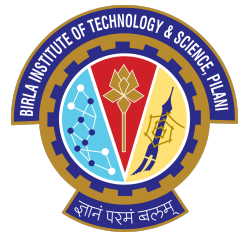
SS-ZG548: ADVANCED DATA MINING

Lecture-13: Clustering on Data Stream, Big Data



Dr. Kamlesh Tiwari
Assistant Professor
Department of Computer Science and Information Systems Engineering,
BITS Pilani, Rajasthan-333031 INDIA

March 11, 2018 (WILP @ BITS-Pilani Jan-Apr 2018)

Recap: Big Data

We are in the era of Big Data

- Daily $\sim 2.5 \times 10^{18}$ bytes, mostly unstructured (80%). 90% generated in last 2 year
- 3V's (Volume, Variety, Velocity) and more (Veracity: noise and error, Value and Viability)
- **Distributed computing** is required.
- One need to deal with issues of synchronization, deadlocks, data dependency, mutual exclusion, replication, reliability, scalability *etc*.
- Use **MapReduce/Hadoop** or **spark**

Recap: PK-Means [1]

Involves 1) Seed selection, 2) Assignment, 3) Centroid computation

Map

- With array of centers, computes closest center for each sample
- Intermediate values/output: $< key, value >$ (Key: index of the closed centre, value: sample)

Combiner

- Combines intermediate data of each map task and stores locally
- Partial sum the values assigned to the same cluster
  - Record number of samples in each cluster and
  - Sum of values at each dimension
- Key: key & Value: string of num and sums

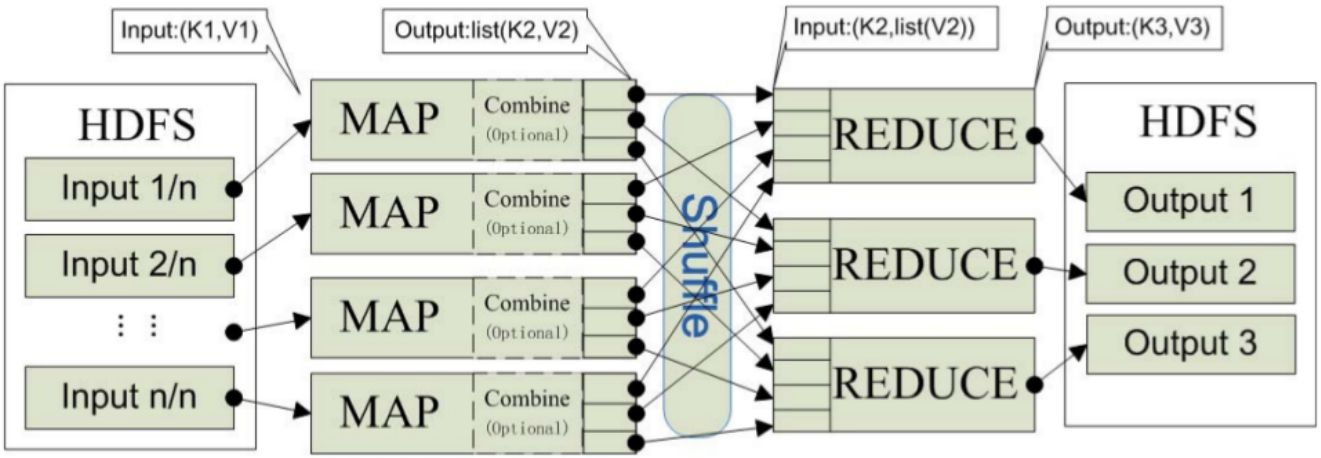
Reducer

- With output of combiner, computes new centers

[1] Du, Zhihua and Wang, Yiwei and Ji, Zhen, "PK-means: A new algorithm for gene clustering" in Computational Biology and Chemistry, pages=243–247, vol 32(4), Elsevier 2008

MR-DBSCAN

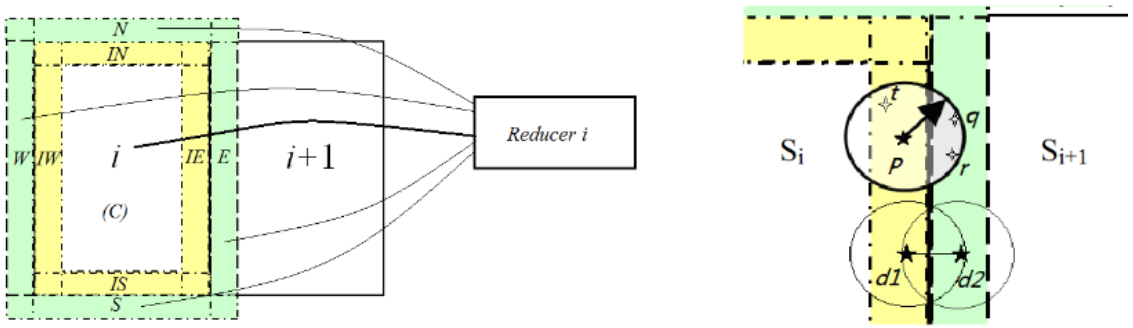
- MR-DBSCAN [2] involves following steps
  1. Preprocessing
  2. Local DBSCAN
  3. Find Merging Mapping
- Uses quadtree, a spacial data structure
- Extended regions (ϵ-extended) is taken in partition



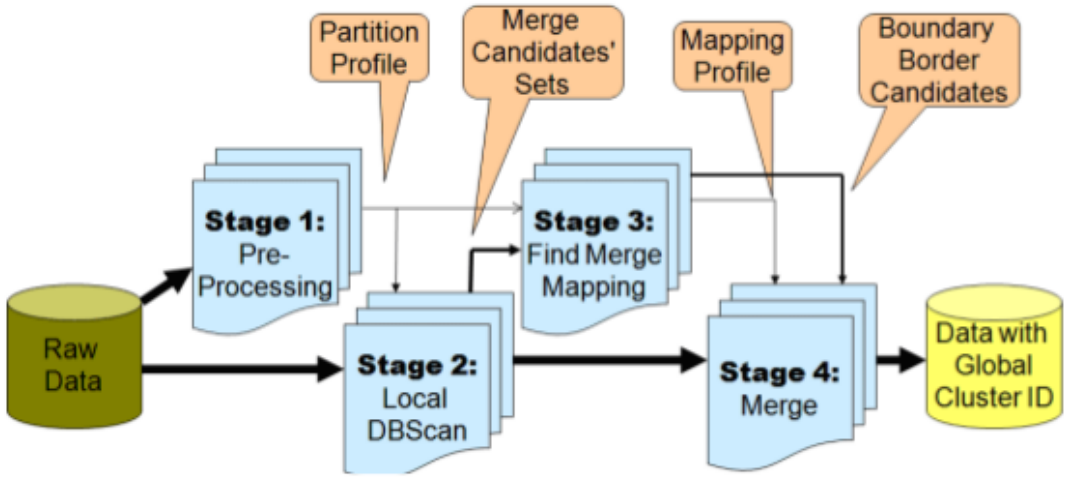
[2] MR-DBSCAN: An Efficient Parallel Density-based Clustering Algorithm using MapReduce He, Yaobin et.al. Parallel and Distributed Systems (ICPADS) 473–480, IEEE 2011

MR-DBSCAN

- Extended regions (ϵ-extended) is taken in partition

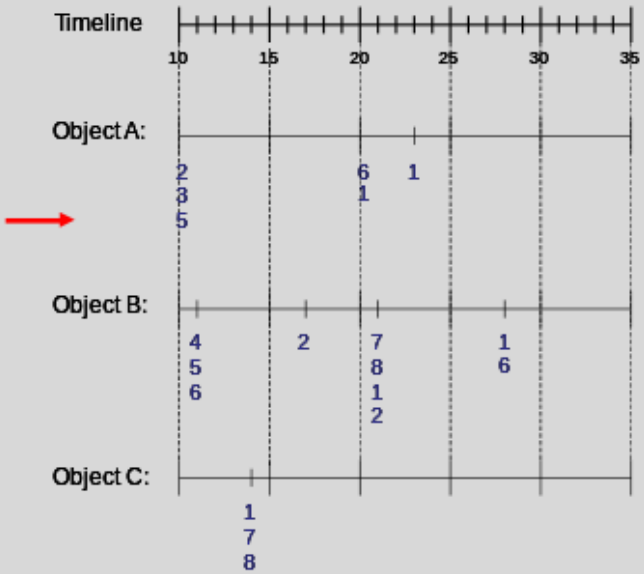


- Cross connection files are processed during reduce
- This makes the algorithm data parallel



Sequence Data

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



- Ordered list of transactions $S =< e_1, e_2, e_3, ... >$
- Each element is a list of events $e_i = \{i_1, i_2, ..., i_k\}$
- Element is attributed to a specific time
- Length of sequence S is $|S|$ and is number of elements of sequence
- A k-sequence contains k elements

Sequence Data Example

Database	Sequence	Element (Trxn)	Event (Item)
Customer	Purchase history of a given customer	A set of items bought by a customer at time t	Books, diary products, etc
Web Data	Browsing activity of a particular Web visitor	List of files viewed by a Web visitor after a single 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

Subsequence

- A sequence $< a_1, a_2, ..., a_n >$ is contained in another sequence $< b_1, b_2, ...b_m >$ where $m \geq n$ if there exists integer $i_1 < i_2 < i_3 < ... < i_n$ such that $a_1 \subseteq b_{i_1}$, $a_2 \subseteq b_{i_2}$, $a_3 \subseteq b_{i_3}$, ... , $a_n \subseteq b_{i_n}$

Data sequence	Subsequence	Contain?
< {2,4} {3,5,6} {8} >	< {2} {3,5} >	Yes
< {1,2} {3,4} >	< {1} {2} >	No
< {2,4} {2,4} {2,5} >	< {2} {4} >	Yes
<{1,2}{4,5}{2,5,8}>	<{2} {8}>	Yes

- The support of a sequence W is defined as the fraction of all data sequence that contain W
- A sequential pattern is a frequent sequence (i.e., a sequence whose support is grater than or equal to *minsup*)

Sequential Pattern Mining

Given a set of sequences and minsup; report all subsequence with support grater then minsup

Consider sequences $A =< \{1,2,4\},\{2,3\},\{5\} >$
$B =< \{1,2\},\{2,3,4\} >$ $C =< \{1,2\},\{2,3,4\},\{2,4,5\} >$
$D =< \{2\},\{3,4\},\{4,5\} >$ $E =< \{1,3\},\{2,4,5\} >$

Sequence	Support
$< \{1,2\} >$	60%
$< \{2,4\} >$	80%
$< \{1\},\{2\} >$	80%
$< \{1,2\},\{2,3\} >$	60%

For n events $i_1, i_2, ..., i_n$.
1-sequence: $< \{i_1\} >, < \{i_2\} >, < \{i_n\} >$
2-sequence: $< \{i_k, i_j\} >, < \{i_k\},\{i_j\} >$
3-seq $< \{i_k, i_j, i_l\} >, < \{i_k\},\{i_j, i_l\} >, < \{i_k, i_j\},\{i_l\} >, < \{i_k\},\{i_j\},\{i_l\} >$

Generalized Sequential Pattern (GSP)[3]

S-1 Make first pass over the sequence database to yield all 1-element frequent sequences

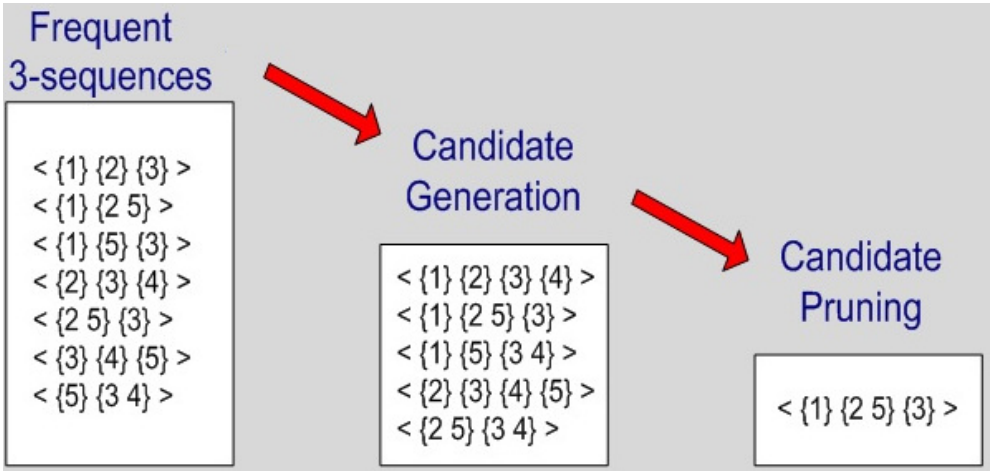
S-2 Repeat until new frequent sequences are found

- **Candidate Generation:** merge pairs found in $k-1^{th}$ pass
  - ⋆ $<\{i_1\}>$ and $<\{i_2\}>$ yields $<\{i_1, i_2\}>$ and $<\{i_1\},\{i_2\}>$
  - ⋆ w_1 and w_2 can be merged if subsequences obtained by removal of first element of w_1 and last element of w_2 are same
- **Candidate Pruning:** Prune candidates containing infrequent $k-1$ subsequeces
- **Support Counting:** Make new pass to find support for new candidates
- **Candidate Elimination:** Eliminate candidate k-sequences whose support is less than minsup

$<\{1\},\{2,3\}\{4\}>$ and $<\{2,3\},\{4,5\}>$	$<\{1\},\{2,3\},\{4,5\}>$
$<\{1\},\{2,3\}\{4\}>$ and $<\{2,3\},\{4\},\{5\}>$	$<\{1\},\{2,3\},\{4\},\{5\}>$
$<\{1\},\{2,6\}\{4\}>$ and $<\{1\},\{2,6\},\{4\}>$	Can not be merged

[3] Generalized Sequential Pattern (GSP), Srikant and Agrawal, In EDBT 1996

GSP: Candidate Generation



Issues:

- Huge number of candidate sets. n frequent 1-length candidate would generate $n^2 + \frac{n*(n-1)}{2}$ 2-length candidate
- Multiple scans of the database
- Mining n-length sequential patterns need $\sum_{i=1}^{n} {}^{n}C_i = 2^n - 1$ number of short candidates. It is exponential

One can use prefix projections approach similar to FP-Growth

Generalized Sequential Pattern (GSP)[4]

S-1 Make first pass over the sequence database to yield all 1-element frequent sequences

S-2 Repeat until new frequent sequences are found

- **Candidate Generation:** merge pairs found in $k-1^{th}$ pass
  - $< \{i_1\} >$ and $< \{i_2\} >$ yields $< \{i_1, i_2\} >$ and $< \{i_1\}, \{i_2\} >$
  - w_1 and w_2 can be merged if subsequences obtained by removal of first element of w_1 and last element of w_2 are same
- **Candidate Pruning:** Prune candidates containing infrequent $k-1$ subsequeces
- **Support Counting:** Make new pass to find support for new candidates
- **Candidate Elimination:** Eliminate candidate k-sequences whose support is less than minsup

$< \{1\}, \{2,3\}\{4\} >$ and $< \{2,3\}, \{4,5\} >$	$< \{1\}, \{2,3\}, \{4,5\} >$
$< \{1\}, \{2,3\}\{4\} >$ and $< \{2,3\}, \{4\}, \{5\} >$	$< \{1\}, \{2,3\}, \{4\}, \{5\} >$
$< \{1\}, \{2,6\}\{4\} >$ and $< \{1\}, \{2,6\}, \{4\} >$	Can not be merged

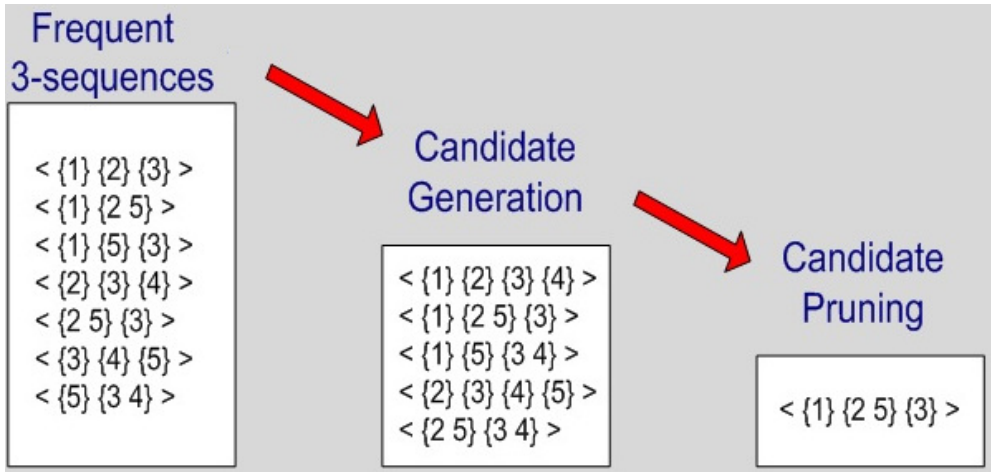
[4] Generalized Sequential Pattern (GSP), Srikant and Agrawal, In EDBT 1996

Recap: Pruning in GSP

S-1 First pass to yield all 1-element frequent sequences

S-2 Repeat until new frequent sequences are found

- **Candidate Generation:** merge pairs found in $k-1^{th}$ pass. w_1 and w_2 can be merged if subsequences obtained by removal of first element of w_1 and last element of w_2 are same
- **Candidate Pruning:** Prune candidates that contain a subsequence which is infrequent in $k-1$ subsequeces
- **Support Counting:** Need new pass to database
- **Candidate Elimination:** Involves thresholding based on minsup



Candidate Generation

Frequent 3-sequences

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

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

Candidate Generation

Frequent 3-sequences

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

Candidate Generation

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

Candidate Pruning

Candidate Generation

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

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

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

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

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

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

Candidate Pruning

Candidate Generation

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

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

Candidate Pruning

Candidate Generation

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

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

Frequent 3-sequences

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

Candidate Pruning

Candidate Generation

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

< {1} {2} {3} {4} >	< {1} {2} {3} {4} >	< {2} {3} {4} > ✓
	< {1} {2} {3} {4} >	< {1} {3} {4} > ✗
	< {1} {2} {3} {4} >	< {1} {2} {4} >
	< {1} {2} {3} {4} >	< {1} {2} {3} >
< {1} {2 5} {3} >	< {1} {2 5} {3} >	< {2 5} {3} >
	< {1} {2 5} {3} >	< {1} {5} {3} >
	< {1} {2 5} {3} >	< {1} {2} {3} >
	< {1} {2 5} {3} >	< {1} {2 5} >
< {1} {5} {3 4} >	< {1} {5} {3 4} >	< {5} {3 4} >
	< {1} {5} {3 4} >	< {1} {3 4} >
	< {1} {5} {3 4} >	< {1} {5} {4} >
	< {1} {5} {3 4} >	< {1} {5} {3} >
< {2} {3} {4} {5} >	< {2} {3} {4} {5} >	< {3} {4} {5} >
	< {2} {3} {4} {5} >	< {2} {4} {5} >
	< {2} {3} {4} {5} >	< {2} {3} {5} >
	< {2} {3} {4} {5} >	< {2} {3} {4} >
< {2 5} {3 4} >	< {2 5} {3 4} >	< {5} {3 4} >
	< {2 5} {3 4} >	< {2} {3 4} >
	< {2 5} {3 4} >	< {2 5} {4} >
	< {2 5} {3 4} >	< {2 5} {3} >

Frequent 3-sequences

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

Candidate Pruning

Candidate Generation

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

< {1} {2} {3} {4} >	< {1} {2} {3} {4} >	< {2} {3} {4} > ✓
	< {1} {2} {3} {4} >	< {1} {3} {4} > ✗
	< {1} {2} {3} {4} >	< {1} {2} {4} > ✗
	< {1} {2} {3} {4} >	< {1} {2} {3} >
< {1} {2 5} {3} >	< {1} {2 5} {3} >	< {2 5} {3} >
	< {1} {2 5} {3} >	< {1} {5} {3} >
	< {1} {2 5} {3} >	< {1} {2} {3} >
	< {1} {2 5} {3} >	< {1} {2 5} >
< {1} {5} {3 4} >	< {1} {5} {3 4} >	< {5} {3 4} >
	< {1} {5} {3 4} >	< {1} {3 4} >
	< {1} {5} {3 4} >	< {1} {5} {4} >
	< {1} {5} {3 4} >	< {1} {5} {3} >
< {2} {3} {4} {5} >	< {2} {3} {4} {5} >	< {3} {4} {5} >
	< {2} {3} {4} {5} >	< {2} {4} {5} >
	< {2} {3} {4} {5} >	< {2} {3} {5} >
	< {2} {3} {4} {5} >	< {2} {3} {4} >
< {2 5} {3 4} >	< {2 5} {3 4} >	< {5} {3 4} >
	< {2 5} {3 4} >	< {2} {3 4} >
	< {2 5} {3 4} >	< {2 5} {4} >
	< {2 5} {3 4} >	< {2 5} {3} >

Frequent 3-sequences

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

Candidate Pruning

< {1} {2} {3} {4} >	< {1} {2} {3} {4} >	< {2} {3} {4} > ✓
	< {1} {2} {3} {4} >	< {1} {3} {4} > ✗
	< {1} {2} {3} {4} >	< {1} {2} {4} > ✗
	< {1} {2} {3} {4} >	< {1} {2} {3} > ✓
< {1} {2 5} {3} >	< {1} {2 5} {3} >	< {2 5} {3} >
	< {1} {2 5} {3} >	< {1} {5} {3} >
	< {1} {2 5} {3} >	< {1} {2} {3} >
	< {1} {2 5} {3} >	< {1} {2 5} >
< {1} {5} {3 4} >	< {1} {5} {3 4} >	< {5} {3 4} >
	< {1} {5} {3 4} >	< {1} {3 4} >
	< {1} {5} {3 4} >	< {1} {5} {4} >
	< {1} {5} {3 4} >	< {1} {5} {3} >
< {2} {3} {4} {5} >	< {2} {3} {4} {5} >	< {3} {4} {5} >
	< {2} {3} {4} {5} >	< {2} {4} {5} >
	< {2} {3} {4} {5} >	< {2} {3} {5} >
	< {2} {3} {4} {5} >	< {2} {3} {4} >
< {2 5} {3 4} >	< {2 5} {3 4} >	< {5} {3 4} >
	< {2 5} {3 4} >	< {2} {3 4} >
	< {2 5} {3 4} >	< {2 5} { 4} >
	< {2 5} {3 4} >	< {2 5} {3} >

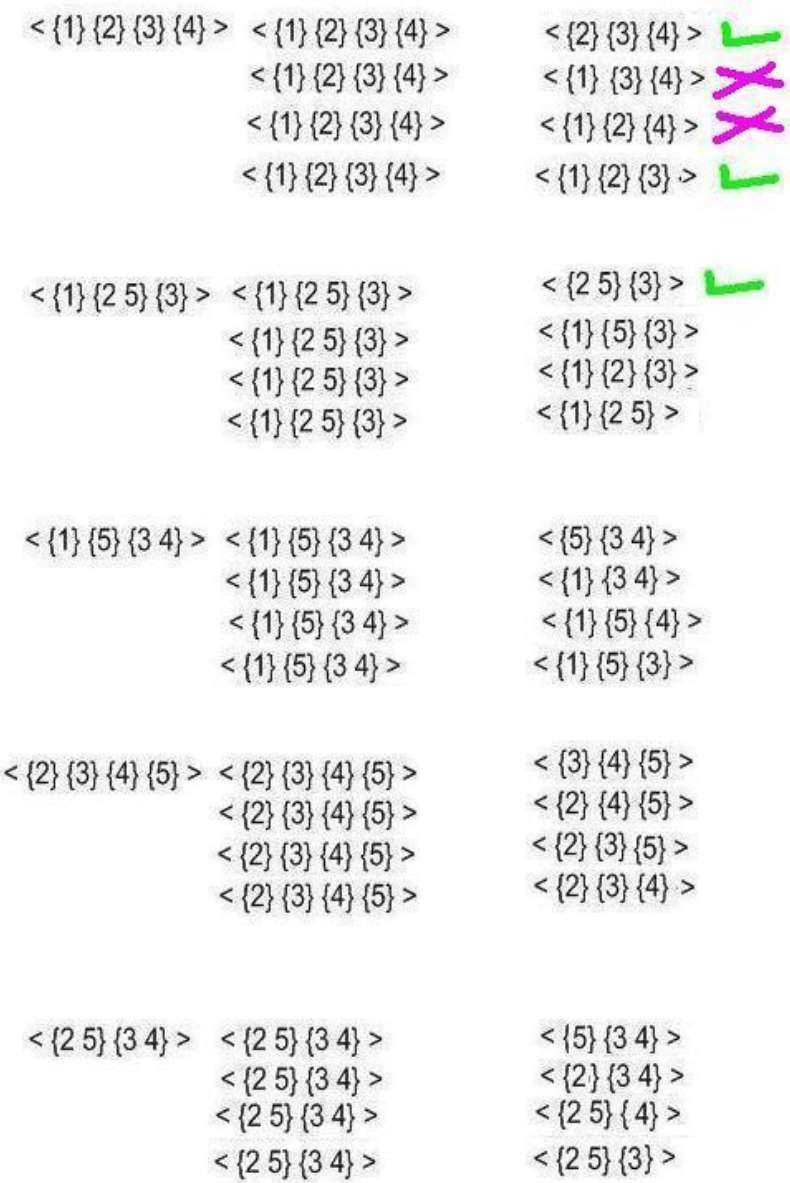
Candidate Generation

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

Frequent 3-sequences

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

Candidate Pruning



Candidate Pruning

Candidate Generation

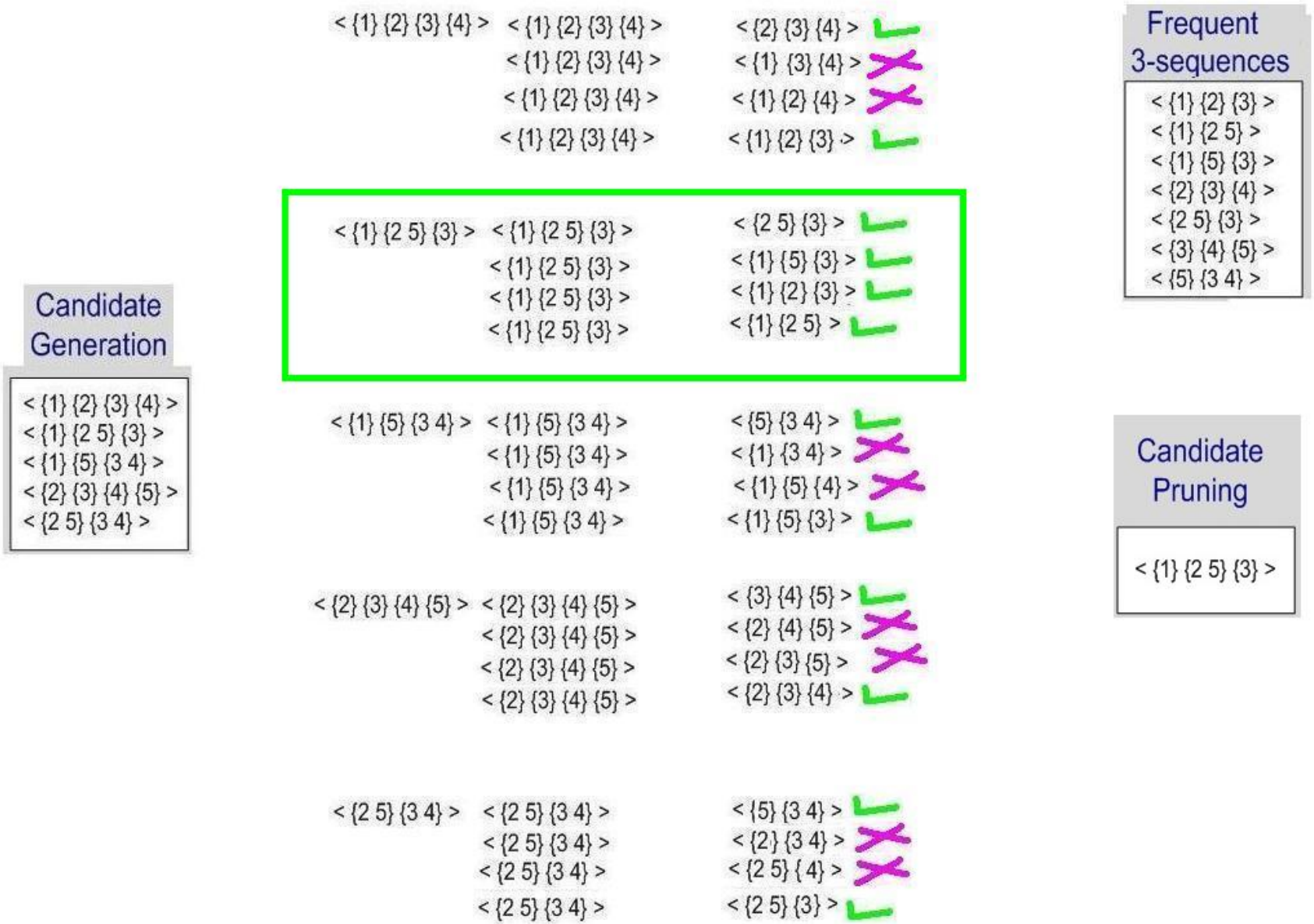
< {1} {2} {3} {4} >
< {1} {2 5} {3} >
< {1} {5} {3 4} >
< {2} {3} {4} {5} >
< {2 5} {3 4} >

< {1} {2} {3} {4} >	< {1} {2} {3} {4} >	< {2} {3} {4} > ✓
	< {1} {2} {3} {4} >	< {1} {3} {4} > ✗
	< {1} {2} {3} {4} >	< {1} {2} {4} > ✗
	< {1} {2} {3} {4} >	< {1} {2} {3} > ✓
< {1} {2 5} {3} >	< {1} {2 5} {3} >	< {2 5} {3} > ✓
	< {1} {2 5} {3} >	< {1} {5} {3} > ✓
	< {1} {2 5} {3} >	< {1} {2} {3} > ✓
	< {1} {2 5} {3} >	< {1} {2 5} > ✓
< {1} {5} {3 4} >	< {1} {5} {3 4} >	< {5} {3 4} > ✓
	< {1} {5} {3 4} >	< {1} {3 4} > ✗
	< {1} {5} {3 4} >	< {1} {5} {4} > ✗
	< {1} {5} {3 4} >	< {1} {5} {3} > ✓
< {2} {3} {4} {5} >	< {2} {3} {4} {5} >	< {3} {4} {5} > ✓
	< {2} {3} {4} {5} >	< {2} {4} {5} > ✗
	< {2} {3} {4} {5} >	< {2} {3} {5} > ✗
	< {2} {3} {4} {5} >	< {2} {3} {4} > ✓
< {2 5} {3 4} >	< {2 5} {3 4} >	< {5} {3 4} > ✓
	< {2 5} {3 4} >	< {2} {3 4} > ✗
	< {2 5} {3 4} >	< {2 5} {4} > ✗
	< {2 5} {3 4} >	< {2 5} {3} > ✓

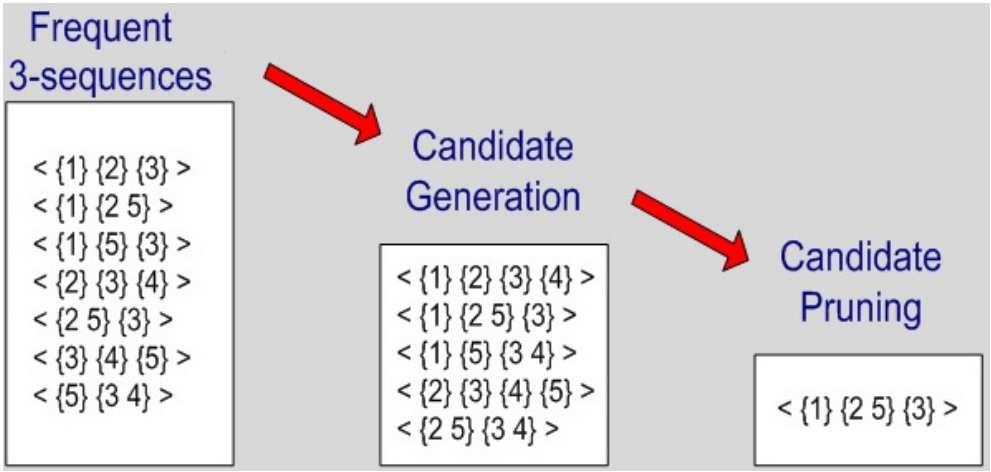
Frequent 3-sequences

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

Candidate Pruning



GSP: Candidate Generation



Issues:

- Huge number of candidate sets. n frequent 1-length candidate could generate $n^2 + \frac{n*(n-1)}{2}$ 2-length candidate
- Multiple scans of the database
- Mining n-length sequential patterns need $\sum_{i=1}^{n} {}^{n}C_i = 2^n - 1$ number of short candidates. It is exponential

One can use prefix projections approach similar to FP-Growth

Thank You!

Thank you very much for your attention!

Queries ?