FREQUENT PATTERN MINING—ADVANCED

Hari Sundaram

hs1@illinois.edu

http://sundaram.cs.illinois.edu

adapted from slides by Jiawei Han and Kevin Chang

Multi-Level

Constraint Based

High Dimensional Data, Colossal Patterns

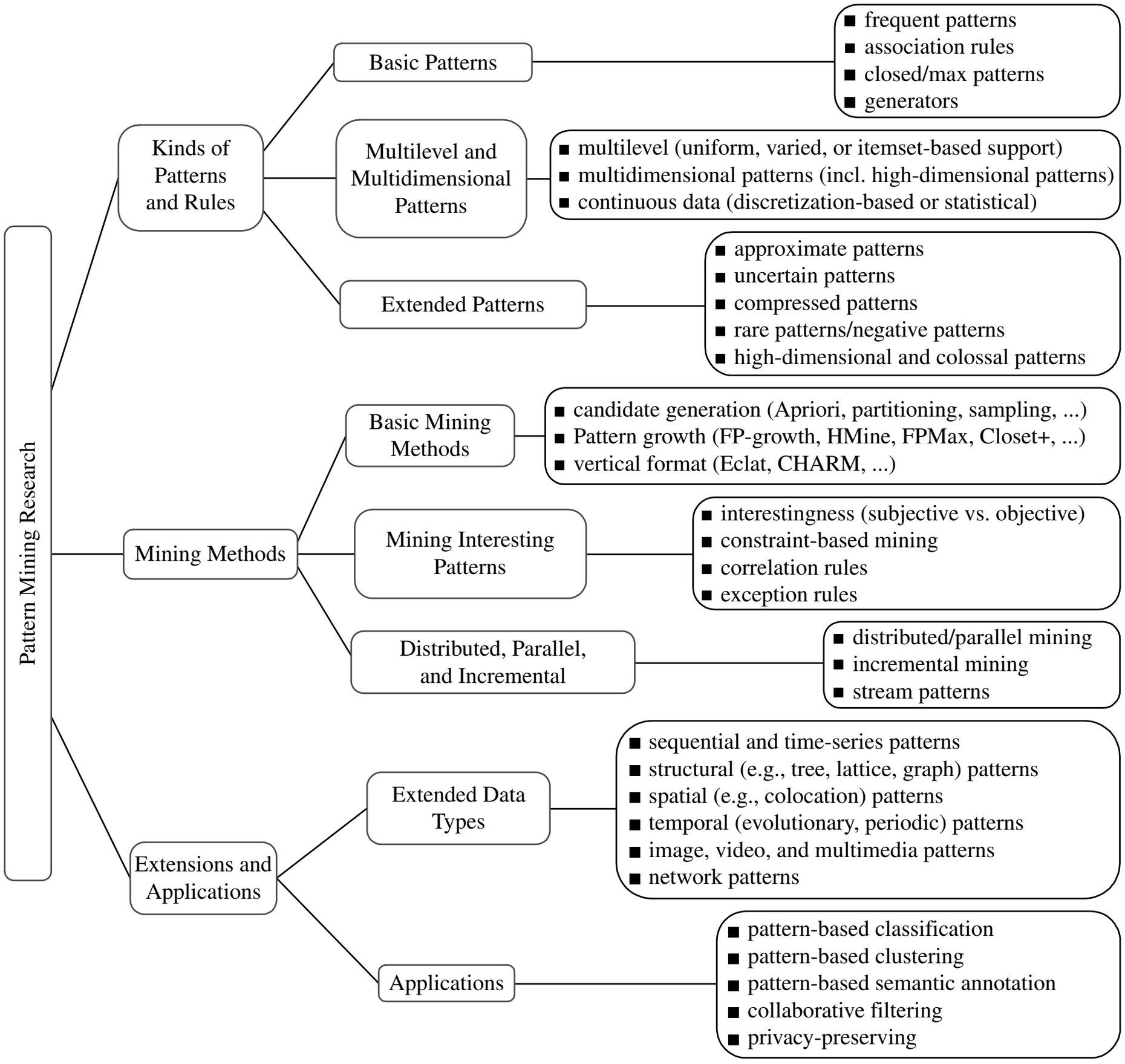
Approximate Patterns

Sequential Patterns

Graph Patterns

Summary

A ROADMAP


Pattern Mining Research
Kinds of Patterns and Rules
Basic Patterns
frequent patterns
association rules
closed/max patterns
generators
Multilevel and Multidimensional Patterns
multilevel (uniform, varied, or itemset-based support)
multidimensional patterns (incl. high-dimensional patterns)
continuous data (discretization-based or statistical)
Extended Patterns
approximate patterns
uncertain patterns
compressed patterns
rare patterns/negative patterns
high-dimensional and colossal patterns
Mining Methods
Basic Mining Methods
candidate generation (Apriori, partitioning, sampling, ...)
Pattern growth (FP-growth, HMine, FPMax, Closet+, ...)
vertical format (Eclat, CHARM, ...)
Mining Interesting Patterns
interestingness (subjective vs. objective)
constraint-based mining
correlation rules
exception rules
Distributed, Parallel, and Incremental
distributed/parallel mining
incremental mining
stream patterns
Extensions and Applications
Extended Data Types
sequential and time-series patterns
structural (e.g., tree, lattice, graph) patterns
spatial (e.g., colocation) patterns
temporal (evolutionary, periodic) patterns
image, video, and multimedia patterns
network patterns
Applications
pattern-based classification
pattern-based clustering
pattern-based semantic annotation
collaborative filtering
privacy-preserving

Multi-Level Association

Multi-Dimensional Association

Quantitative Association Rules

Mining Rare Patterns and Negative Patterns

MULTI-LEVEL

Road Map Constraint Based

High Dimensional Data, Colossal Patterns Approximate Patterns

Sequential Patterns Graph Patterns Summary

Table 7.1 Task-Relevant Data, D

always at the lowest level of the concept hierarchy

TID	*Items Purchased*
T100	Apple 17″ MacBook Pro Notebook, HP Photosmart Pro b9180
T200	Microsoft Office Professional 2010, Microsoft Wireless Optical Mouse 5000
T300	Logitech VX Nano Cordless Laser Mouse, Fellowes GEL Wrist Rest
T400	Dell Studio XPS 16 Notebook, Canon PowerShot SD1400
T500	Lenovo ThinkPad X200 Tablet PC, Symantec Norton Antivirus 2010
…	…

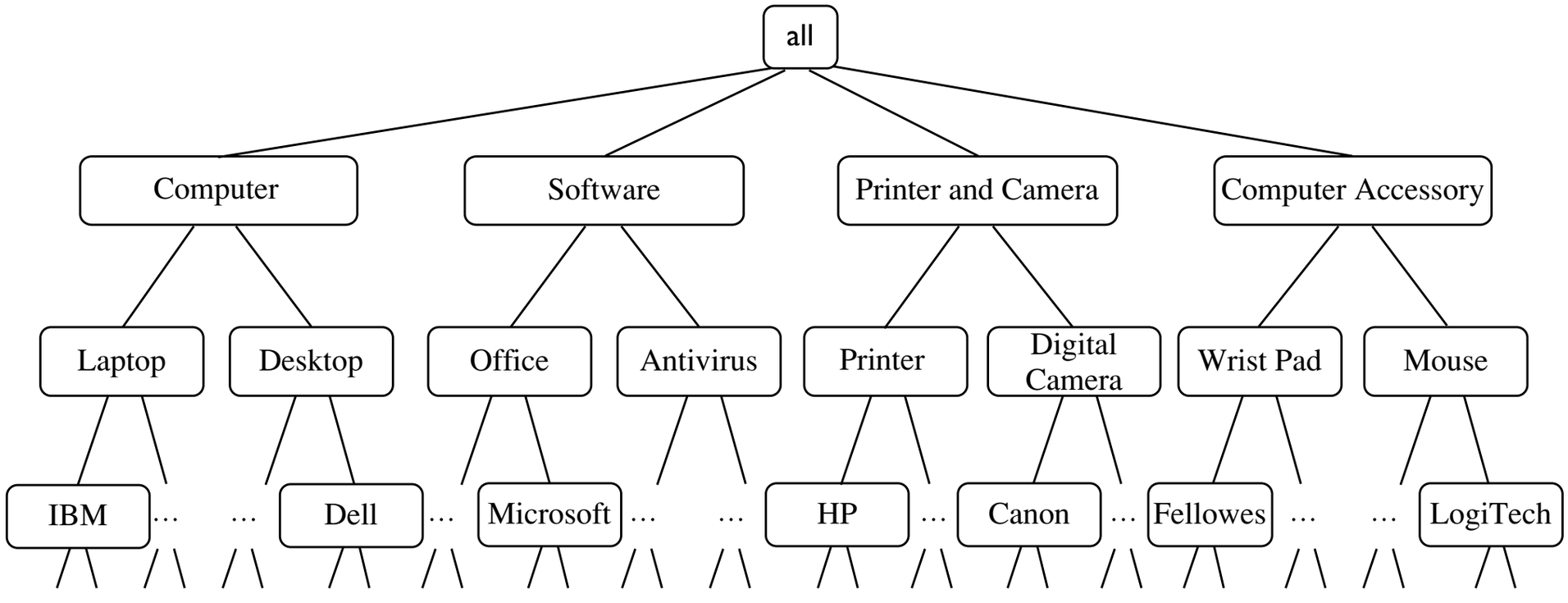
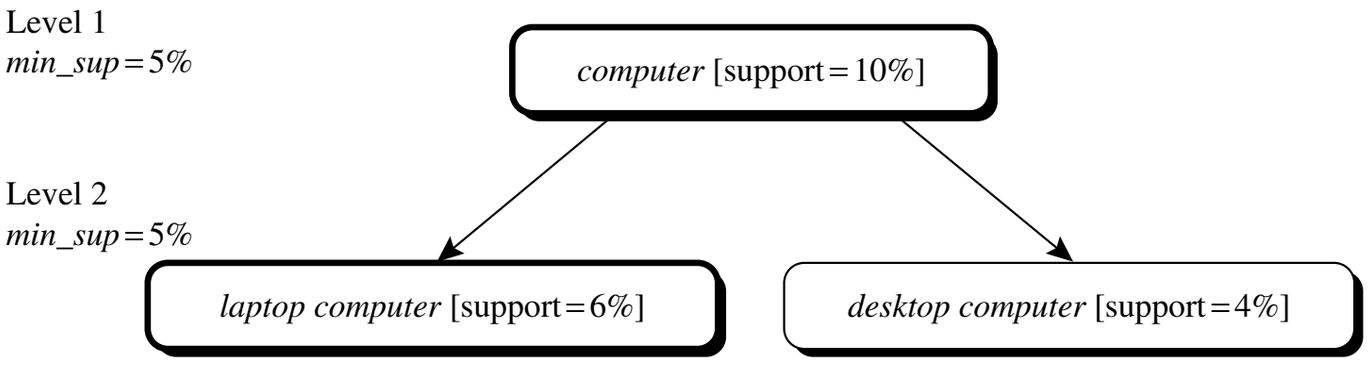


Figure 7.2 Concept hierarchy for *AllElectronics* computer items.

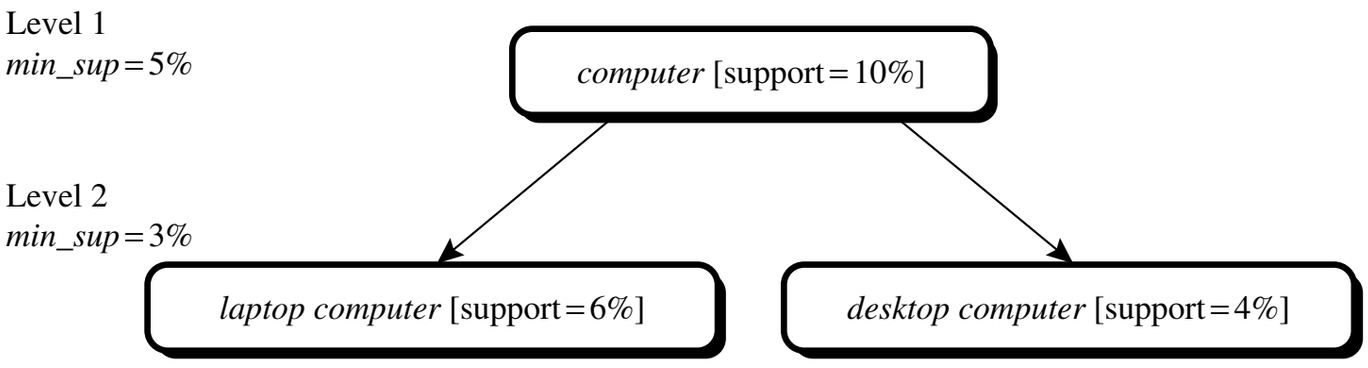
MULTI LEVEL ASSOCIATION RULES



Items often form hierarchies

Flexible support settings

Items at the lower level are expected to have lower support



J. Han and Y. Fu. Discovery of multiple-level association rules from large databases. In VLDB, volume 95, pages 420–431, 1995.

R. Srikant and R. Agrawal. Mining generalized association rules. IBM Research Division, 1995.

Some items are more valuable but less frequent

Flexible min-support thresholds

Use non-uniform, group-based min-support

{diamond, watch, camera}: 0.05%;
{bread, milk}: 5%; …

Some rules may be redundant due to "ancestor" relationships between items

Redundancy Filtering

milk → wheat bread [support = 8%, confidence = 70%]

2% milk → wheat bread [support = 2%, confidence = 72%]

redundant if 25% of the milk sold is "2% milk"

Multi-Level Association

Multi-Dimensional Association

Quantitative Association Rules

Mining Rare Patterns and Negative Patterns

MULTIDIMENSIONAL

Road Map Constraint Based

High Dimensional Data, Colossal Patterns Approximate Patterns

Sequential Patterns Graph Patterns Summary

SINGLE VS. MULTIDIMENSIONAL

Single-dimensional rules:

buys(X, "milk") → buys(X, "bread")

Multi-dimensional rules: at least two dimensions or predicates

Inter-dimension assoc. rules (no repeated predicates)

age(X,"19-25") ∧ occupation(X,"student") → buys(X,"coke")

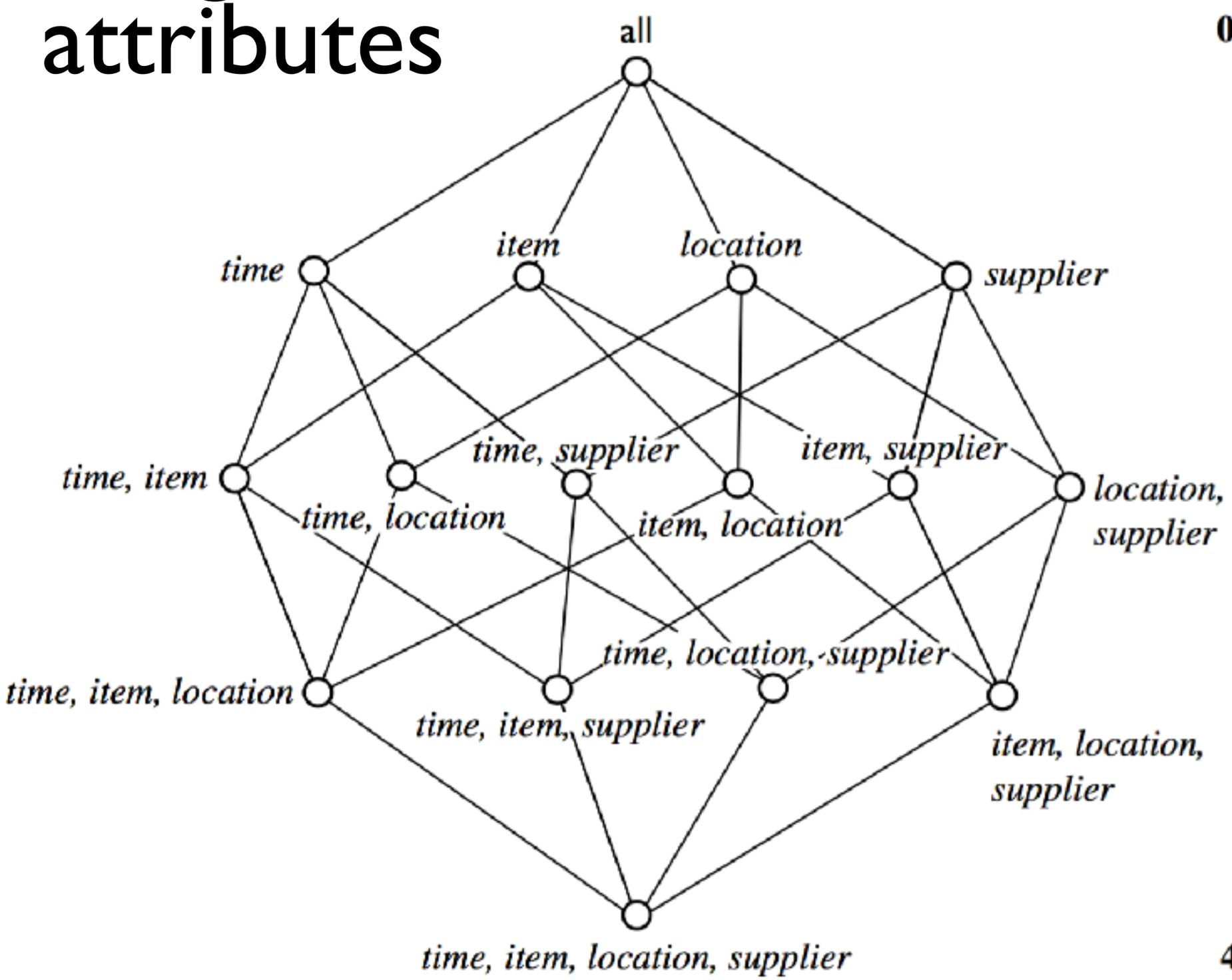
hybrid-dimension assoc. rules (repeated predicates)

age(X,"19-25") ∧ buys(X, "popcorn") → buys(X, "coke")

categorical attributes

Finite number of possible values, no ordering among values

categorical attributes



numerical attributes

discretization, clustering

Multi-Level Association

Multi-Dimensional Association

Quantitative Association Rules

Mining Rare Patterns and Negative Patterns

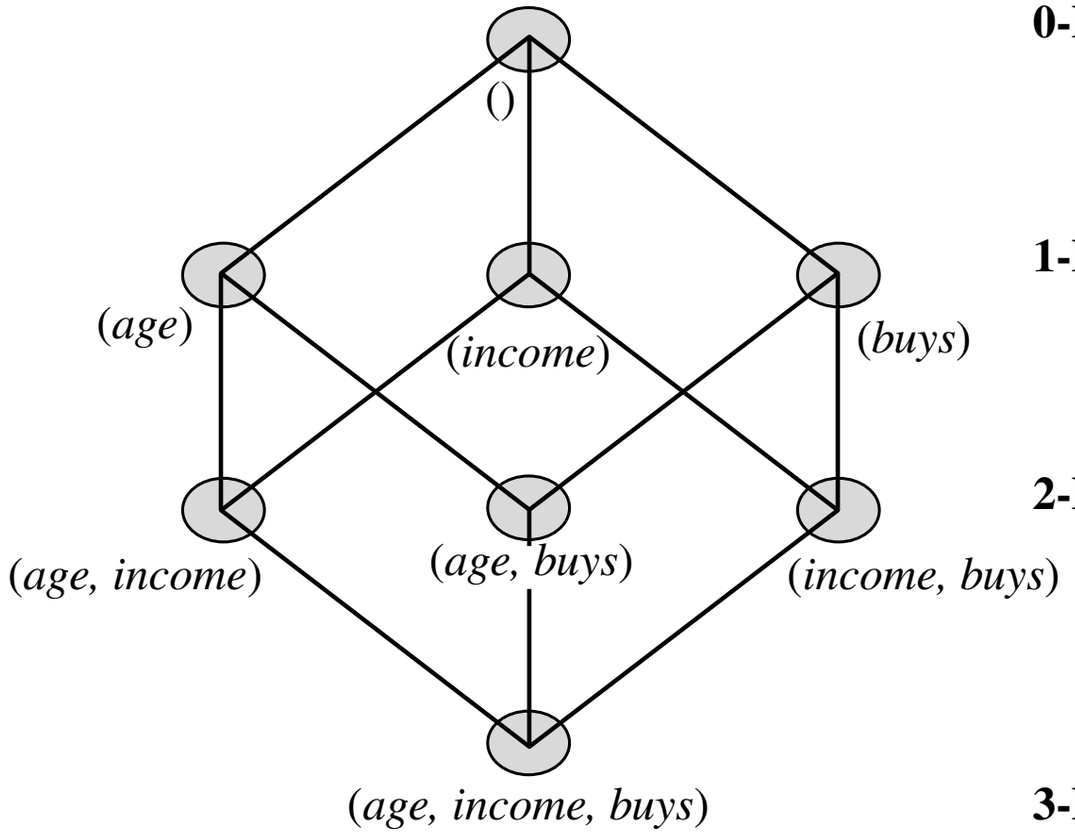
MULTIDIMENSIONAL

Road Map Constraint Based

High Dimensional Data, Colossal Patterns Approximate Patterns

Sequential Patterns Graph Patterns Summary

Static discretization based on predefined concept hierarchies

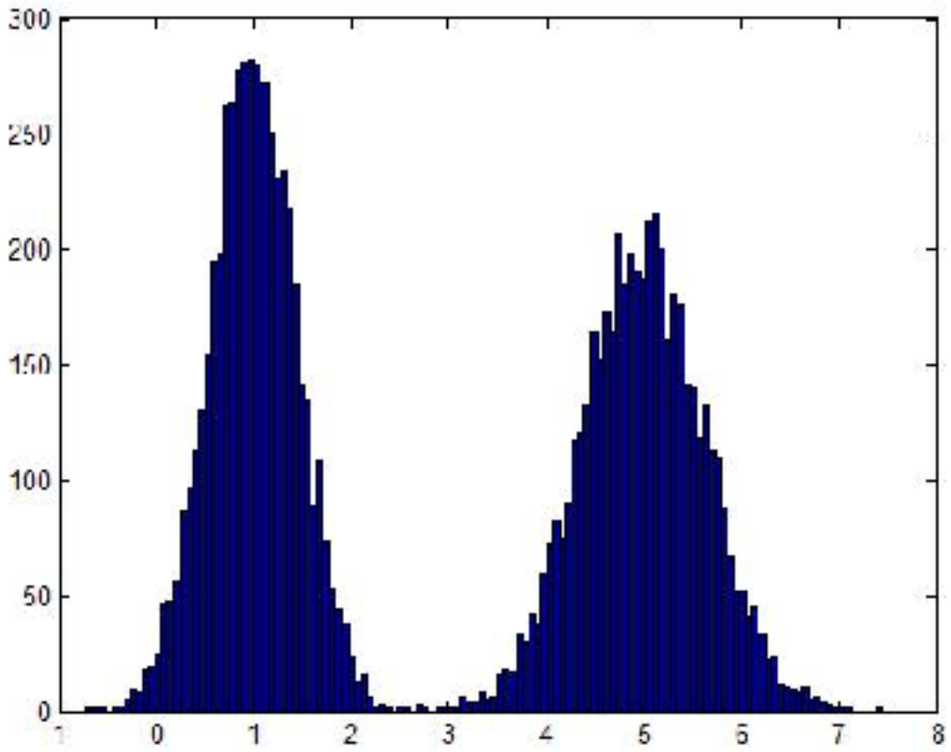


0-D (apex) cuboid

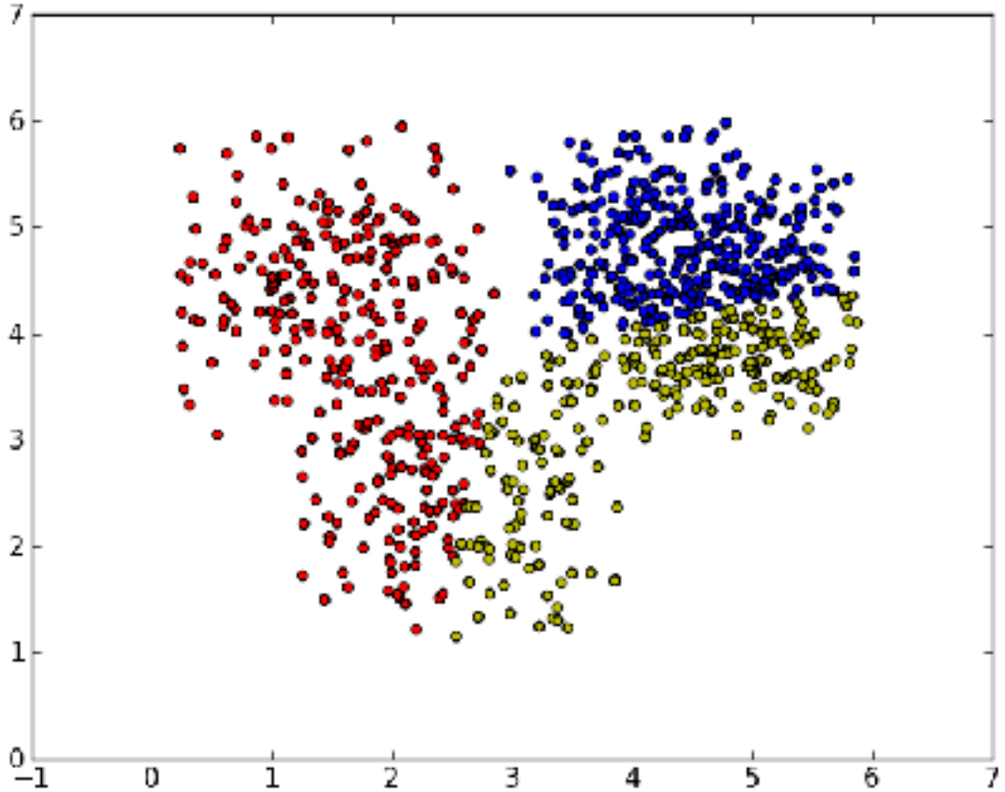
1-D cuboids

2-D cuboids

3-D (base) cuboid



Dynamic discretization based on data distribution



Clustering

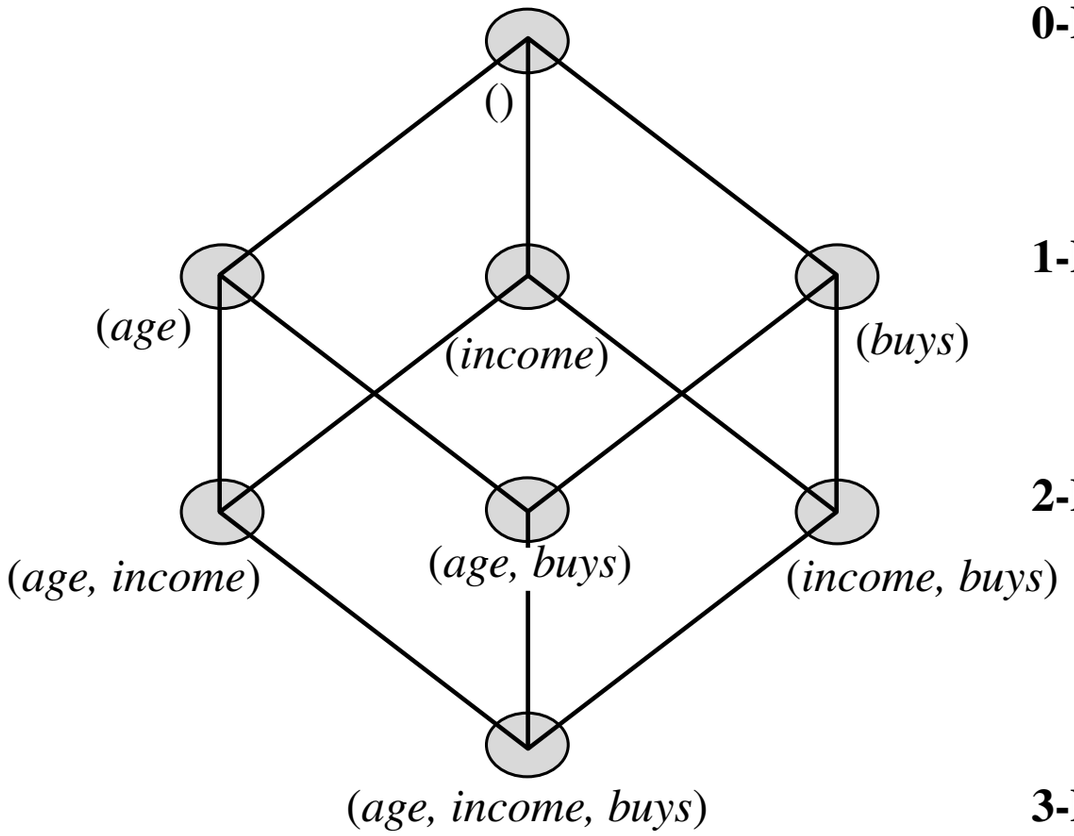
distance based association

Miller, R.J. and Yang, Y., 1997. Association rules over interval data. ACM SIGMOD Record, 26(2), pp.452-461.

statistical inference

Sex = female → Wage: mean=$7/hr (overall mean = $9)

STATIC DISCRETIZATION



0-D (apex) cuboid

1-D cuboids

2-D cuboids

3-D (base) cuboid

- Discretized prior to mining using concept hierarchy.
- Numeric values are replaced by ranges
- In relational database, finding all frequent k-predicate sets will require k or k+1 table scans
- Data cube is well suited for mining
- The cells of an n-dimensional cuboid correspond to the predicate sets
- Mining from data cubes can be much faster

RULES FROM STATISTICAL INFERENCE

Finding extraordinary and therefore interesting phenomena, e.g.,

(Sex = female) → Wage: mean=$7/hr (overall mean = $9)

LHS: a subset of the population

RHS: an extraordinary behavior of this subset

The rule is accepted only if a statistical test (e.g., Z-test) confirms the inference with high confidence

RULES FROM STATISTICAL INFERENCE

Subrule: highlights the extraordinary behavior of a subset of the population of the super rule

E.g., (Sex = female) ∧ (South = yes) → mean wage = $6.3/hr

RULES FROM STATISTICAL INFERENCE

Two forms of rules

- Categorical → quantitative rules, or
  Quantitative → quantitative rules

  E.g., Education in [14-18] (yrs) → mean wage = $11.64/hr

Open problem: Efficient methods for LHS containing two or more quantitative attributes

Multi-Level Association

Multi-Dimensional Association

Quantitative Association Rules

Mining Rare Patterns and Negative Patterns

MULTIDIMENSIONAL

Road Map Constraint Based

High Dimensional Data, Colossal Patterns Approximate Patterns

Sequential Patterns Graph Patterns Summary

RARE PATTERNS

Rare patterns: Very low support but interesting

E.g., buying Rolex watches

Mining: Setting individual-based or special group-based support threshold for valuable items

NEGATIVE PATTERNS

Negative patterns

Since it is unlikely that one buys Ford Expedition (an SUV car) and Toyota Prius (a hybrid car) together, Ford Expedition and Toyota Prius are likely negatively correlated patterns

Negatively correlated patterns that are infrequent tend to be more interesting than those that are frequent

NEGATIVE CORRELATED PATTERNS:1

Definition 1 (support-based)

If itemsets X and Y are both frequent but rarely occur together, i.e.,

$sup(X \cup Y) < sup(X) * sup(Y)$

Then X and Y are negatively correlated

Problem: A store sold two needle 100 packages A and B, only one transaction containing both A and B.

With total of 200 transactions:

$s(A \cup B) = 0.005$, $s(A) * s(B) = 0.25$, $s(A \cup B) < s(A) * s(B)$

With total of 10^5 transactions:

$s(A \cup B) = 1/10^5$, $s(A) * s(B) = 1/10^3 * 1/10^3$, $s(A \cup B) > s(A) * s(B)$

NEGATIVE CORRELATED PATTERNS:2

Definition 2 (negative itemset-based)

X is a negative itemset if

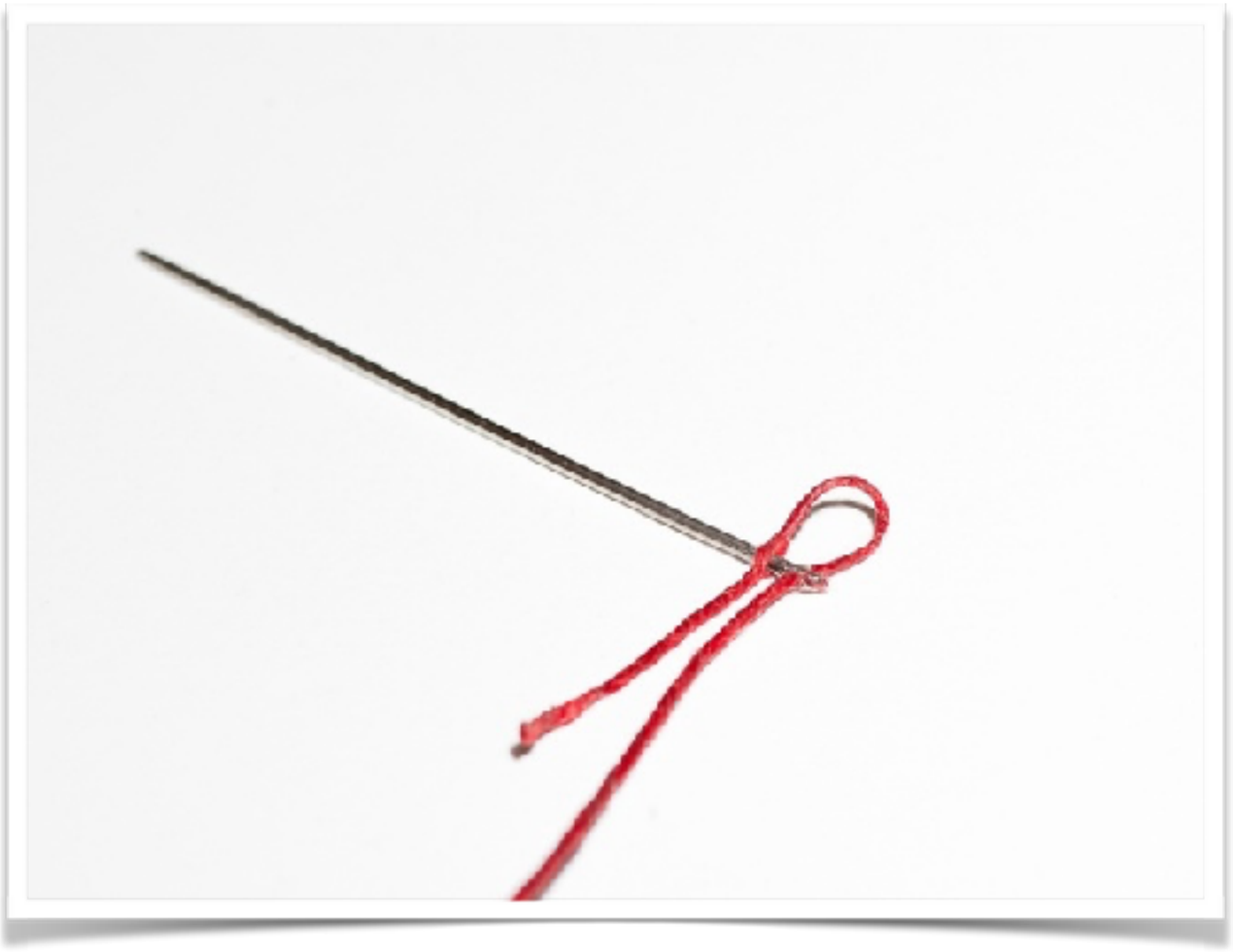
1. $X = \bar{A} \cup B$, where B is a set of positive items, and $\bar{A}$ is a set of negative items, $|\bar{A}| \geq 1$, and
2. $sup(X) \geq \mu$

Itemset X is negatively correlated, if

$$s(X) < \prod_{i=1}^{k} s(x_i), x_i \in X, s(x_i) \text{ is the support of } x_i$$

This definition suffers a similar null-invariant problem

NEGATIVE CORRELATED PATTERNS:3



Definition 3 (Kulzynski measure-based)

If itemsets X and Y are frequent, but (P(X|Y) + P(Y|X))/2 < ϵ, where ϵ is a negative pattern threshold, then X and Y are negatively correlated.

For the same needle package problem, when no matter there are 200 or 10^5 transactions, if ϵ = 0.05, we have

(P(A|B) + P(B|A))/2 = (0.01 + 0.01)/2 < ϵ

CONSTRAINT BASED MINING

Road Map Multidimensional

High Dimensional Data, Colossal Patterns Approximate Patterns

Sequential Patterns Graph Patterns Summary

Finding all the patterns in a database autonomously?

Too many patterns!

mining should be an interactive process

User directs what to be mined using a data mining query language (or a graphical user interface)

QUERY DIRECTED MINING

User flexibility: provides constraints on what to be mined

Optimization: explores such constraints for efficient mining—constraint-based mining: constraint-pushing, similar to push selection first in DB query processing

Note: goal is to still find all the answers satisfying constraints, not finding some answers in "heuristic search"

dimension

region, price , brand etc.

knowledge

association, correlation etc.

interestingness

sup≥3%, confidence≥60%

what kinds of constraints?

data

which data to mine (via SQL)

rule / pattern

cheap items (price < $10)

triggers big sales (sum > $200)

META RULE GUIDED MINING

Meta-rule can be in the rule form with partially instantiated predicates and constants

$P_1(X,Y) \wedge P_2(X,W) \rightarrow$ buys(X, "iPad")

The resulting rule derived can be

age(X, "15-25") $\wedge$ profession(X, "student") $\rightarrow$ buys(X, "iPad")

In general, it can be in the form of

$P_1 \wedge P_2 \wedge \ldots \wedge P_l \rightarrow Q_1 \wedge Q_2 \wedge \ldots \wedge Q_r$

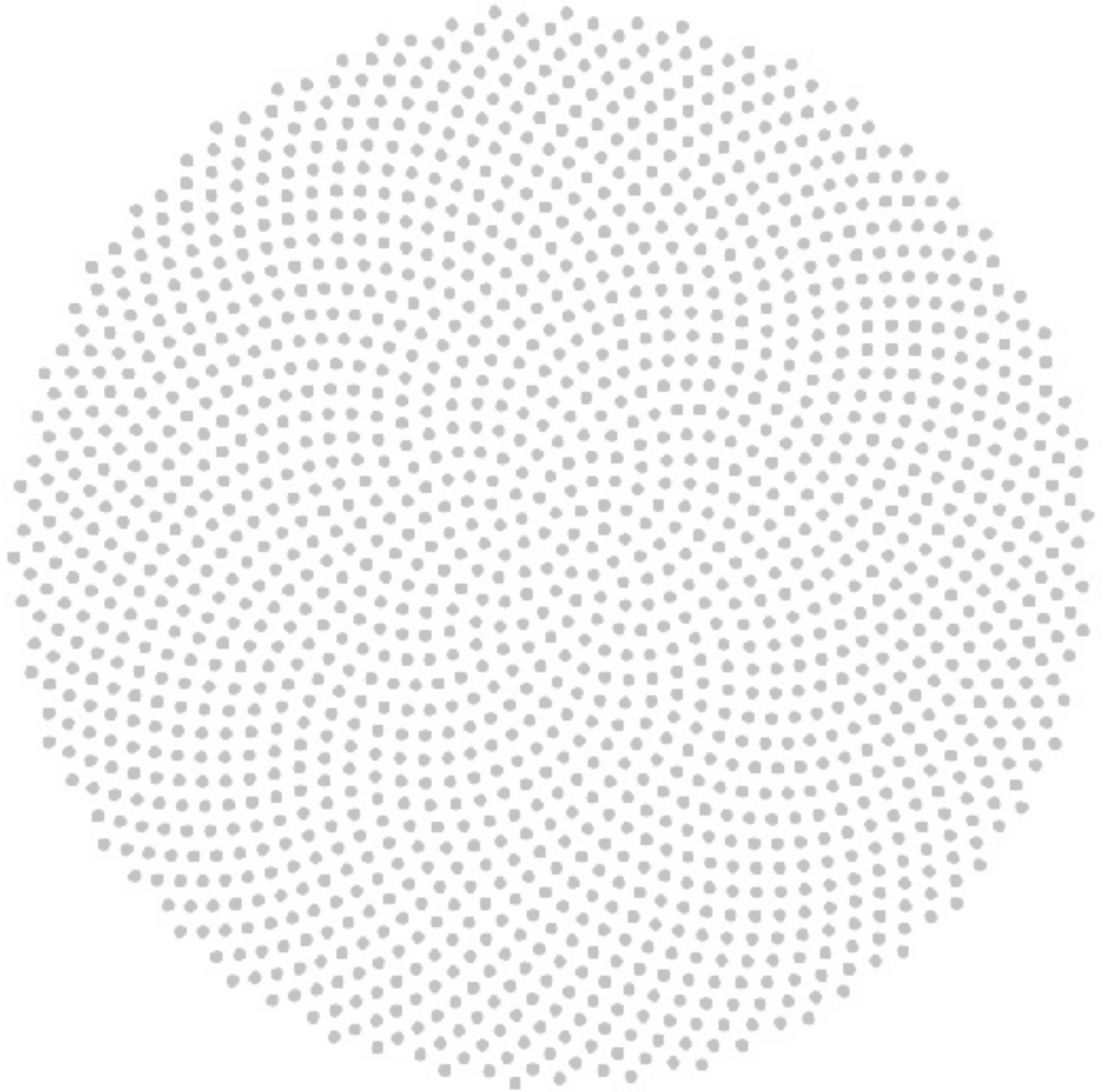
Method:

Find frequent (l+r) predicates (based on min-support threshold)

When possible, push constraints deeply into the mining process

Use confidence, correlation, and other measures when possible

PATTERN PRUNING CONSTRAINTS



Anti-monotonic: If constraint c is violated, its further mining can be terminated

Monotonic: If c is satisfied, no need to check c again

Succinct: c must be satisfied, so one can start with the data sets satisfying c

Convertible: c is neither monotonic nor anti-monotonic, but it can be converted into them if items in the transaction can be properly ordered

DATA PRUNING CONSTRAINTS

Succinct: Data space can be pruned at the initial pattern mining process

Anti-monotonic: If a transaction t does not satisfy c, t can be pruned from its further mining

sup=2

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

PSP: ANTI MONOTONICITY

A constraint C is anti-monotone if the super pattern satisfies C, all of its sub-patterns do so too

In other words, anti-monotonicity: If an itemset S **violates** the constraint, so does any of its supersets

1. sum(S.price)≤v, anti-monotone
2. range(S.profit)≤15, anti-monotone

   Itemset {a,b} violates C

   So does every superset of {a,b}
3. sum(S.Price)≥v is not anti-monotone
4. count(S) is anti-monotone, core property used in Apriori

sup=2

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

PSP: MONOTONICITY

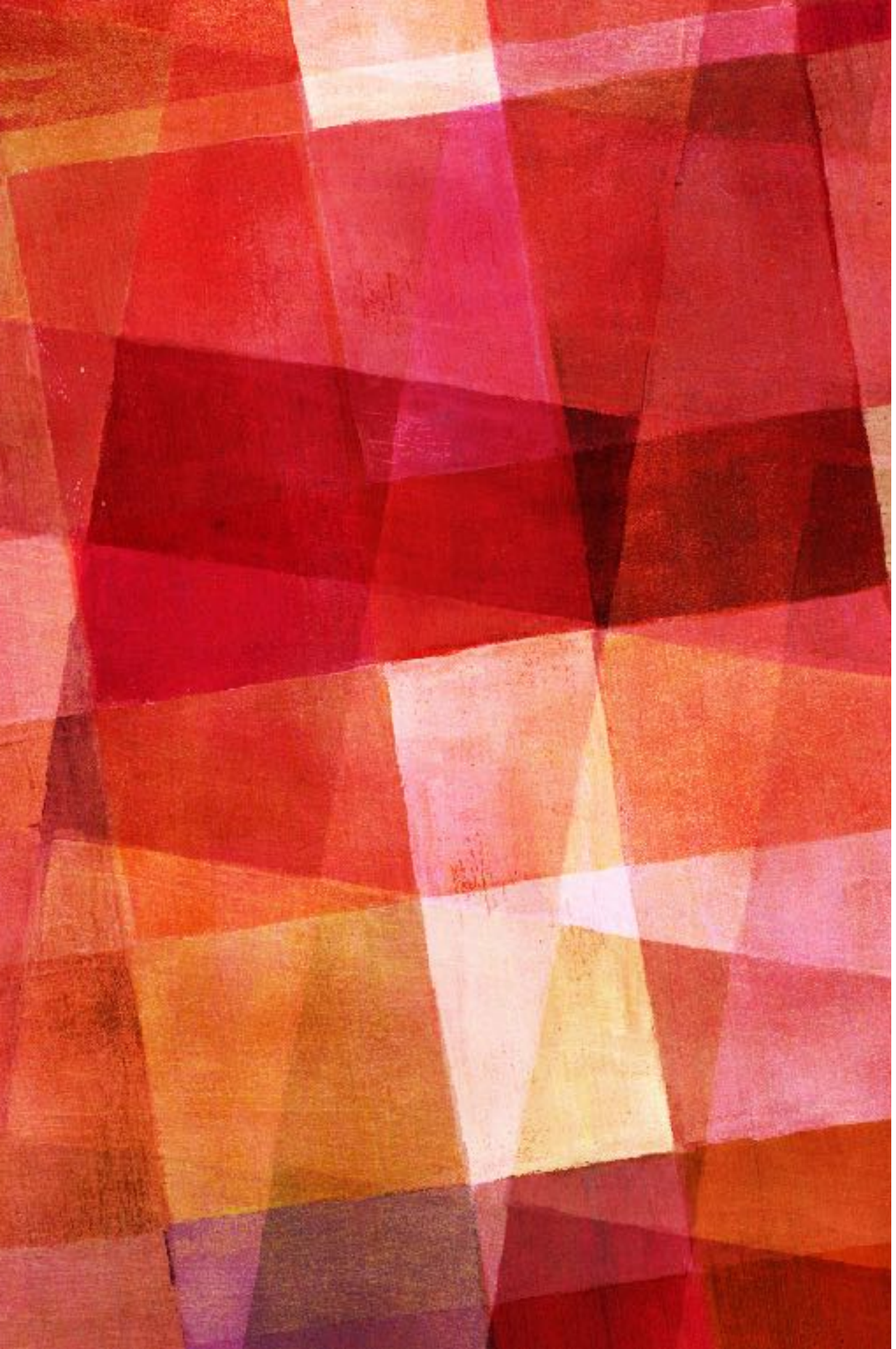
A constraint C is monotone if the pattern satisfies C, all of its super patterns do so too

In other words, monotonicity: If an itemset S **satisfies** the constraint, so does any of its superset

1. sum(S.price) ≥v, monotone
2. range(S.profit)≥15, monotone

Itemset {a,b} satisfies C

So does every superset of {a,b}



PATTERN SPACE MINING: SUCCINCTNESS

Given A_I, the set of items satisfying a succinctness constraint C, then any set S satisfying C is based on A_I , i.e., S contains a subset belonging to A_I

Idea: Without looking at the transaction database, whether an itemset S satisfies constraint C can be determined based on the selection of items

min(S.Price)≤v is succinct

sum(S.Price)≥v is **not** succinct

Optimization: If C is succinct, C is pre-counting prunable

to check for **anti-monotonicity**, we check for superset **violations**.

sum(S.price)≤v

to check for **monotonicity**, we check for superset **satisfactions**.

sum(S.price) ≥v

R. T. Ng, L. V. S. Lakshmanan, J. Han, and A. Pang. Exploratory mining and pruning optimizations of constrained associations rules. In Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data, SIGMOD '98, pages 13–24, New York, NY, USA, 1998. ACM.

S: is a **variable** set

v, V: are **constants**; either a single element (v) or a set (V)

apply the rules to the variables, **not** constants!

Constraint	*Antimonotonic*	*Monotonic*	*Succinct*
$v \in S$	no	yes	yes
$S \supseteq V$	no	yes	yes
$S \subseteq V$	yes	no	yes
$min(S) \leq v$	no	yes	yes
$min(S) \geq v$	yes	no	yes
$max(S) \leq v$	yes	no	yes
$max(S) \geq v$	no	yes	yes
$count(S) \leq v$	yes	no	weakly
$count(S) \geq v$	no	yes	weakly
$sum(S) \leq v\ (\forall a \in S, a \geq 0)$	yes	no	no
$sum(S) \geq v\ (\forall a \in S, a \geq 0)$	no	yes	no
$range(S) \leq v$	yes	no	no
$range(S) \geq v$	no	yes	no
$avg(S)\ \theta\ v, \theta \in \{\leq, \geq\}$	convertible	convertible	no
$support(S) \geq \xi$	yes	no	no
$support(S) \leq \xi$	no	yes	no
$all\_confidence(S) \geq \xi$	yes	no	no
$all\_confidence(S) \leq \xi$	no	yes	no

Thus far, we've been looking at constraints on patterns

What if a **specific** datum is going to cause problems?

Then, if **milk**, **cheese** is the current itemset under consideration, **then coffee cannot appear** along with it.

range(S.price) ≥ $7

pattern monotonic

Assume that milk ($**3**) cheese ($**7**) and a cup of coffee ($**5**) are frequently bought together.

we are rejecting coffee for its value in conjunction with {milk, cheese}; it **can** appear with other itemsets and hence cannot be pruned at the beginning

data anti-monotonicity **cannot be easily used** with apriori

DATA PRUNING: ANTI MONOTONICITY

sup=2

TID	Transaction
10	a, b, c, d, f, h
20	b, c, d, f, g, h
30	b, c, d, f, g
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	-15
e	-30
f	-10
g	20
h	-5

A constraint c is data anti-monotone if:

> a pattern p cannot satisfy a transaction t under constraint c, p's superset cannot satisfy t under constraint c

The key for data anti-monotone is recursive data reduction

1. sum(S.Price)≥v is data anti-monotone
2. min(S.Price)≤v is data anti-monotone
3. range(S.profit)≥25 is data anti-monotone

Itemset {b, c}'s projected DB:

> T10': {d, f, h}, T20': {d, f, g, h}, T30': {d, f, g}
>
> since {T10 ∪ {b,c} } cannot satisfy C, T10 can be pruned

pattern anti-monotonic **sum{S.price} < $5**

apriori + constraint

TID	Items (Price)
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

scan →

Candidates-1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

→

L1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

Candidates-2

itemset
{1,2}
{1,3}
{1,5}
{2,3}
{2,5}
{3,5}

2-Candidates support

itemset	sup.
{1,2}	1
{1,3}	2
{1,5}	1
{2,3}	2
{2,5}	3
{3,5}	2

←

L2

itemset	sup.
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

Candidates-3

itemset
{2,3,5}

unnecessary computations

min{S.price} ≤1

apriori + succinct constraint

∴ Items satisfying **price ≤$1 must** appear in **every** itemset of interest

TID	**Items** (Price)
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

scan →

Candidates-1

itemset	**sup.**
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

→

L1

itemset	**sup.**
{1}	2
{2}	3
{3}	3
{5}	3

Candidates-2

itemset
{1,2}
{1,3}
{1,5}
{2,3}
{2,5}
{3,5}

←

2-Candidates support

itemset	**sup.**
{1,2}	1
{1,3}	2
{1,5}	1
{2,3}	2
{2,5}	3
{3,5}	2

L2

itemset	**sup.**
{1,3}	2
{2,3}	2
{2,5}	3
{3,5}	2

Candidates-3

itemset
{2,3,5}

unnecessary computations

constrained FP-Growth: succinct constraint

min{S.price} ≤1

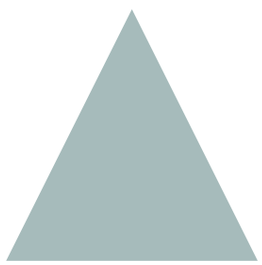
∴ Items satisfying **price ≤$1 must** appear in **every** itemset of interest

TID	**Items** (Price)
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

scan →

remove infrequent items

TID	**Items** (Price)
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5



FP Tree

No need to project items with price 2 or 5 because they are **not** frequent with 1

1 Projected DB

TID	**Items**
100	3
300	2 3 5

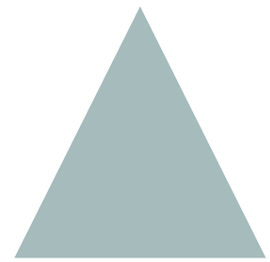
constrained FP-Growth: push a data-antimonotic constraint deep

TID	Items (Price)
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

scan

remove infrequent items

TID	Items (Price)
100	1 3 4
300	1 2 3 5



FP Tree

single branch!

remove transactions because they do not contain items with price $1

min{S.price} ≤1

TID	Transaction
10	a, b, c, d, f
20	b, c, d, f, g, h
30	a, c, d, e, f
40	c, e, f, g

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

CONVERTIBLE CONSTRAINTS

Convert tough constraints into anti-monotone or monotone by properly ordering items

Examine C: avg(S.profit) ≥ $25

Order items in value-descending order

<a, f, g, d, b, h, c, e>

If an itemset afb violates C

So does afbh, afb*

It becomes anti-monotone!

Item	Profit
a	40
b	0
c	-20
d	10
e	-30
f	30
g	20
h	-10

APRIORI AND CONVERTIBLE CONSTRAINTS

A convertible, not monotone nor anti-monotone nor succinct constraint cannot be pushed deep into the an Apriori mining algorithm

Within the level wise framework, no direct pruning based on the constraint can be made

Itemset df violates constraint C: $\mathtt{avg}(X) \geq 25$

Since adf satisfies C, Apriori needs df to assemble adf, df cannot be pruned

But it can be pushed into frequent-pattern growth framework!

TID	Transaction
10	a, f, d, b, c
20	f, g, d, b, c
30	a, f, d, c, e
40	f, g, h, c, e

Item	Value
a	40
f	30
g	20
d	10
b	0
h	-10
c	-20
e	-30

PSP: CONVERTIBLE CONSTRAINTS

C: avg(X)≥25, min_sup=2

List items in every transaction in value descending order R: <a, f, g, d, b, h, c, e>

C is convertible anti-monotone w.r.t. R

Scan TDB once

remove infrequent items (Item h is dropped)

Itemsets a and f are good, …

Projection-based mining

Imposing an appropriate order on item projection

Many tough constraints can be converted into (anti)-monotone

HANDLING MULTIPLE CONSTRAINTS

Different constraints may require different or even conflicting item-ordering

If there exists an order R such that both C_1 and C_2 are convertible w.r.t. R, then there is no conflict between the two convertible constraints

If there exists conflict on order of items

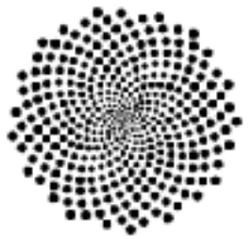
- Try to satisfy one constraint first
- Then using the order for the other constraint to mine frequent itemsets in the corresponding projected database

rule / pattern

cheap items (price < $10) triggers big sales (sum > $200)

meta rule based mining

$$P_1(X,Y) \wedge P_2(X,W) \rightarrow \textbf{buys}(X,\text{“iPad”})$$



anti-monotonic, monotonic, succinct, convertible

anti-monotonic, succinct

SUMMARY

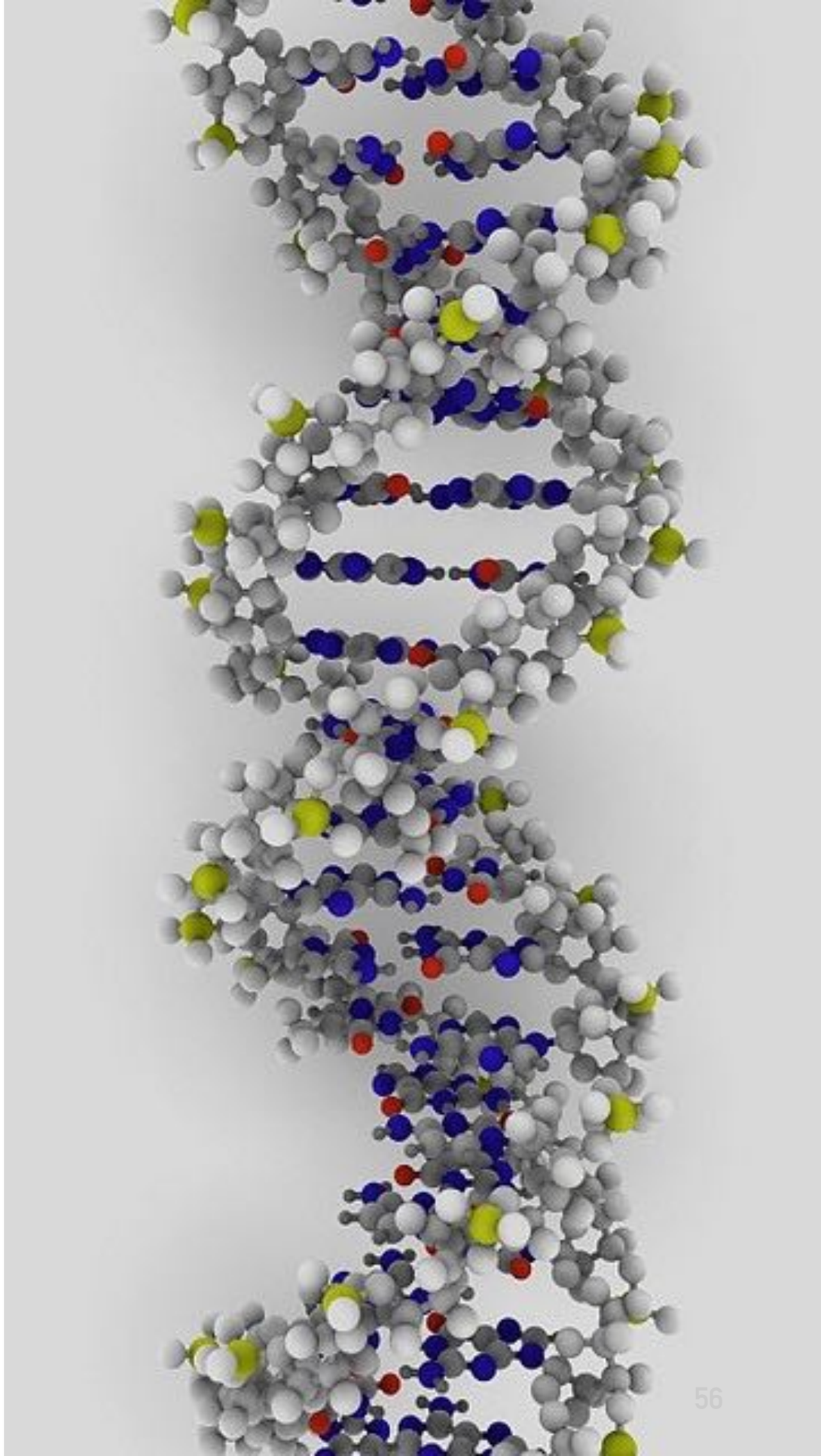
how to push constraints deep into priori, FPGrowth

HIGH DIMENSIONAL DATA, COLOSSAL PATTERNS

Road Map Multidimensional

Constraint Based Mining Approximate Patterns

Sequential Patterns Graph Patterns Summary



can we mine patterns of large <u>size</u>?

Zhu, Feida, Xifeng Yan, Jiawei Han, Philip S. Yu, and Hong Cheng. "Mining colossal frequent patterns by core pattern fusion." In Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, pp. 706-715. IEEE, 2007.

unfortunately,
no!

the curse returns!

2^{100} patterns!

If $(a_1, a_2, \ldots, a_{100})$ is frequent, then $(a_1, a_2, \ldots, a_{100})$, (a_1, a_2), (a_1, a_3), …, (a_1, a_{100}), (a_1, a_2, a_3), … are all frequent!

neither Apriori or FP Growth are of help

downward closure property

Any sub-pattern of a frequent pattern is frequent.

A MOTIVATING EXAMPLE

row/col	1	2	3	4	…	38	39
1	2	3	4	5	…	39	40
2	1	3	4	5	…	39	40
3	1	2	4	5	…	39	40
4	1	2	3	5	…	39	40
5	1	2	3	4	…	39	40
…	…	…	…	…	…	…	…
39	1	2	3	4	…	38	40
40	1	2	3	4	…	38	39
41	41	42	43	44	…	78	79
42	41	42	43	44	…	78	79
…	…	…	…	…	…	…	…
60	41	42	43	44	…	78	79

items on the diagonal have been deleted

Closed/maximal patterns may partially alleviate the problem but not really solve it: We often need to mine scattered large patterns!

Let the minimum support threshold σ= 20

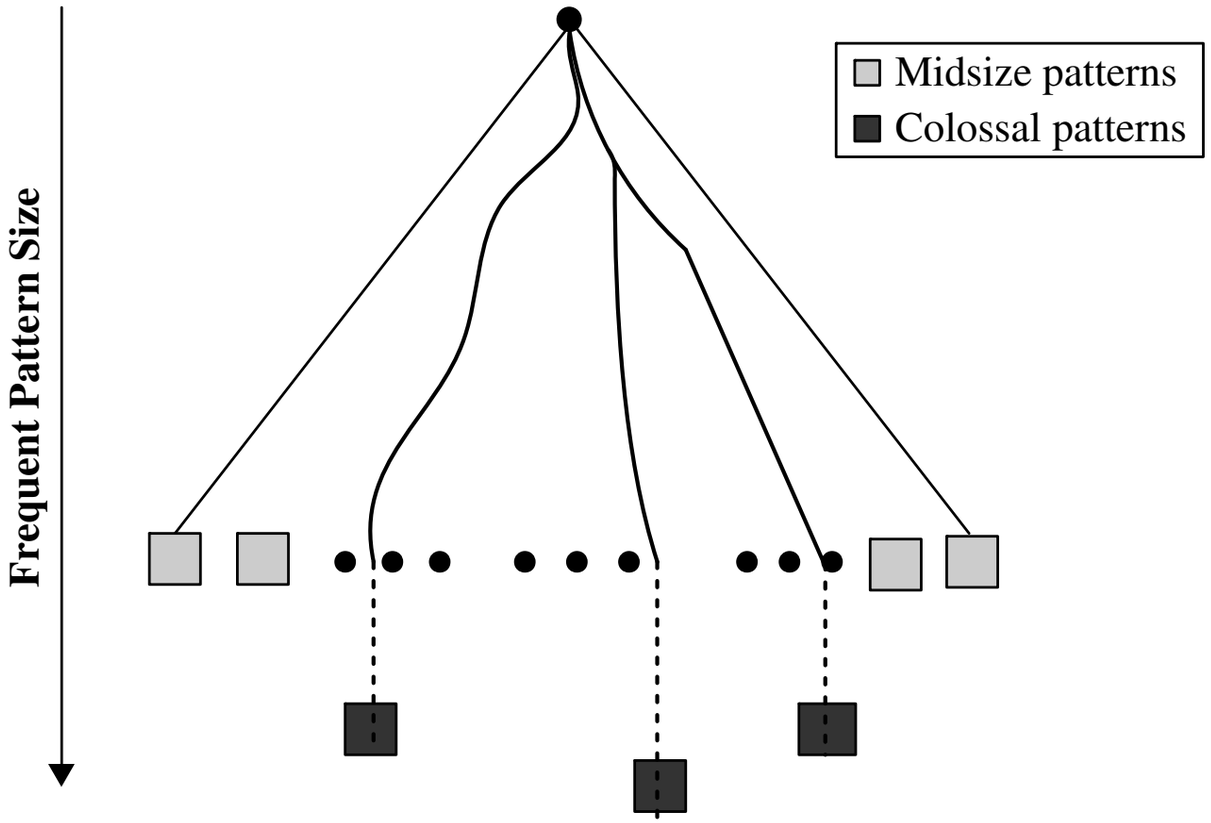
There are $\binom{40}{20}$ frequent patterns of size 20

Each is closed and maximal

patterns = $\binom{n}{n/2} = \sqrt{\frac{2}{\pi n}}2^n$

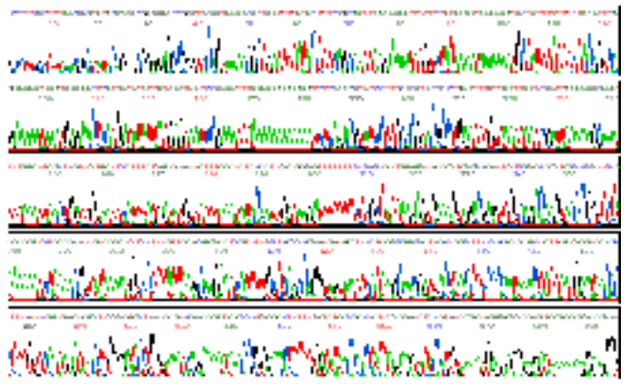
The size of the answer set is exponential to n

COLOSSAL PATTERNS: SMALL BUT INTERESTING



It is often the case that only a small number of patterns are colossal, i.e., of large size

Colossal patterns are usually attached with greater importance than those of small pattern sizes



Micro-array analysis in bioinformatics (when support is low)

Many real-world tasks need discovery of colossal patterns

Biological sequence patterns

Biological/sociological/ information graph pattern mining

No hope for completeness

If the mining of mid-sized patterns is explosive in size, there is no hope to find colossal patterns efficiently by insisting "complete set" mining philosophy

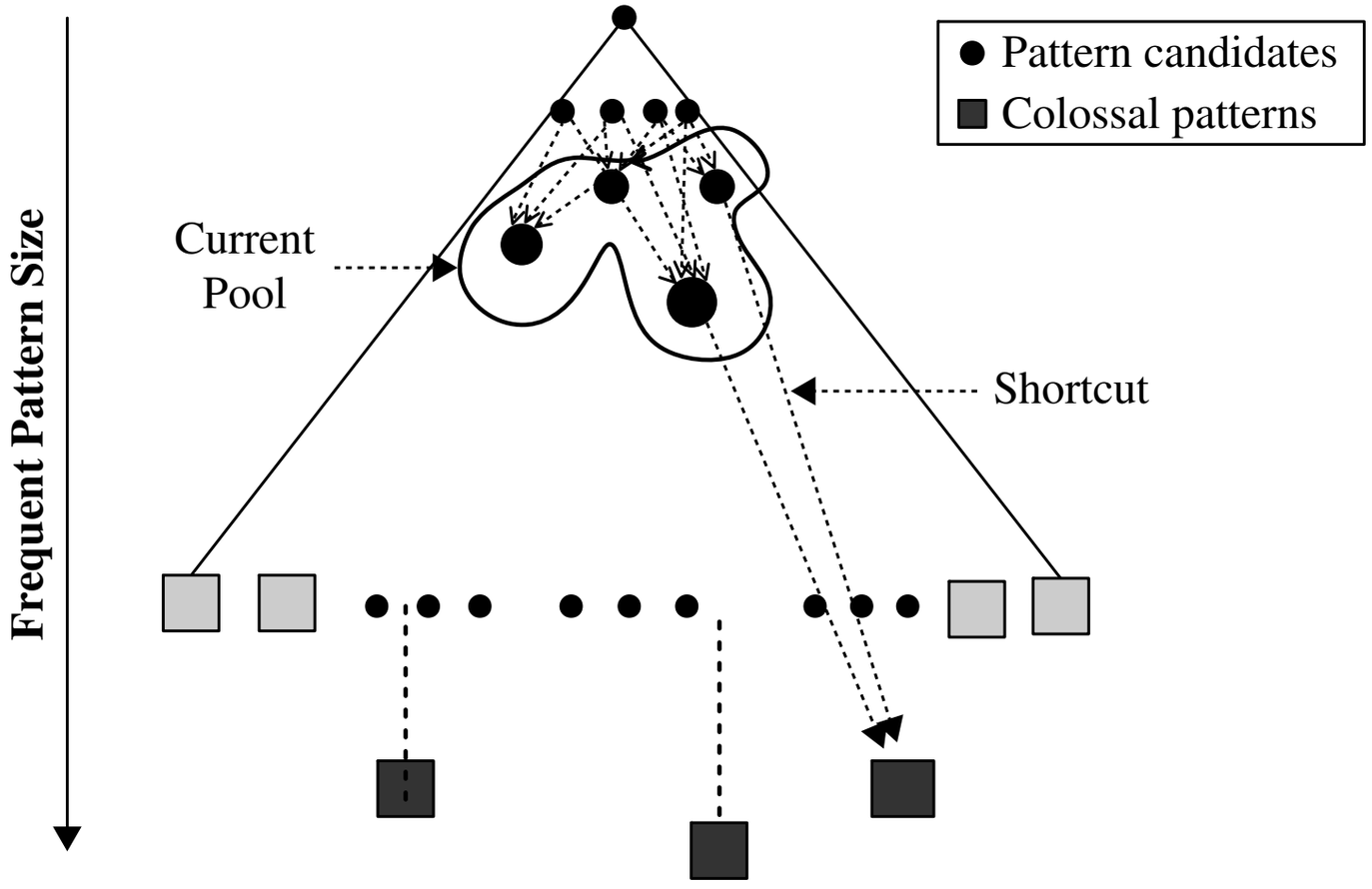
Jumping out of the swamp of the mid-sized results

What we may develop is a philosophy that may jump out of the swamp of mid-sized results that are explosive in size and jump to reach colossal patterns

Striving for mining almost complete colossal patterns

The key is to develop a mechanism that may quickly reach colossal patterns and discover most of them

PATTERN FUSION METHODOLOGY

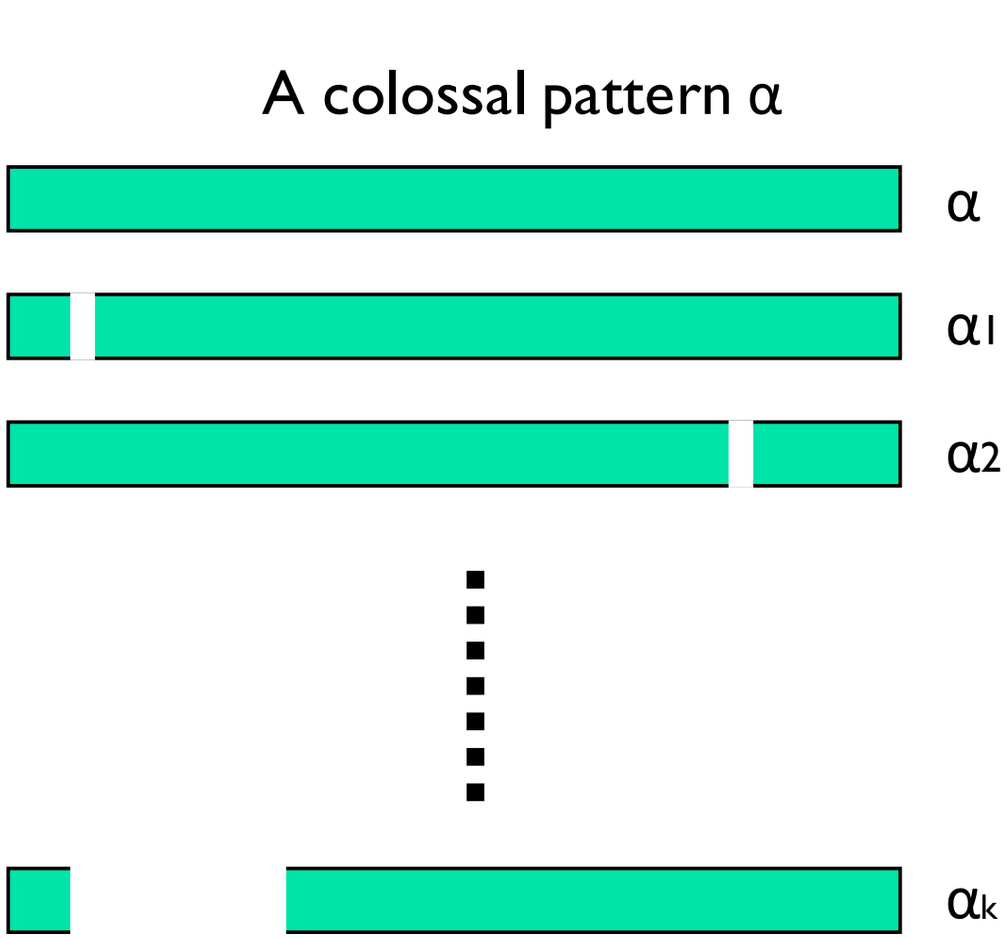


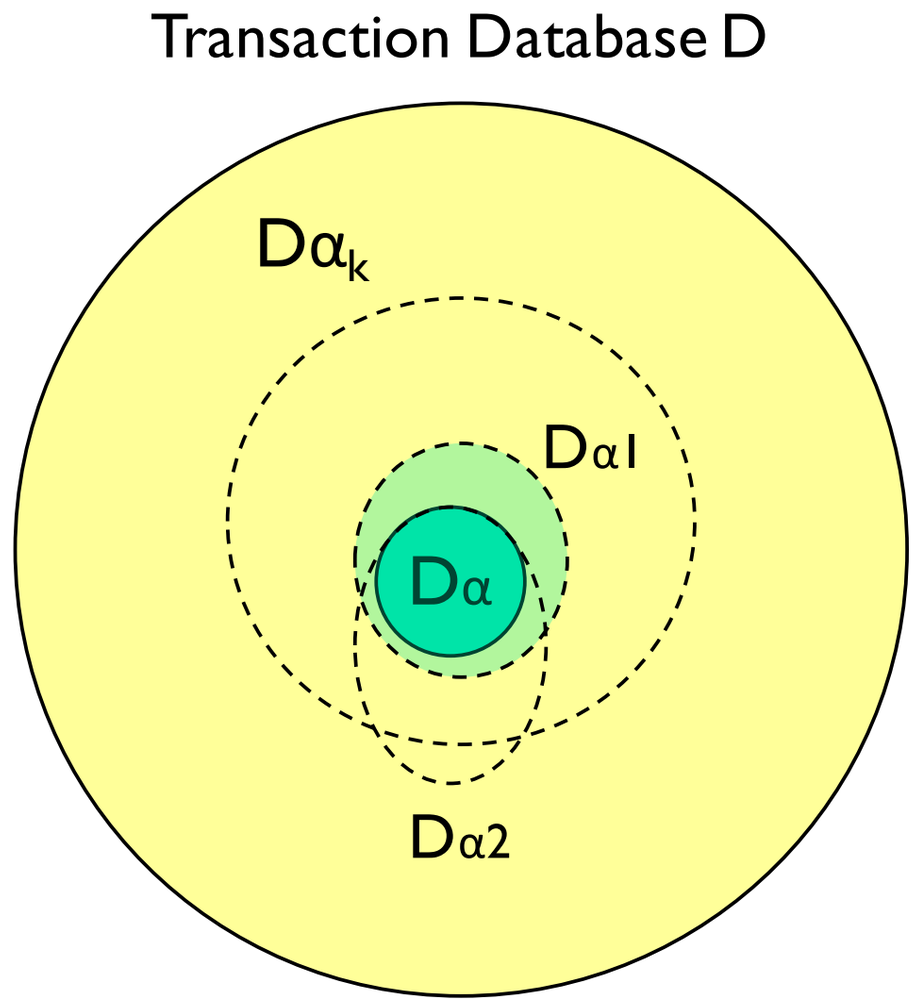
Pattern-Fusion traverses the tree in a bounded-breadth way

- Always pushes down a frontier of a bounded-size candidate pool
- Only a fixed number of patterns in the current candidate pool will be used as the starting nodes to go down in the pattern tree — thus avoids the exponential search space

Pattern-Fusion identifies "shortcuts" whenever possible

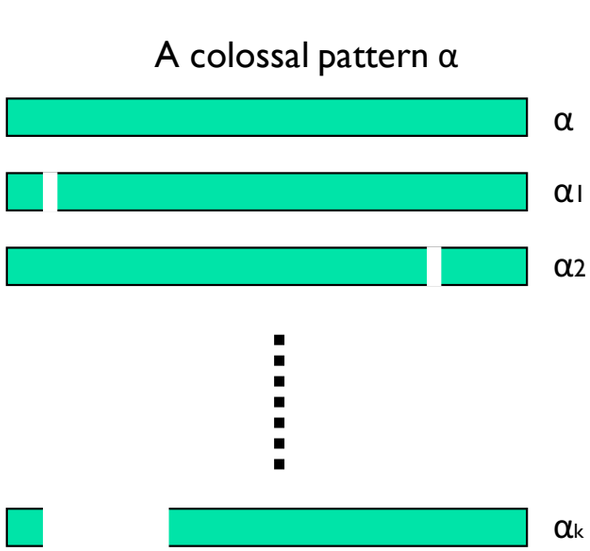
- Pattern growth is not performed by single-item addition but by leaps and bounded: agglomeration of multiple patterns in the pool
- These shortcuts will direct the search down the tree much more rapidly towards the colossal patterns

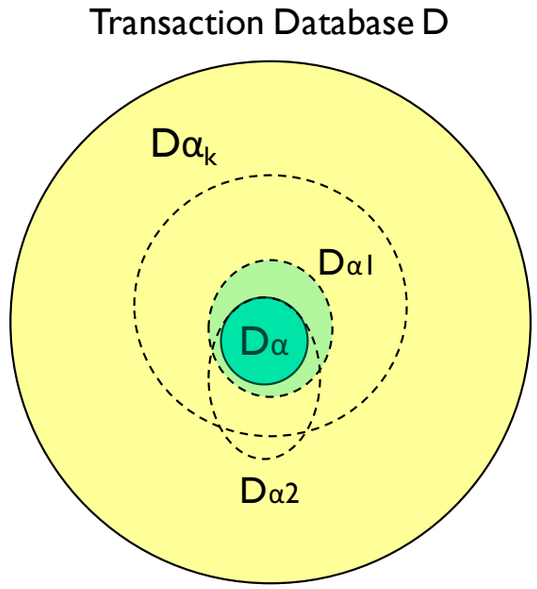




Subpatterns α_1 to α_k cluster tightly around the colossal pattern α by sharing a similar support. We call such subpatterns core patterns of α

CORE PATTERNS





Intuitively, for a frequent pattern α, a subpattern β is a τ-core pattern of α if β shares a similar support set with α, i.e.,

$$\frac{|D_\alpha|}{|D_\beta|} \geq \tau$$

where 0<τ<1 is called the core ratio

Robustness of Colossal Patterns

A colossal pattern is robust in the sense that it tends to have much more core patterns than small patterns

example core patterns

Transaction (# of Ts)	**Core Patterns (τ = 0.5)**
(abe) (100)	(abe), (ab), (be), (ae), (e)
(bcf) (100)	(bcf), (bc), (bf)
(acf) (100)	(acf), (ac), (af)
(abcef) (100)	(ab), (ac), (af), (ae), (bc), (bf), (be) (ce), (fe), (e), (abc), (abf), (abe), (ace), (acf), (afe), (bcf), (bce), (bfe), (cfe), (abcf), (abce), (bcfe), (acfe), (abfe), (abcef)

EXAMPLE CORE PATTERNS

Transaction (# of Ts)	**Core Patterns (τ = 0.5)**
(abe) (100)	(abe), (ab), (be), (ae), (e)
(bcf) (100)	(bcf), (bc), (bf)
(acf) (100)	(acf), (ac), (af)
(abcef) (100)	(ab), (ac), (af), (ae), (bc), (bf), (be) (ce), (fe), (e), (abc), (abf), (abe), (ace), (acf), (afe), (bcf), (bce), (bfe), (cfe), (abcf), (abce), (bcfe), (acfe), (abfe), (abcef)

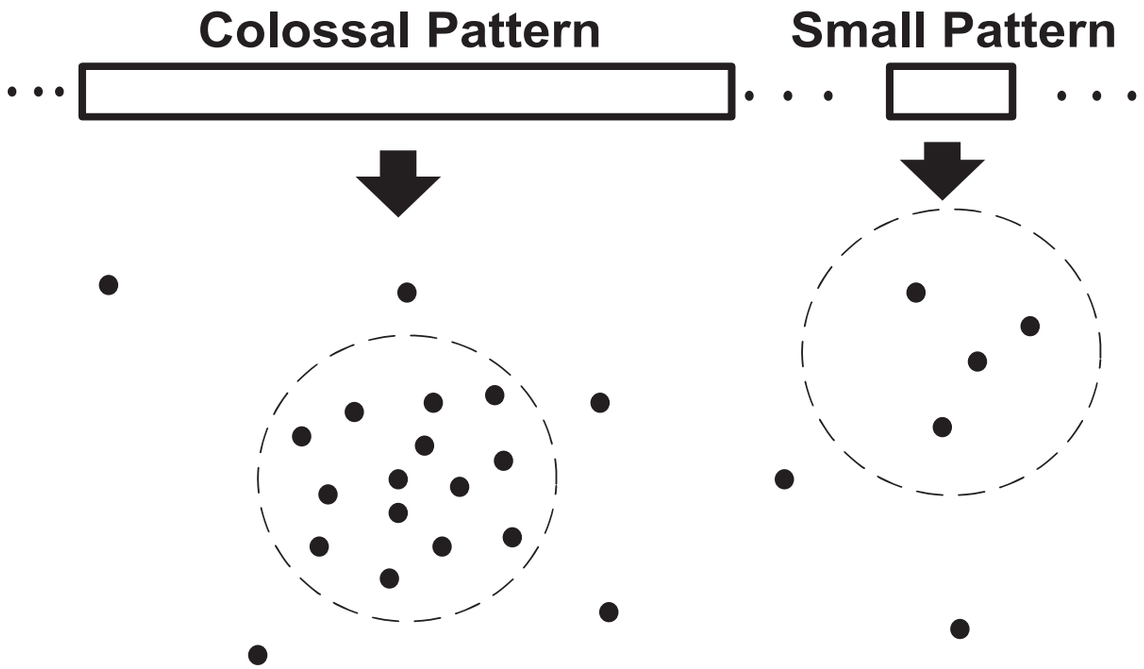
A colossal pattern has far more core patterns than a small-sized pattern

A colossal pattern has far more core descendants of a smaller size c

A random draw from a complete set of patterns of size c would **more likely** to pick a **core** descendant of a colossal pattern

A colossal pattern can be generated by merging a set of core patterns

Due to their robustness, colossal patterns correspond to dense balls $\Omega(2^d)$ in size



A random draw in the pattern space will hit somewhere in the ball with high probability

ROBUSTNESS

A pattern α is (d, τ)-robust if d is the maximum number of items that can be removed from α for the resulting pattern to remain a τ-core pattern of α

For a (d,τ)-robust pattern α, it has $\Omega(2^d)$ core patterns

A colossal patterns tend to have a large number of core patterns

Pattern distance: For patterns α and β, the pattern distance of α and β is defined to be

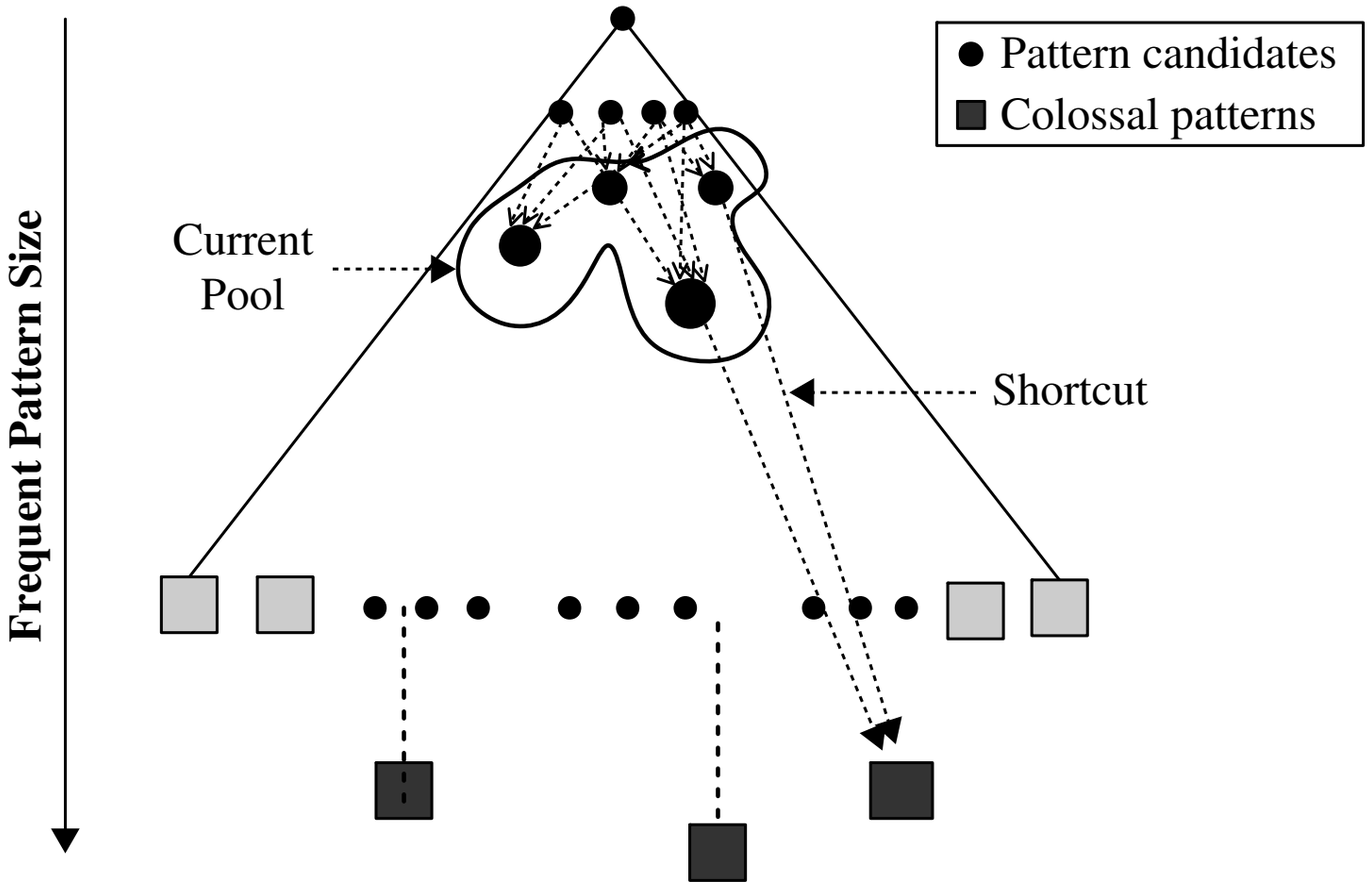
$$Dist(\alpha, \beta) = 1 - \frac{|D_\alpha \cap D_\beta|}{|D_\alpha \cup D_\beta|}$$

If two patterns α and β are both core patterns of a same pattern, they would be bounded by a "ball" of a radius specified by their core ratio τ

$$Dist(\alpha, \beta) \leq 1 - \frac{1}{2/\tau - 1}$$

Once we identify one core pattern, we will be able to find all the other core patterns by a bounding ball of radius r(τ)

KEY INSIGHT: PATTERN FUSION



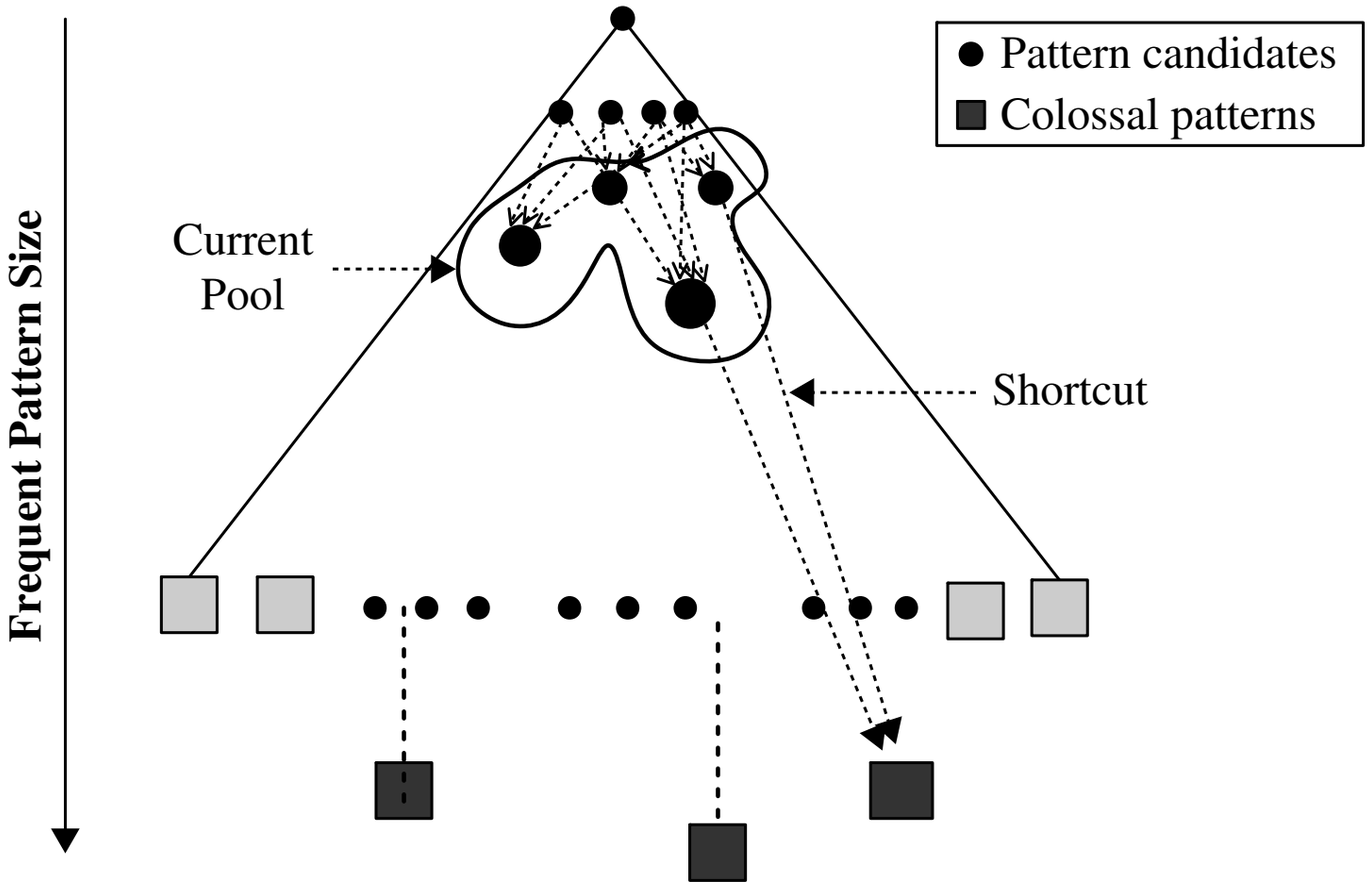
Generate a complete set of frequent patterns up to a small size

Randomly pick a pattern β, and β has a high probability to be a core-descendant of some colossal pattern α

Identify all α's descendants in this complete set, and merge all of them — This would generate a much larger core-descendant of α

In the same fashion, we select K patterns. This set of larger core-descendants will be the candidate pool for the next iteration

PATTERN FUSION: ALGORITHM



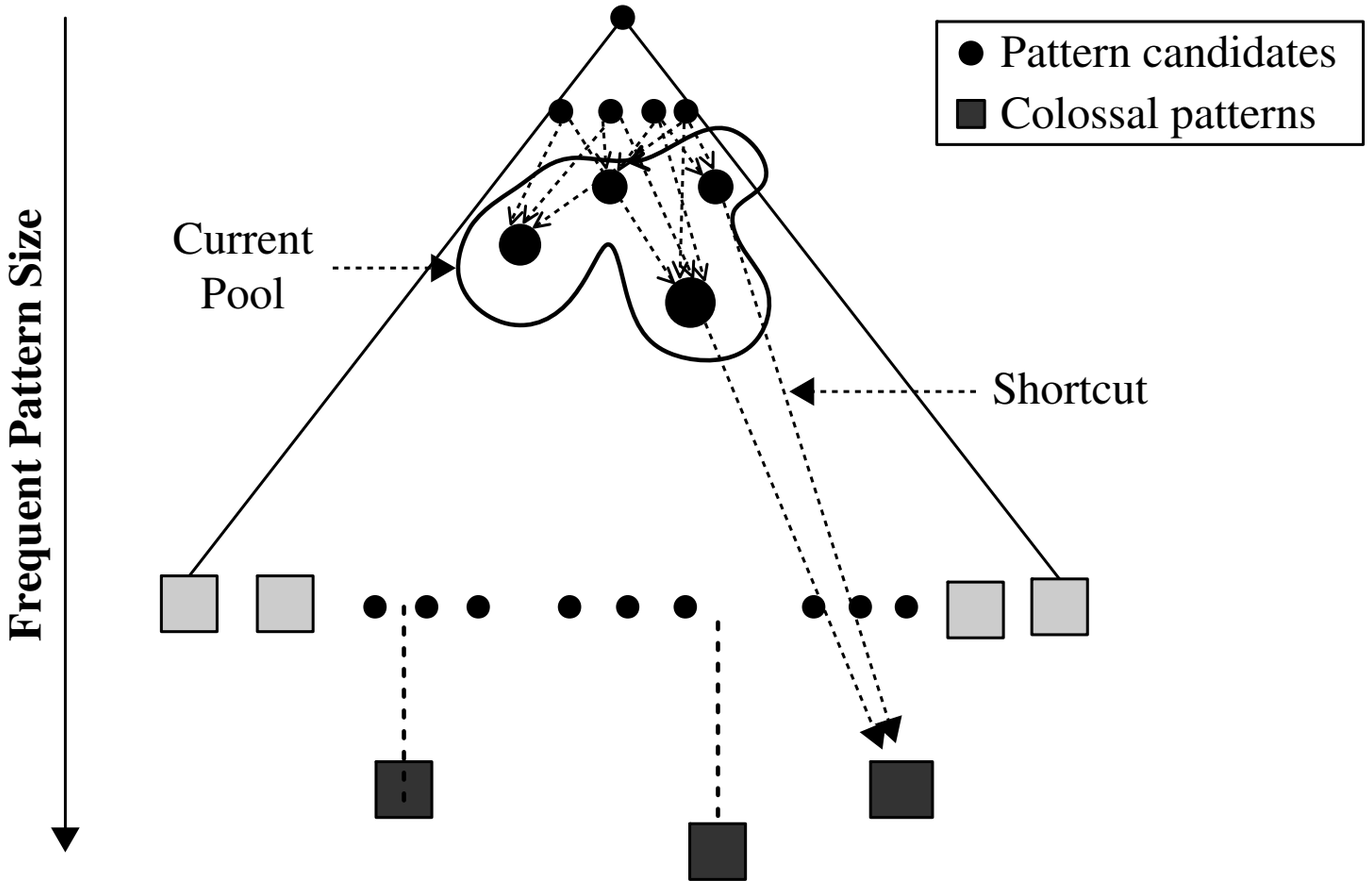
Initialization (Initial pool): Use an existing algorithm to mine all frequent patterns up to a small size, e.g., 3

Iteration (Iterative Pattern Fusion):

- At each iteration, k seed patterns are randomly picked from the current pattern pool
- For each seed pattern thus picked, we find all the patterns within a bounding ball centered at the seed pattern
- All these patterns found are fused together to generate a set of super-patterns. All the super-patterns thus generated form a new pool for the next iteration

Termination: when the current pool contains no more than K patterns at the beginning of an iteration

PATTERN FUSION: EFFICIENCY



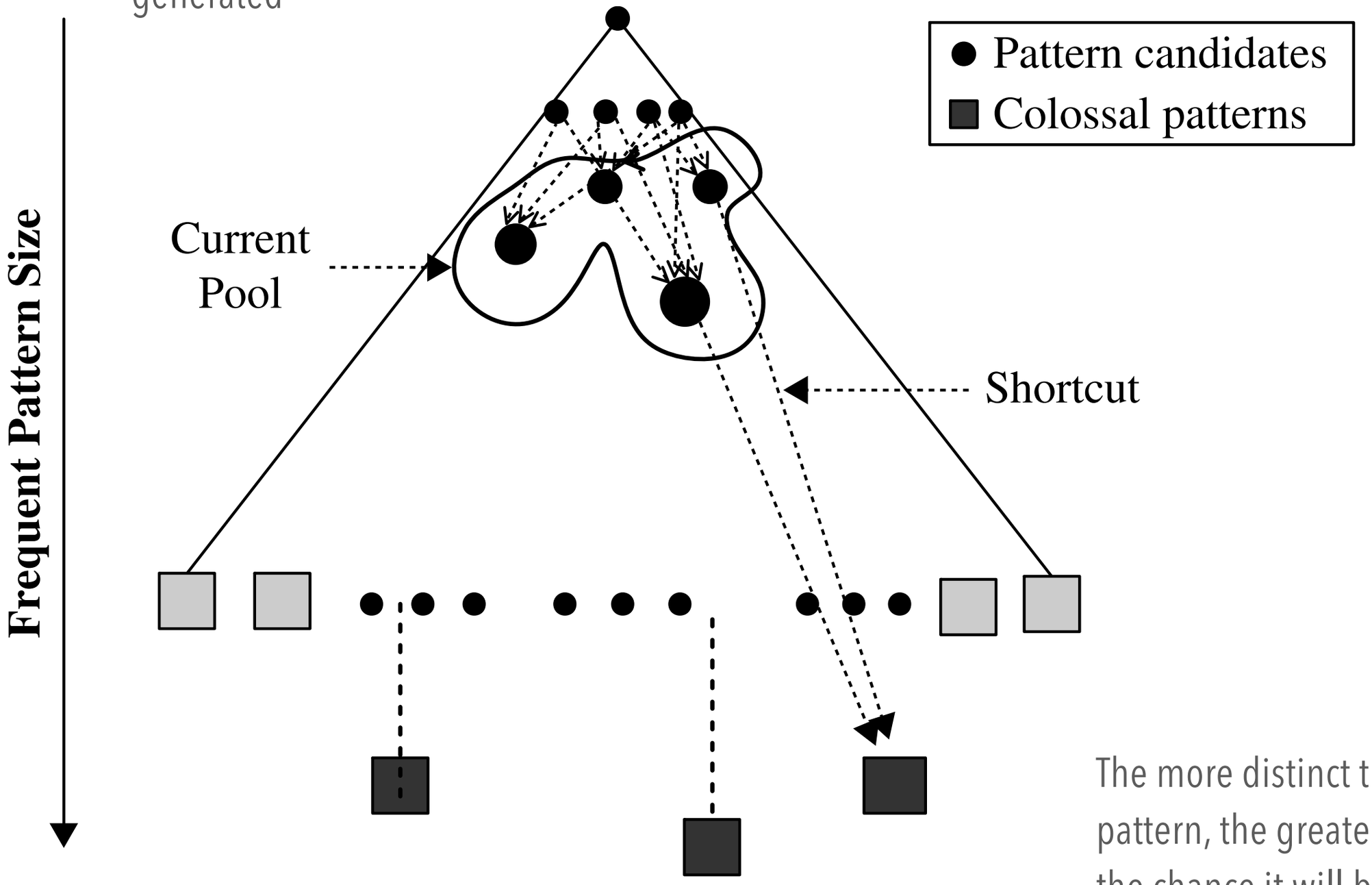
A bounded-breadth pattern tree traversal

- It avoids explosion in mining mid-sized ones
- Randomness comes to help to stay on the right path

Ability to identify "short-cuts" and take "leaps"

- fuse small patterns together in one step to generate new patterns of significant sizes

The larger the pattern, the greater the chance it will be generated



The more distinct the pattern, the greater the chance it will be generated

experiments

DATASETS

Synthetic data set

Diag-n an n x (n-1) table where ith row has integers from 1 to n except i. Each row is taken as an itemset. min_sup=n/2.

Real data set

Replace: A program trace data set collected from the "replace" program, widely used in software engineering research

ALL: A popular gene expression data set, a clinical data on ALL-AML leukemia (www.broad.mit.edu/tools/data.html).

Each item is a column, representing the activitiy level of gene/protein in the sample

Frequent pattern would reveal important correlation between gene expression patterns and disease outcomes

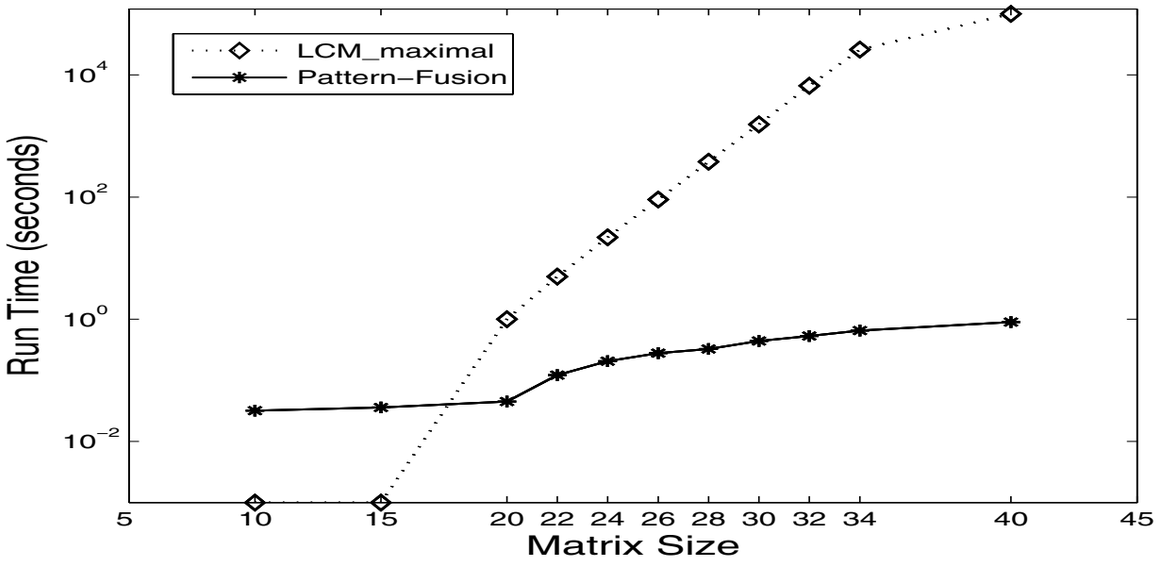


Figure 6. Run Time on $Diag_n$

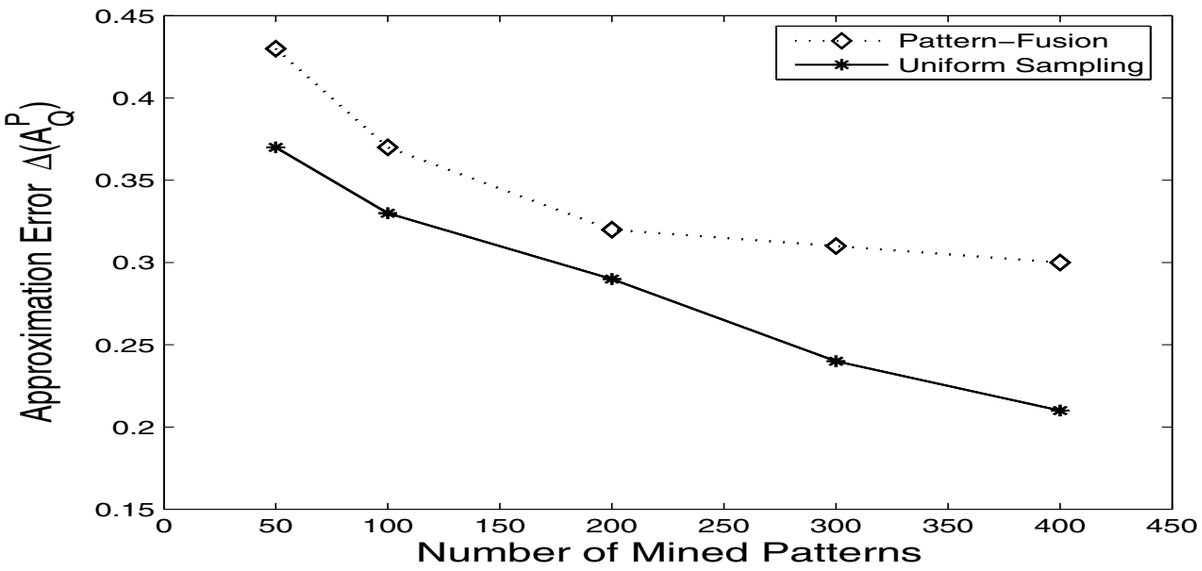


Figure 7. Approximation Error on $Diag_n$

The mining result is compared with the complete set S each of which is a pattern of size 20. Since the complete set S is too big, the complete set is randomly sampled for comparison.

RESULTS ON DIAG-N

LCM run time increases exponentially with pattern size n

LCM maximal: This is a maximal pattern mining algorithm. It is easy to observe that as n increases, the running time of LCM maximal increases exponentially since the number of patterns equals $\binom{n}{n/2}$ (**min sup** = n/2), rendering the time cost unaffordable even for a moderate value of n.

Pattern-Fusion finishes efficiently

The approximation error of Pattern-Fusion (with min-sup 20) in comparison with the complete set) is rather close to uniform sampling (which randomly picks K patterns from the complete answer set)

Definition 10 (Approximation Error) *The* approximation error *of an approximation* A_Q^P *is denoted as* $\Delta(A_Q^P)$,

$$\Delta(A_Q^P) = \frac{\sum_{i=1}^{m} r_i}{m}$$

where $r_i = \max_{\beta \in Q_i} \frac{Edit(\beta, \alpha_i)}{|\alpha_i|}$

The approximation error $\Delta(A_Q^P)$ gives the average maximum pattern distance between *any* pattern in the complete set Q and some pattern in the mining result set P. Hence, the smaller the approximation error, the better P approximates Q, in the sense that P has better representatives of the patterns in Q. See the following example.

Deep Dive!

Zhu, Feida, Xifeng Yan, Jiawei Han, Philip S. Yu, and Hong Cheng. "Mining colossal frequent patterns by core pattern fusion." In Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on, pp. 706-715. IEEE, 2007.

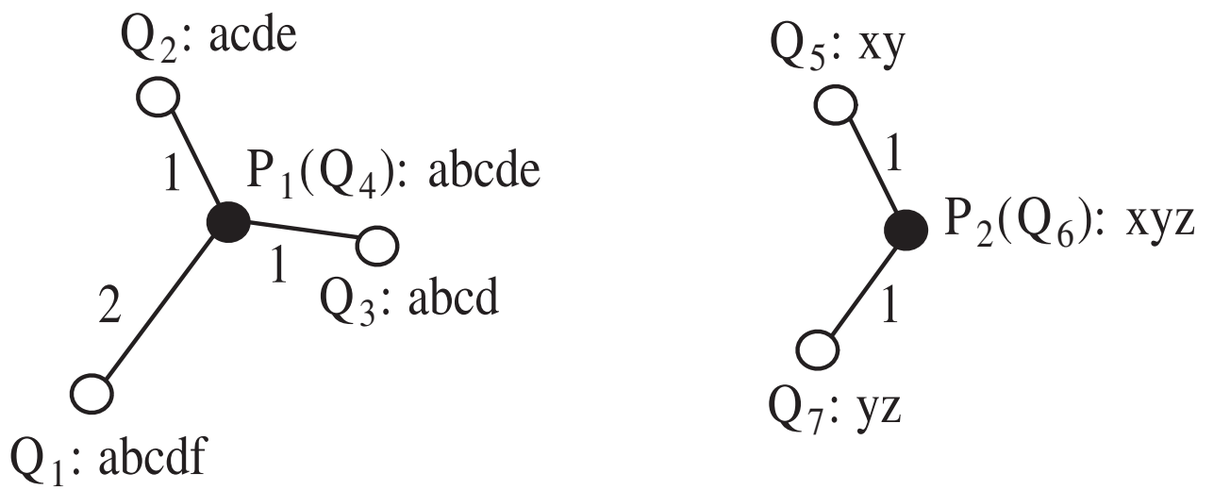


Figure 5. Pattern Set Approximation A_Q^P

$$\Delta(A_Q^P) = \frac{\sum_{i=1}^{m} r_i}{m}$$

where $r_i = \max_{\beta \in Q_i} \frac{Edit(\beta, \alpha_i)}{|\alpha_i|}$

Example 1 *Suppose there is a complete set* $Q = \{Q_1, \ldots, Q_7\}$*, as shown in Figure 5. The approximation set* $P = \{P_1, P_2\}$*, where* $P_1 = Q_4$*, and* $P_2 = Q_6$*. By definition,* Q_1 *is the pattern with the largest distance from* P_1 *in* P_1*'s cluster. Since* $Edit(Q_1, P_1) = 2$ *and* $|P_1| = 5$*, the approximation error of* P_1 *equals* $\frac{2}{5}$*. Similarly,* Q_5 *and* Q_7 *are of equal distance to* P_2 *in* P_2*'s cluster. Since* $Edit(Q_5, P_2) = 1$ *and* $|P_2|$*=3,* $r_2 = \frac{1}{3}$*. Therefore,* $\Delta(A_Q^P) = (\frac{2}{5} + \frac{1}{3})/2 = \frac{11}{30} = 0.37$*. This means, on average, any pattern in* Q *is at most* $0.37 \times 5 \approx 2$ *items away from some pattern in* P*.*

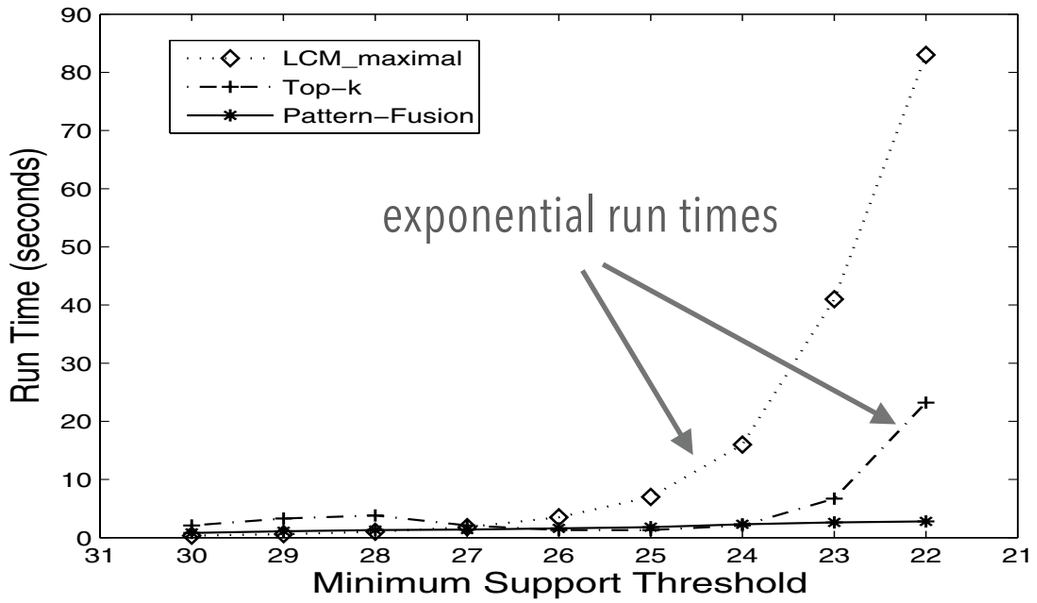
Pattern Size	110	107	102	91	86	84	83
The complete set	1	1	1	1	1	2	6
Pattern-Fusion	1	1	1	1	1	1	4
Pattern Size	82	77	76	75	74	73	71
The complete set	1	2	1	1	1	2	1
Pattern-Fusion	0	2	0	1	1	1	1

Comparing Pattern-Fusion's mining result (K = 100) against the complete set of frequent patterns of size > 70, which are the colossal ones for this data. In fact, Pattern-Fusion is able to get all the largest colossal patterns with size greater than 85.

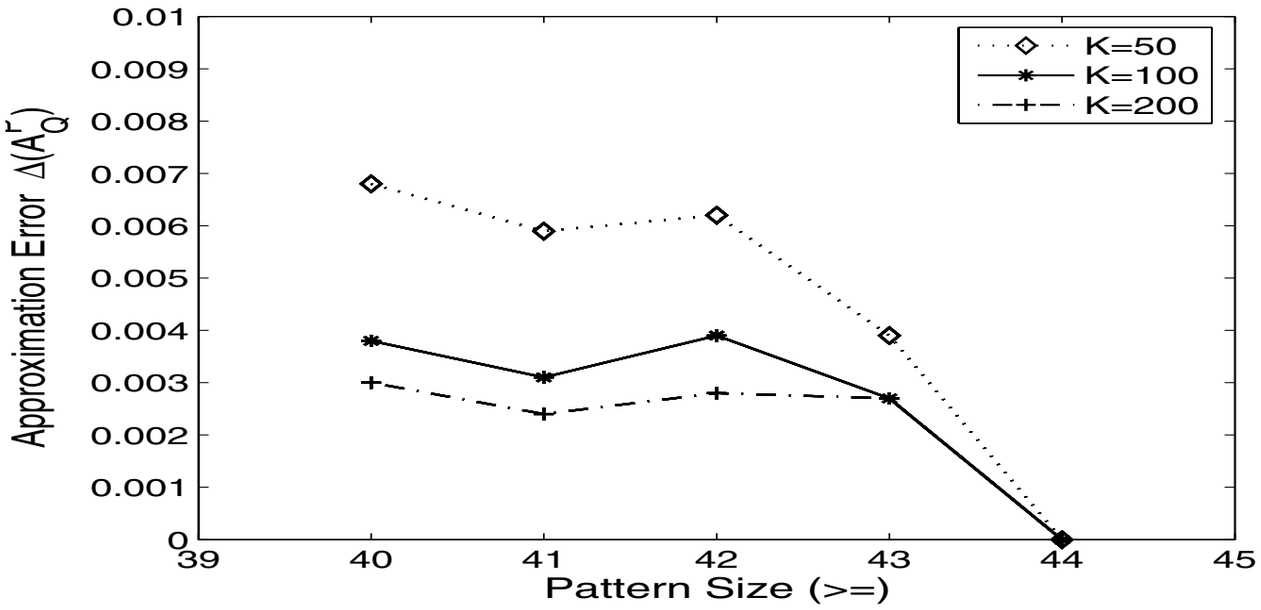
RESULTS ON ALL

ALL: A popular gene expression data set with 38 transactions, each with 866 columns

There are 1736 items in total



RESULTS ON REPLACE



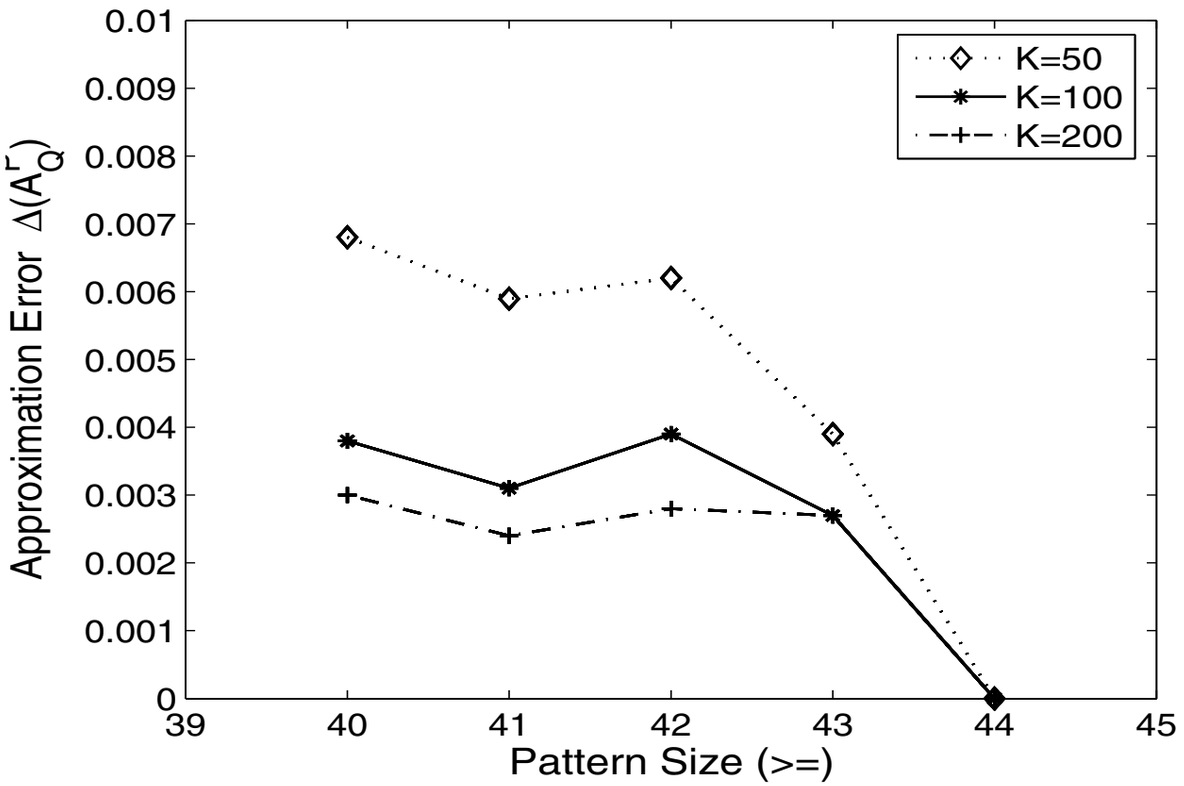
A program trace data set, recording 4395 calls and transitions

The data set contains 4395 transactions with 57 items in total

With support threshold of 0.03, the largest patterns are of size 44

They are all discovered by Pattern-Fusion with different settings of K and τ, when started with an initial pool of 20948 patterns of size ≤3

RESULTS ON REPLACE



Approximation error when compared with the complete mining result

Example. Out of the total 98 patterns of size ≥42, when K=100, Pattern-Fusion returns 80 of them

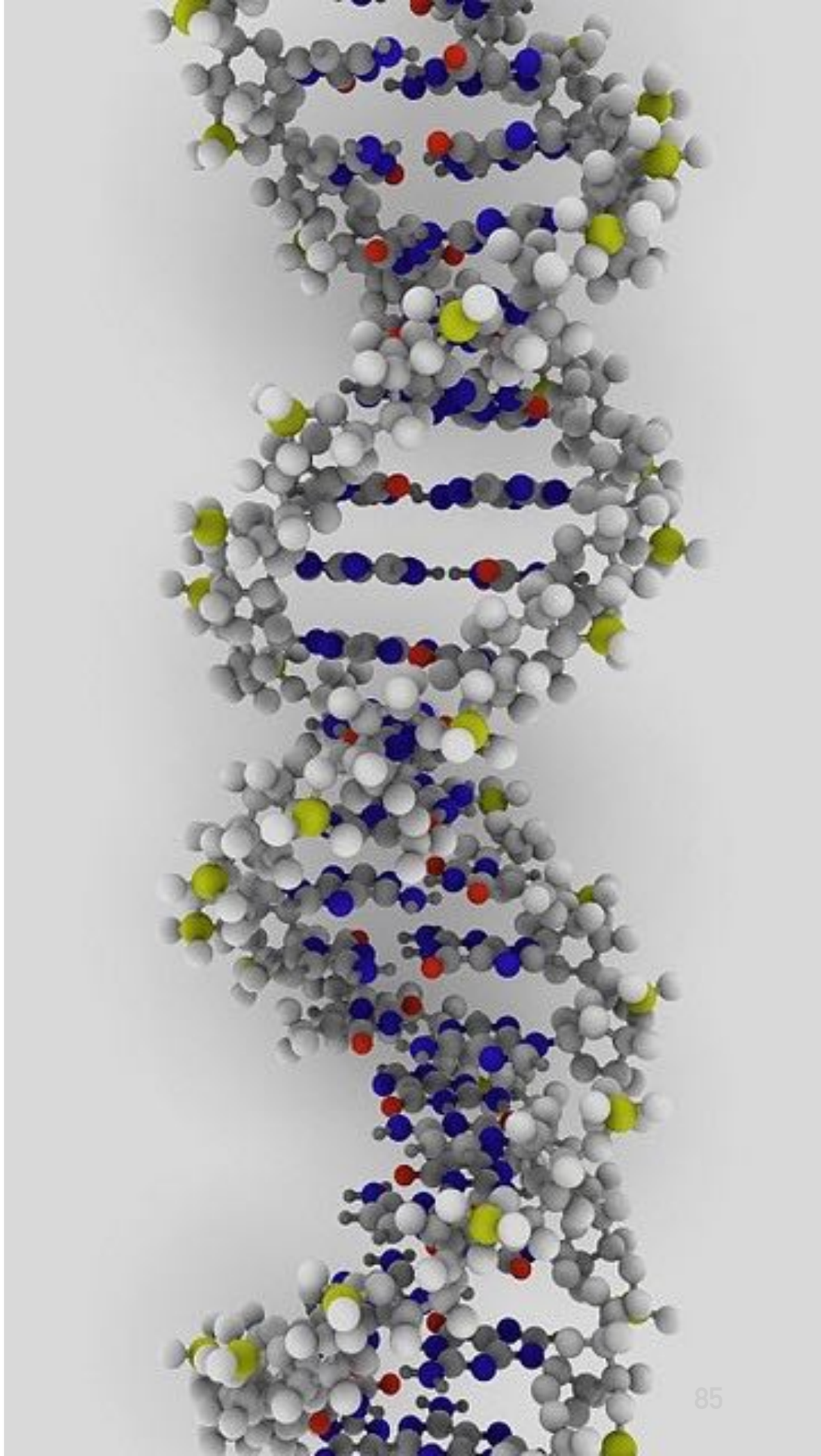
A good approximation to the colossal patterns in the sense that any pattern in the complete set is on average at most 0.17 items away from one of these 80 patterns

APPROXIMATE PATTERNS

Road Map Multidimensional

Constraint Based Mining High Dimensional Patterns

Sequential Patterns Graph Patterns Summary



ID	Item-Sets	Support
P1	{38,16,18,12}	205227
P2	{38,16,18,12,17}	205211
P3	{39,38,16,18,12,17}	101758
P4	{39,16,18,12,17}	161563
P5	{39,16,18,12}	161576

Closed frequent pattern

P1, P2, P3, P4, P5

Emphasize too much on support

no compression

Max-pattern, P3: info loss

A desirable output: P2, P3, P4

Xin, D., Han, J., Yan, X. and Cheng, H., 2005, August. Mining compressed frequent-pattern sets. In Proceedings of the 31st international conference on Very large data bases (pp. 709-720). VLDB Endowment.

DELTA CLUSTERING

Why compressed patterns?

too many, but less meaningful

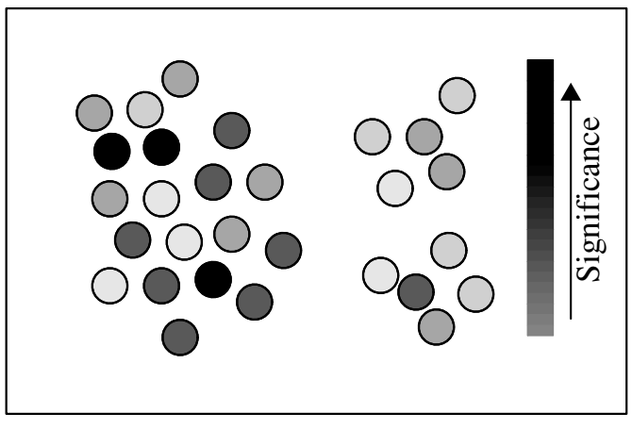
Pattern distance measure

$$D(P_1, P_2) = 1 - \frac{|T(P_1) \cap T(P_2)|}{|T(P_1) \cup T(P_2)|}$$

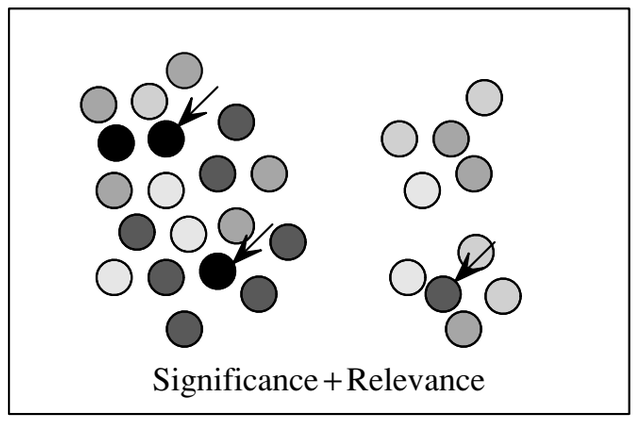
δ-clustering: For each pattern P, find all patterns which can be expressed by P and their distance to P are within δ (δ-cover)

All patterns in the cluster can be represented by P

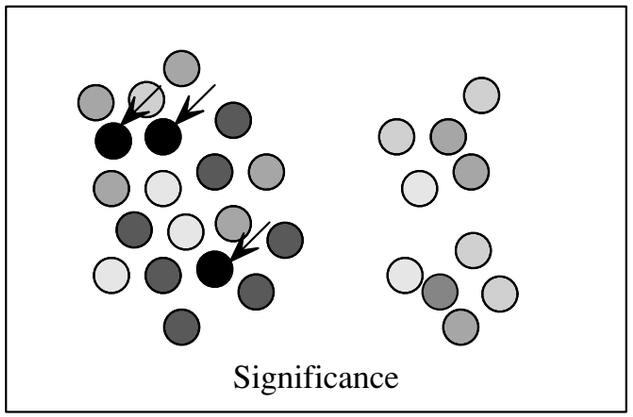
REDUNDANCY AWARE CLUSTERING



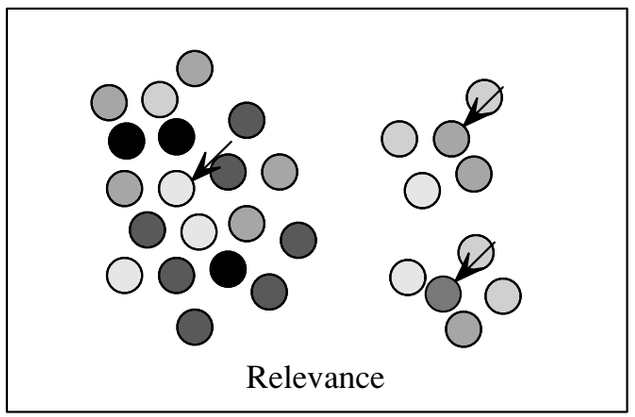
(a)



(b)



(c)



(d)

Desired patterns: high significance & low redundancy

Propose the MMS (Maximal Marginal Significance) for measuring the combined significance of a pattern set

Xin, D., Cheng, H., Yan, X. and Han, J., 2006, August. Extracting redundancy-aware top-k patterns. In Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 444-453). ACM.

SEQUENTIAL PATTERNS

Road Map Multidimensional

Constraint Based Mining High Dimensional Patterns

Approximate Patterns Graph Patterns Summary

SEQUENTIAL PATTERNS

Transaction databases, time-series databases vs. sequence databases

Frequent patterns vs. (frequent) sequential patterns

Applications of sequential pattern mining

- Customer shopping sequences:
  - First buy computer, then CD-ROM, and then digital camera, within 3 months.
- Medical treatments, natural disasters (e.g., earthquakes), science & eng. processes, stocks and markets, etc.
- Telephone calling patterns, Weblog click streams
- Program execution sequence data sets
- DNA sequences and gene structures

< (ef) (ab) (df) c b >

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 a set of sequences, find the complete set of frequent subsequences

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

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

sequential patterns

Given a set of sequences, find the complete set of frequent subsequences

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

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

Sequential pattern mining: find the complete set of patterns, satisfying the minimum support threshold

MINING ALGORITHMS

Concept introduction and an initial Apriori-like algorithm

- Agrawal & Srikant: Mining sequential patterns, ICDE'95

Requirement: efficient, scalable, complete, minimal database scans, and be able to incorporate various kinds of user-specific constraints

Representative algorithms

- GSP (Generalized Sequential Patterns): Srikant & Agrawal @ EDBT'96)
- Vertical format-based mining: SPADE (Zaki@Machine Leanining'00)
- Pattern-growth methods: PrefixSpan (Pei, Han et al. @ICDE'01)

Constraint-based sequential pattern mining (SPIRIT: Garofalakis, Rastogi, Shim@VLDB'99; Pei, Han, Wang @ CIKM'02)

Mining closed sequential patterns: CloSpan (Yan, Han et al. @SDM'03)

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)>

APRIORI PROPERTY

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 → <hab> and <(ah)b> are infrequent

GSP MINING ALGORITHM

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)>

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

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

L-1 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

36

	<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>

L-2 PATTERNS

51 GSP candidates

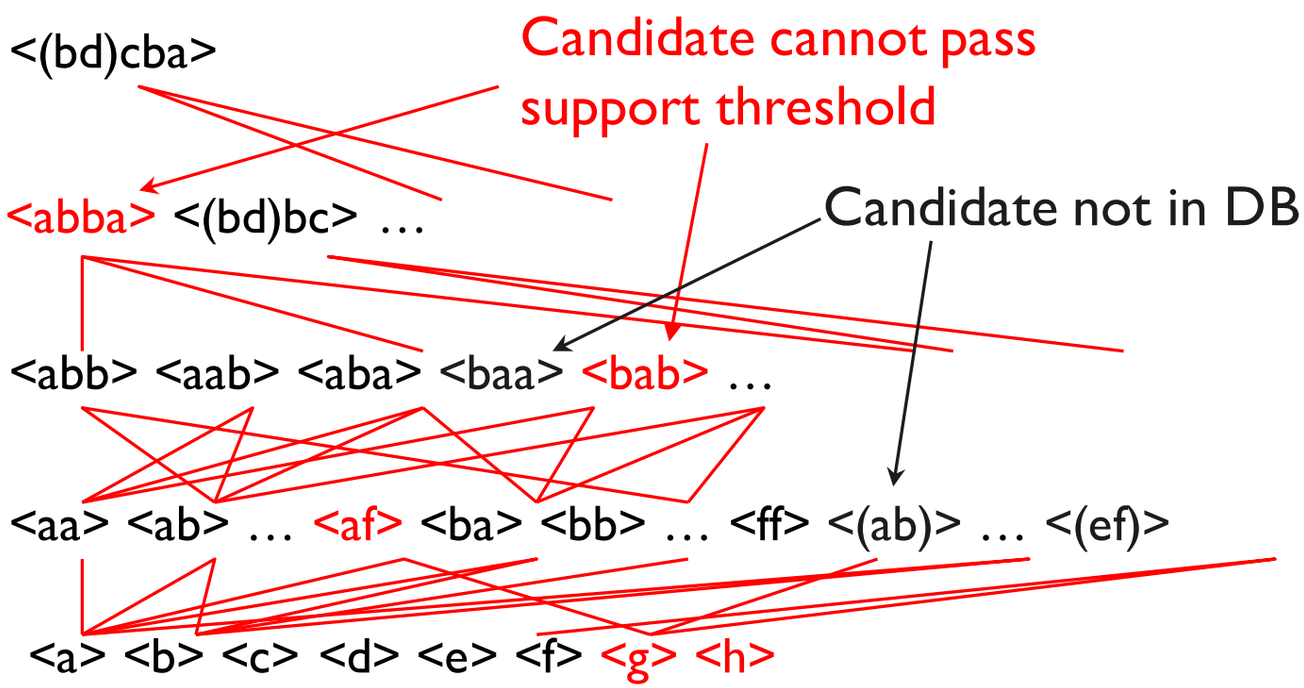
Without Apriori property,

8*8+8*7/2=92 candidates

	<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>						

15

THE GSP PROCESS



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)>

1st scan: 8 candidates, 6 length-1 sequence pattern

2nd scan: 51 candidates, 19 length-2 sequence pattern; 10 candidates not in DB at all

3rd scan: 46 candidates; 19 length-3 sequence patterns; 20 candidates not in DB at all

4th scan: 8 candidates; 6 length-4 sequence patterns

5th scan: 1 candidate; 1 length-5 sequence pattern

SID: sequence id EID: event id

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

Shopping example:

Events are itemsets bought during one shopping trip

Time-ordering the events will generate the **sequence**

SPADE

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

M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. Machine Learning, 42(1):31–60, 2001.

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			

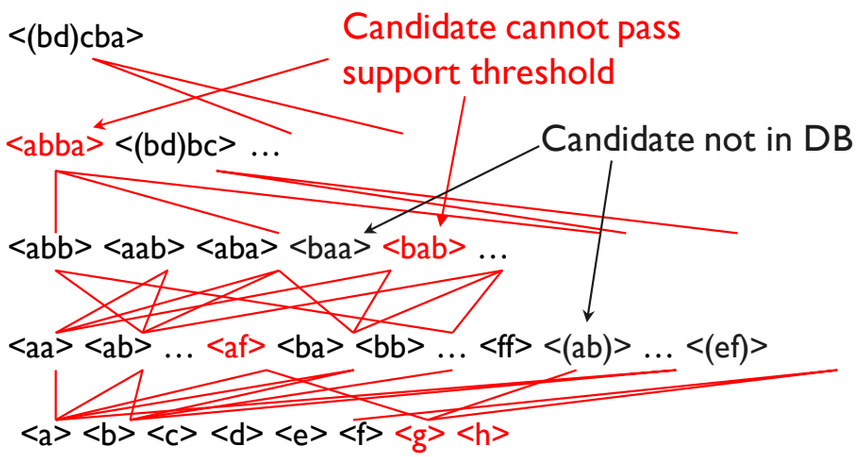
ab			ba			···
SID	EID (a)	EID(b)	SID	EID (b)	EID(a)	···
1	1	2	1	2	3	
2	1	3	2	3	4	
3	2	5				
4	3	5				

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

BOTTLENECKS: GSP, SPADE

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

SPADE



GSP

A huge set of candidates could be generated

1,000 frequent length-1 sequences generates huge number of length-2 candidates! (~1.5M!)

Multiple scans of database while mining

Mining long sequential patterns by growing from shorter patterns

- Needs an exponential number of short candidates
- A length-100 sequential pattern needs 10^{30} candidate sequences!

PREFIX MINING

min_sup=2

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

The **number** of instances of items in a sequence is called the length of the sequence.

J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M.-C. Hsu. Mining sequential patterns by pattern-growth: the prefixspan approach. IEEE Transactions on Knowledge and Data Engineer- ing, 16(11):1424–1440, Nov 2004.

The set of items in the database: {a; b; c; d; e; f; g}.

A sequence <a(abc)(ac)d(cf)> has five elements: (a), (abc), (ac), (d) and (cf), where items a and c appear more than once, respectively, in different elements. It is a **9**-sequence since there are nine instances appearing in that sequence. Item a appears **three** times in this sequence, so it contributes **3** to the length of the sequence. However, the whole sequence <a(abc)(ac)d(cf)> contributes only **1** to the support of <a>.

Also, sequence <a(bc)df> is a subsequence of <a(abc)(ac)d(cf)>.

Since both sequences 10 and 30 contain subsequence s=<(ab)c>, s is a sequential pattern of length 3 (i.e., 3-pattern).

Can we mine sequential patterns by extension of the FP-tree structure?

NO!

It is easy to explore the sharing among a set of unordered items, but it is difficult to explore the sharing of common data structures among a set of ordered items.

For example, a set of frequent itemsets {abc, cbad, ebadc, cadb} share the same tree branch {abcde} in the FP-tree.

However, if they were a set of sequences, there is no common prefix subtree structure that can be shared among them because one cannot change the order of items to form sharable prefix subsequences.

Definition: **Prefix**

Definition 1 (Prefix). *Suppose all the items within an element are listed alphabetically. Given a sequence* $\alpha = \langle e_1 e_2 \cdots e_n \rangle$ *(where each* e_i *corresponds to a frequent element in* S*), a sequence* $\beta = \langle e'_1 e'_2 \cdots e'_m \rangle$ $(m \leq n)$ *is called a prefix of* α *if and only if 1)* $e'_i = e_i$ *for* $(i \leq m-1)$*; 2)* $e'_m \subseteq e_m$*; and 3) all the frequent items in* $(e_m - e'_m)$ *are alphabetically after those in* e'_m*.*

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

For example, $\langle a \rangle$, $\langle aa \rangle$, $\langle a(ab) \rangle$, and $\langle a(abc) \rangle$ are *prefixes* of sequence $s = \langle a(abc)(ac)d(cf) \rangle$, but neither $\langle ab \rangle$ nor $\langle a(bc) \rangle$ is considered as a prefix if every item in the prefix $\langle a(abc) \rangle$ of sequence s is frequent in S.

Definition: Suffix

Definition 2 (Suffix). *Given a sequence* $\alpha = \langle e_1 e_2 \cdots e_n \rangle$ *(where each* e_i *corresponds to a frequent element in S). Let* $\beta = \langle e_1 e_2 \cdots e_{m-1} e'_m \rangle$ $(m \leq n)$ *be the prefix of* α*. Sequence* $\gamma = \langle e''_m e_{m+1} \cdots e_n \rangle$ *is called the suffix of* α *with regards to prefix* β*, denoted as* $\gamma = \alpha/\beta$*, where* $e''_m = (e_m - e'_m)$*. We also denote* $\alpha = \beta \cdot \gamma$*. Note, if* β *is not a subsequence of* α*, the suffix of* α *with regards to* β *is empty.*

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

For example, for the sequence $s = \langle a(abc)(ac)d(cf) \rangle$, $\langle (abc)(ac)d(cf) \rangle$ is the *suffix* with regards to the prefix $\langle a \rangle$, $\langle (\_bc)(ac)d(cf) \rangle$ is the *suffix* with regards to the prefix $\langle aa \rangle$, and $\langle (\_c)(ac)d(cf) \rangle$ is the *suffix* with regards to the prefix $\langle a(ab) \rangle$.

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

Prefix	*Suffix* (Prefix-Based *Projection*)
<a>	<(abc)(ac)d(cf)>
<aa>	<(_bc)(ac)d(cf)>
<a(ab)>	<(_c)(ac)d(cf)>

PREFIX MINING

Prefix and suffix

Given sequence <a(abc)(ac)d(cf)>

Prefixes: <a>, <aa>, <a(ab)> and <a(abc)>

PrefixSpan Mining framework

Step 1: find length-1 sequential patterns

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

Step 2: divide search space. The complete set of seq. pat. can be partitioned into 6 subsets:

The ones having prefix <a>;

The ones having prefix <b>;

…

The ones having prefix <f>

Notice that (_b) means that the last element in the prefix, which is a, together with b, form one element.

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

identify **frequent 1** sequential items in <a>'s projected database

Moreover, in a sequence containing <a>, only the subsequence prefixed with the **first occurrence** of <a> should be considered.

PREFIX <A>

Only need to consider projections w.r.t. <a>

<a>-projected database:

<(abc)(ac)d(cf)>

<(_d)c(bc)(ae)>

<(_b)(df)cb>

<(_f)cbc>

Find all the length-2 seq. pat. Having prefix <a>: <aa>, <ab>, <(ab)>, <ac>, <ad>, <af>

Further partition into 6 subsets

Having prefix <aa>;

…

Having prefix <af>

completeness of prefix span

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>

<a>-projected database

<(abc)(ac)d(cf)>

<(_d)c(bc)(ae)>

<(_b)(df)cb>

<(_f)cbc>

<b>-projected database

Length-2 sequential patterns

<aa>, <ab>, <(ab)>,

<ac>, <ad>, <af>

<aa>-proj. db

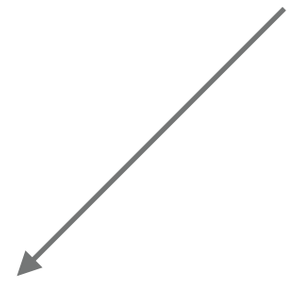
<ab>-proj. db

<c>, <d>, <e>, <f>

Major strength of PrefixSpan:

- No candidate sequence needs to be generated
- Projected databases keep shrinking

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



<a>-projected database

<(abc)(ac)d(cf)>

<(_d)c(bc)(ae)>

<(_b)(df)cb>

<(_f)cbc>

SPEED UP

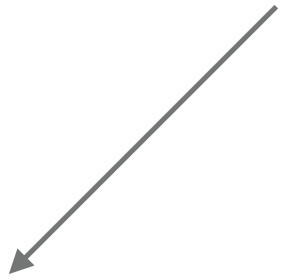
Major cost of PrefixSpan: Constructing projected databases

- Postfixes of sequences often appear repeatedly in recursive projected databases

When (projected) database can be held in main memory, use pointers to form pseudo-projections

- Pointer to the sequence
- Offset of the postfix

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



<a>-projected database

<(abc)(ac)d(cf)>

<(_d)c(bc)(ae)>

<(_b)(df)cb>

<(_f)cbc>

PSEUDO PROJECTION

Pseudo-projection avoids physically copying postfixes

- Efficient in running time and space when database can be held in main memory

However, it is not efficient when database cannot fit in main memory

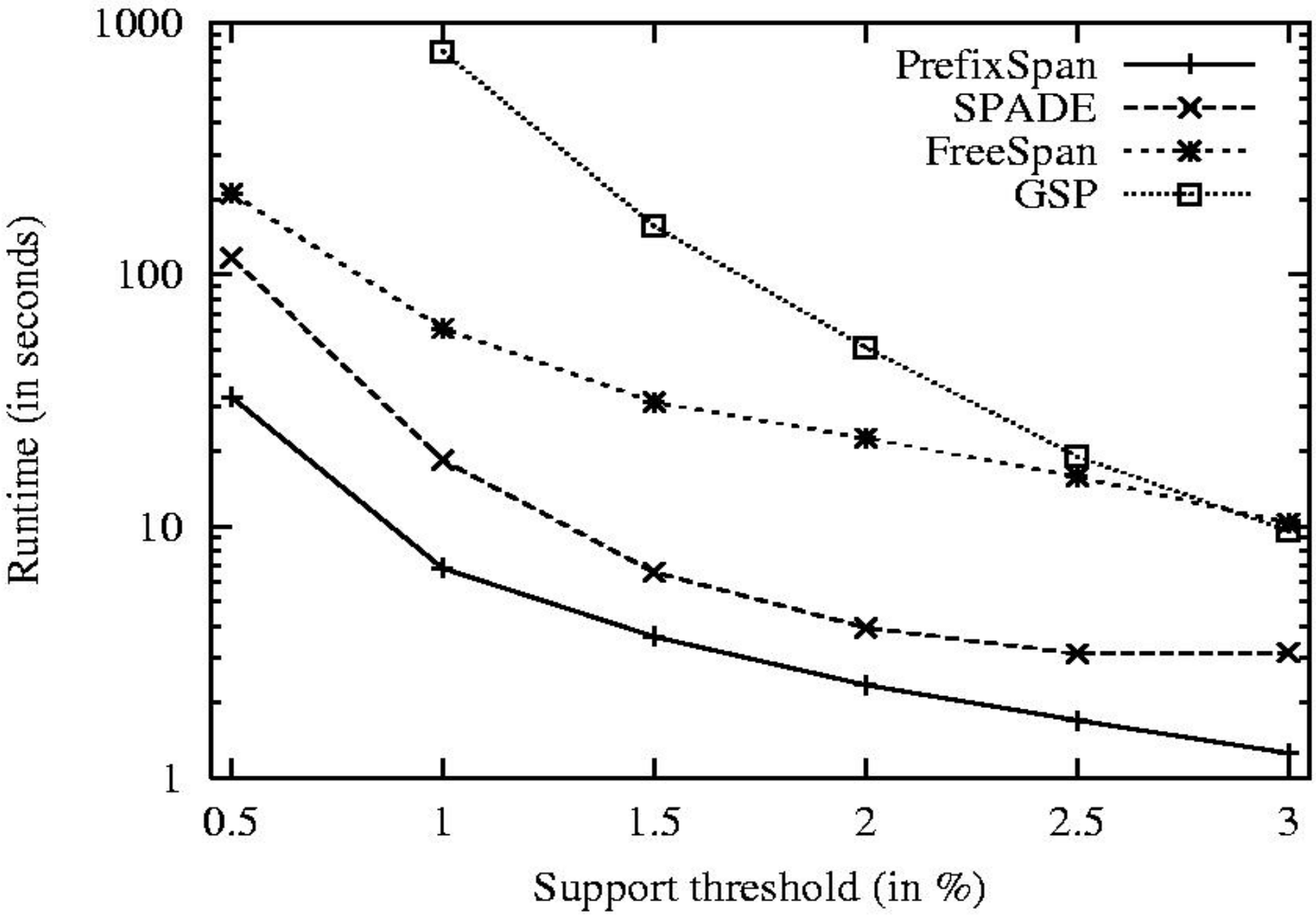
- Disk-based random accessing is very costly

Suggested Approach:

- Integration of physical and pseudo-projection
- Swapping to pseudo-projection when the data set fits in memory

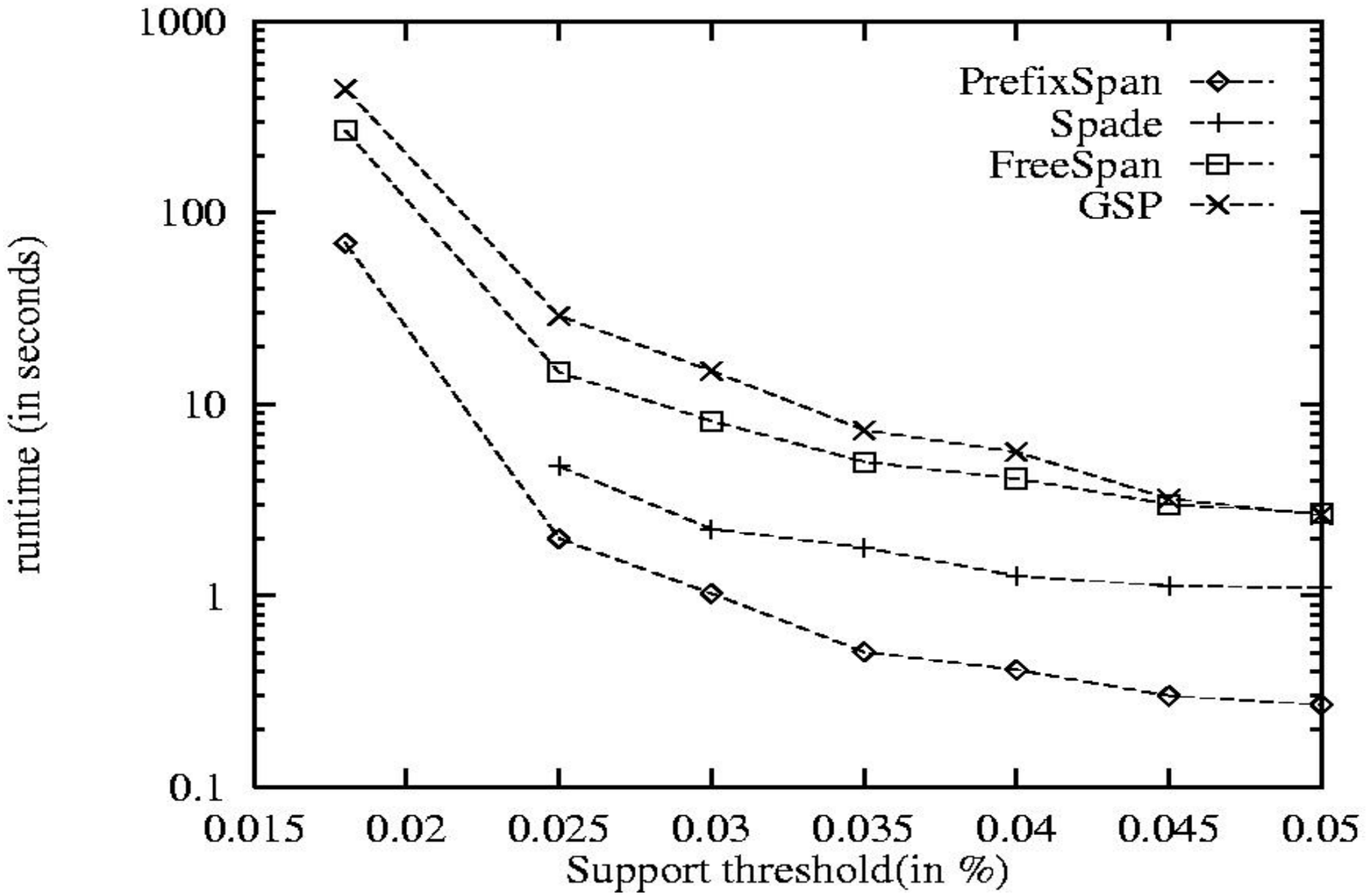
comparisons

Performance comparison on data set C10T8S8I8



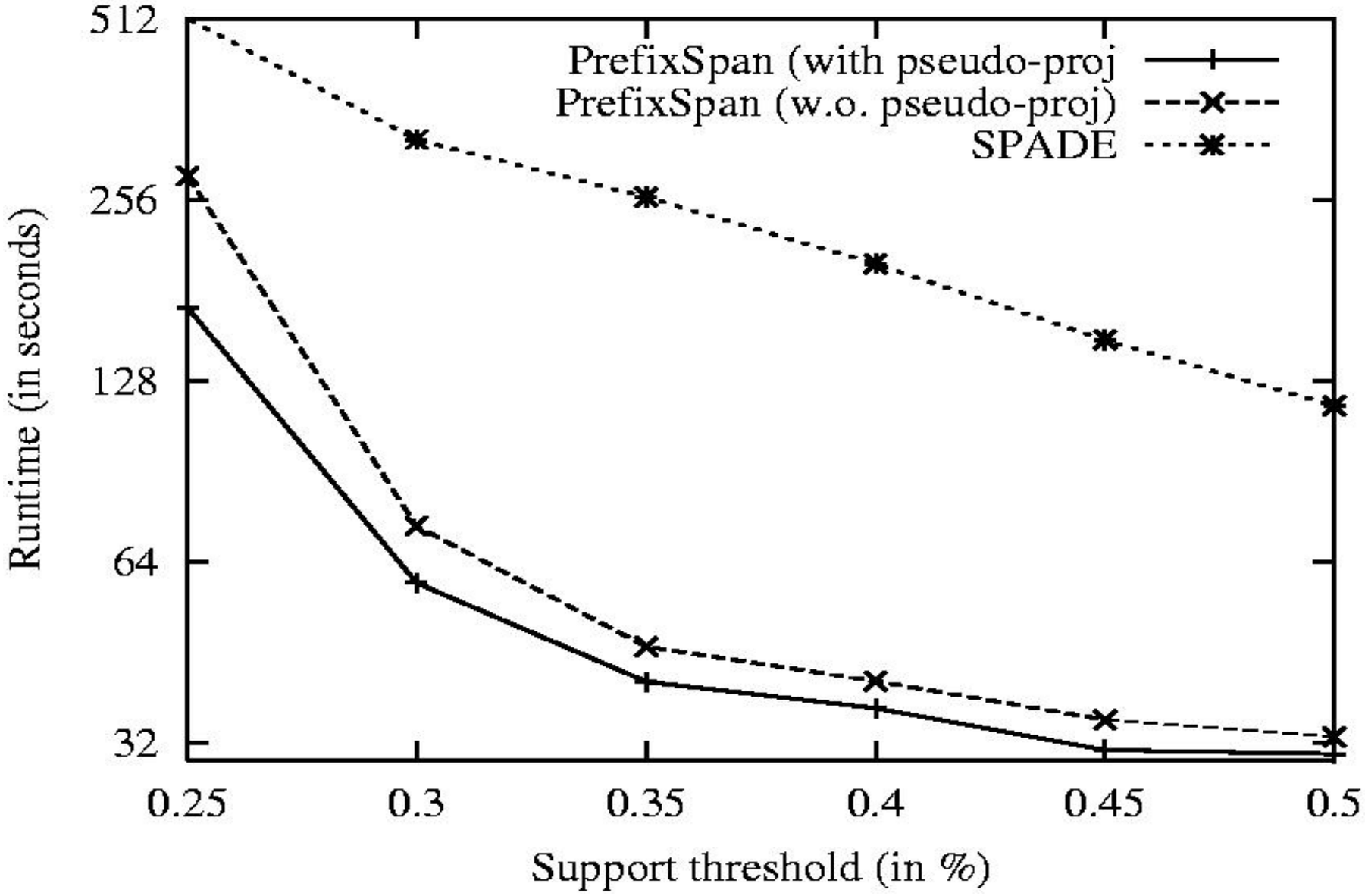
comparisons

Performance comparison on the gazelle data set

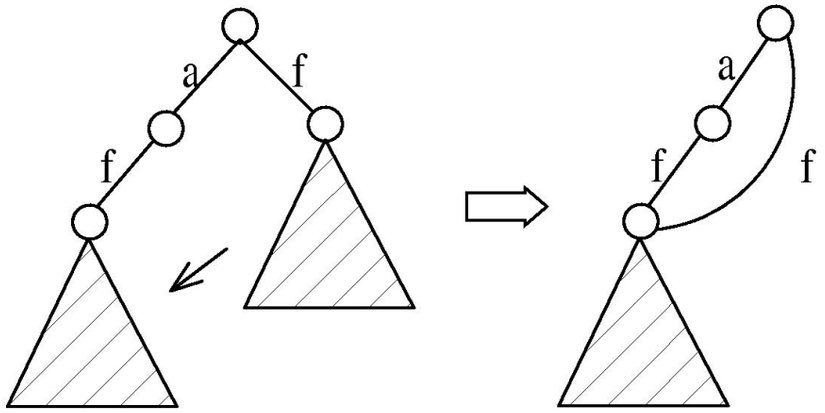


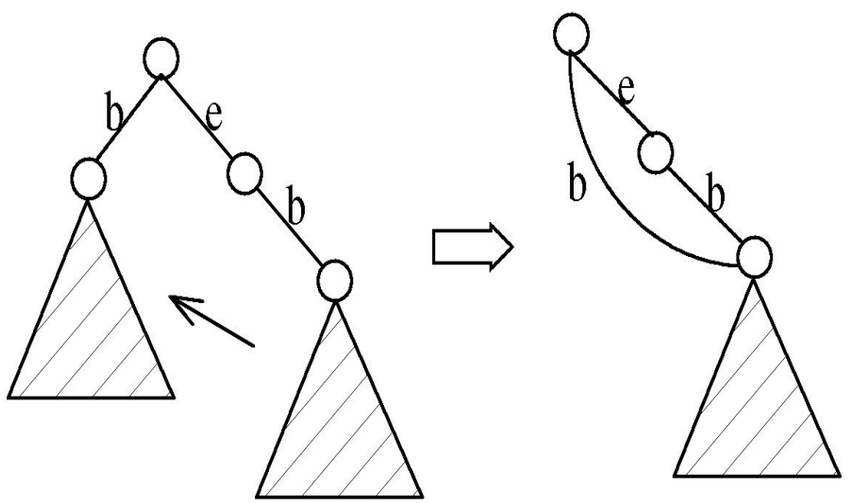
comparisons

Performance comparison: effect of pseudo projection



X. Yan, J. Han, and R. Afshar. Clospan: Mining: Closed sequential patterns in large datasets. In Proceed- ings of the 2003 SIAM International Conference on Data Mining, pages 166–177, 2003.





CLOSPAN: MINING CLOSED SEQUENTIAL PATTERNS

A closed sequential pattern s: there exists no superpattern s' such that s' ⊃ s, and s' and s have the same support

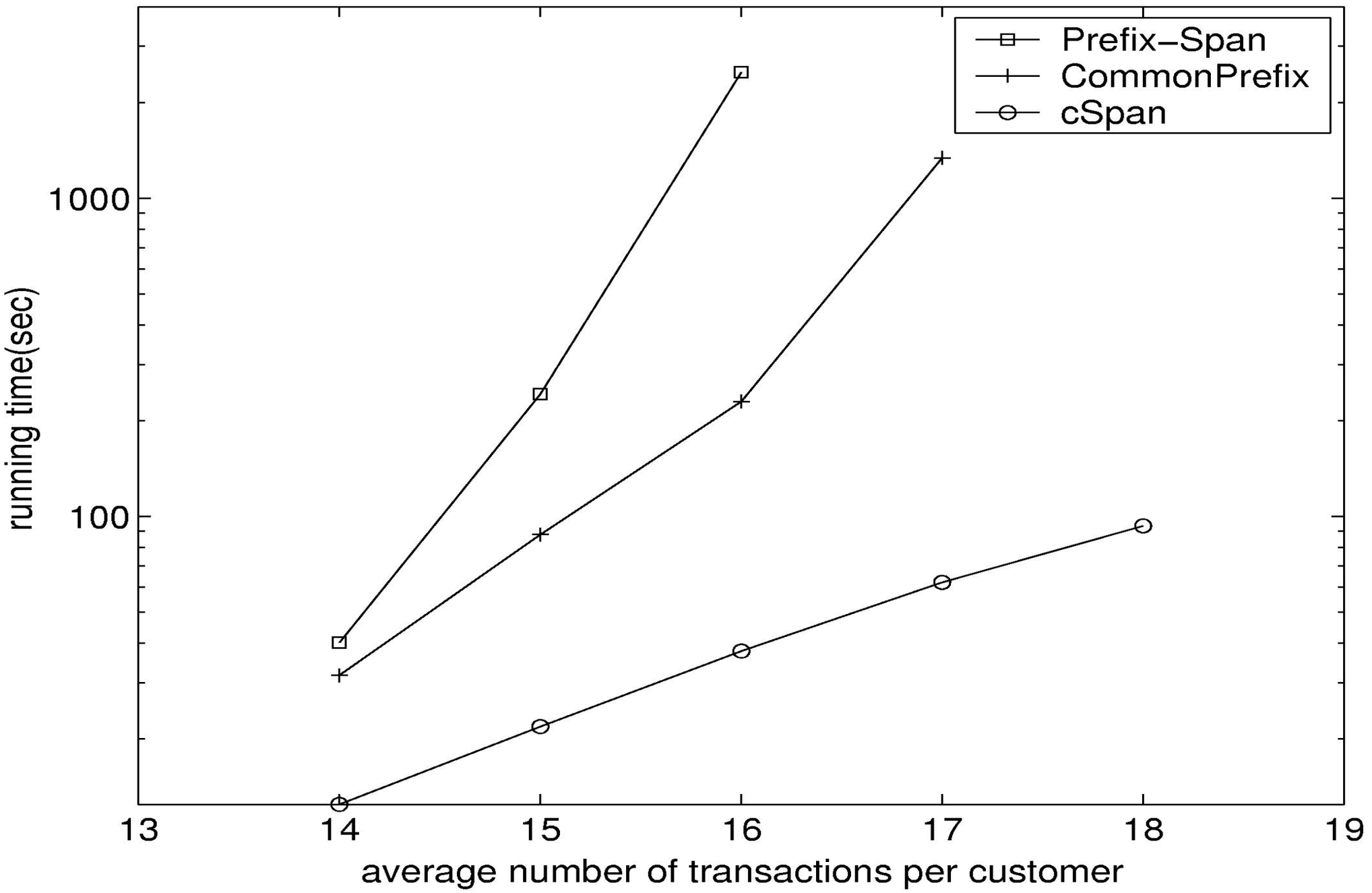
Which one is closed? <abc>: 20, <abcd>:20, <abcde>: 15

Why mine close seq. patterns?

Reduces the number of (redundant) patterns but attains the same expressive power

Property: If s'⊃s, s' is closed iff two project DBs have the same size (total number of elements)

Using Backward Subpattern and Backward Superpattern pruning to prune redundant search space


Prefix–Span
CommonPrefix
cSpan
1000
100
running time(sec)
13
14
15
16
17
18
19
average number of transactions per customer

CONSTRAINT BASED MINING

Constraint-based sequential pattern mining

Constraints: User-specified, for focused mining of desired patterns

How to explore efficient mining with constraints? — Optimization

Classification of constraints

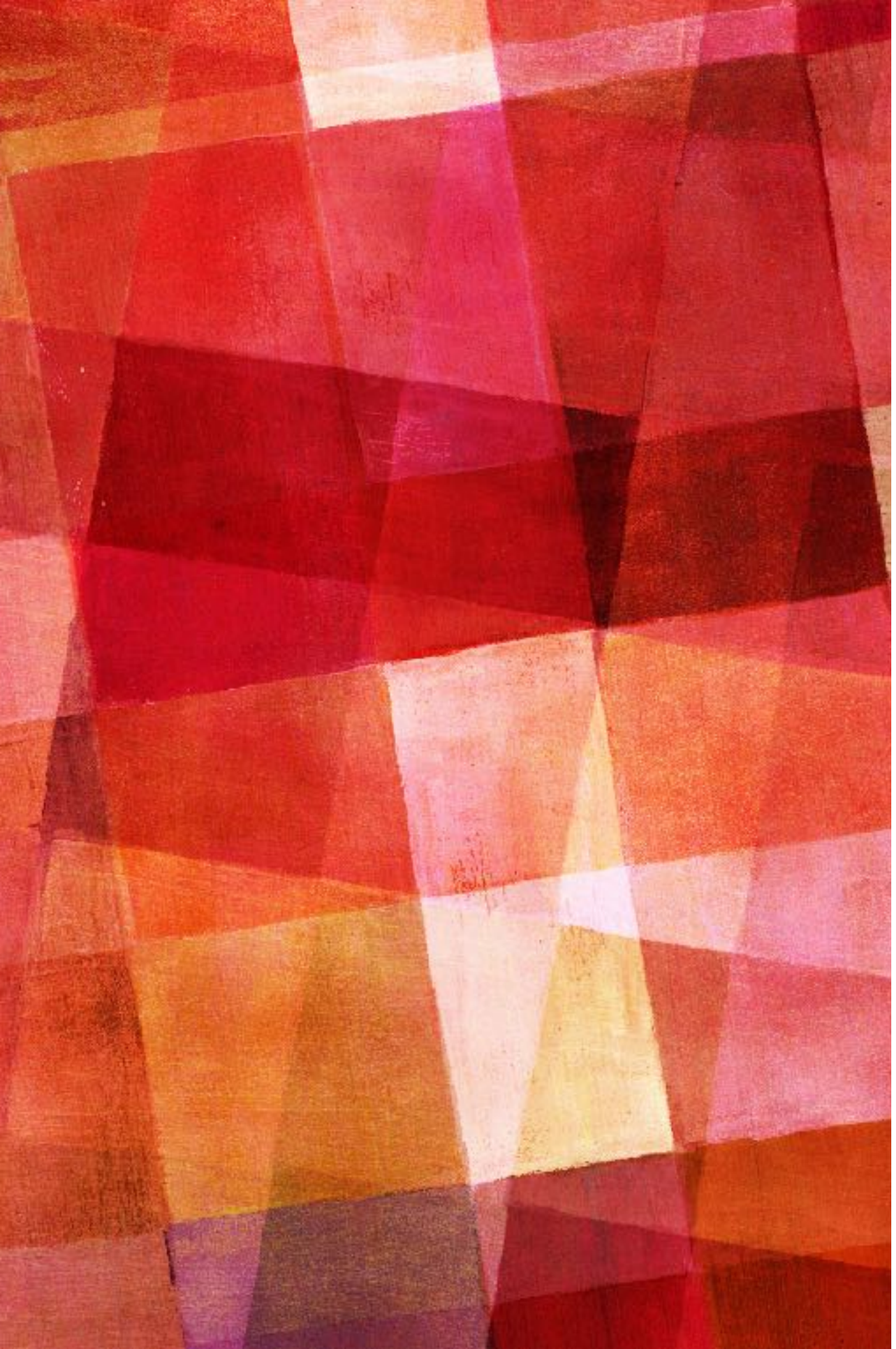
Anti-monotone: E.g., value_sum(S) < 150, min(S) > 10

Monotone: E.g., count (S) > 5, S ⊇{PC, digital_camera}

Succinct: E.g., length(S) ≥10, S ∈ {Pentium, MS/Office, MS/Money}

Convertible: E.g., value_avg(S) < 25, profit_sum (S) > 160, max(S)/avg(S) < 2, median(S) – min(S) > 5

Inconvertible: E.g., avg(S) – median(S) = 0



SEQUENCES TO STRUCTURE

Sets, sequences, trees, graphs, and other structures

Transaction DB: Sets of items

{{i1, i2, …, im}, …}

Seq. DB: Sequences of sets:

{<{i1, i2}, …, {im, in, ik}>, …}

Sets of Sequences:

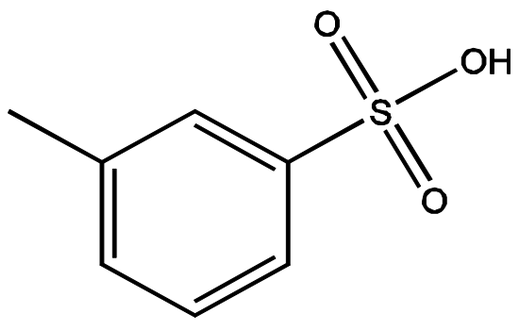
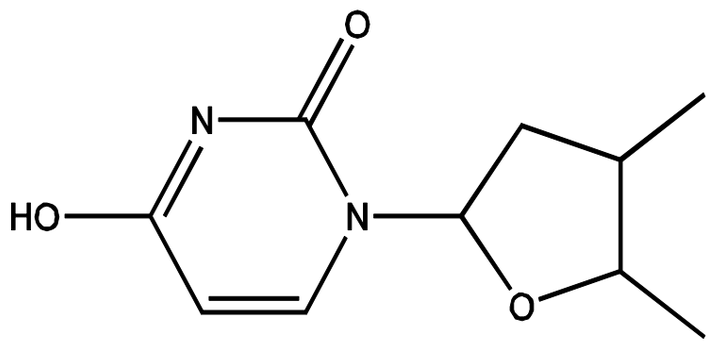
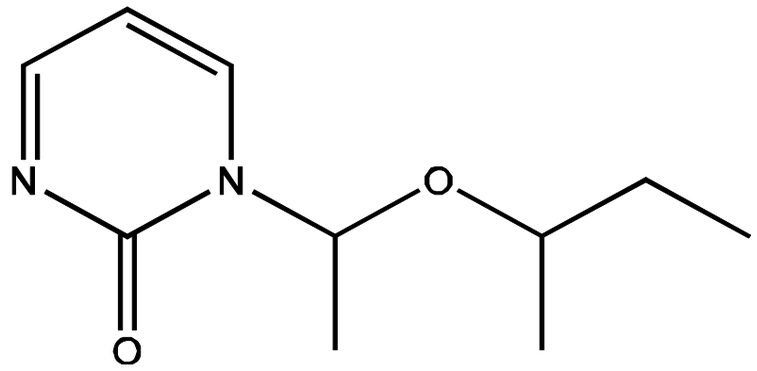
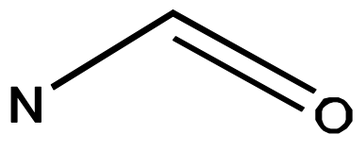
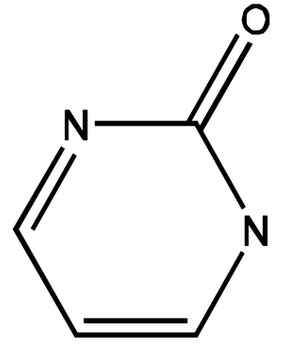
{{<i1, i2>, …, <im, in, ik>}, …}

Sets of trees: {t1, t2, …, tn}

Sets of graphs (mining for frequent subgraphs):

{g1, g2, …, gn}

Mining structured patterns in XML documents, bio-chemical structures, etc.

GRAPH PATTERNS

Road Map Multidimensional

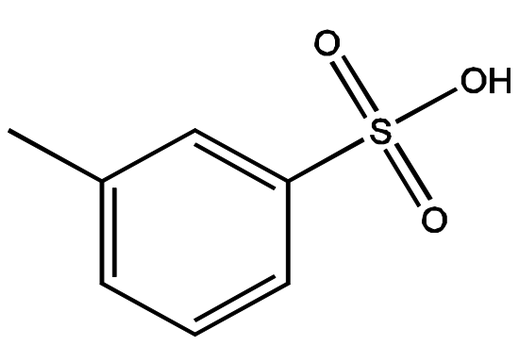
Constraint Based Mining High Dimensional Patterns

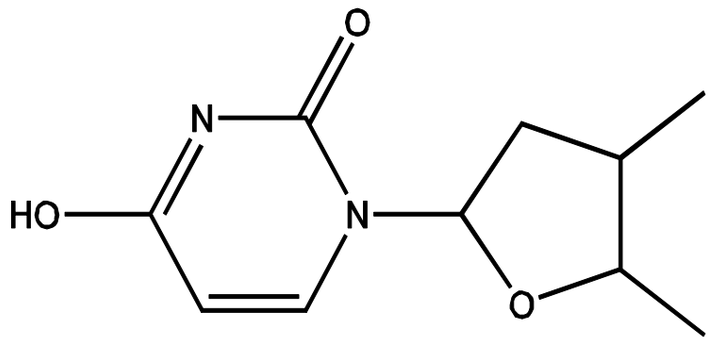
Approximate Patterns Sequential Patterns Summary

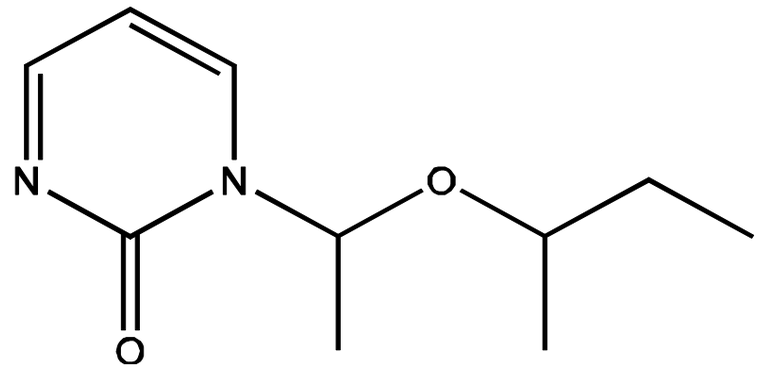
Given a labeled graph dataset $D = \{G_1, G_2, \ldots, G_n\}$, the supporting graph set of a subgraph g is $D_g = \{G_i \mid g \subseteq G_i, G_i \in D\}$.

support(g) $= |D_g| / |D|$

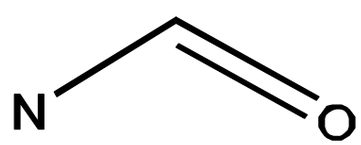
A (sub)graph g is frequent if **support(g) ≥ min_sup**



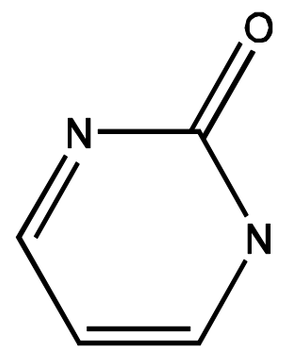




Frequent **Sub-graph** patterns



min_sup=2



frequent sub-graphs

Bioinformatics

Gene networks, protein interactions, metabolic pathways

Chem-informatics:
Mining chemical compound structures

Software engineering: program execution flow analysis

Web graphs, XML structures, semantic Web, information networks

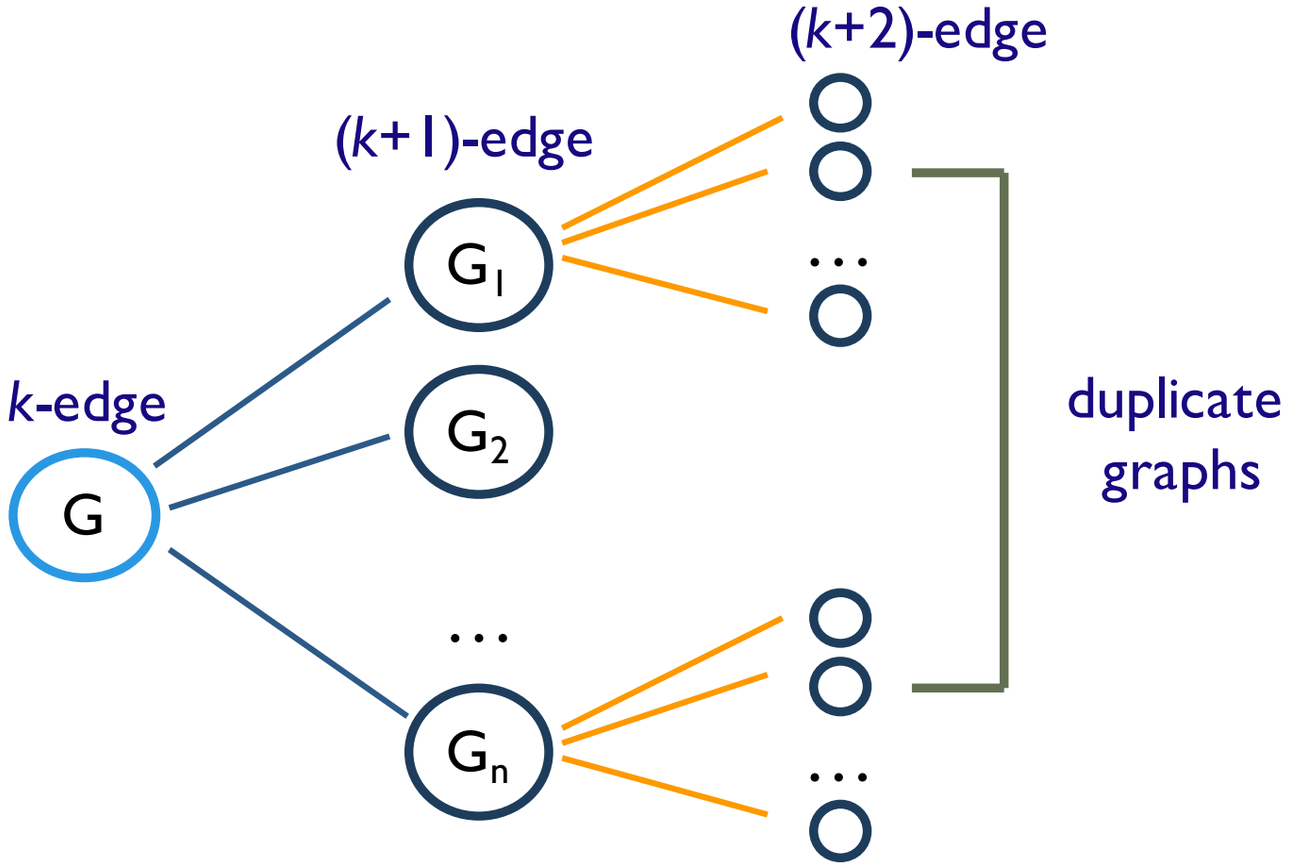
Many Applications

Social **networks**, web communities, tweets, …

Cell phone networks, computer networks, …

Building blocks for **graph classification, clustering, compression, comparison, and correlation** analysis

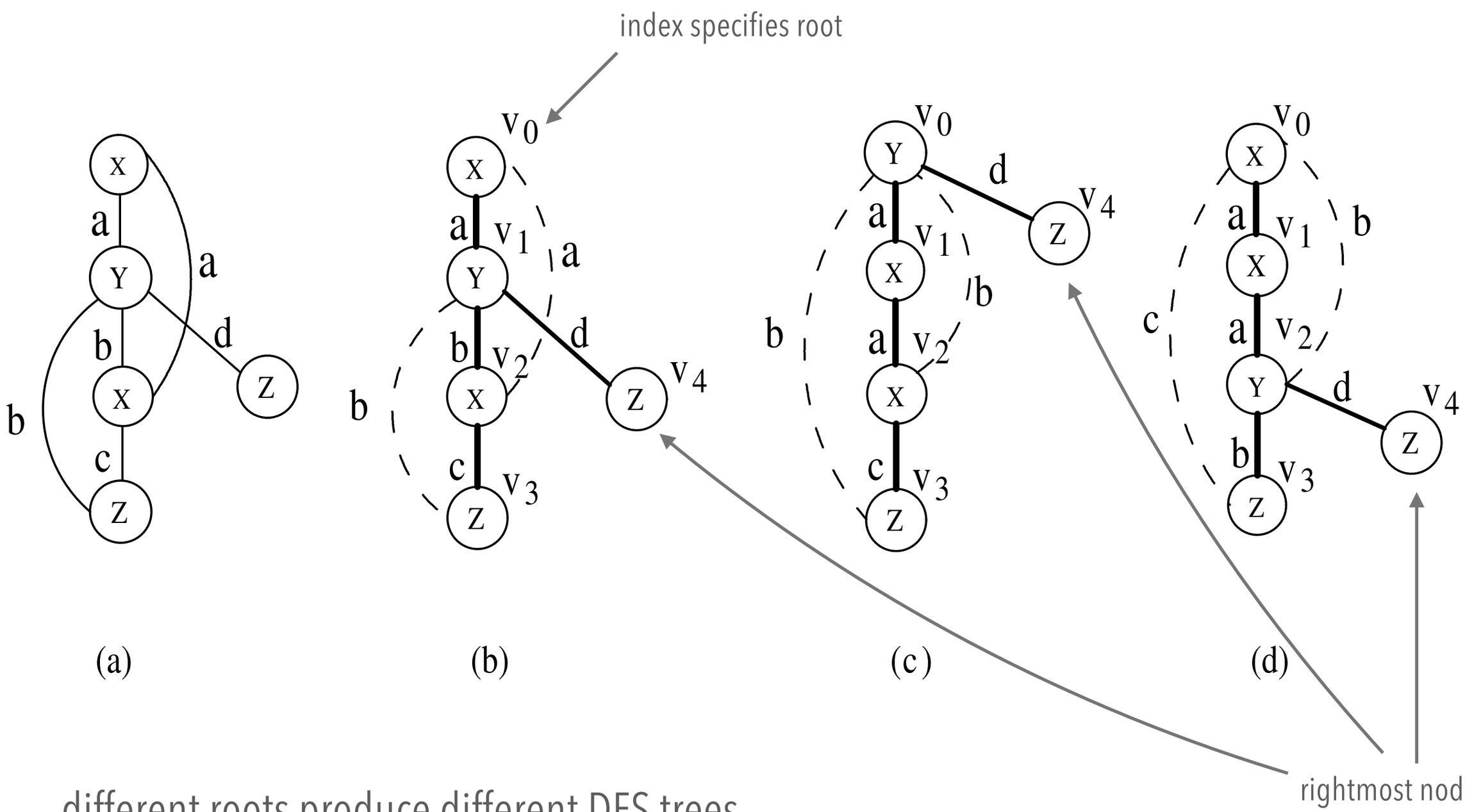
PATTERN GROWTH APPROACHES



- Depth-first growth of subgraphs from k-edge to (k+1)-edge, then (k+2)-edge subgraphs
- Major challenge
  - Generating many duplicate subgraphs
- Major idea to solve the problem
  - Define an order to generate subgraphs
  - DFS spanning tree: Flatten a graph into a sequence using depth-first search

X. Yan and J. Han. gspan: graph-based substructure pattern mining. In Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on, pages 721–724, 2002.

DFS Search Trees



different roots produce different DFS trees

DFS Codes

Forward Edge and Backward Edge. Given G_T, the forward edge (*tree edge* [3]) set contains all the edges in the DFS tree, and the backward edge (*back edge* [3]) set contains all the edges which are not in the DFS tree. For simplicity, (i, j) is an ordered pair to represent an edge. If $i < j$, it is a forward edge; otherwise, a backward edge. A linear order, $\prec_T$ is built among all the edges in G by the following rules (assume $e_1 = (i_1, j_1), e_2 = (i_2, j_2)$): (i) if $i_1 = i_2$ and $j_1 < j_2$, $e_1 \prec_T e_2$; (ii) if $i_1 < j_1$ and $j_1 = i_2$, $e_1 \prec_T e_2$; and (iii) if $e_1 \prec_T e_2$ and $e_2 \prec_T e_3$, $e_1 \prec_T e_3$.

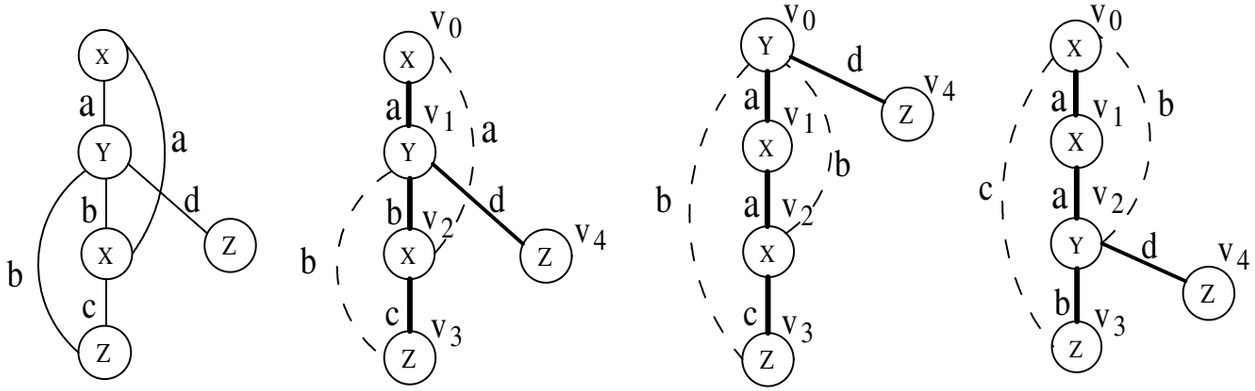
Definition 1 (DFS Code) *Given a DFS tree T for a graph G, an edge sequence (e_i) can be constructed based on $\prec_T$, such that $e_i \prec_T e_{i+1}$, where $i = 0, \ldots, |E| - 1$. (e_i) is called a DFS code, denoted as code(G, T).*

DFS Codes

there must be a **minimum** DFS tree

order on DFS trees

$$\gamma < \alpha < \beta$$



(a) (b) (c) (d)

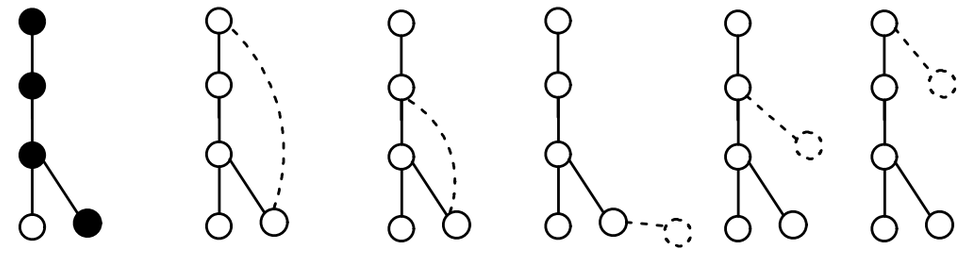
edge	(Fig 1b) α	(Fig 1c) β	(Fig 1d) γ
0	$(0, 1, X, a, Y)$	$(0, 1, Y, a, X)$	$(0, 1, X, a, X)$
1	$(1, 2, Y, b, X)$	$(1, 2, X, a, X)$	$(1, 2, X, a, Y)$
2	$(2, 0, X, a, X)$	$(2, 0, X, b, Y)$	$(2, 0, Y, b, X)$
3	$(2, 3, X, c, Z)$	$(2, 3, X, c, Z)$	$(2, 3, Y, b, Z)$
4	$(3, 1, Z, b, Y)$	$(3, 0, Z, b, Y)$	$(3, 0, Z, c, X)$
5	$(1, 4, Y, d, Z)$	$(0, 4, Y, d, Z)$	$(2, 4, Y, d, Z)$

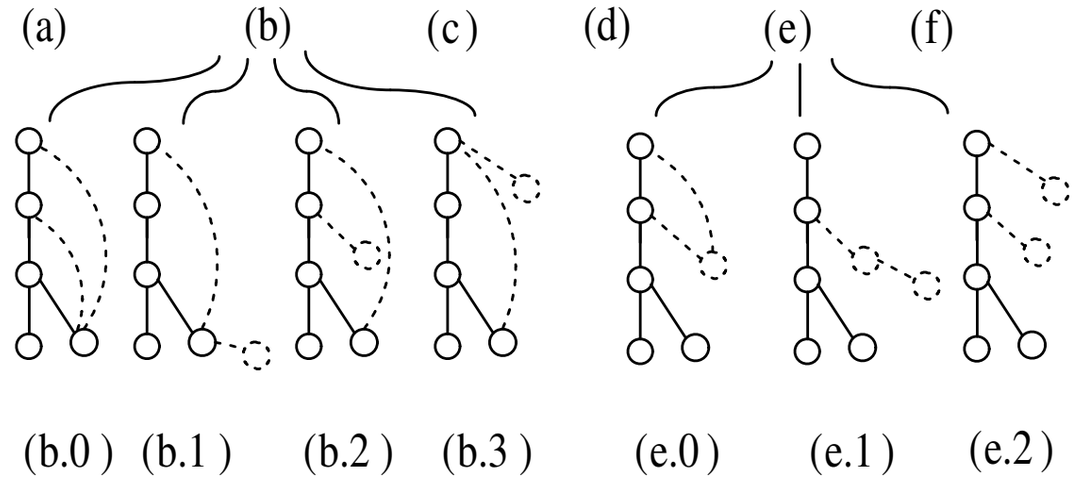
Table 1. DFS codes for Fig. 1(b)-(d)

For simplicity, an edge can be presented by a 5-tuple, $(i, j, l_i, l_{(i,j)}, l_j)$, where l_i and l_j are the labels of v_i and v_j respectively and $l_{(i,j)}$ is the label of the edge between them. For example, (v_0, v_1) in Fig. 1(b) is represented by $(0, 1, X, a, Y)$. Table 1 shows the corresponding DFS codes for Fig. 1(b), 1(c), and 1(d).

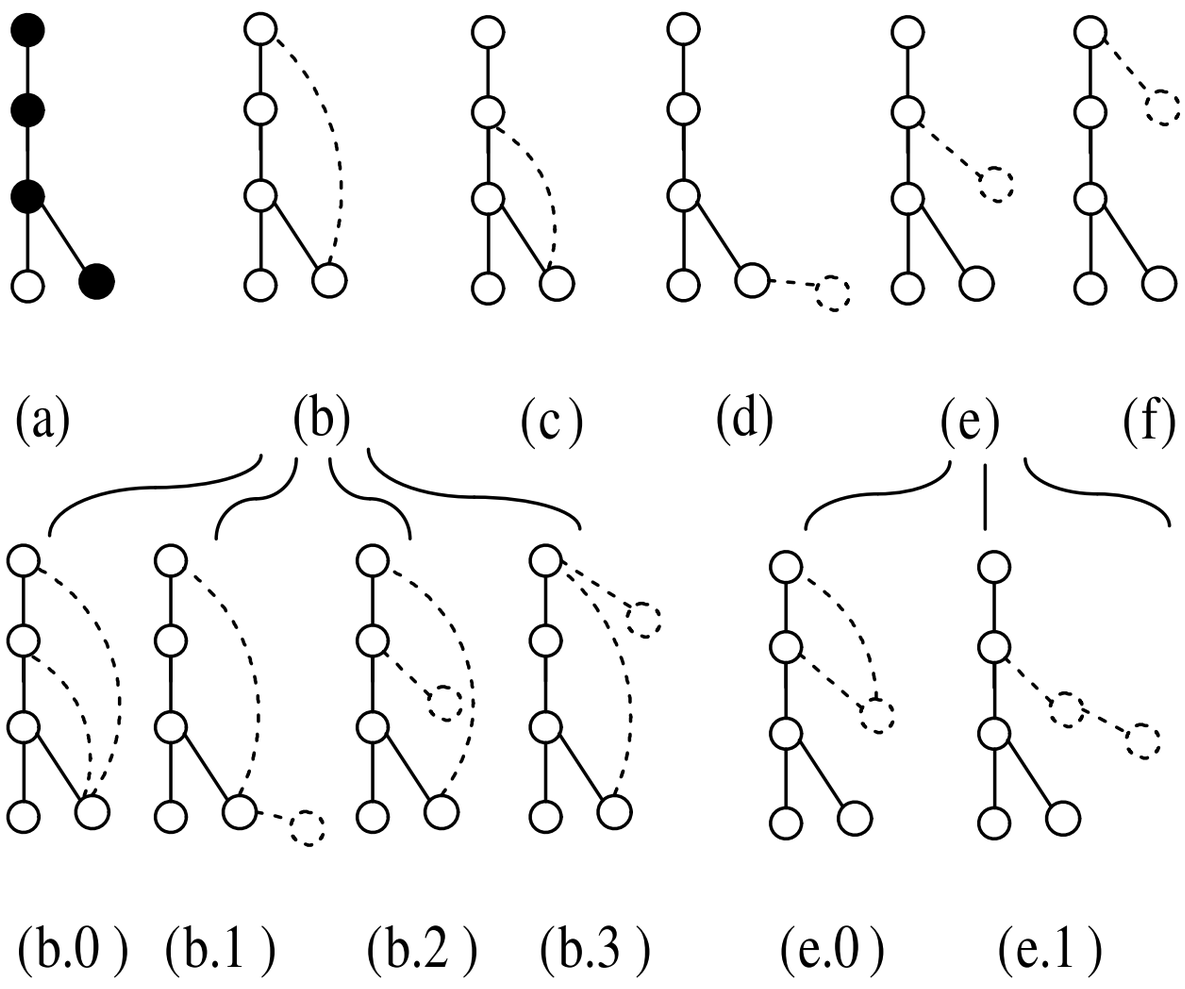
Growth

Given a DFS code $\alpha = (a_0, a_1, ..., a_m)$, any valid DFS code $\beta = (a_0, a_1, ..., a_m, b)$, is called α's **child**, and α is called β's **parent**. In fact, to construct a valid DFS code, b must be an edge which only grows from the vertices on the rightmost path. In Fig. 2, the graph shown in 2(a) has several potential children with one edge growth, which are shown in 2(b)-(f) (assume the darkened vertices constitute the rightmost path). Among them, 2(b), 2(c), and 2(d) grow from the rightmost vertex while 2(e) and 2(f) grow from other vertices on the rightmost path. 2(b.0)-(b.3) are children of 2(b), and 2(e.0)-(e.2) are children of 2(e). Backward edges can only grow from the rightmost vertex while forward edges can grow from vertices on the rightmost path. This restriction is similar to TreeMinerV's equivalence class extension [8] and FREQT's rightmost expansion [2] in frequent tree discovery. The enumeration order of these children is enhanced by the DFS lexicographic order, i.e., it should be in the order of 2(b), 2(c), 2(d), 2(e), and 2(f).





GROWTH



Right-most path extension in subgraph pattern growth

- Right-most path: The path from root to the right-most leaf (choose the vertex w. the smallest index at each step)
- Reduce generation of duplicate subgraphs

Completeness: The Enumeration of graphs using right-most path extension is complete

DFS Code: Flatten a graph into a sequence using depth-first search