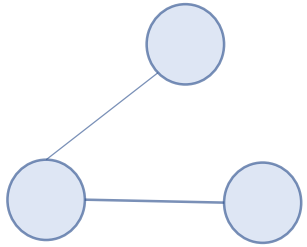
Mining Attribute Evolution Rules in Dynamic Attributed Graphs

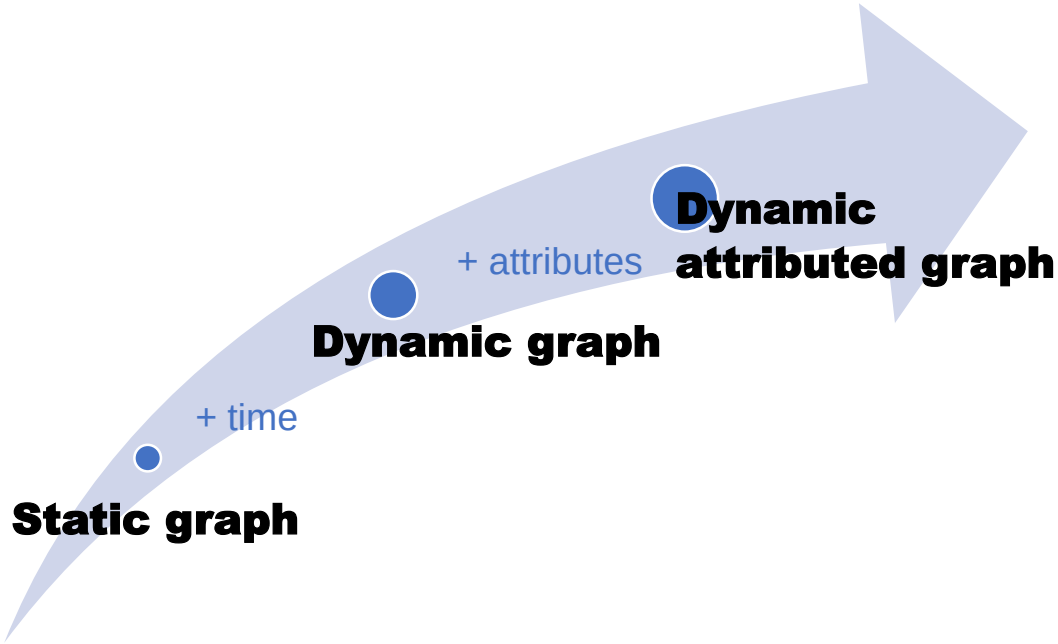
Philippe Fournier-Viger[1]
Ganghuan He[1]
Jerry Chun-Wei Lin[2]
Heitor Murilo Gomes[3]

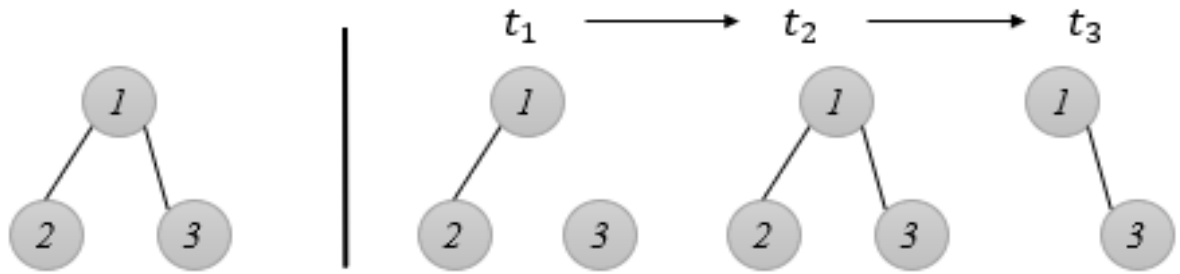
1 Harbin Institute of Technology (Shenzhen), China
2 Western Norway University of Applied Sciences, Norway
3 University of Waikato , New Zealand



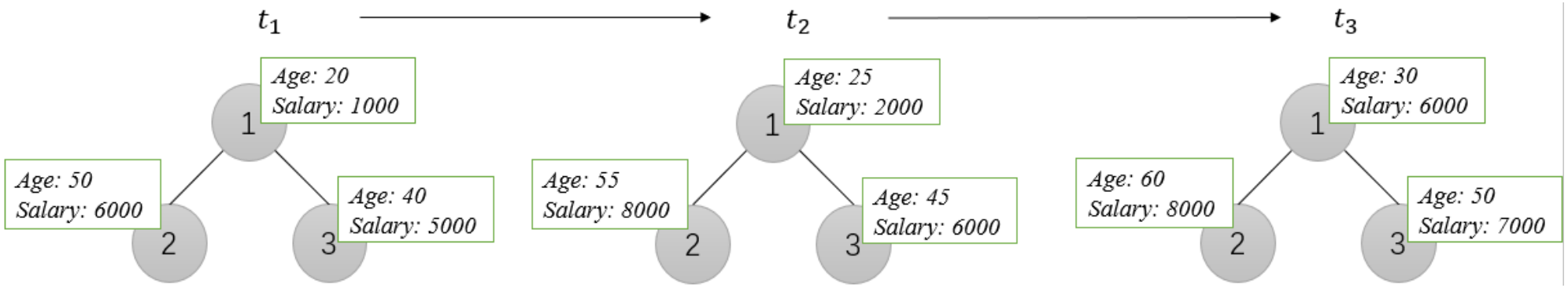
DAWAK 2020

Introduction · What's a dynamic attributed graph?





a) A static graph *b) A dynamic graph*



c) A dynamic attributed graph

Introduction · Previous work

1. Pattern mining in dynamic attributed graph

- Cheng et al. (2017) proposed to mining **recurrent pattern** in dynamic attributed graph.
- Desmier et al. (2012) proposed to mining **cohesive co-evolution patterns**.
- Fournier-Viger et al. (2019) proposed to mining **significant trend sequences**.

2. Rule mining in dynamic graph

- Berlingerio et al. (2009) proposed to mining **graph evolution rules**.
- Leung et al. (2010) proposed to mining **interesting link formation rules in social networks.**

Introduction · Limitations

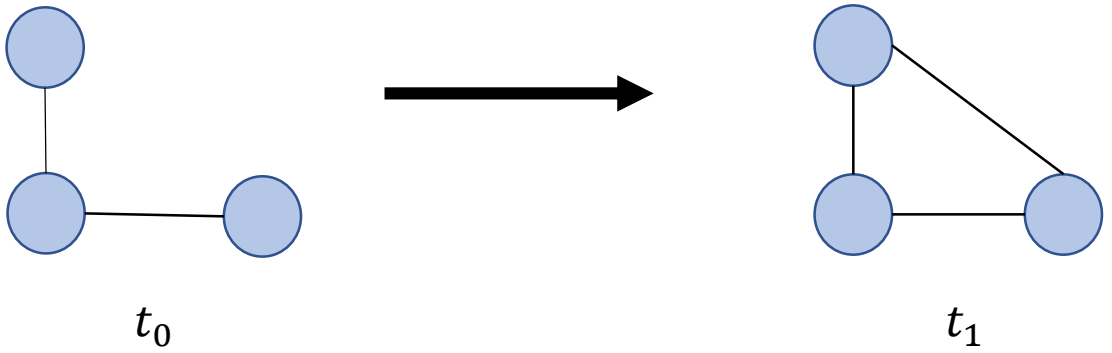
1. Pattern mining in dynamic attributed graph

- All vertices must follow a same trend and may be disconnected (recurrent pattern)
- Set of vertices is fixed (cohesive co-evolution patterns)
- Focus on influence of a vertex on its neighbors (significant trend sequences)

2. Rule mining in dynamic graph

- Only consider topology evolution rather than label evolution.

An example of graph evolution rule

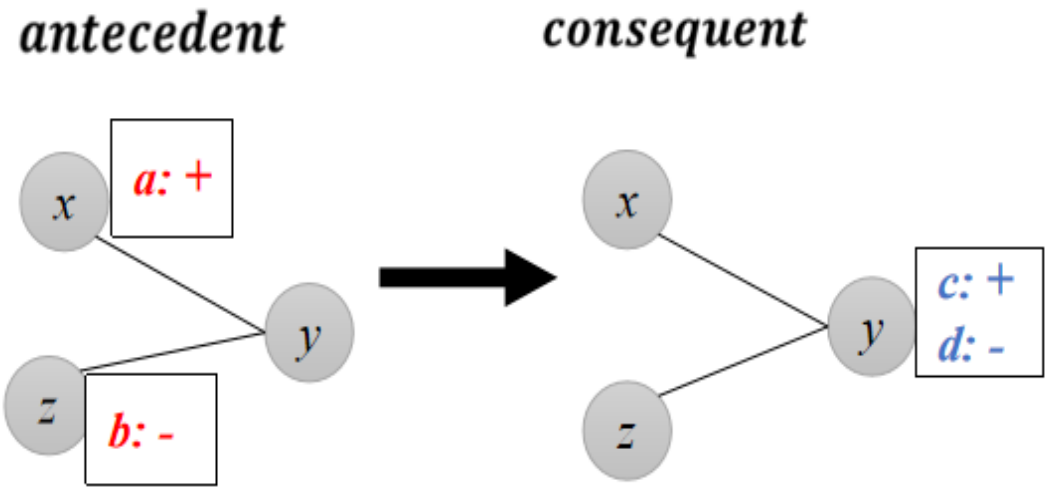


Introduction · Proposal

We propose to mine **attribute evolution rules (AERs)** in dynamic attributed graphs, which focus on finding out evolution rules describing how attribute values change for nodes of a subgraph.

Pattern characteristics

- Two parts: antecedent → consequent
- Two consecutive time intervals
- A connected subgraph



An attribute evolution rule

Introduction · Possible usage of AERs

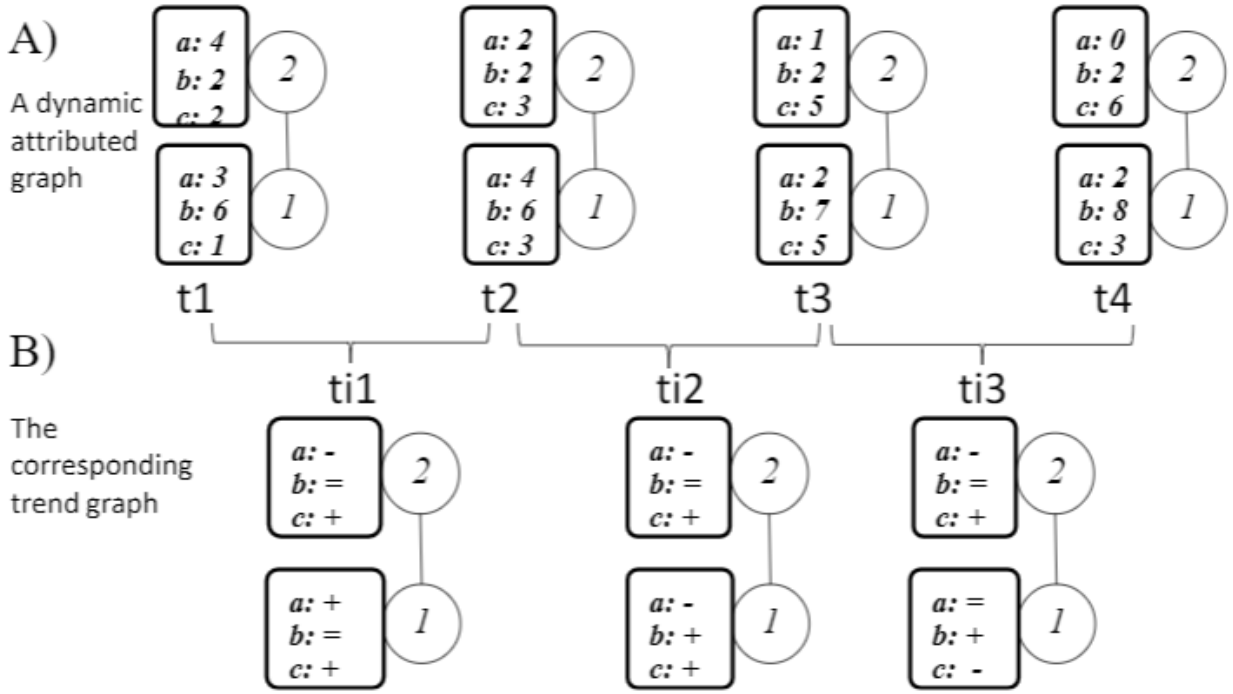
- The correlation analysis between attributes in a subgraph.
- Compression of graph
- Prediction for attributes in nodes of a graph.
- Recommendation for next timestamp behavior for social network.
- …

2. Definitions

Definitions

1. Trend graph:

Discretization of continuous attributes for dynamic attributed graph. The k-th graph G'_k of a trend graph indicates how attribute values have changed in the time interval from timestamp k to k+ 1



2. Attribute evolution rule:

An attribute evolution rule is a tuple R: (V,E,λ_{before},λ_{after})that indicates how the attribute values of a connected subgraph(V,E) have evolved for two consecutive time intervals in a trend graph G′.

Definitions · Measures

Problem: if define the **support** of a pattern as its number of occurrences in single graph, the powerful downward closure property cannot be directly used to reduce the search space.

Solution:

- maximum independent set-based support (Vanetik 2003).
- MNI (Minimum Node Image) support (Bringmann 2007).
- …

Definitions · Measure

Our problem can be considered as a multiple single graphs sequence. So a modified confidence for is defined based on MNI.

An **embedding** of a rule R is a mapping between the vertices of the rule and a set of vertices of two consecutive time intervals of a trend graph

c-support:

$\text{support}(R, G) = \sum_{j=1,2,\ldots,max-1} min_{v \in V_{consequent}} (|embedding(R, G_j)_{j,j+1}|)$

In short, the least number of distinct **consequent** nodes of its embeddings

Definitions · Correlation measure

C-Confidence: If the antecedent occurs, how likely is it will be followed by the consequent.

Lift: The *influence* of the antecedent on the consequent.

$$\mathrm{conf}(\mathrm{R}, \mathcal{G}') = \frac{\mathrm{Support}(\mathrm{R}, \mathcal{G}')}{\mathrm{Support}(\mathrm{Consequent\ of\ R}, \mathcal{G}')}$$

$$lift(R, \mathcal{G}') = \frac{conf(R, \mathcal{G}')}{expectedConf(antecedent\ of\ R, \mathcal{G}')}$$

3. AER-Miner

AER-Miner

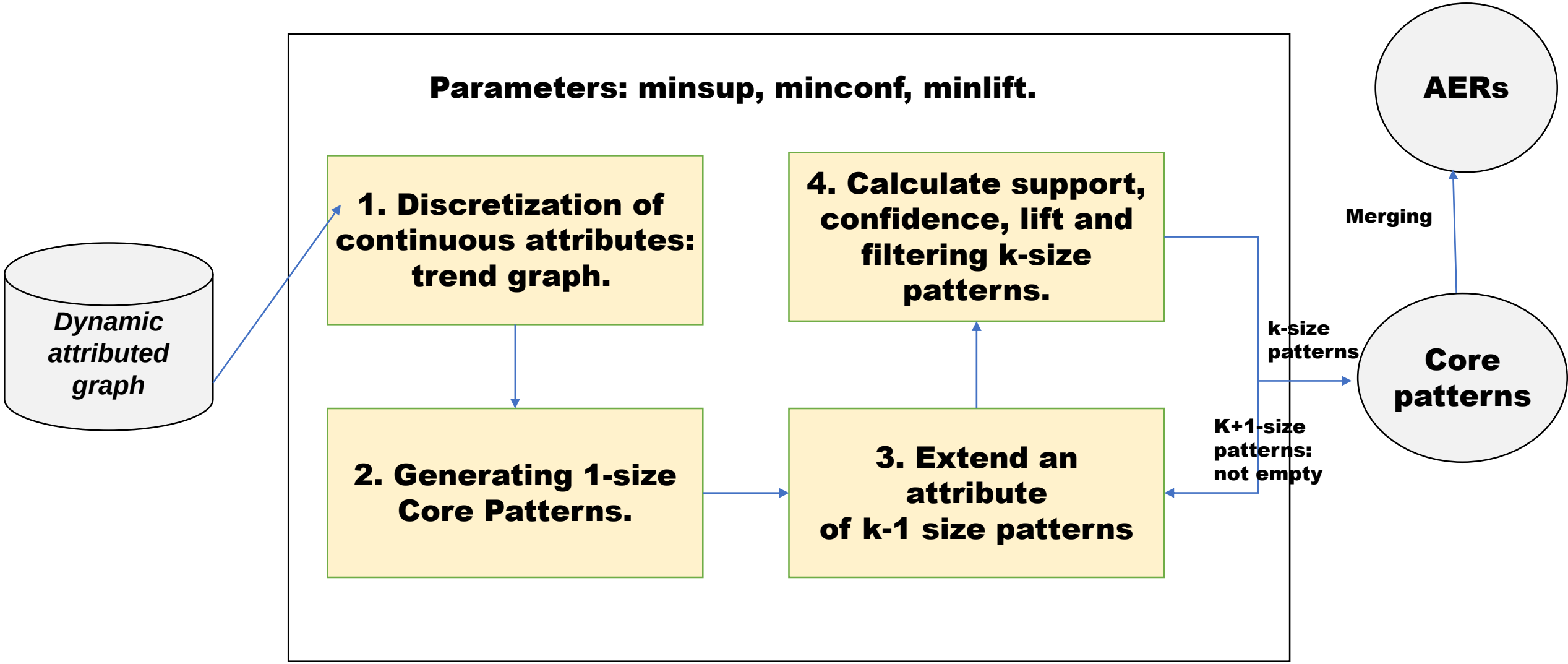
The structure of an AER can be relatively complex. Thus, rather than trying to enumerate all AERs directly, the proposed algorithm first finds core patterns, which are a simplified form of AERs. Then, it merges core patterns.

Core pattern:

Core pattern is an AER composed of a consequent node and several antecedent nodes, where each consequent-antecedent node pair is connected.

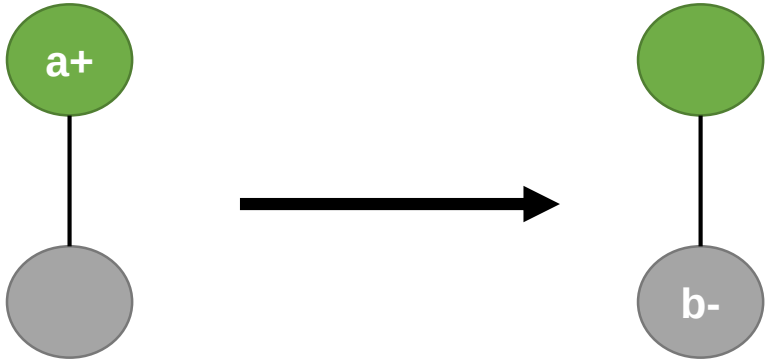
Moreover, each node is described using a single attribute.

AER-Miner

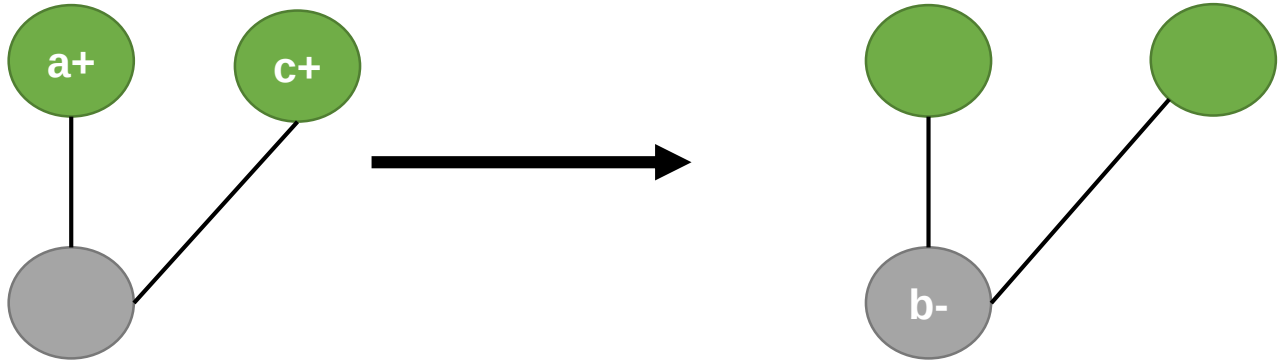

Parameters: minsup, minconf, minlift.
Dynamic attributed graph
1. Discretization of continuous attributes: trend graph.
2. Generating 1-size Core Patterns.
3. Extend an attribute of k-1 size patterns
4. Calculate support, confidence, lift and filtering k-size patterns.
K+1-size patterns: not empty
k-size patterns
Core patterns
Merging
AERs

AER-Miner

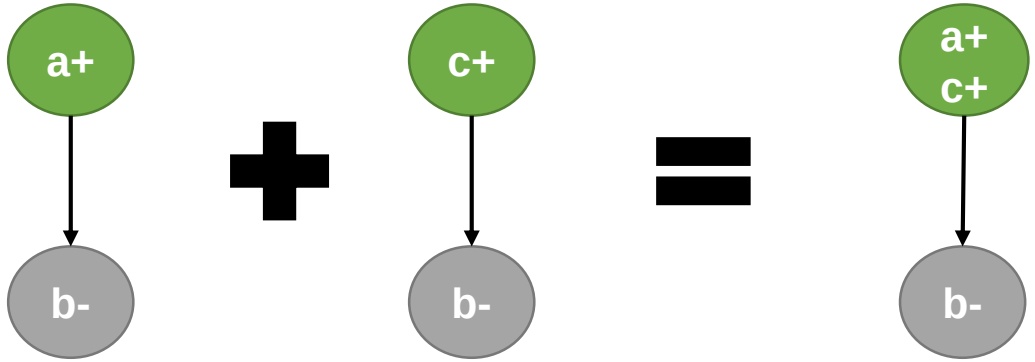
Generating 1-size Core Patterns:



Extend an attribute on k-1 size patterns :



Merging core patterns to AERs:



AER-Miner • Pruning strategies

To reduce search space:

1. Measures: support, lift
2. When extending an attribute following a breadth-first search, the attribute order is lexicographical order
3. Limit the size k of patterns
4. Ignore uninteresting attribute trends (e.g. no change)

AER-Miner • Pseudocode

Algorithm 1: The AER-Miner algorithm

input : a dynamic attributed graph $\mathcal{G}$ or trend graph $\mathcal{G}'$, the $minsup$, $minlift$ and $minconf$ thresholds
output: all the valid attribute evolution rules

```
1  Initialize a core pattern Map map_candidates <core pattern,instances> for
   growing patterns. Initially, each pattern contains one consequent attribute.
2  Initialize a list list_patterns for storing all possible core patterns.
3  while map_candidates ≠ ∅ do
4      map_{k+1sizecandidate} ⇐ ∅
5      foreach candidate ∈ map_candidates do
6          pattern ← candidate.key
7          instances ← candidate.value
8          foreach attr ∈ attributelist do
9              if attr ≥ the last attribute of pattern and attr ≠'=' then
10                 newPattern ← pattern ∪ attr
11                 newInstances ← extendInstances(pattern, attr, instaces)
12                 support ⇐ sizeofnewInstance
13                 lift, confidence ← calLiftAndConfi(pattern, attr, instaces)
14                 if support ≥ minsup and lift ≥ minlift then
15                     put < newPattern, newInstance >∈ map_{k+1sizecandidate}
16                 end
17             end
18         end
19     end
20     list_patterns ← list_patterns ∪ map_{k+1sizecandidate}
21 end
22 list_patterns ← filterPatterns(list_patterns, minconf)
23 Return AERs = list_patterns ∪ mergePatterns(list_patterns)
```

4. Experiment and analysis

Experiment and analysis

Data

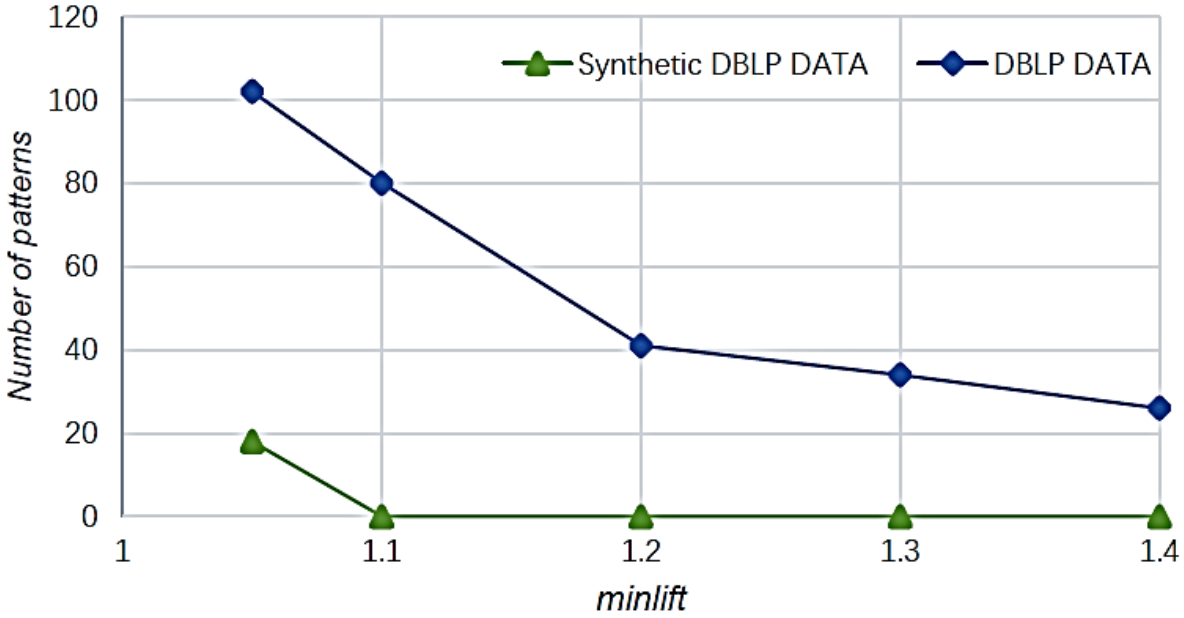
Dataset	|Vertices|	|AvgEdgesPerTimestamp|	Timestamps|	|Attributes|
DBLP	2,723	10,737	9	43
US Flight	280	1,206	8	8
synthetic-DBLP	2,723	10,737	9	43
synthetic-USFlight	280	1,206	8	8

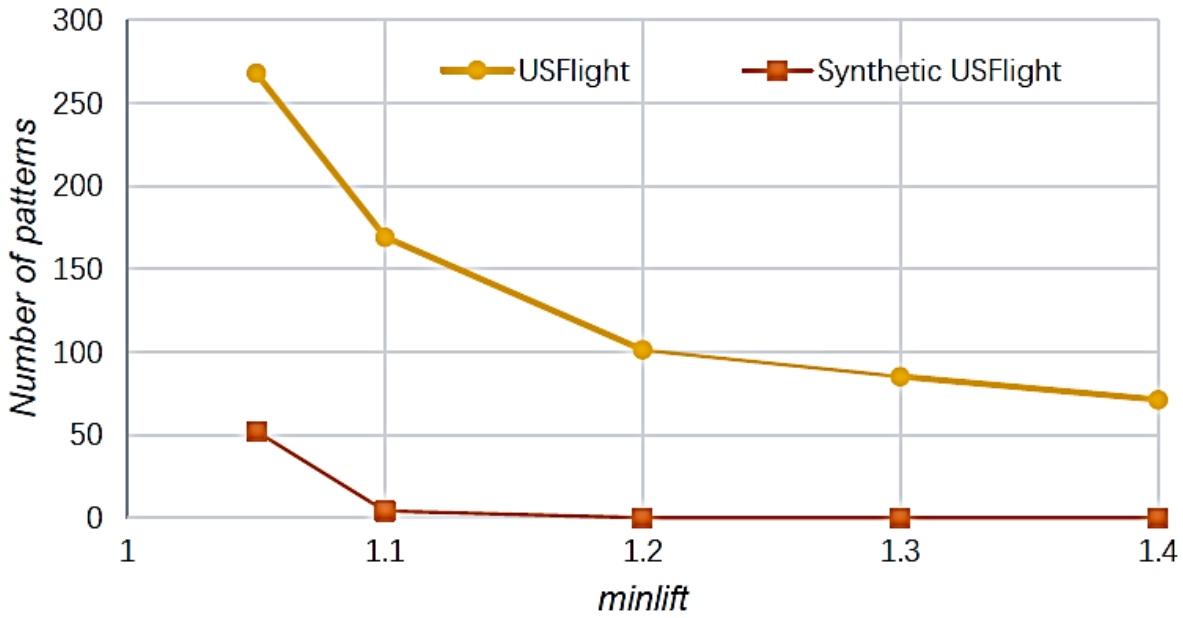
1. DBLP is a co-authorship dataset containing data from the DBLP online bibliographic service.

2. US Flight contains data about US air traffic during the Katrina hurricane period

3. Two **synthetic** dataset: Same topology with above dataset, but attribute values of each vertex were randomly generated following a Gaussian distribution.

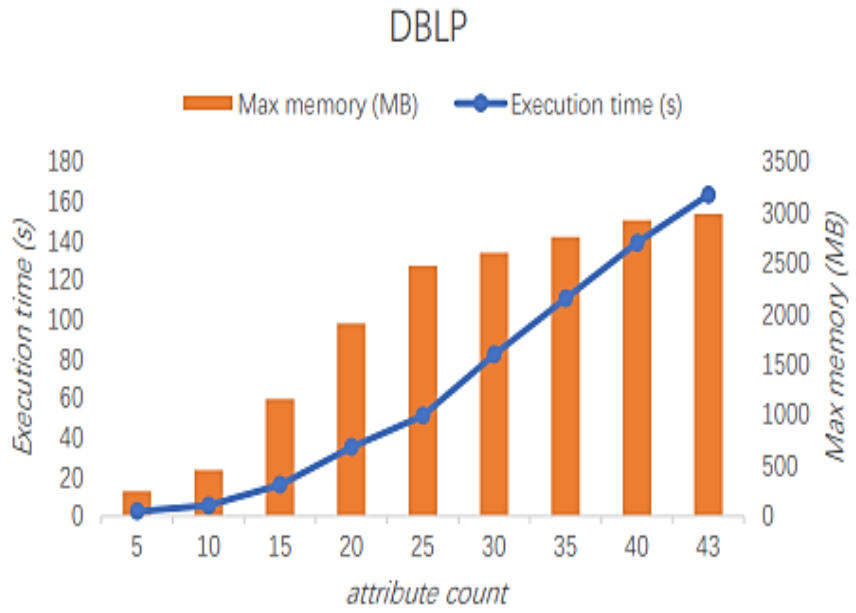
Statistical validation experiment

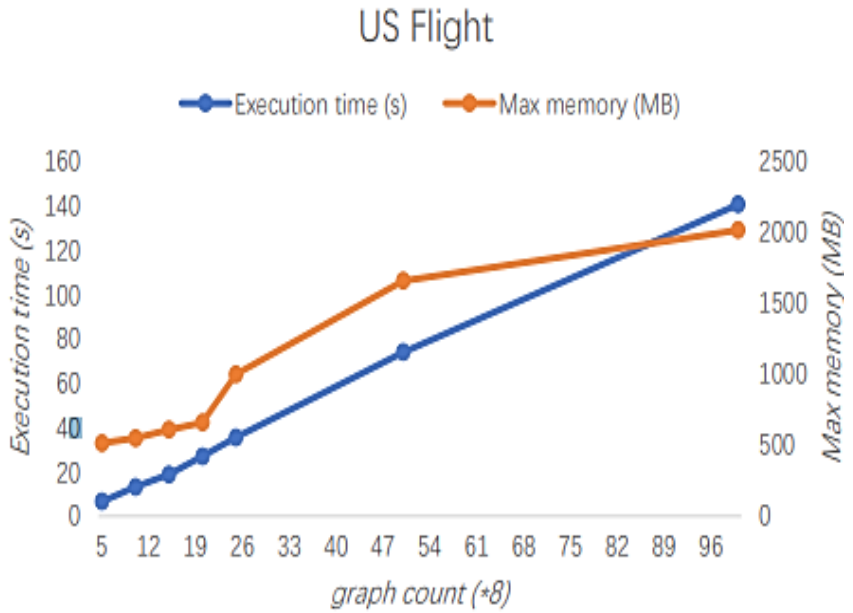


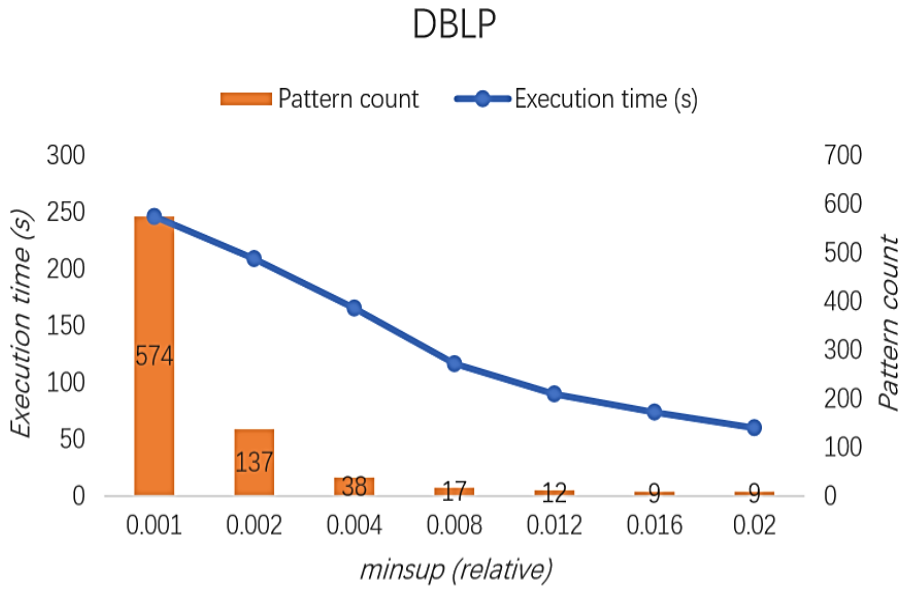


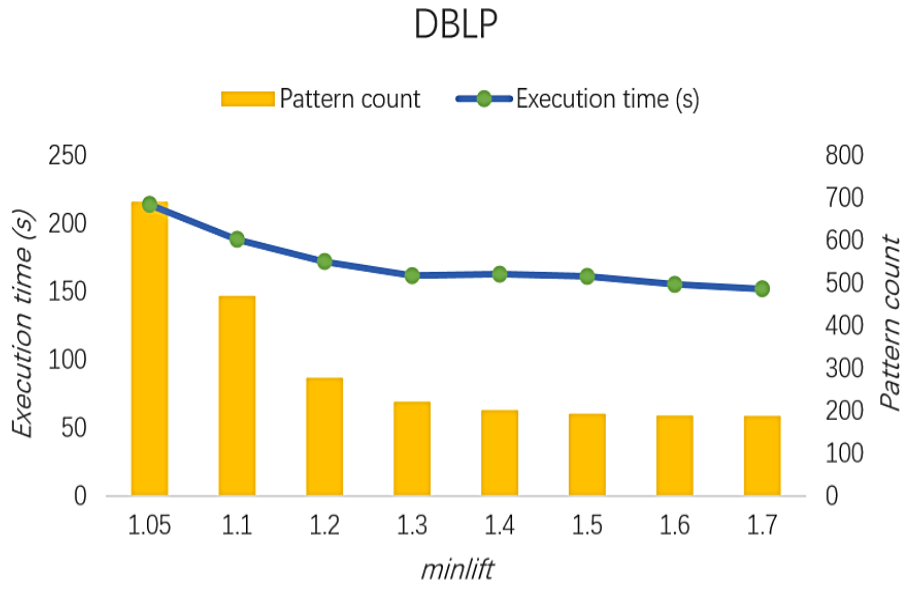
- No patterns found for synthetic datasets when *minlift* is large enough
- When minlift is small (<1.1), some random or noise AERs also generated in synthetic datasets

Quantitative experiments









Qualitative assessment: interesting real patterns

DBLP pattern : $< (\boldsymbol{PVLDB}+), (\boldsymbol{PVLDB}+), (\boldsymbol{PVLDB}+) > \rightarrow < (\boldsymbol{VLDB}-) >$

US-Flight pattern: $< (\boldsymbol{NbDeparture}-) > \rightarrow < (\boldsymbol{NbCancelation}+) >$

It indicates that departure cancellations caused by the hurricane are strongly correlated with a flight cancellation increase at the next timestamp

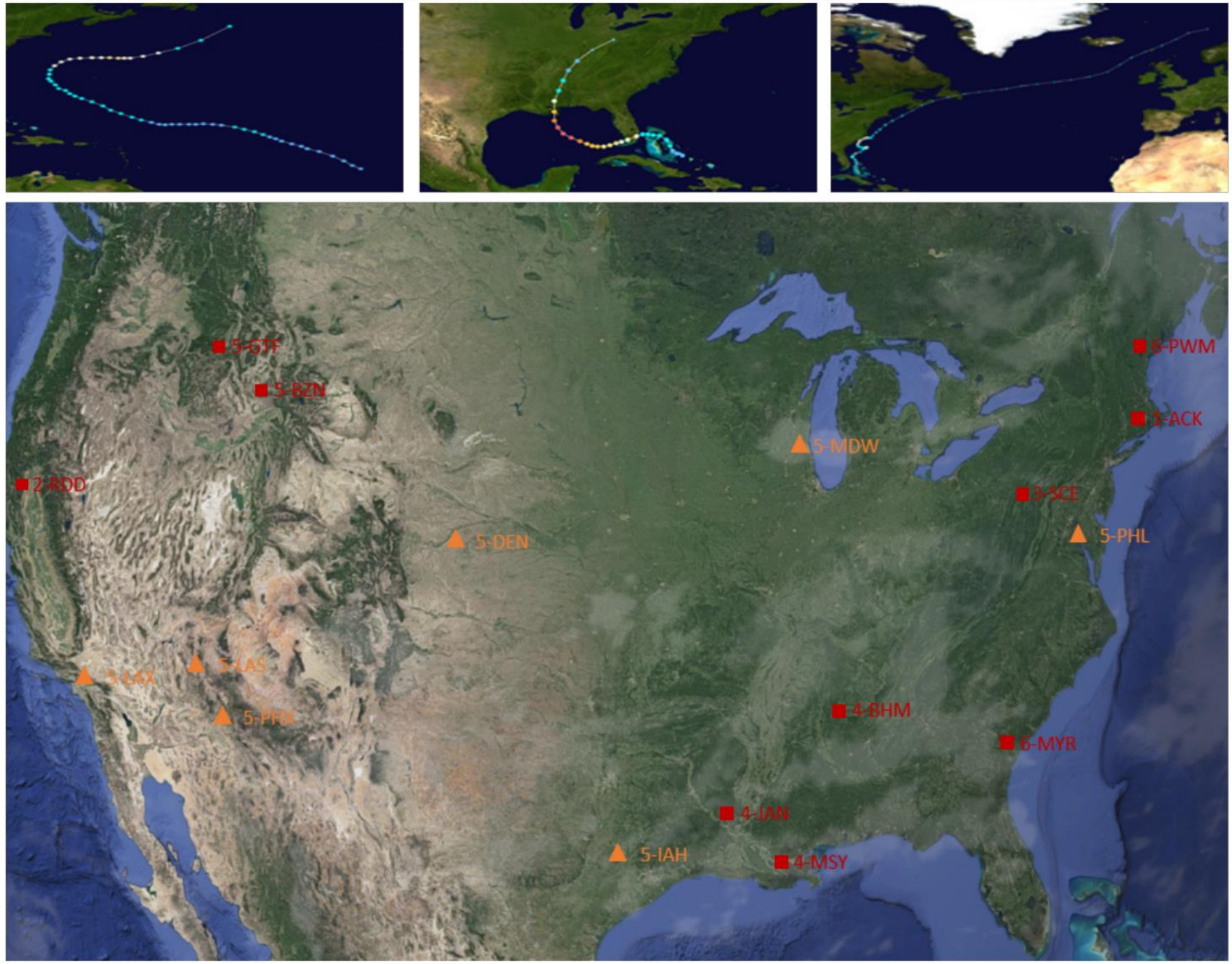
5. Conclusion

Conclusion

- Novel type of patterns, attribute evolution rules, indicating how changes of attribute values of multiple vertices may influence those of others.
- An efficient algorithm named AER-Miner.
- Experiments on real data have shown that AER-Miner is efficient and that AERs can provide interesting insights about real-life dynamic attributed graphs.

Q & A

Open source Java data mining software, 150 algorithms
http://www.phillippe-fournier-viger.com/spmf/


5-GTF
5-BZN
2-RDD
5-DEN
5-LAX
5-LAS
5-PHX
5-MDW
6-PWM
1-ACK
3-SCE
5-PHL
4-BHM
6-MYR
4-JAN
5-IAH
4-MSY

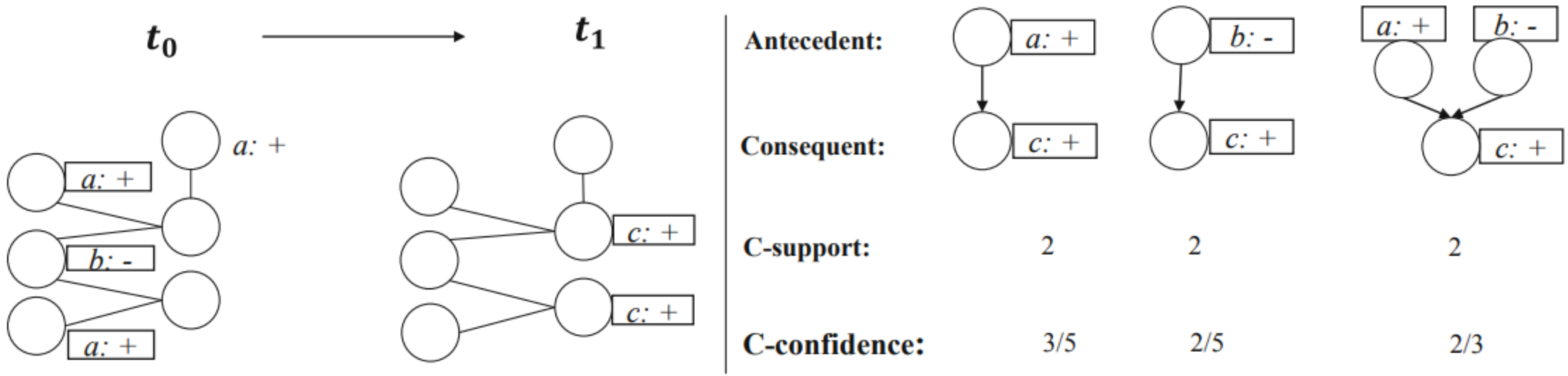


Fig. 3. The c-support and c-confidence of three AERs