An Exhaustive Covering Approach to Parameter-free Mining of Non-redundant Discriminative Itemsets

Yoshitaka Kameya
Meijo University

Outline

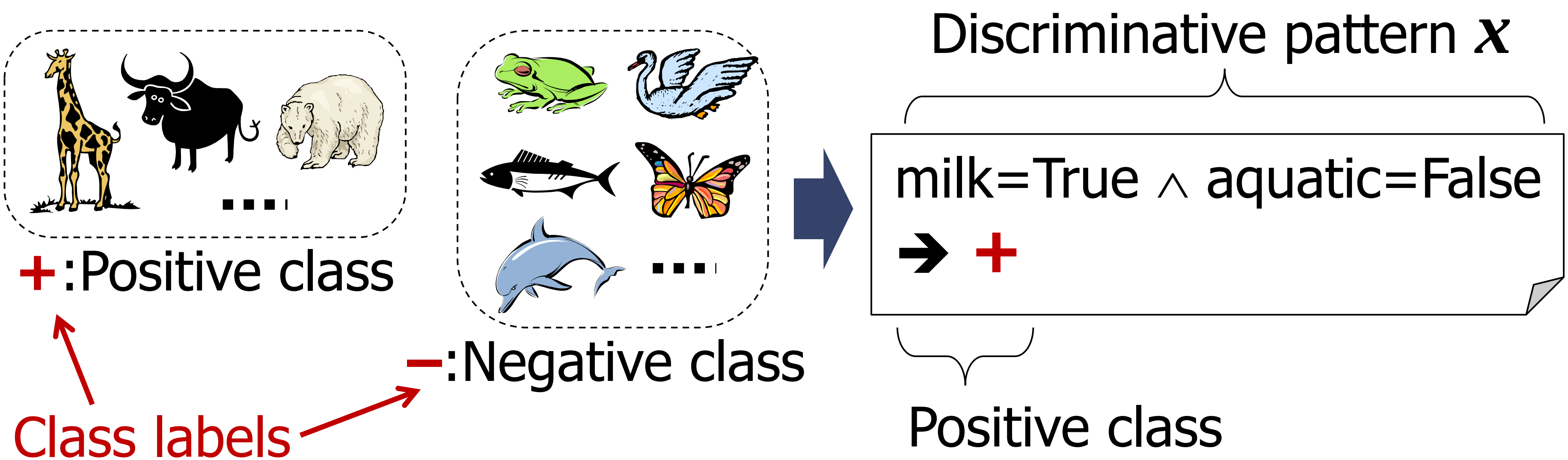
- Background
- Our propsal
- Experiments

Outline

- Background
- Our propsal
- Experiments

Background: Discriminative Patterns (1)

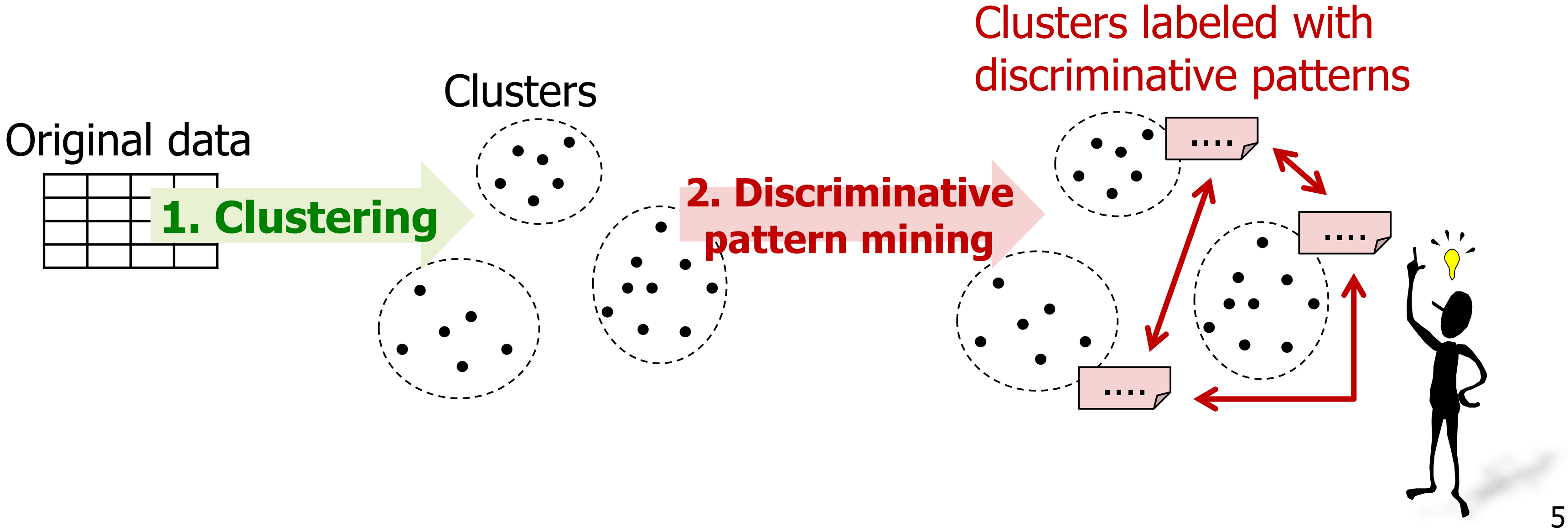
- Discriminative patterns:
  - Show differences between two groups (classes)
  - Used for:
    - Characterizing the positive class
    - Building more precise classifiers



Background: Discriminative Patterns (2)

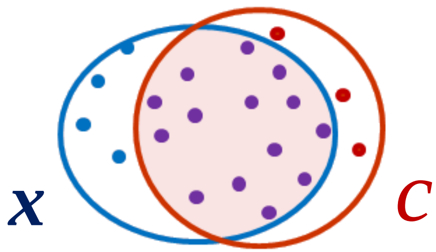
- Discriminative patterns tend to be more meaningful than frequent patterns (thanks to class labels)
- Are class labels always available?
  - Comparing groups is a standard starting point in data analysis
  - Clustering can find groups (classes)

    → Cluster labeling

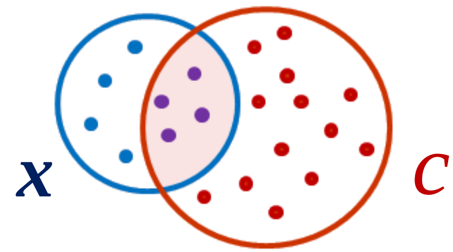


Background: Discriminative Patterns (3)

- Quality score: Measures the overlap between pattern x and positive class c



Quality is **high**



Quality is **low**

- Most of popular quality scores are *not* anti-monotonic:
  - Confidence, Lift
  - Support difference, Weighted relative accuracy, Leverage
  - F-score, Dice, Jaccard
  - ...

➔ Branch & bound pruning is often used
[Morishita+ 00][Zimmarmann+ 09][Nijssen+ 09]

Background: Coping with redundancy (1)

- **Example**: Item A is relevant to the positive class
  → Patterns containing A tend to be top-ranked in the candidate list (most of them are redundant)

Dataset

TID	Class	Transaction
1	+	{A, B, D, E}
2	+	{A, B, C, D, E}
3	+	{A, C, D, E}
4	+	{A, B, C}
5	+	{B}
6	−	{A, B, D, E}
7	−	{B, C, D, E}
8	−	{C, D, E}
9	−	{A, D, E}
10	−	{A, D}

Positive Transactions (TIDs 1–5)

Negative Transactions (TIDs 6–10)

Top-15 patterns (+1 due to tie score)

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

Background: Coping with redundancy (2)

- Set-inclusion-based constraints
  - Closedness [Pasquier+ 99]
  - Productivity [Bayardo 00][Webb 07]

Background: Coping with redundancy (2)

- Set-inclusion-based constraints
  - Closedness [Pasquier+ 99]
  - Productivity [Bayardo 00][Webb 07]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

Closedness:
For patterns covering the same (positive) transactions,
pick the largest one

Background: Coping with redundancy (2)

- Set-inclusion-based constraints
  - Closedness [Pasquier+ 99]
  - Productivity [Bayardo 00][Webb 07]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

Closedness:
For patterns covering the same (positive) transactions,
pick the largest one

Background: Coping with redundancy (2)

- Set-inclusion-based constraints
  - Closedness [Pasquier+ 99]
  - Productivity [Bayardo 00][Webb 07]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

16 patterns → **8** patterns

Background: Coping with redundancy (2)

- Set-inclusion-based constraints
  - Closedness [Pasquier+ 99]
  - Productivity [Bayardo 00][Webb 07]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

Productivity:
If a super-pattern has no higher quality, remove it

Background: Coping with redundancy (2)

- Set-inclusion-based constraints
  - Closedness [Pasquier+ 99]
  - Productivity [Bayardo 00][Webb 07]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

Productivity:
If a super-pattern has no higher quality, remove it

Background: Coping with redundancy (2)

- Set-inclusion-based constraints
  - Closedness [Pasquier+ 99]
  - Productivity [Bayardo 00][Webb 07]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

16 patterns → **4** patterns

Background: Coping with redundancy (2)

- Set-inclusion-based constraints
  - Productivity + Closedness [Kameya+ 13]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

16 patterns → **3** patterns

Background: Coping with redundancy (3)

- The best-covering constraint
  - In the same spirit of the HCC (highest confidence covering) constraint in HARMONY [Wang+ 05]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

Best-covering: Every pattern must be the best to at least one positive transaction

Background: Coping with redundancy (3)

- The best-covering constraint
  - In the same spirit of the HCC (highest confidence covering) constraint in HARMONY [Wang+ 05]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

Best-covering: Every pattern must be the best to at least one positive transaction

Background: Coping with redundancy (3)

- The best-covering constraint
  - In the same spirit of the HCC (highest confidence covering) constraint in HARMONY [Wang+ 05]

Rank	Pattern	F-score	TIDs Covered
1	{A, C}	0.75	2, 3, 4
2	{B}	0.73	1, 2, 4, 5
3	{A}	0.67	1, 2, 3, 4
3	{A, B}	0.67	1, 2, 4
5	{A, D, E}	0.60	1, 2, 3
5	{A, E}	0.60	1, 2, 3
5	{C}	0.60	2, 3, 4
8	{A, B, C}	0.57	2, 4
8	{A, C, D}	0.57	2, 3
8	{A, C, D, E}	0.57	2, 3
8	{A, C, E}	0.57	2, 3
12	{A, D}	0.55	1, 2, 3
13	{A, B, D}	0.50	1, 2
13	{A, B, D, E}	0.50	1, 2
13	{A, B, E}	0.50	1, 2
13	{B, C}	0.50	2, 4

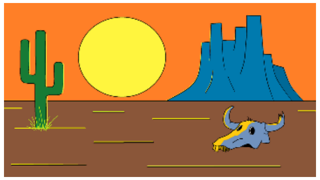
Original dataset

TID	Class	Transaction
1	+	{A, **B**, D, E}
2	+	{**A**, **B**, **C**, D, E}
3	+	{**A**, **C**, D, E}
4	+	{**A**, **B**, **C**}
5	+	{**B**}
6	–	{A, **B**, D, E}
7	–	{**B**, C, D, E}
8	–	{C, D, E}
9	–	{A, D, E}
10	–	{A, D}

16 patterns → **2** patterns

Background: Control parameters

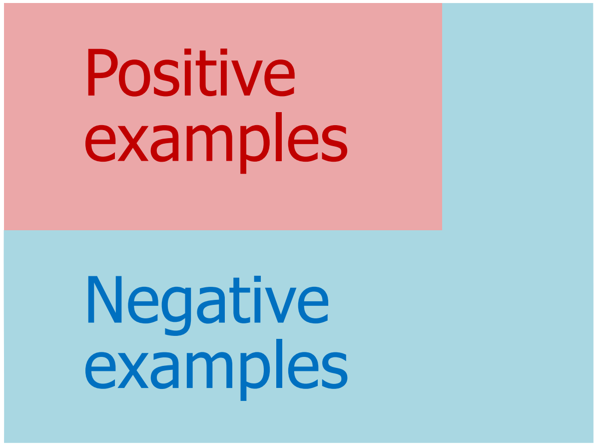
- Minimum support (minsup) σ_{min} is a sensitive control parameter


or


- Top-k mining [Han+ 02]:
  - k = "# of output patterns"
  - k is fairly easy to specify because we usually know how many patterns we can handle
    (k is more *human-centric* than σ_{min})
  - However, we do *not* exactly know *in advance* how many *useful* patterns we can mine
  - Is it possible to remove even k ?

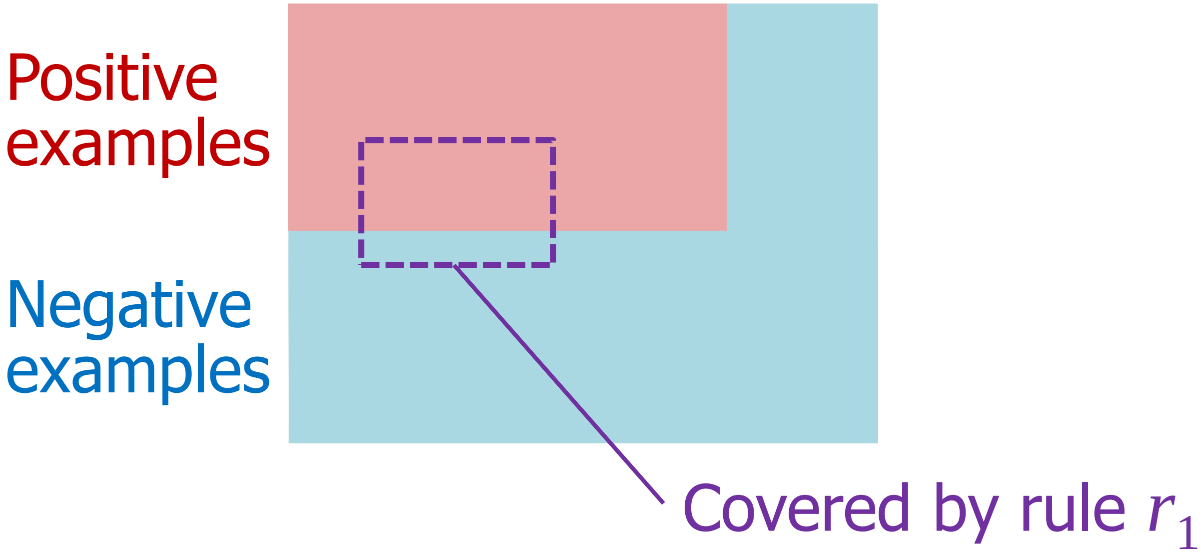
Background: Sequential covering (1)

- Sequential covering:
  - One traditional way for building a rule-based classifier
- Procedure:
  - Iterate until there are no uncovered positive examples
    - Induce a new rule r
    - Remove all positive examples covered by r



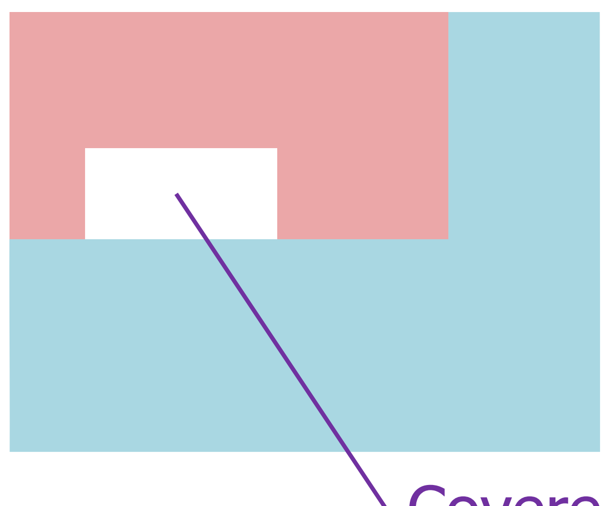
Background: Sequential covering (1)

- Sequential covering:
  - One traditional way for building a rule-based classifier
- Procedure:
  - Iterate until there are no uncovered positive examples
    - Induce a new rule r
    - Remove all positive examples covered by r



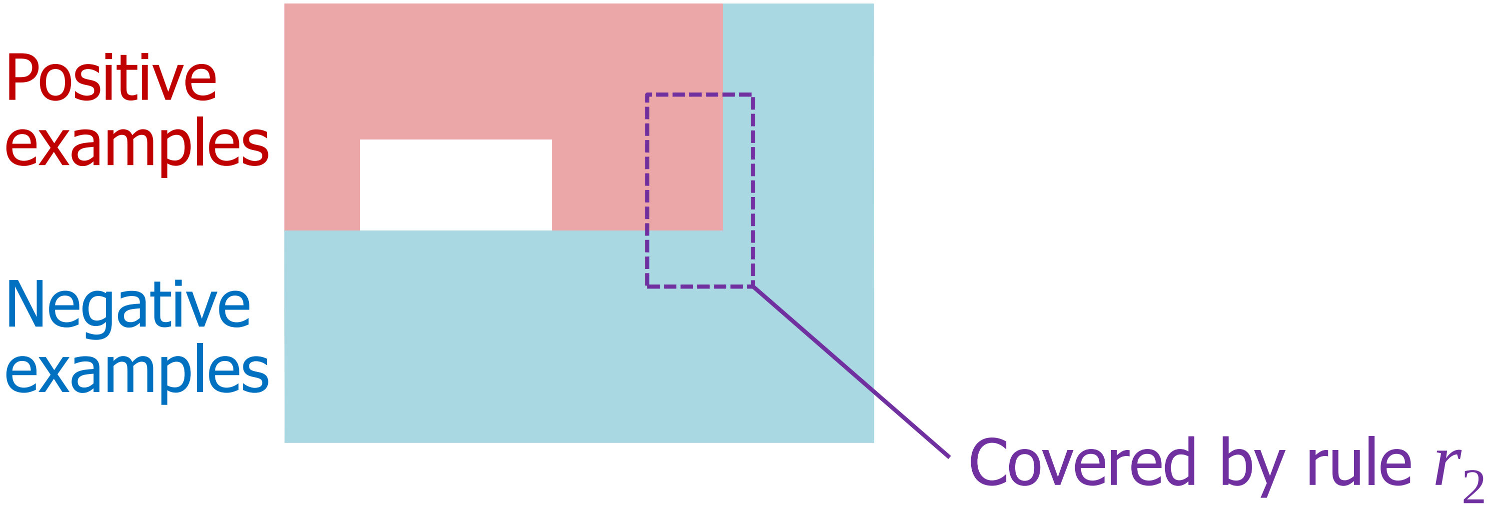
Background: Sequential covering (1)

- Sequential covering:
  – One traditional way for building a rule-based classifier
- Procedure:
  – Iterate until there are no uncovered positive examples
    - Induce a new rule *r*
    - Remove all positive examples covered by *r*



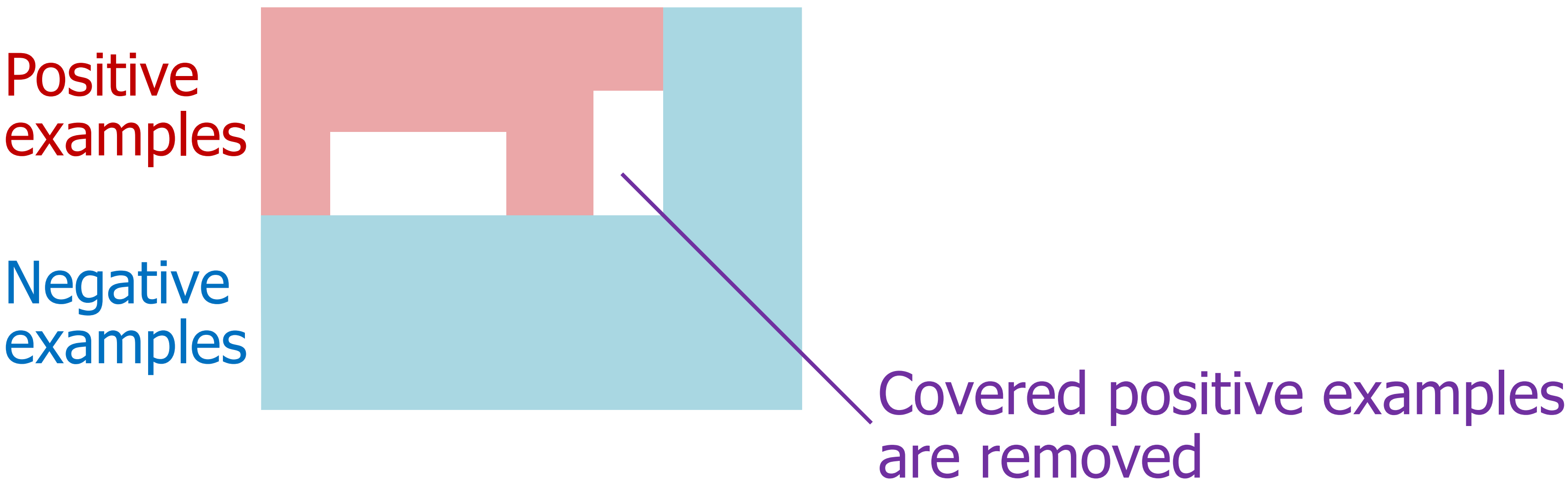
Background: Sequential covering (1)

- Sequential covering:
  - One traditional way for building a rule-based classifier
- Procedure:
  - Iterate until there are no uncovered positive examples
    - Induce a new rule r
    - Remove all positive examples covered by r



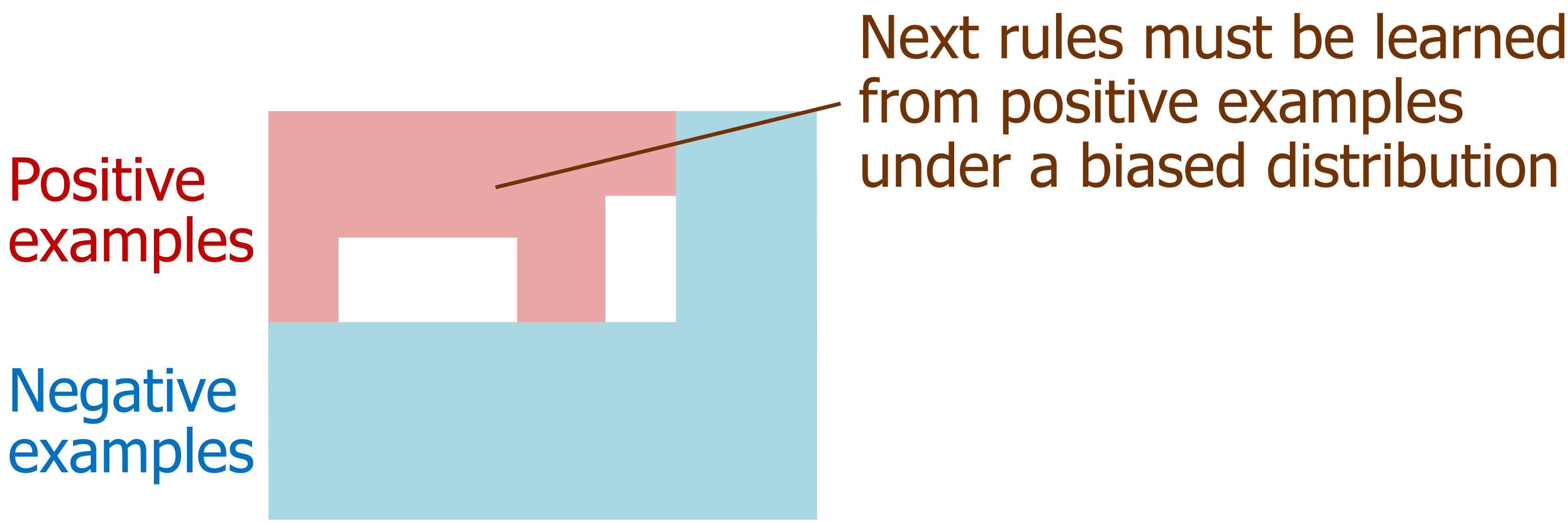
Background: Sequential covering (1)

- Sequential covering:
  - One traditional way for building a rule-based classifier
- Procedure:
  - Iterate until there are no uncovered positive examples
    - Induce a new rule r
    - Remove all positive examples covered by r



Background: Sequential covering (2)

- Problems in removing positive examples:
  - Lately-generated rules may not be meaningful
  - The number of positive examples decreases [Domingos 94]
    → Lately-generated rules may not be statistically reliable



Our proposal

- ExCover: an efficient and exact method for finding non-redundant discriminative itemsets
- Features:
  - Exhaustive search
    unlike sequential covering
  - Best-covering constraint
    tighter than productivity → fewer redundant patterns
  - No control parameters limiting the search space

Outline

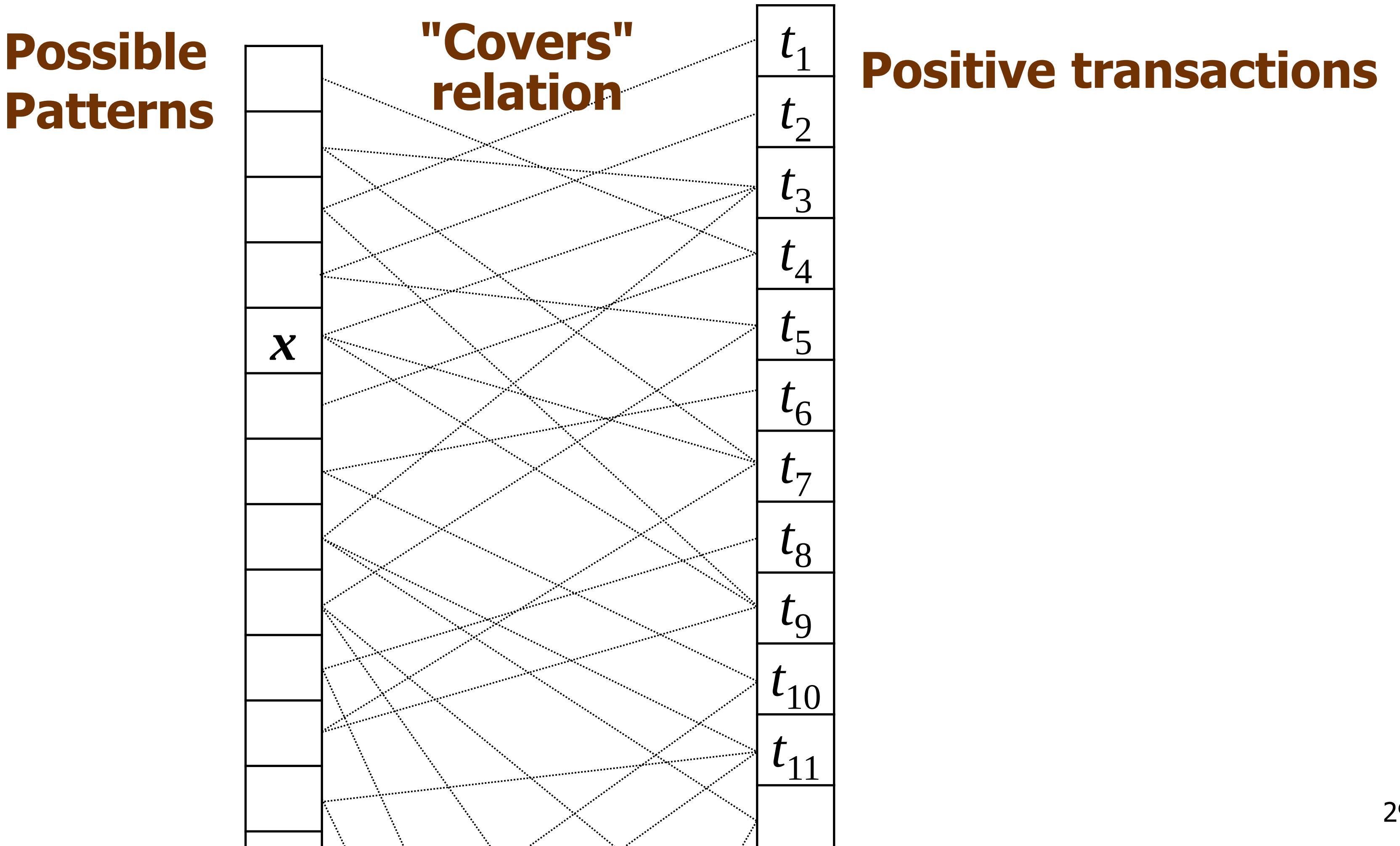
- ✓ Background
- Our proposal
  - Best-covering constraint
  - ExCover
- Experiments

Outline

✓ Background
- Our proposal
  – Best-covering constraint
  – ExCover
- Experiments

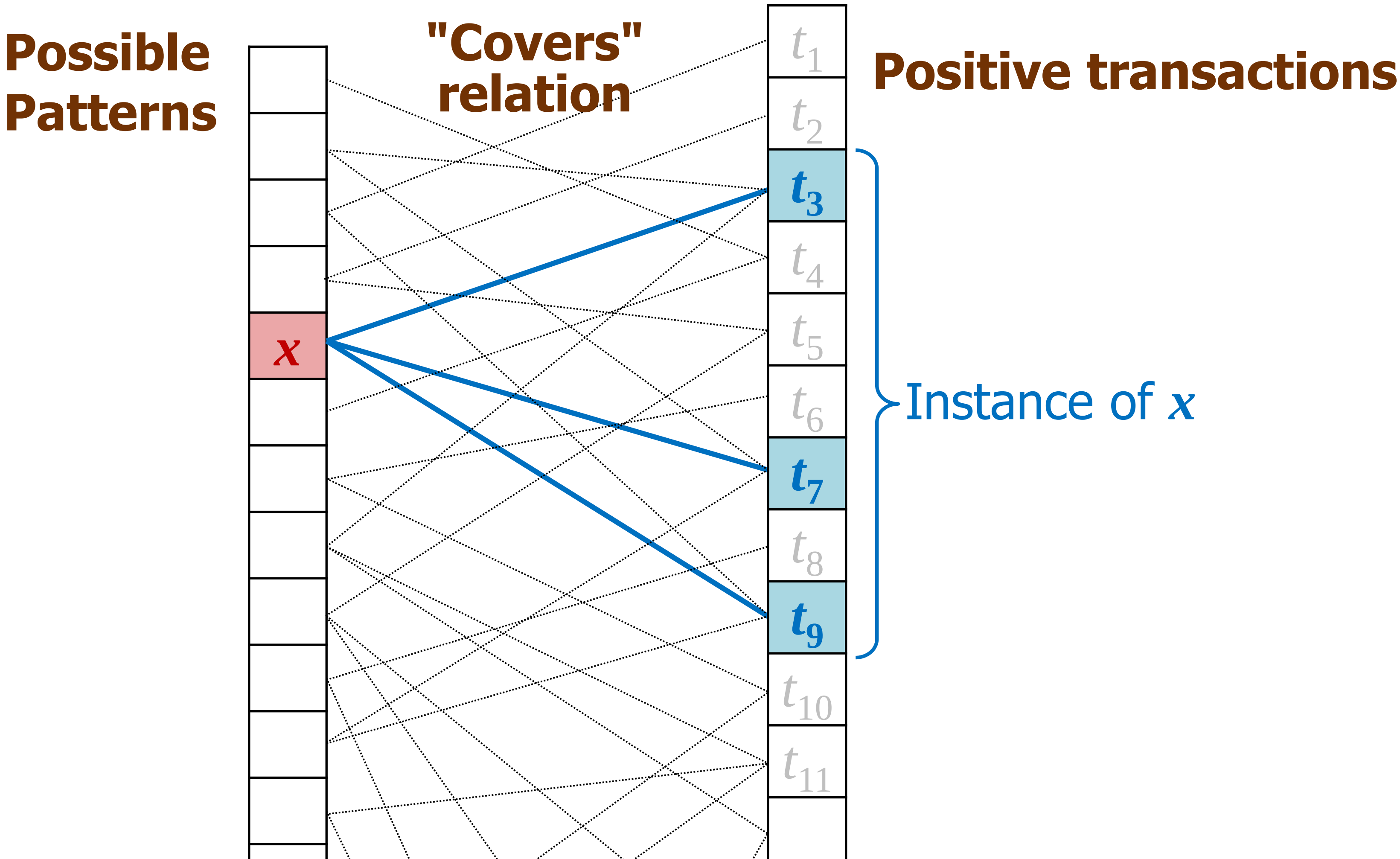
Best-covering constraint (1)

- Best-covering constraint:
  "Every pattern must have the highest quality for at least one positive transaction it covers"



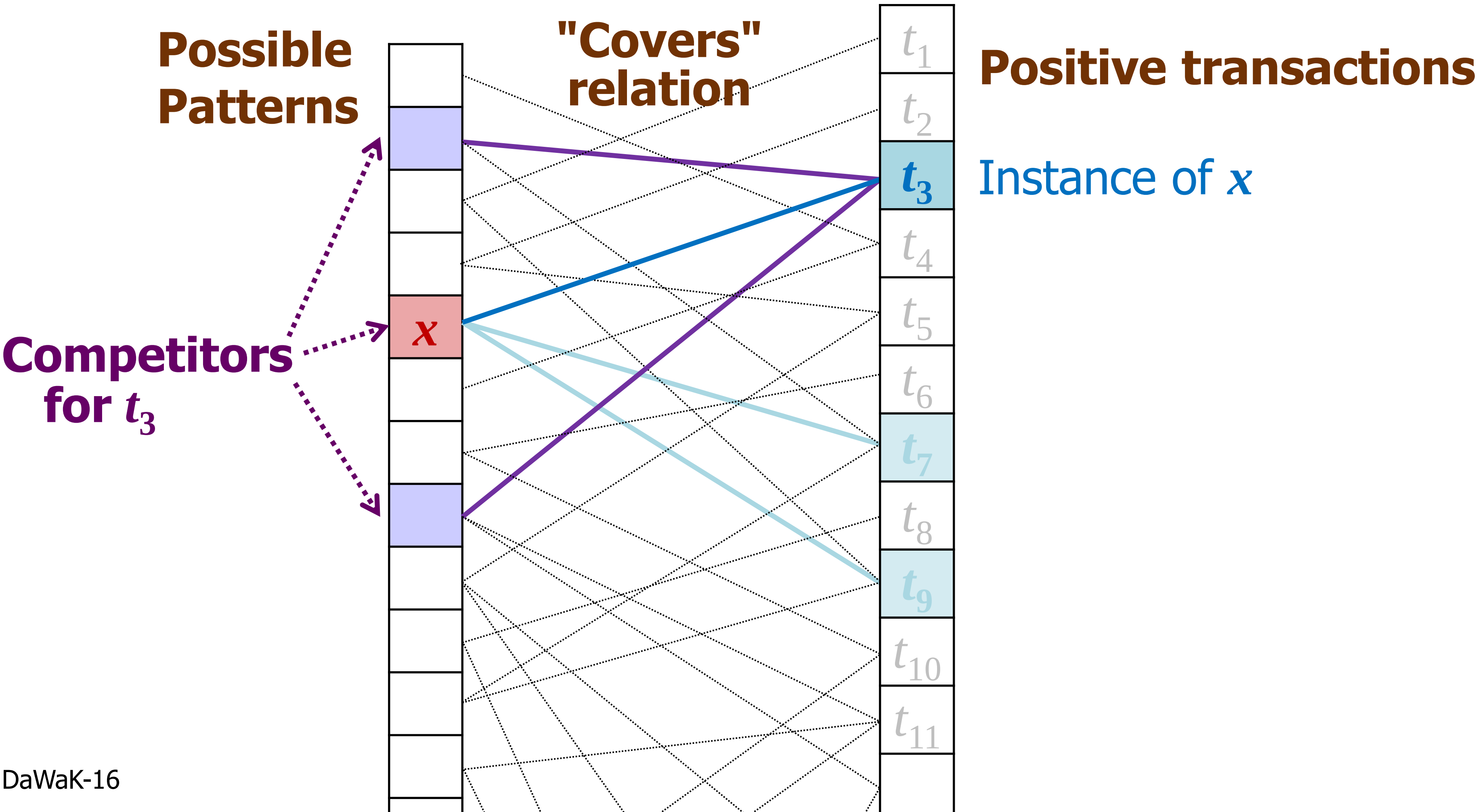
Best-covering constraint (2)

- Best-covering constraint:
  "Every pattern must have the highest quality for at least one positive transaction it covers"



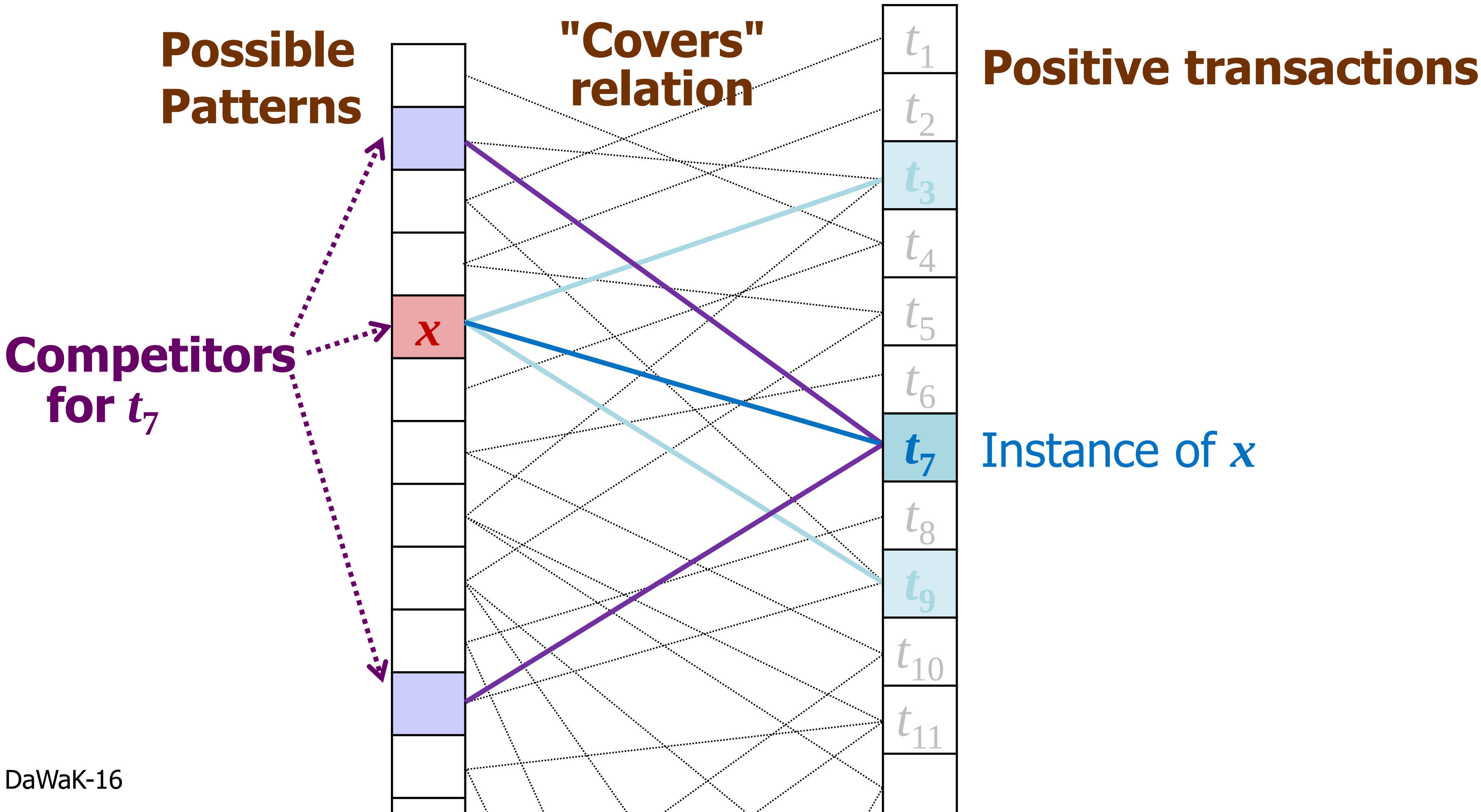
Best-covering constraint (3)

- Best-covering constraint:
  "Every pattern must have the highest quality for at least one positive transaction it covers"



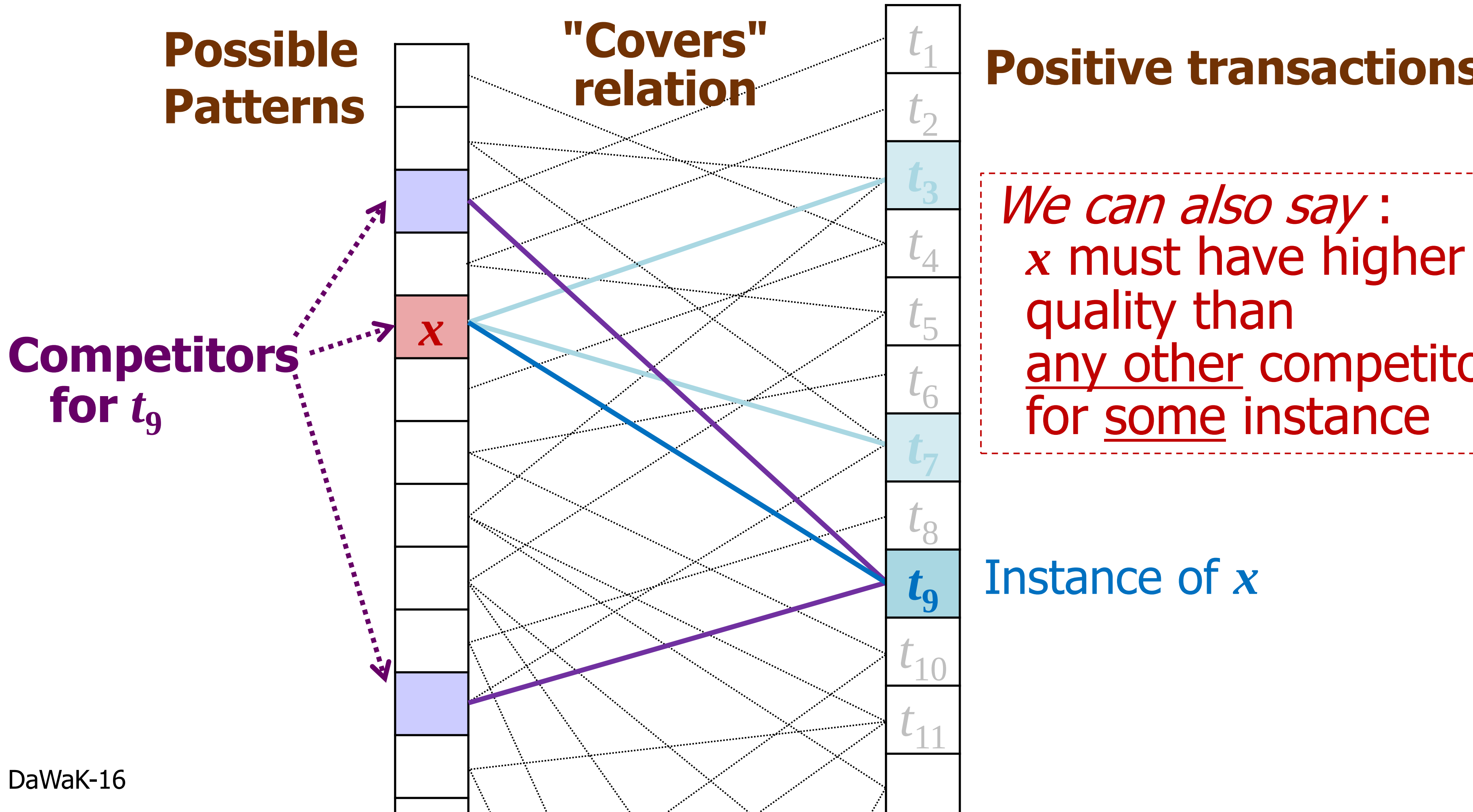
Best-covering constraint (3)

- Best-covering constraint:
  "Every pattern must have the highest quality for at least one positive transaction it covers"



Best-covering constraint (3)

- Best-covering constraint:
  "Every pattern must have the highest quality for at least one positive transaction it covers"



We can also say :
x must have higher quality than any other competitors for some instance

Best-covering constraint (4)

- **Tightness**:
  Best-covering is tighter than productivity

  <u>Sketch of proof</u>
  - Sub-pattern of x is always a competitor of x
  - If x is best-covering, its sub-pattern must have lower quality
  - Productivity: x must have higher quality than <u>its sub-patterns</u>

  Best-covering:
  x must have higher quality than
  <u>any other competitors for some instance</u>

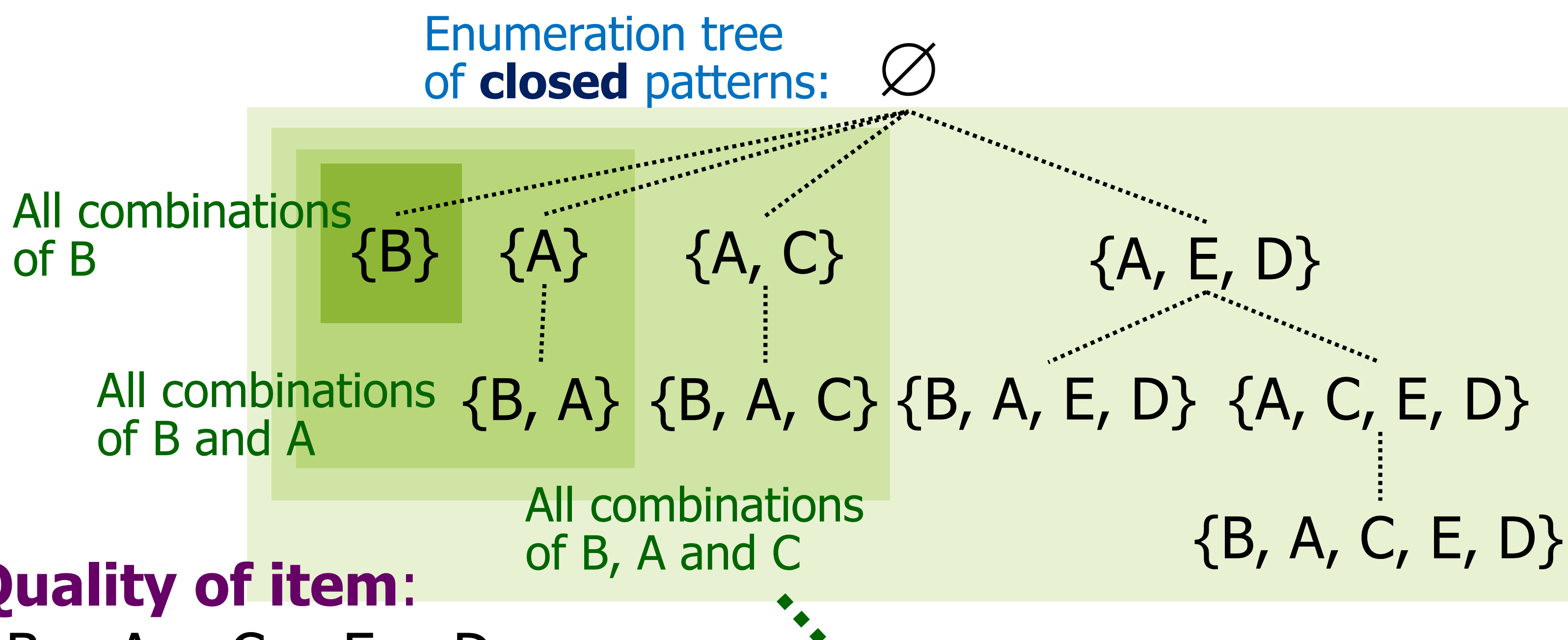
- **Branch & bound pruning**:
  We can safely prune x and its descendants
  when the <u>upper bound</u> of x's quality is lower than
  the quality of *any* competitor of x

Outline

✓ Background
- Our proposal
  ✓ Best-covering constraint
  – ExCover
- Experiments

ExCover: Search space

- Basic strategy:
  - Depth-first search by a variant [Kameya+ 13] of LCM [Uno+ 04]:
    - Only visits patterns closed on positive transactions
      → The closedness constraint is *built-in*
    - Visits earlier shorter patterns including high quality items
      → There is more chance of pruning



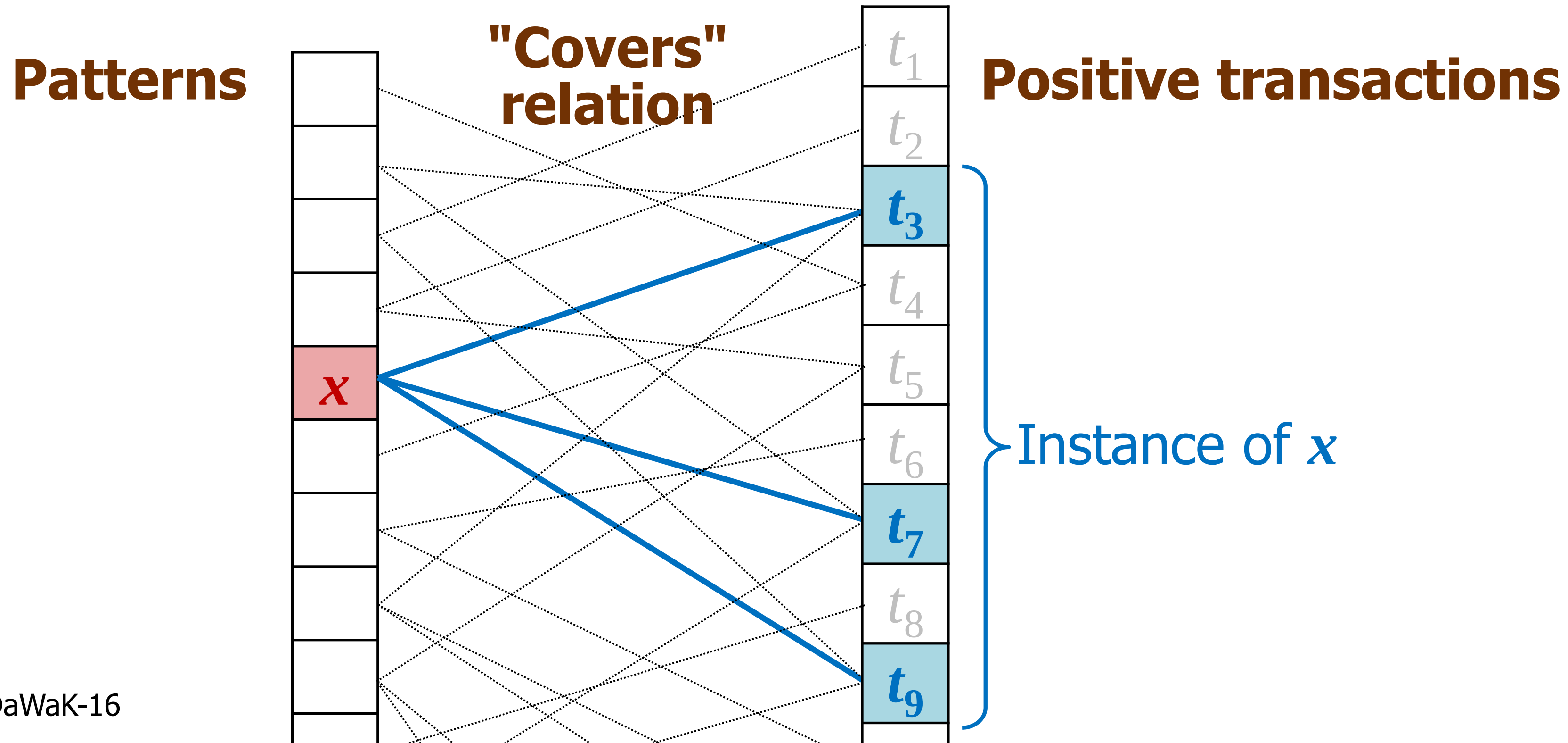
Quality of item:
B > A > C > E > D

ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate patterns are maintained in the *candidate table* following the best-covering constraint

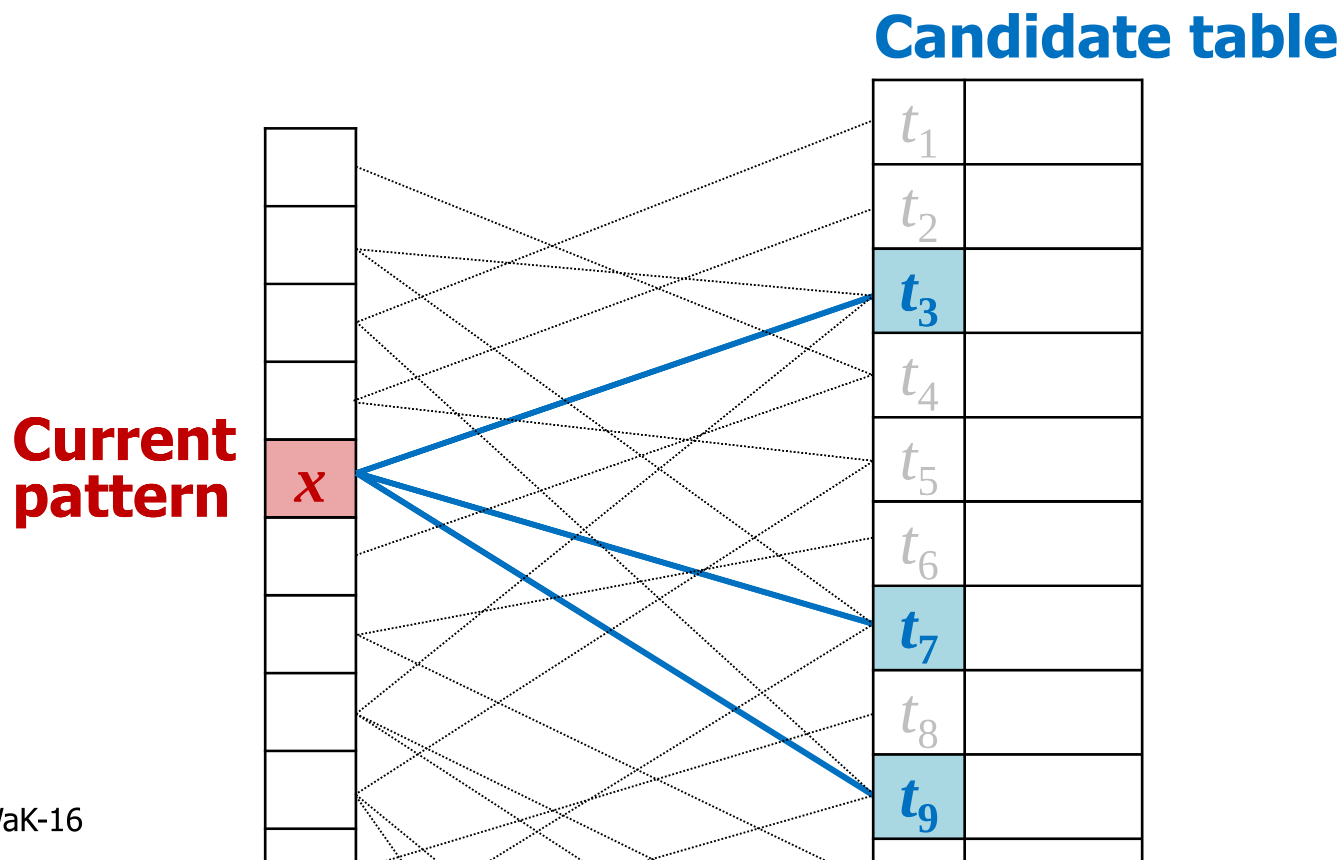
ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate patterns are maintained in the *candidate table* following the best-covering constraint



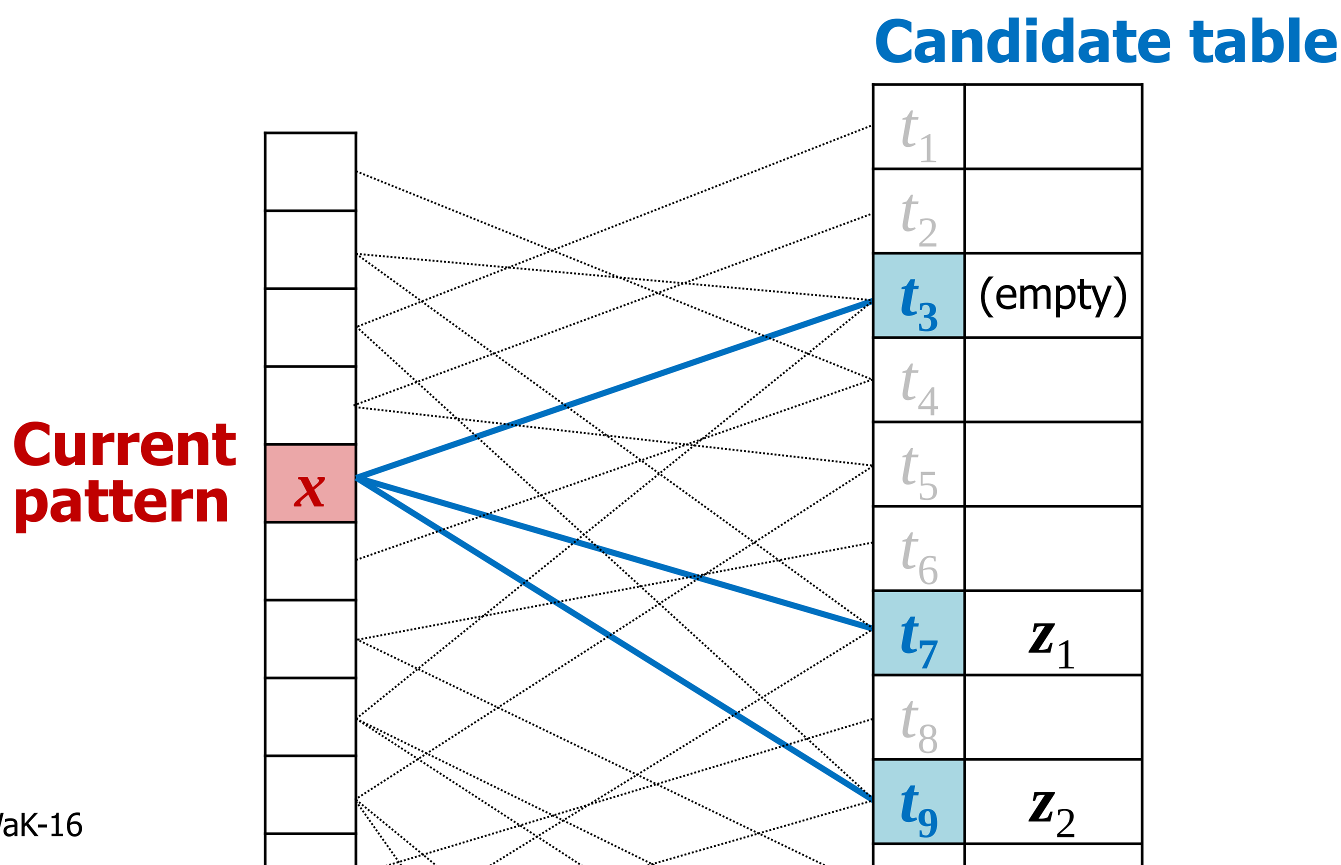
ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



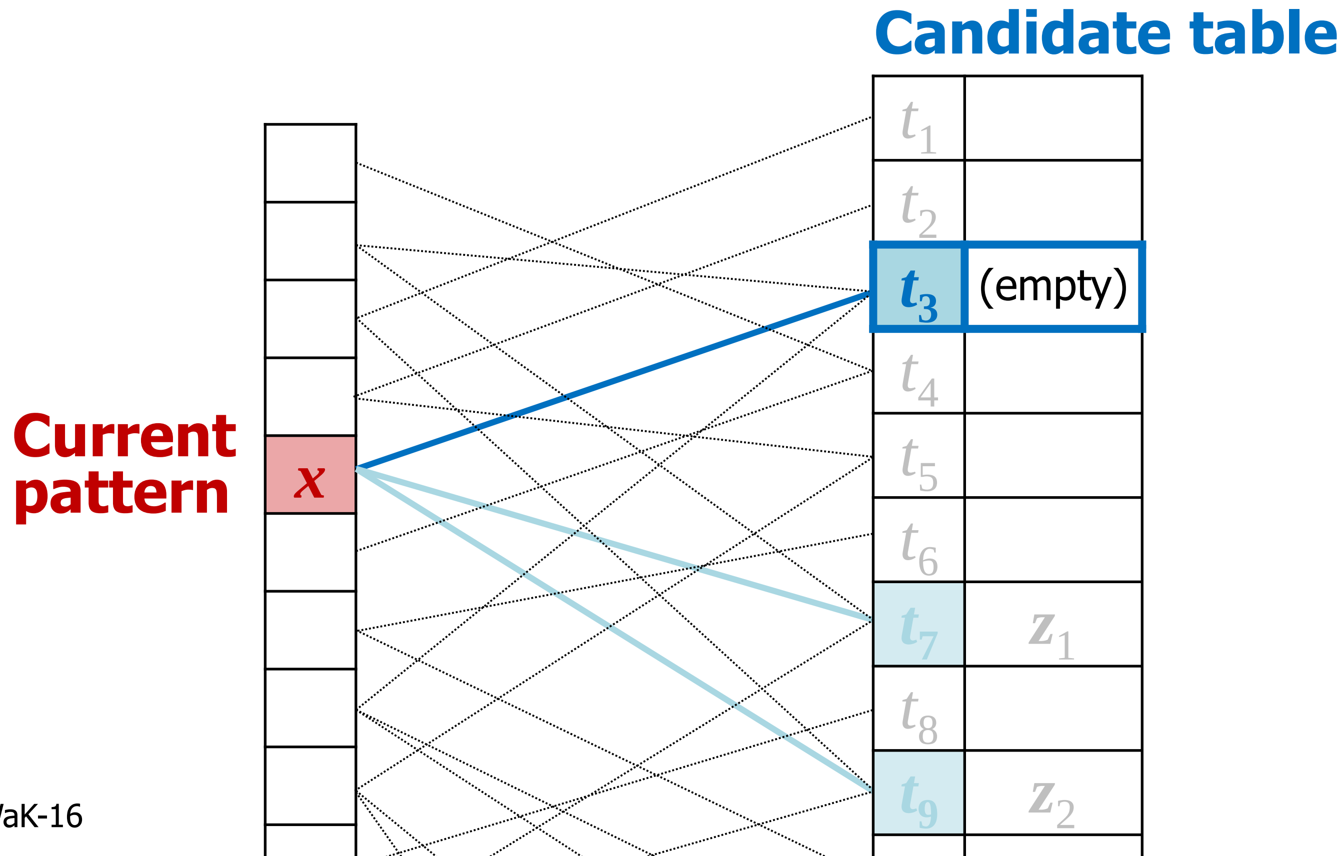
ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



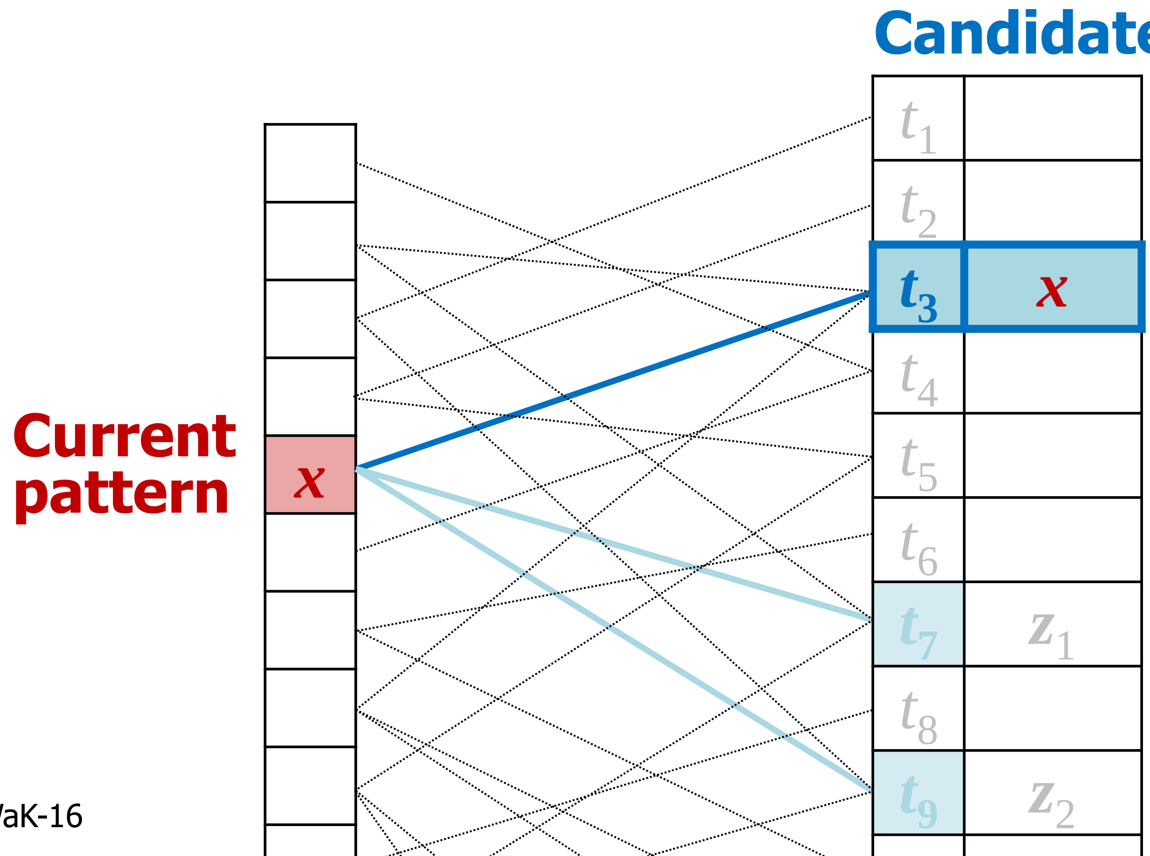
ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



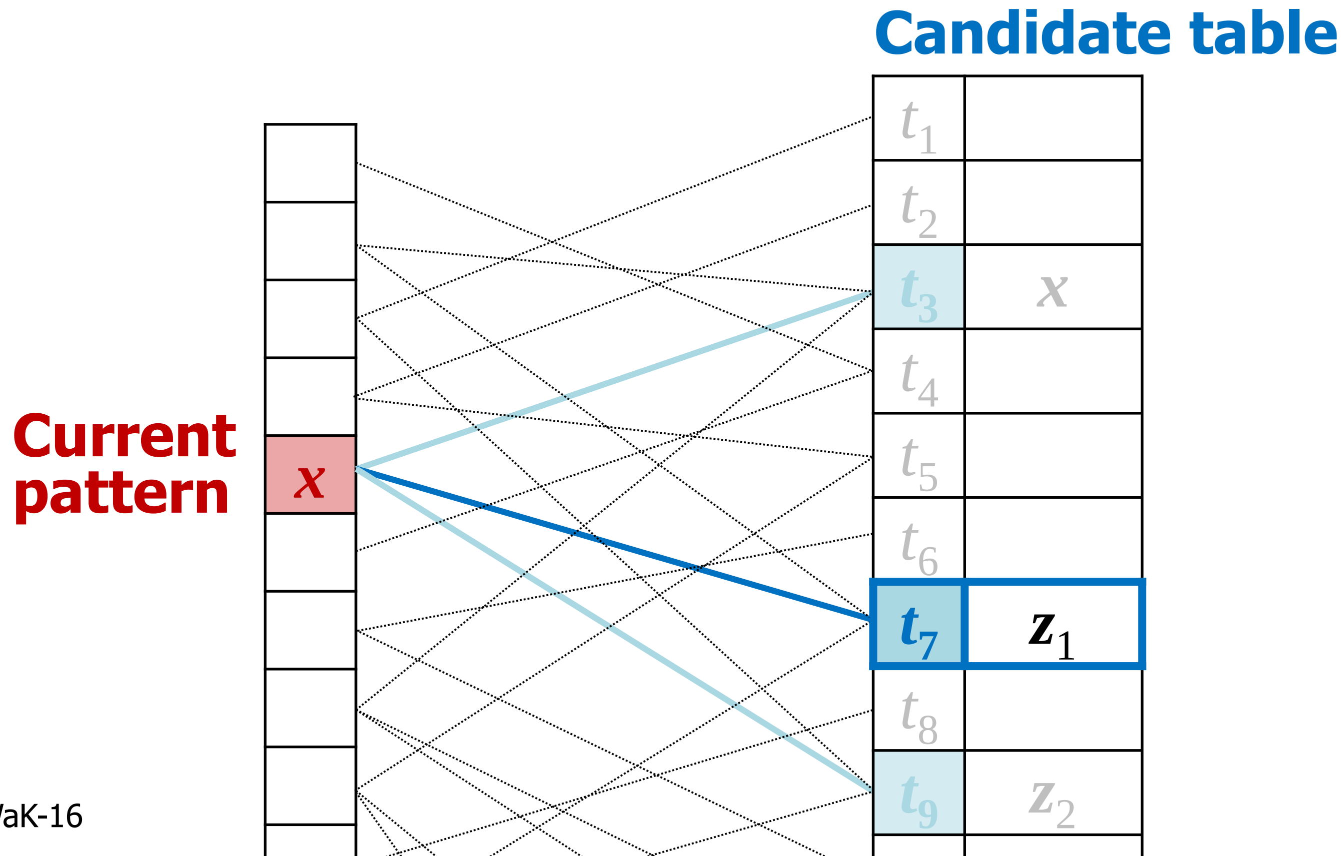
ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



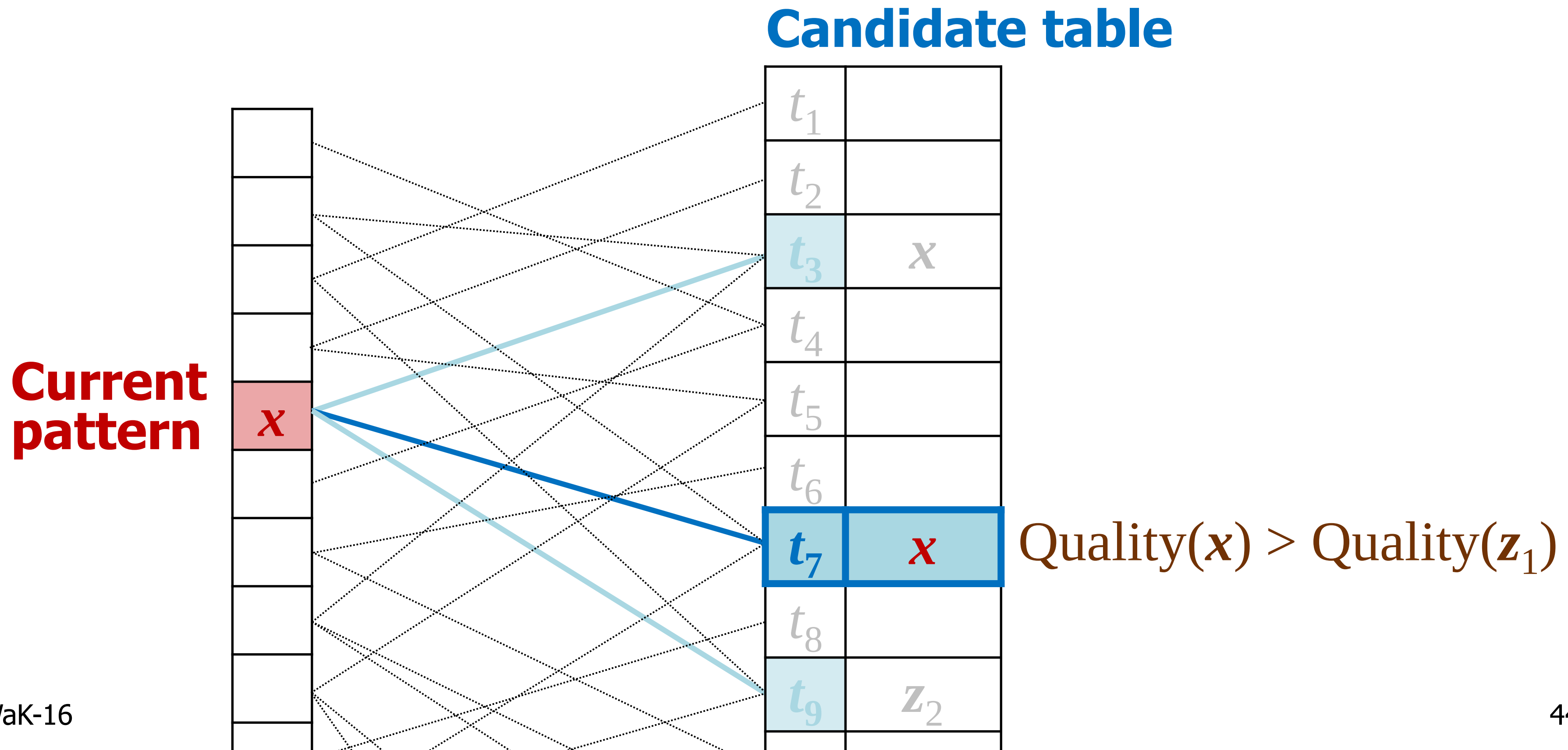
ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



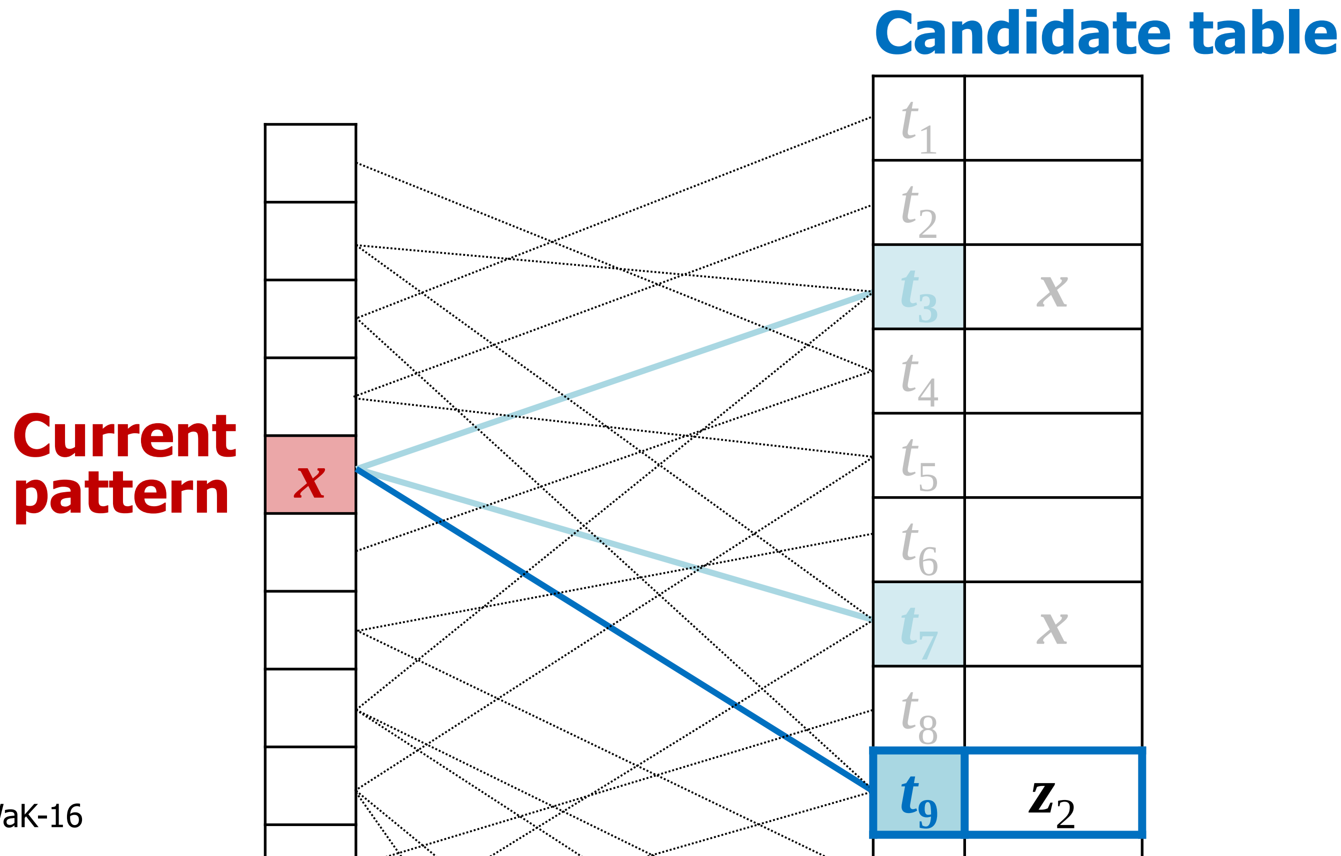
ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



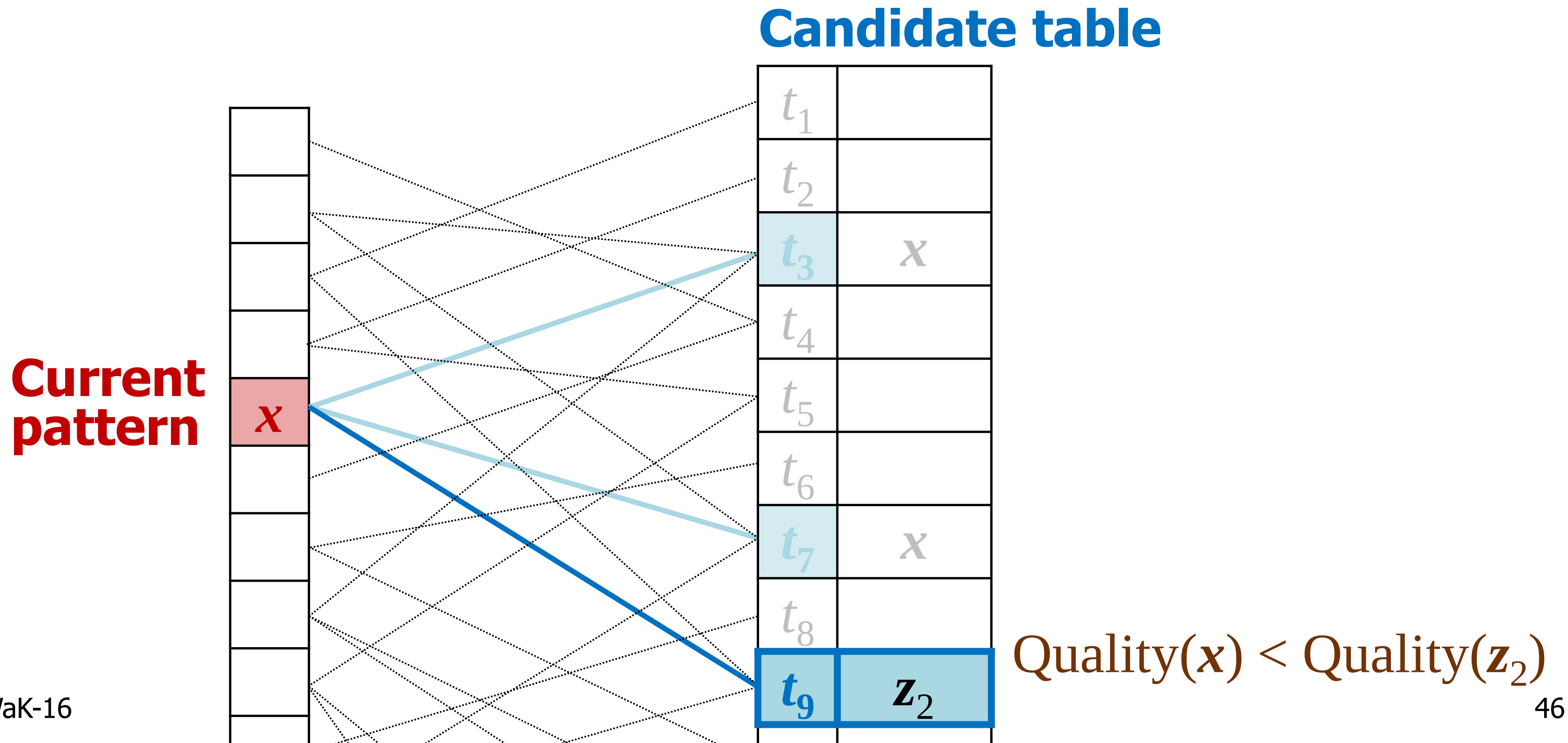
ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



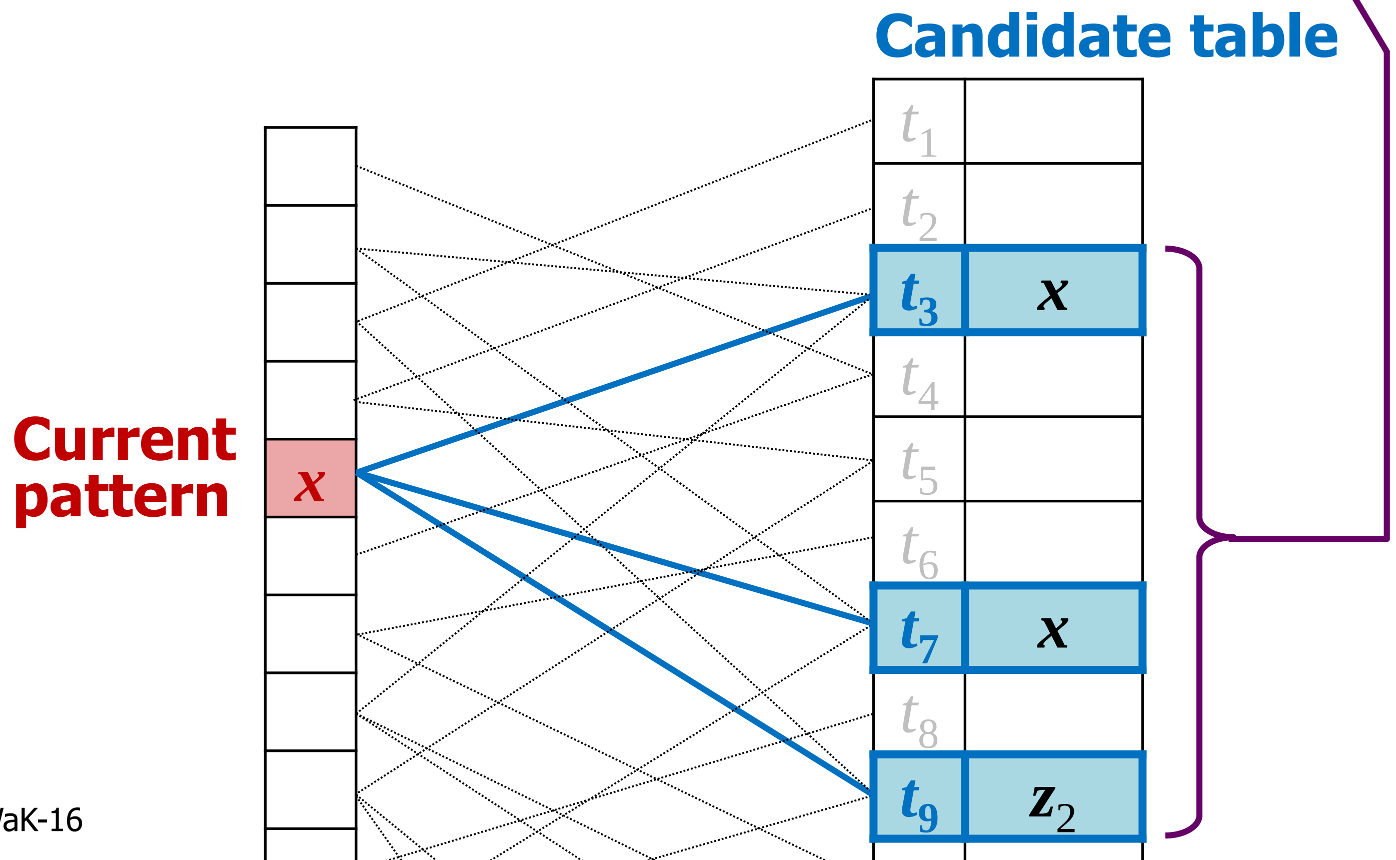
ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



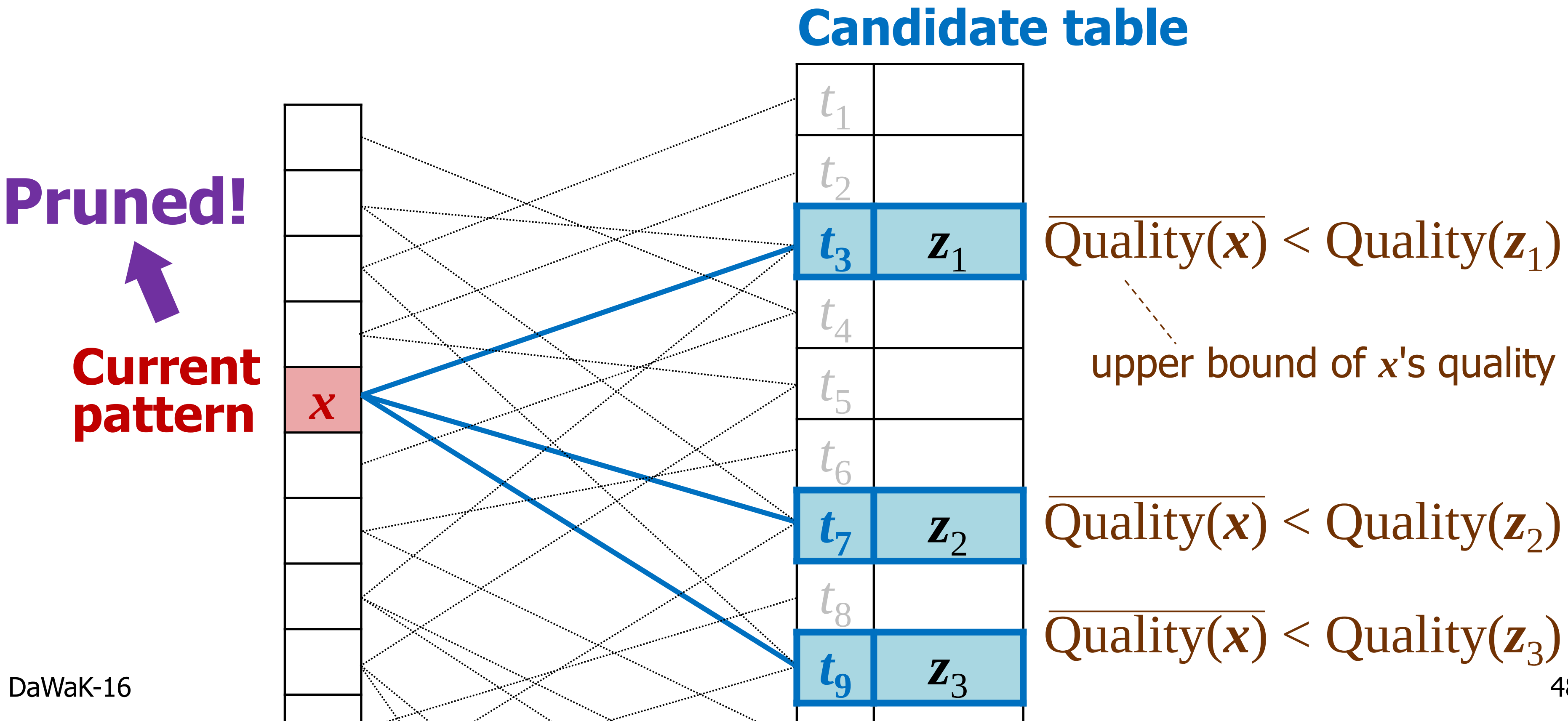
ExCover: Candidate table

- Basic strategy (cont'd):
  – Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



ExCover: Candidate table

- Basic strategy (cont'd):
  - Top-1 (Top-k with $k = 1$) mining **concurrently** for each positive transaction
    - Candidate table is a map:
      Positive transaction t → **Best competitor(s)** for t



ExCover: Property

- ExCover is...
  - Exhaustive
    - Only performs safe branch & bound pruning
  - Parameter-free
    - Conducts concurrent top-1 mining

Fixed inside the algorithm

ExCover: Related work

- HARMONY [Wang+ 05]
  - Uses the same strategy as that of ExCover
  - However its original paper does not mention on redundancy
  - Uses confidence $p(c \mid \boldsymbol{x})$ as the quality score
    - Confidence prefers highly specific patterns → Not easy to have its upper bound
    - User-specified minsup σ_{min} is required for pruning

Outline

- ✓ Background
- ✓ Our proposal
  - ✓ Best-covering constraint
  - ✓ ExCover
- Experiments

Experiments: Outline

- We use datasets from UCI ML Repository
- Experiment 1:
  - Detailed analysis on redundancy among patterns using the Mushroom dataset
- Experiment 2:
  - Analysis on search performance using 16 datasets preprocessed by the CP4IM project:

Dataset	#Trans.	#Items
anneal	812	93
audiology	216	148
australian-credit	653	125
german-credit	1,000	112
heart-cleveland	296	95
hepatitis	137	68
hypothyroid	3,247	88
kr-vs-kp	3,196	73

Dataset	#Trans.	Items
lymph	148	68
mushroom	8,124	110
primary-tumor	336	31
soybean	630	50
splice-1	3,190	287
tic-tac-toe	958	28
vote	435	48
zoo-1	101	36

Experiment 1: Mushroom

Covers
4,112 out of
4,208 positive
transactions

Productivity + Closedness + Top-k [Kameya+ 13] (k = 30)

Rank	Pattern	F-score
1	{odor=n, veil-type=p}	0.881
2	{gill-size=b, stalk-surface-above-ring=s, veil-type=p}	0.866
3	{gill-size=b, stalk-surface-below-ring=s, veil-type=p}	0.837
4	{gill-size=b, veil-type=p}	0.798
5	{stalk-surface-above-ring=s, veil-type=p}	0.776
6	{ring-type=p, veil-type=p}	0.771
7	{stalk-surface-below-ring=s, veil-type=p}	0.744
8	{veil-type=p}	0.682

Covers remaining
96 positive transactions

Experiment 1: Mushroom

Specifying $k < 5$ loses information from 96 positive transactions!

Productivity + Closedness + Top-k [Kameya+ 13] (k = 30)

Rank	Pattern	F-score
1	{odor=n, veil-type=p}	0.881
2	{gill-size=b, stalk-surface-above-ring=s, veil-type=p}	0.866
3	{gill-size=b, stalk-surface-below-ring=s, veil-type=p}	0.837
4	{gill-size=b, veil-type=p}	0.798
5	{stalk-surface-above-ring=s, veil-type=p}	0.776
6	{ring-type=p, veil-type=p}	0.771
7	{stalk-surface-below-ring=s, veil-type=p}	0.744
8	{veil-type=p}	0.682

Covers remaining 96 positive transactions

ExCover

Rank	Pattern	F-score
1	{odor=n, veil-type=p}	0.881
2	{gill-size=b, stalk-surface-above-ring=s, veil-type=p}	0.866
3	{stalk-surface-above-ring=s, veil-type=p}	0.776

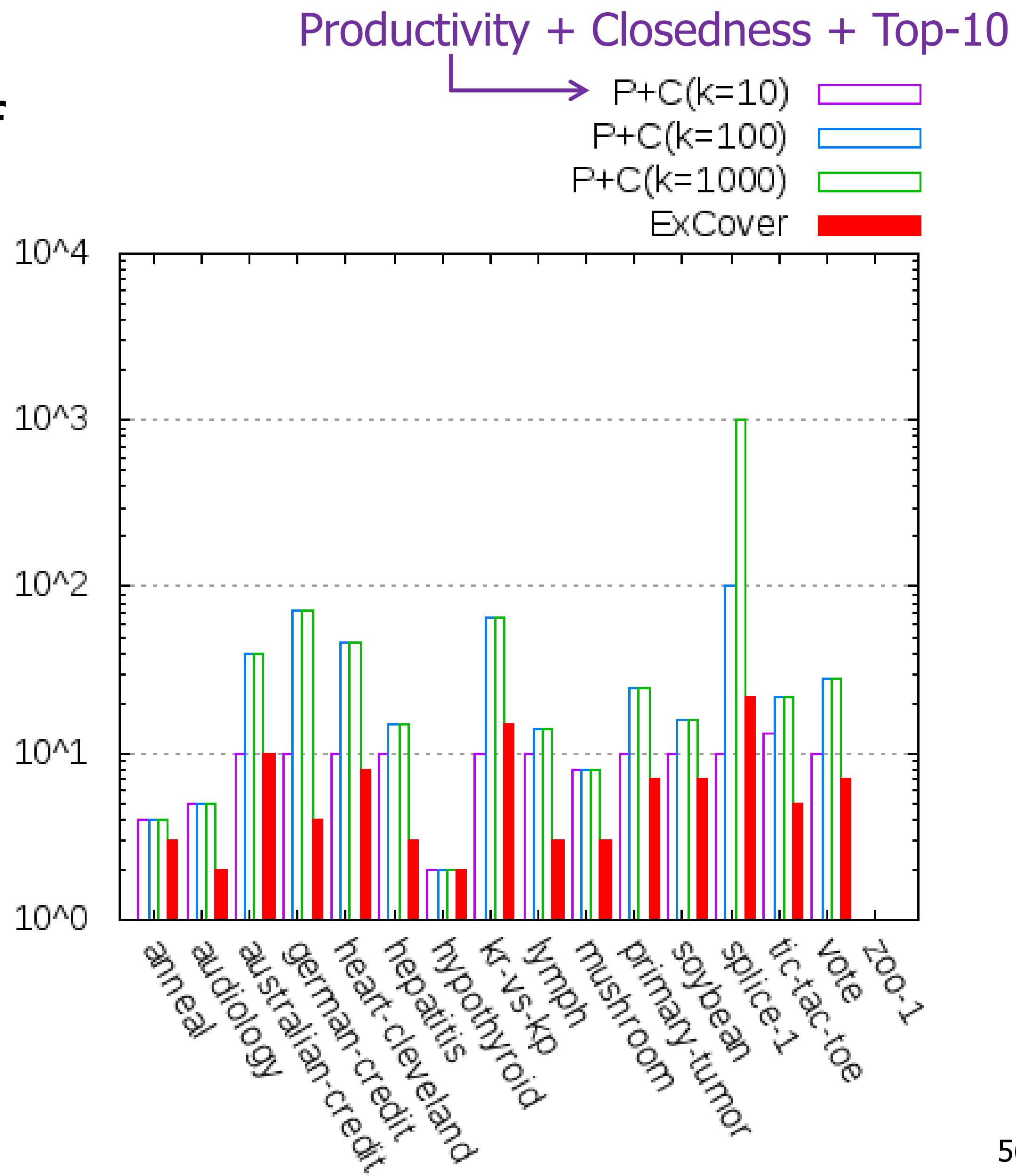
We only need 3 best-covering patterns to summarize the entire dataset

Experiment 2: Settings

- 16 datasets preprocessed by the CP4IM project
- Previous method in comparison [Kameya+ 13]:
  - Productivity + Closedness + Top-k
  - k was chosen from 10, 100 and 1,000

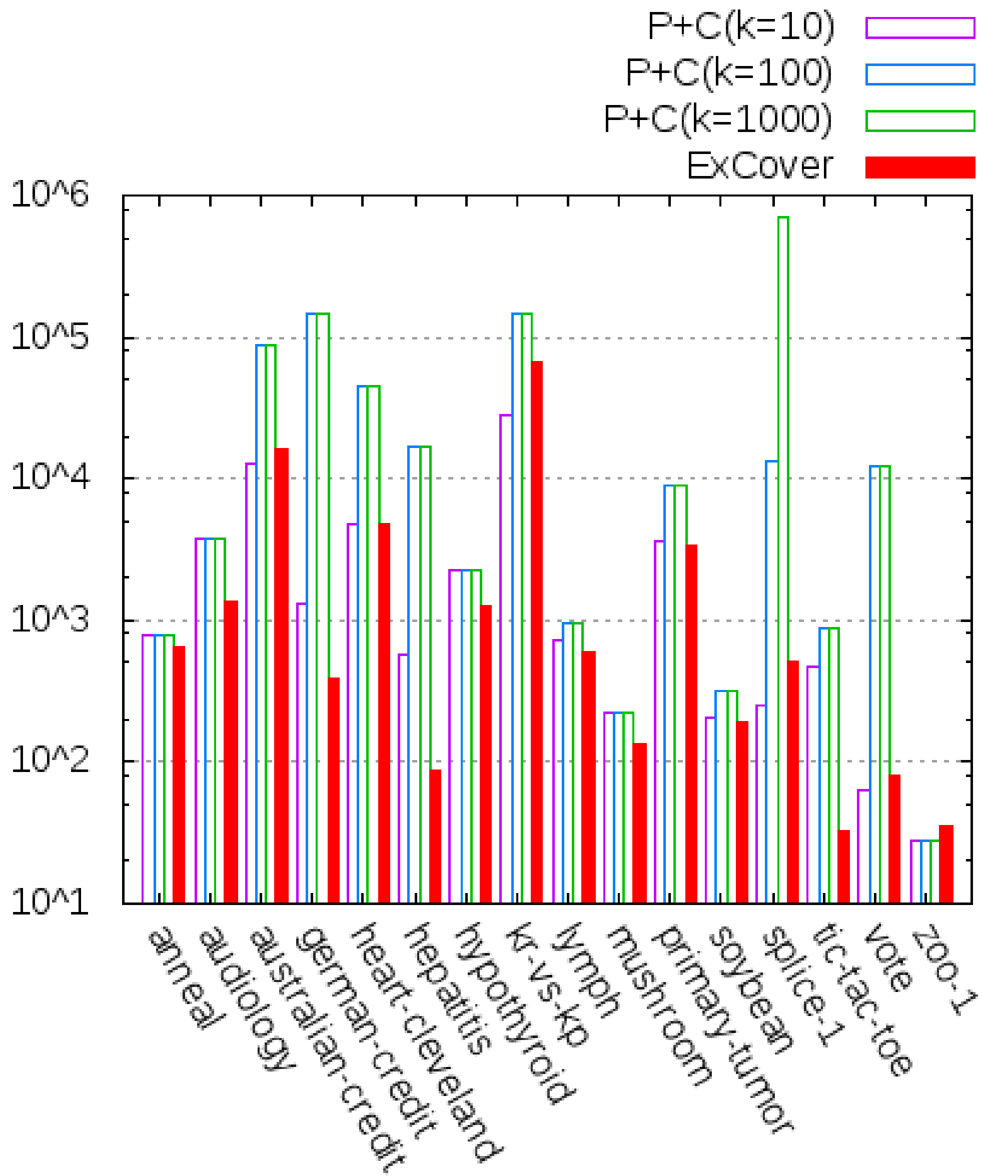
Experiment 2: #Patterns

- ExCover outputs a more compact set of patterns
- # of output patterns was moderate and did not vary



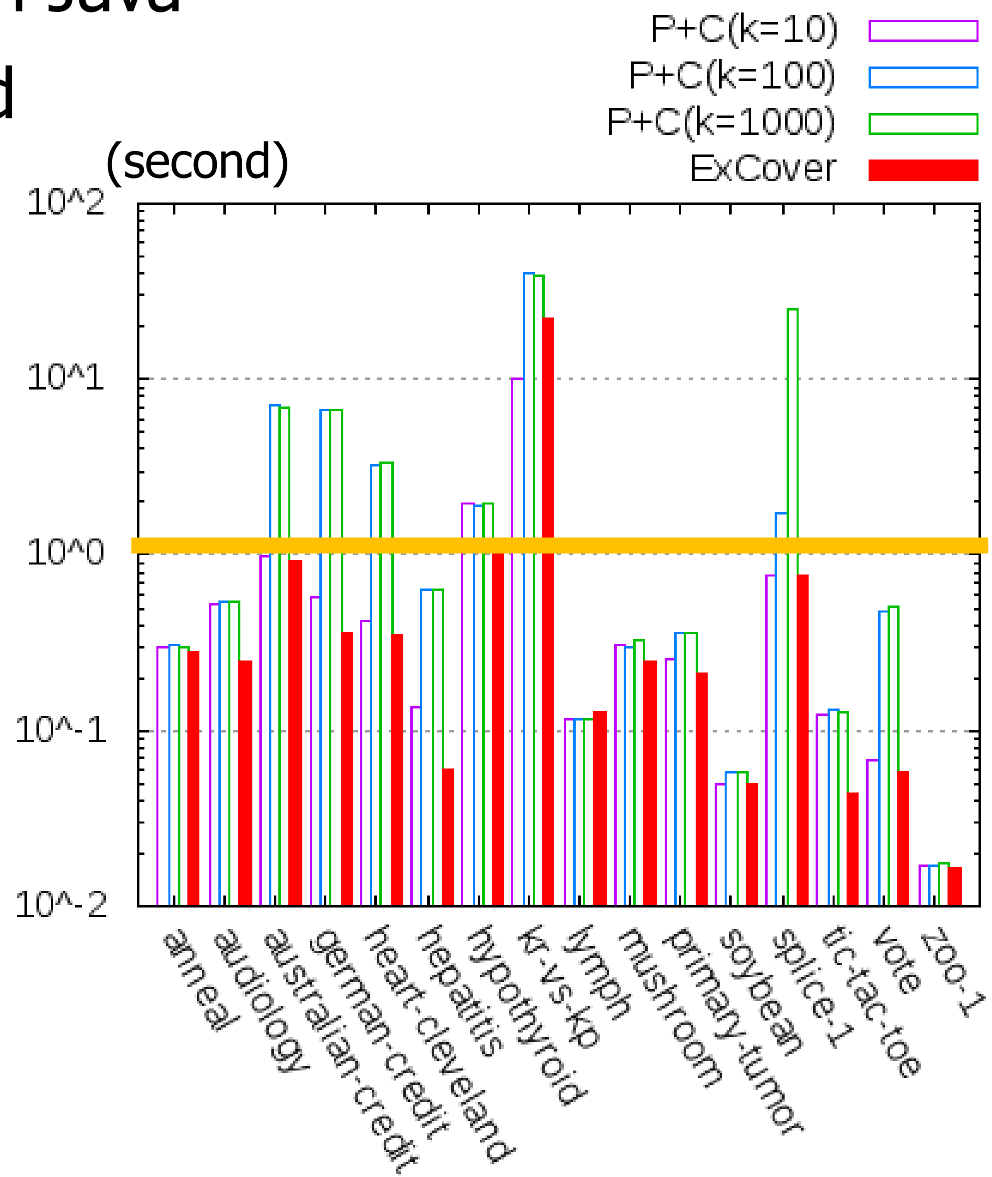
Experiment 2: Search space

- Search space =
  # of visited patterns
  in depth-first search



Experiment 2: Running time

- Our implementation: In Java
- Running time averaged over 30 runs
- For most datasets, ExCover finishes within one second



Summary

- ExCover: an efficient and exact method for finding non-redundant discriminative itemsets
  - Works under the best-covering constraint
  - Requires no control parameters limiting the search space
  - Finds a more compact set of patterns in a shorter time

Future work

- Transactions including numeric values
- Building classifiers from best-covering patterns
- Sequence pattern mining