pattern is not as clear cut as itemset support, due to the repetition of the items in the data sequence
- Many choices, two most important are
  - COBJ: At least one occurrence of a given sequence in an object's timeline. Even though the sequence occurs many time in the object's timeline it is counted only once.
  - CWIN: One occurrence per sliding window.
  - A sliding time window of fixed length (maxspan) is moved across an object's timeline, one unit at a time. The support is incremented each time the sequence is encountered in the sliding window.
  - CMINWIN: Number of minimal windows of occurrence.
  - CDIST_O: Distinct occurrences with possibility of event-timestamp overlap.
  - CDIST:Distinct occurrences with no event-timestamp overlap allowed.

Object's Timeline



Sequence: (p) (q)

Method	Support Count
COBJ	1
CWIN	6
CMINWIN	4
CDIST_O	8
CDIST	5

Assume:

x_g = 2 (max-gap)

n_g = 0 (min-gap)

ws = 0 (window size)

m_s = 2 (maximum span)

Take home message

- The Sequential pattern mining is to find the subsequences which satisfy minimum support.
- Modified Apriori is used to extract the frequent subsequences
- GSP and SPADE are the basic algorithms for sequence mining
- Window constarints can be used to mine some specific patterns which are time based.