Case Western Reserve University

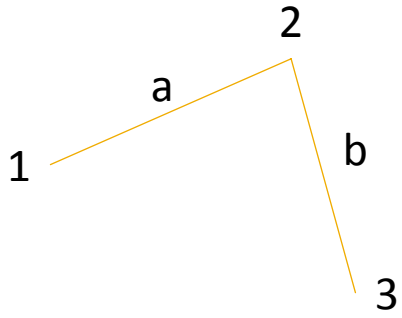
RAM: Randomized Approximate Graph Mining

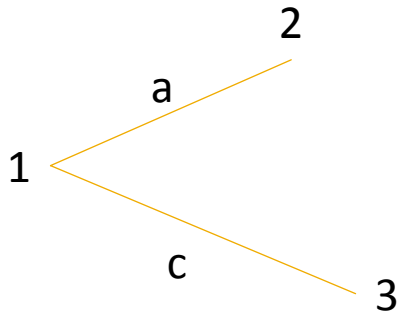
Why approximate graph mining

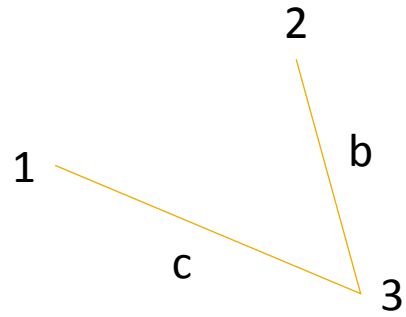
- Incomplete or inaccurate data
  - biological data are known to be inaccurate, e.g., gene expression profiles, protein interaction networks, metabolic pathways.
  - human social interactions can be difficult to define.
  - In procedure dependency graphs, inaccuracy exists with missing paths, conditions, and cases in the field of software engineering

What is approximate graph mining?

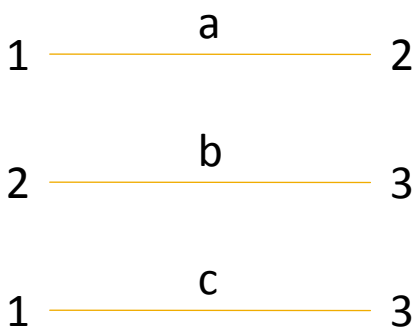
Given a database of 3 graphs, and a minimum support of 2:



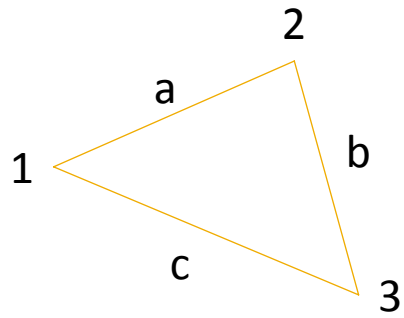




Regular graph mining:

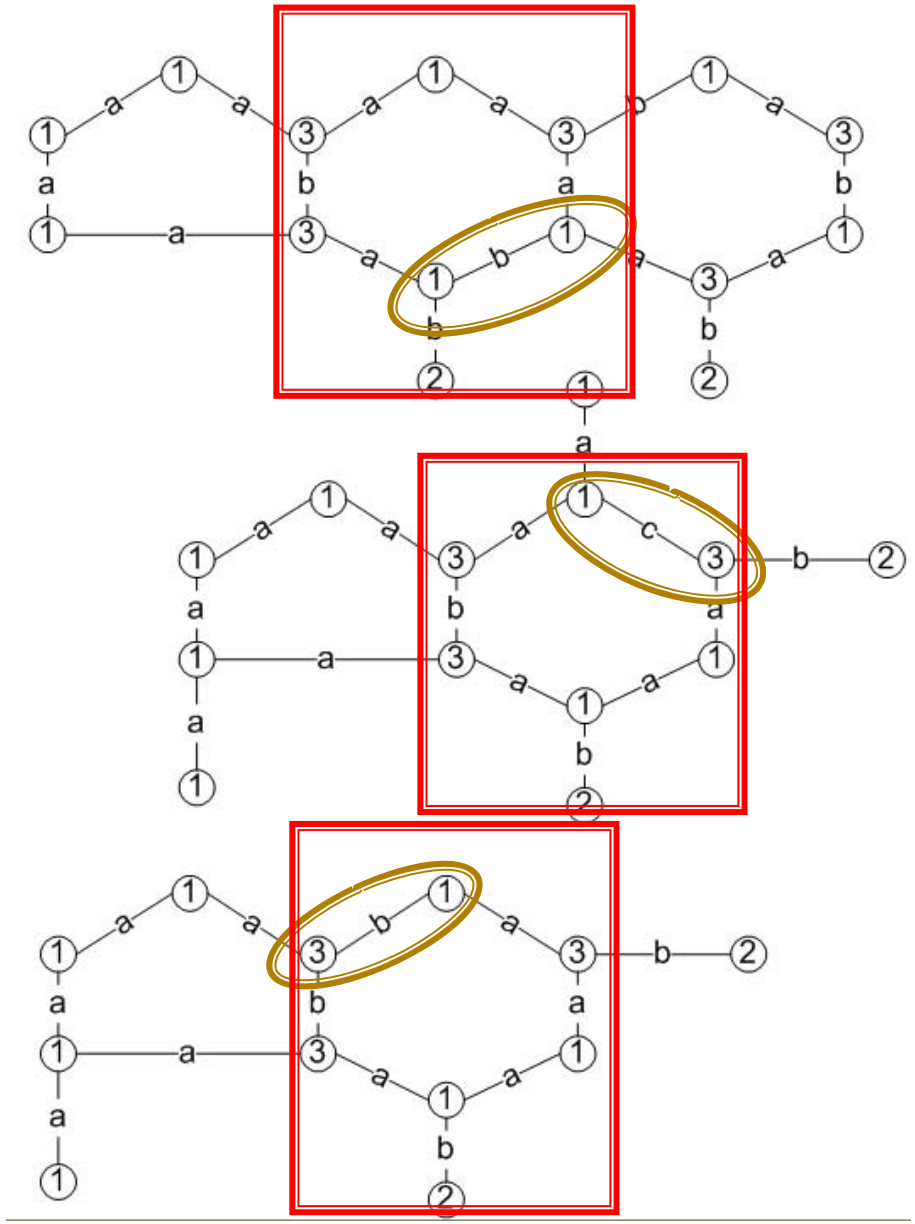


Approximate graph mining:

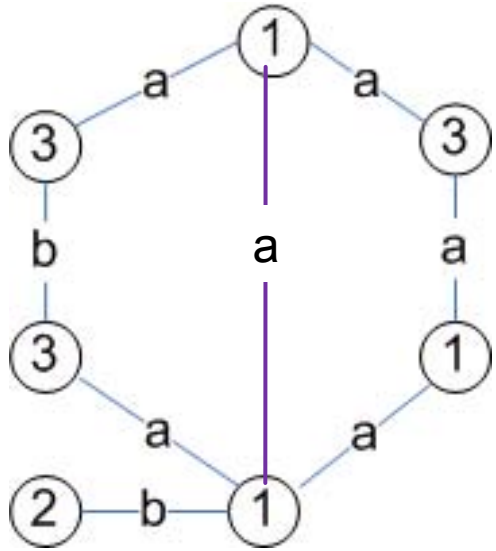


Basics

- Support (γ) – occurrences of pattern in database
- Variability (β) – maximum edges that can be missing from a match
- Tolerance (α) – maximum times an edge can be missing from the match set



Candidate Graph:



γ = 3

β = 1

α = 2

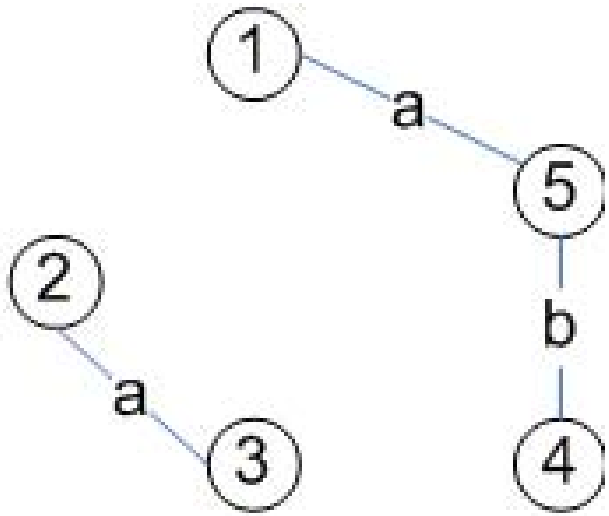
Defining aFG

- Labeled graph $G = \{V, E, \Sigma_V, \Sigma_E, L_G\}$
  - V: Vertices
  - E: Edges
  - Σ_V: Vertex labels
  - Σ_E: Edge labels
  - L_G : maps V and E to Σ_V and Σ_E

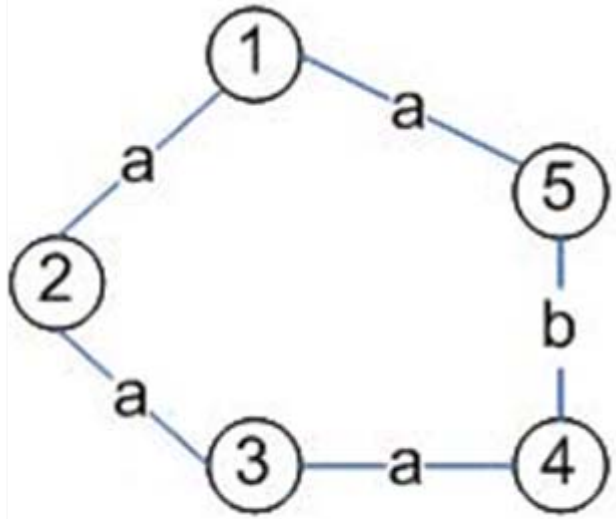
Defining aFG

β edge isomorphism

$p \approx_\beta q$ ($\beta \geq 2$)



Graph p



Graph q

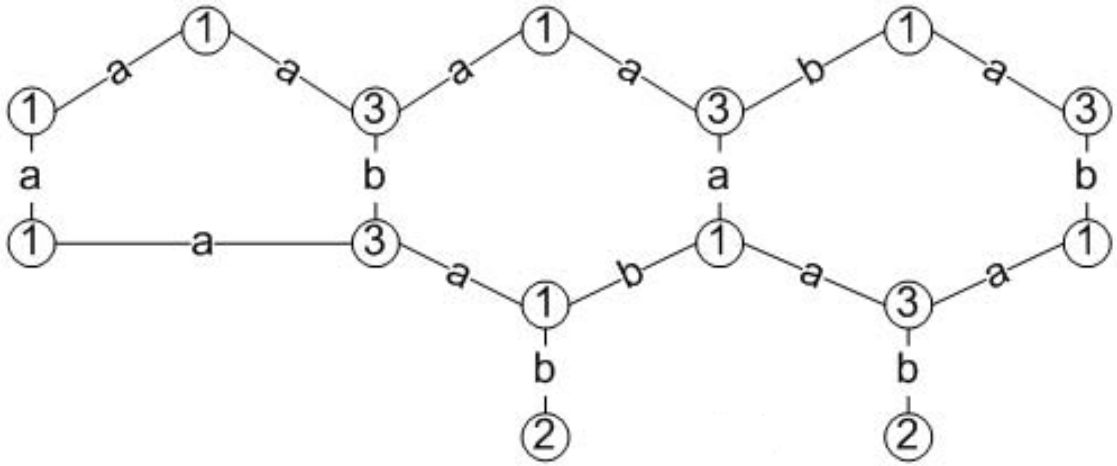
Defining aFG

β edge subgraph isomorphism

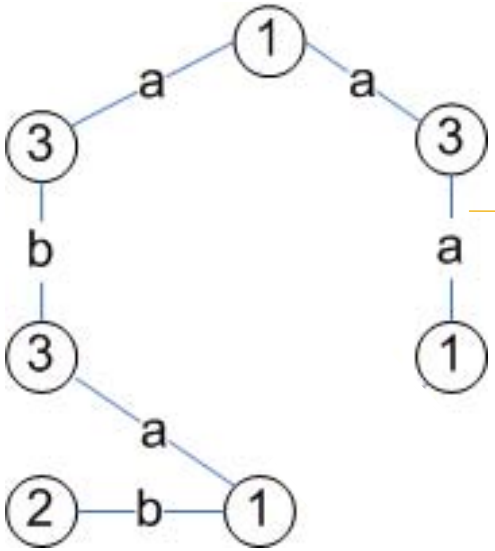
$$\mathbf{G} \subseteq_{\beta} \mathbf{G'}$$

if there is a $\mathbf{G''} \subseteq \mathbf{G'}$ where $\mathbf{G''} \approx_{\beta} \mathbf{G}$

DB graph (G'):

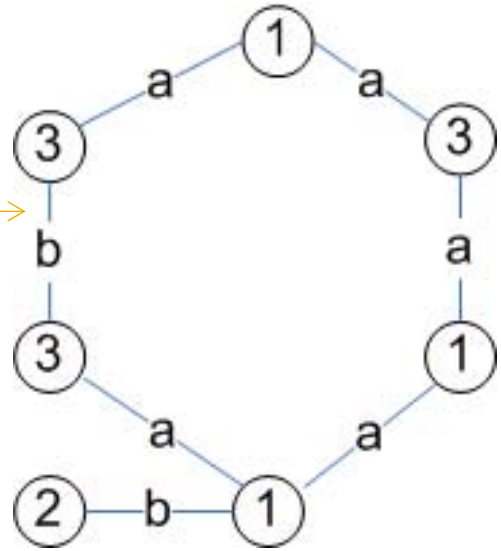


Match (G" $\subseteq$ G'):



G" $\approx_\beta$ G

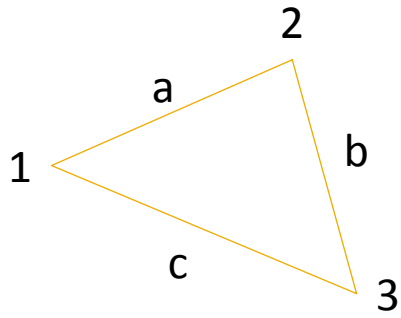
Candidate Graph (G):



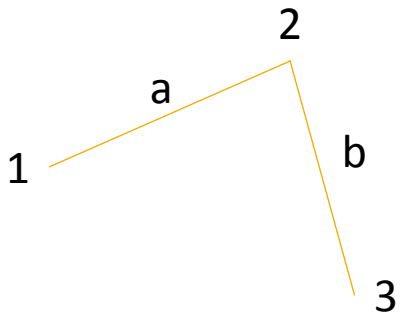
G $\subseteq_\beta$ G'

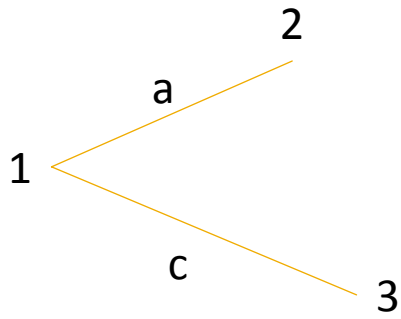
Defining aFG

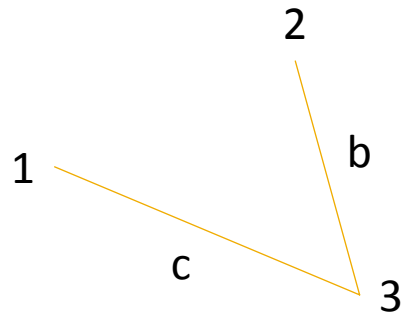
Given $\beta = 1$, what is the support for:



In database:

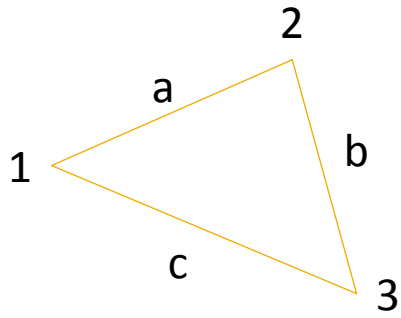




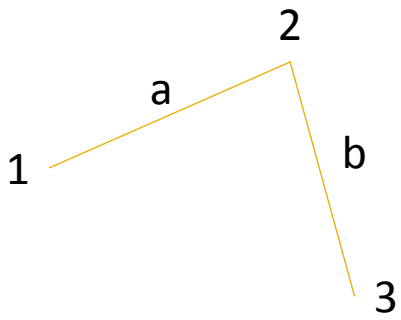


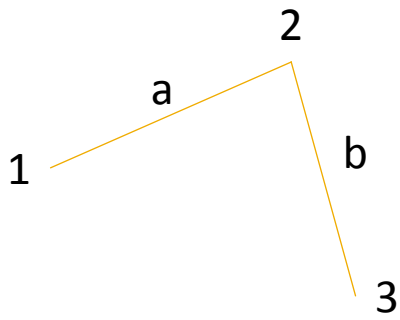
Defining aFG

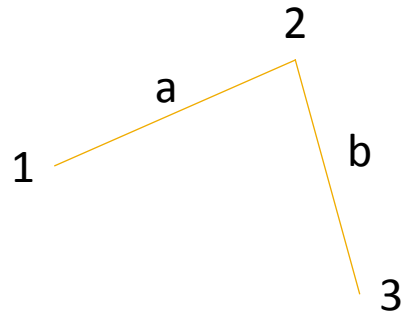
Given $\beta = 1$, what is the support for:



In database:

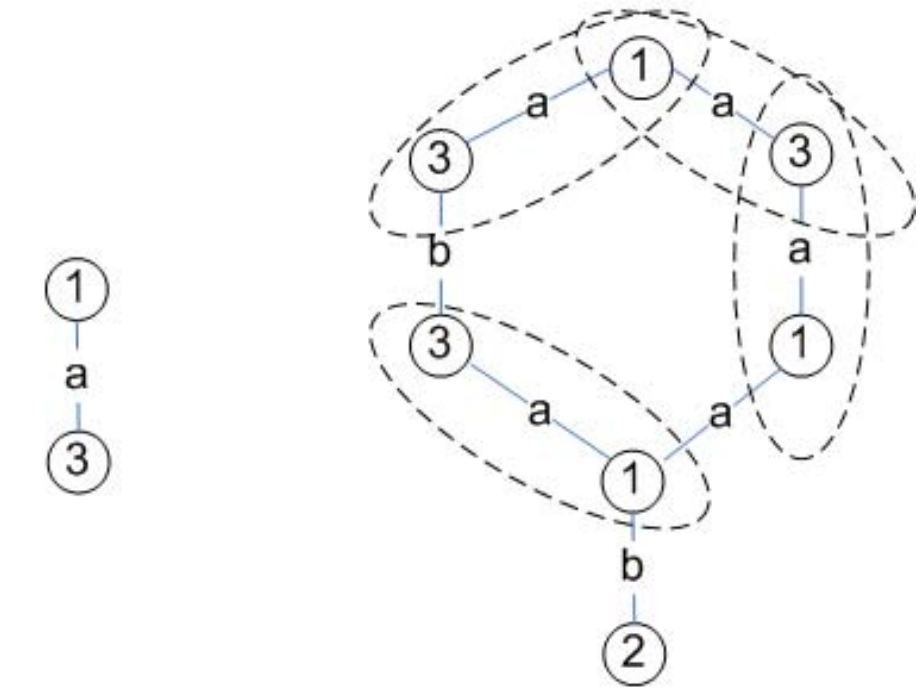






Defining aFG

Counting of edges: $|E_g(e)| = 4$



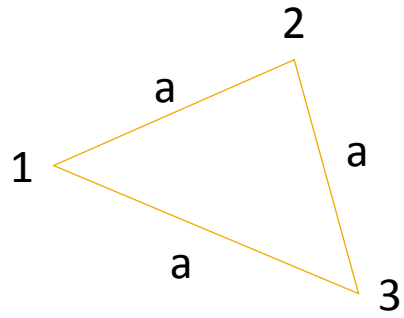
Defining aFG

A graph **g** is **β edge γ frequent** if:

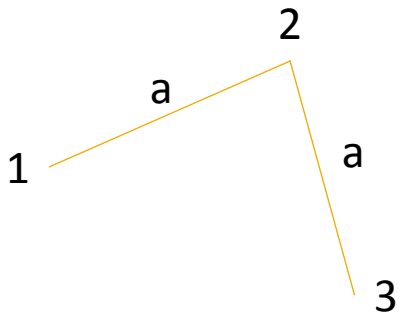
1. $|\text{Support}(D, g, \beta)| \geq \gamma$
2. Each edge **e** in **g** occurs $|E_g(e)|$ times in the elements of S
   - $S \subseteq \text{Support}(D, g, \beta)$
   - $|S| \geq \gamma - \alpha$

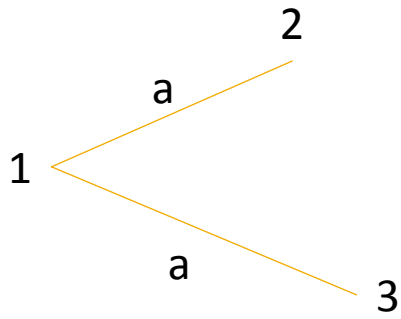
Why is the Tolerance(α) necessary?

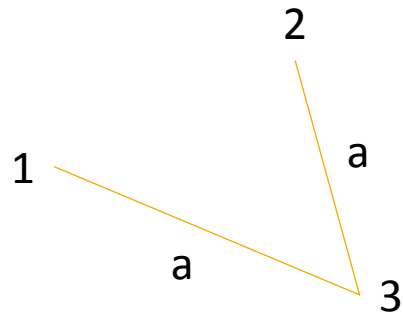
Given β= 1, γ=3, without considering α :



In database graph 1, 2 ,3, the matches are:







the pattern will be considered as frequent.

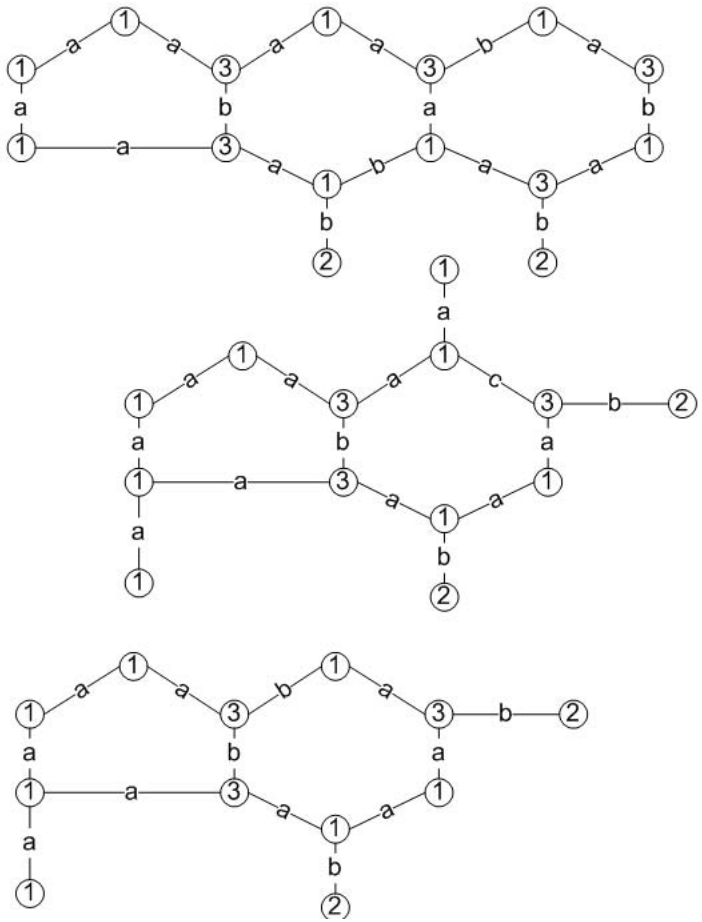
Defining aFG

- Problem defined: find all connected graphs which are β edge γ frequent
- Any questions so far?

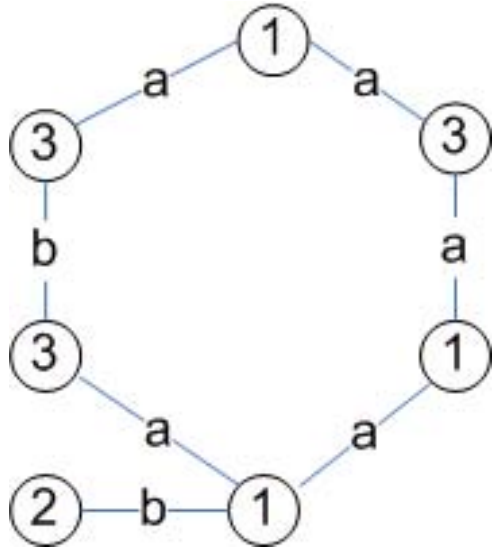
Properties of aFG

- Possible absence of exact match
- Maximal representation - β edge γ frequent – maximal graph is meaningful
- Apriori Property

Possible absence of exact match



Absence of exact match:

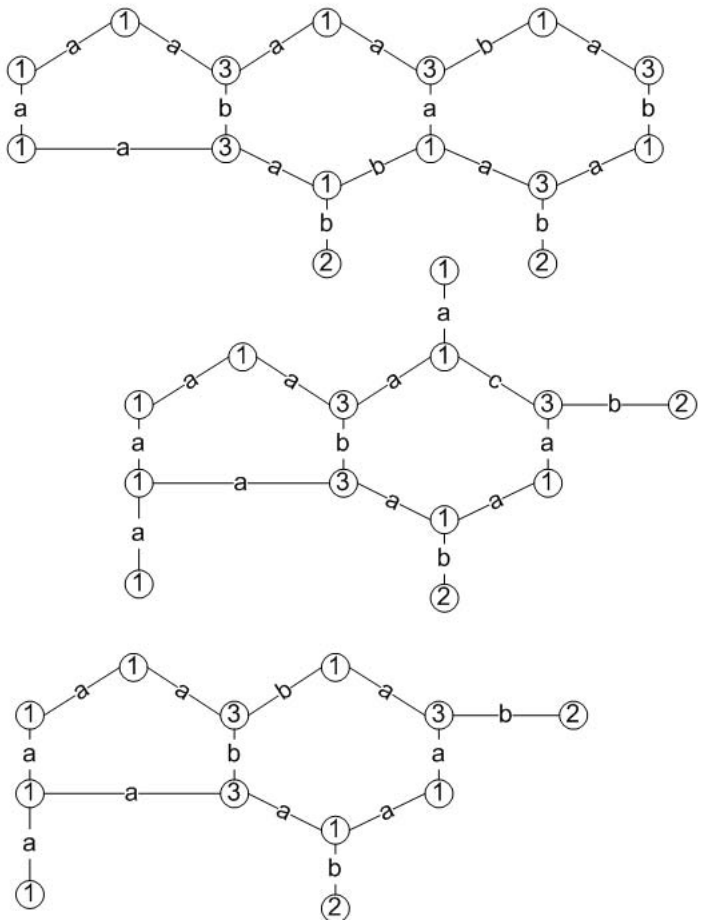


β = 1

α = 2

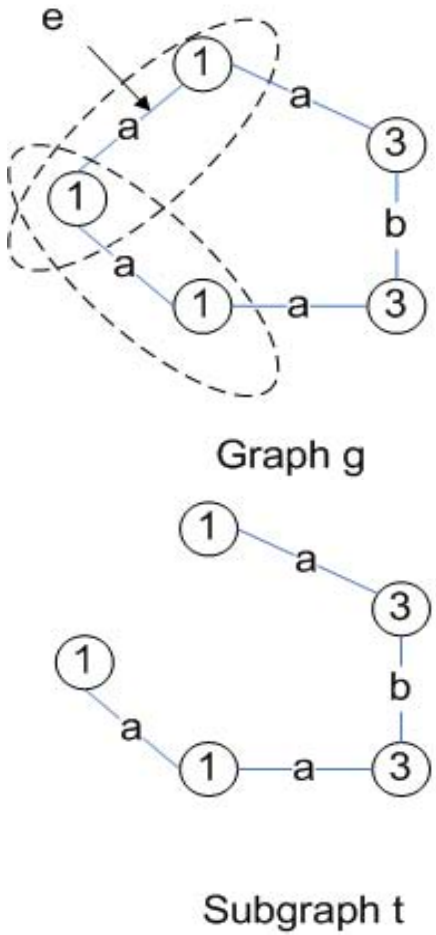
γ = 3

Maximal representation



β = 1
α = 2
γ = 3

Maximal Representation:



Apriori Property

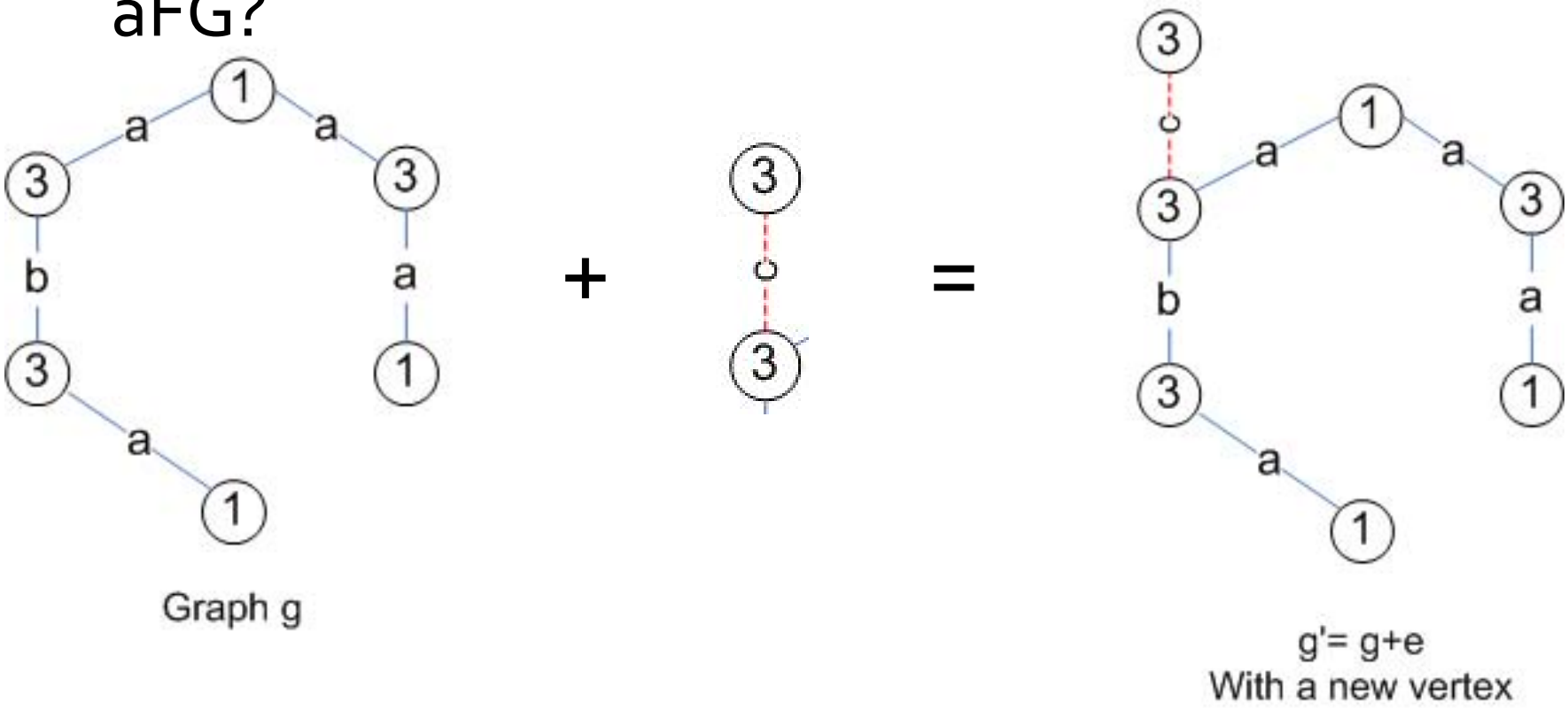
- For a frequent approximate graph *g*, *any subgraph of g is also an approximate frequent* pattern

The basic algorithm

- Find all approximately frequent edges by enumeration (support $\geq \gamma - \alpha$)
- Find a maximal approximately frequent tree.
- Add edges inside the tree to find a maximal aFGs.
- Use tree canonical forms for frequent trees, and canonical matrix for aFGs.

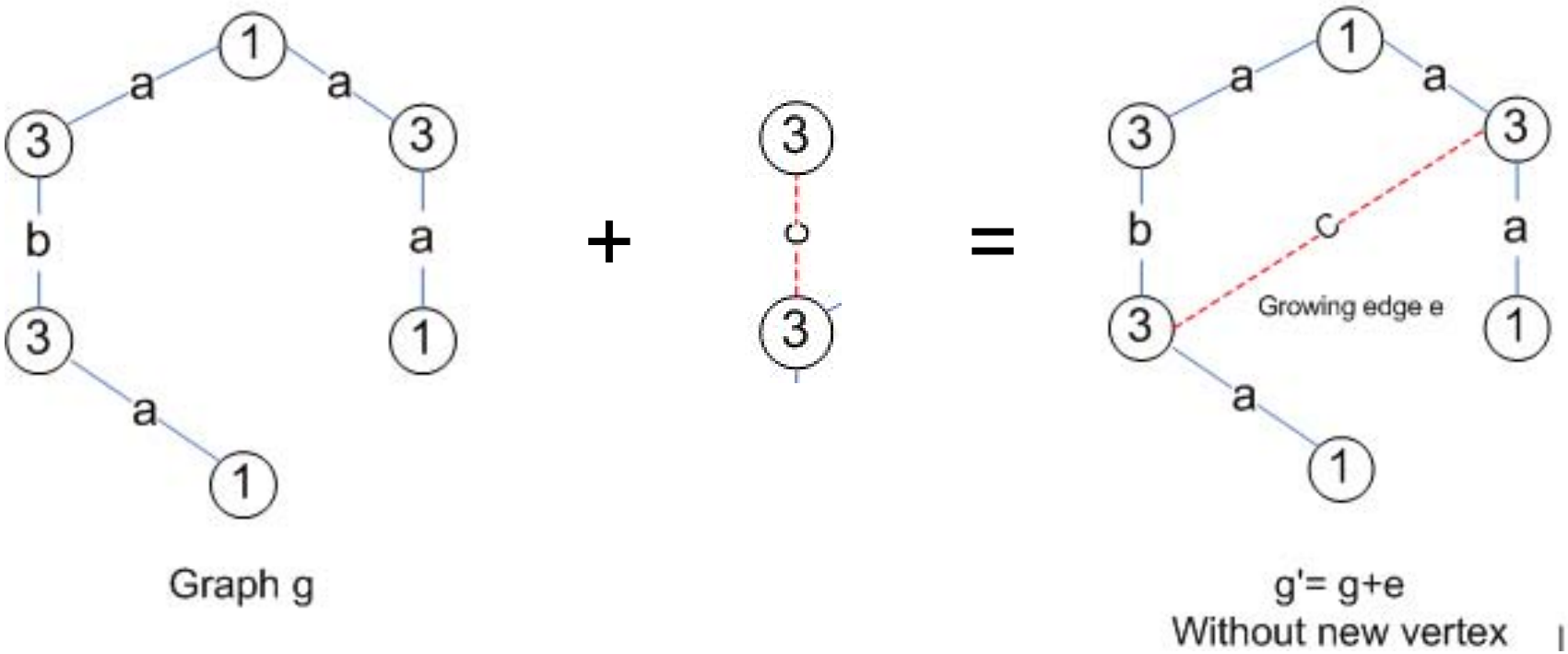
Finding Candidate Trees

- Combine edges together to "grow" tree
- How to find the new tree ($g + e_{uv}$ by u) is still a aFG?



Finding Candidate Graphs

- Add edge w/o vertex into maximal tree
- How to find the new tree ($g+e_{uv}$) is still an aFG?



Drawbacks of the Basic Algorithm

- In approximate graph mining, taking edge relaxation into account, the average size of maximal frequent approximate pattern grows.
- Consequently, the number of non-maximal frequent approximate graphs is even larger.
- The calculation of graph canonical forms is time consuming.

Hashing Graphs

- Instead of using canonical forms, we use a hash value to distinct FAGs.
- Polynomial time calculation, O(1) lookup
- RAM uses 6 functions, for a 6-D hash table

Hash Functions

- Requirements:
  - Isomorphic graphs must have the same feature set
  - Minimize the number of patterns which may share the same feature values.
  - The feature can be calculated in polynomial time.

Hash Functions (cont'd)

- Use different properties for each one:
  - Sums and Products, modulo prime number
  - Edge and vertex counts, minimum spanning trees, shortest paths between nodes, degrees.

- What happens if two graphs share the same hash vectors?

RAM: Randomized Mining

- RAM does lose patterns.
- Minimize losses with multiple runs
- Confidence: 80% in 1 run$\rightarrow$ 99% with 3 runs
  - $1 - p^q$ with p confidence and q runs

Results:Metabolic dataset

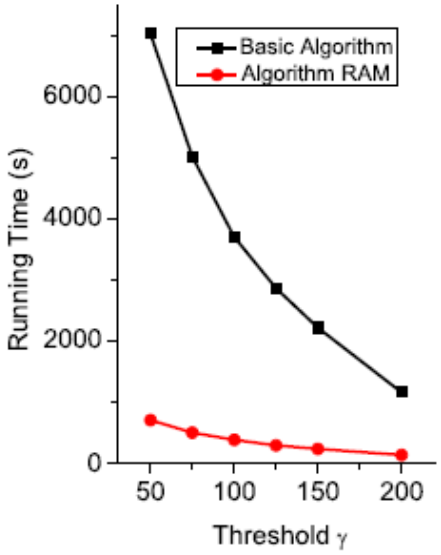
- Efficiency and effectiveness of RAM compared with basic algorithm.

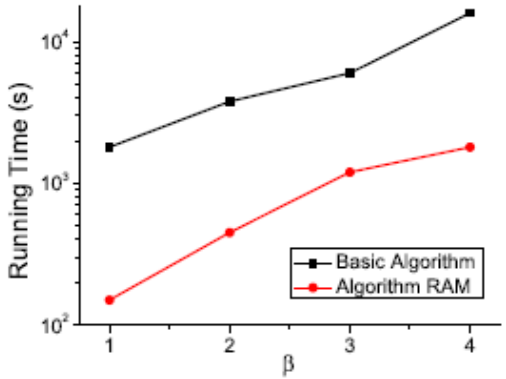
(γ, β, α)	Exe.Time of Algorithm RAM	Basic Algorithm
(25,1,2)	9 s	25 s
(25,2,2)	54 s	126 s
(35,1,2)	8 s	17 s
(35,2,2)	45 s	104 s
(γ, β, α)	No. of Max. Patterns of RAM	Basic Algorithm
(25,1,2)	123	123
(25,2,2)	461	467
(35,1,2)	107	107
(35,2,2)	431	431

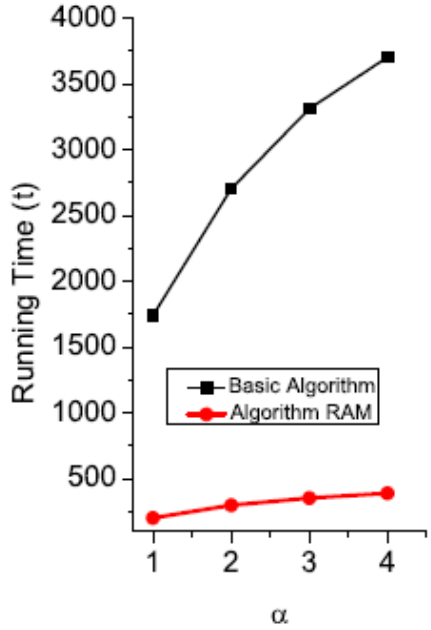
- β = 2, α = 1, γ = 25, RAM found 123 maximal patterns, and the largest approximate pattern contained 14 edges and 14 vertices. The exact graph mining method only found 24 maximal patterns, and the average edge and vertex count of the exact patterns were 10 and 9, respectively.

Results: synthetic dataset

- Efficiency and effectiveness of RAM compared with basic algorithm.

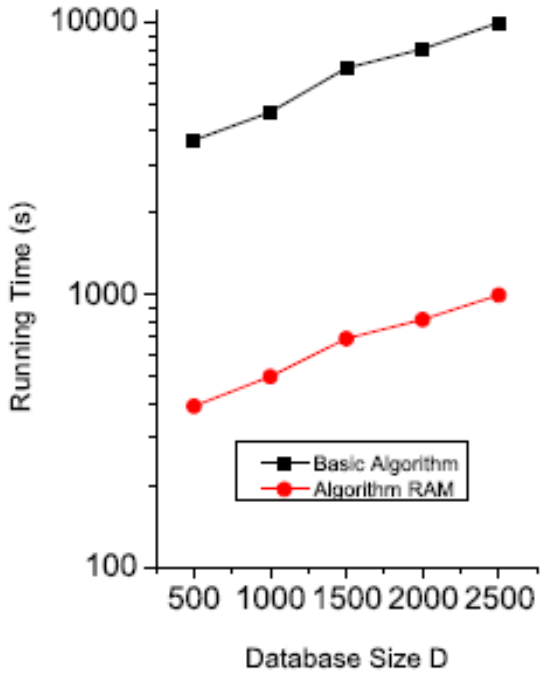


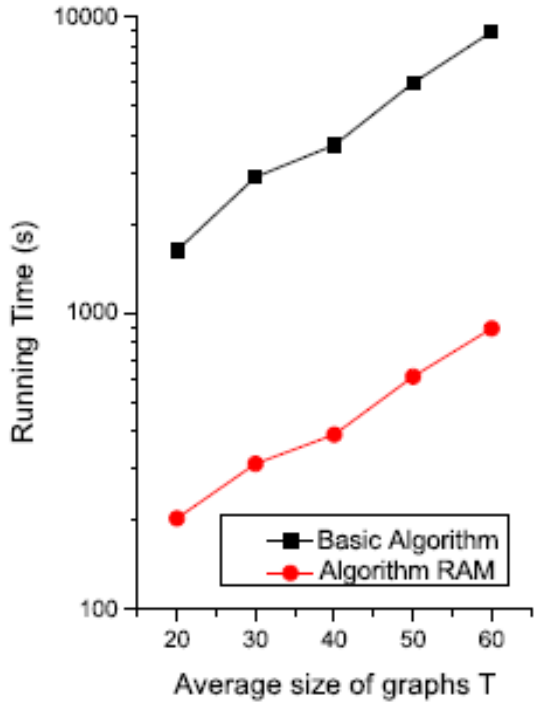




Results: synthetic dataset

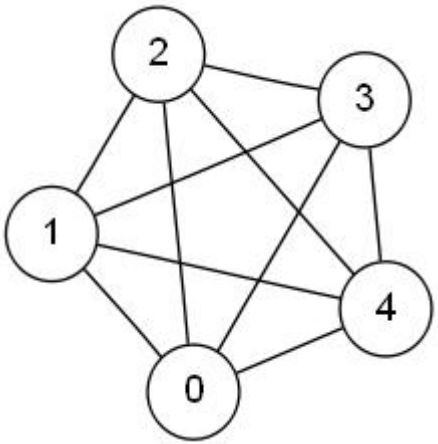
- Efficiency and effectiveness of RAM compared with the basic algorithm.



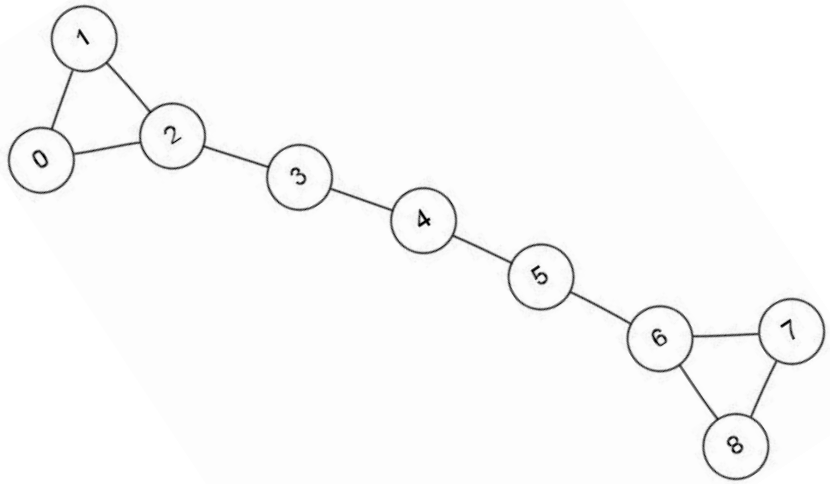


Challenges

- Complexity of β-edge subgraph isomorphism testing
- Possibility of missing less connected aFGs:



vs.



Questions?

- ✓ Overview of approximate graph mining
- ✓ Defining frequent approximate subgraphs (aFG)
- ✓ Elements of the mining algorithm
- ✓ Challenges

- ? Questions?

Overview of Presentation

- ✓ Overview of approximate graph mining
- ✓ Defining frequent approximate subgraphs (aFG)
- ✓ Elements of the mining algorithm
- Challenges

Overview of Presentation

- Overview of approximate graph mining
- Defining frequent approximate subgraphs (aFG)
- Elements of the mining algorithm
- Challenges

Finding Candidate Trees

- We are to find:

- Variability :
  whether there are no less than γ database graphs which contain at least one embedding of g and a vertex v′ with label L(v). If each embedding in a database graph has exactly β edge difference from g, and none of them are connected to v′ with label $L(e_{uv})$, this database graph is not effective.

- Tolerance:
  whether there are no less than γ-α database graphs which contain at least one embedding of g and a vertex v′ with label L(v). If none of the embeddings are connected to v' with label $L(e_{uv})$, and exactly contain all the edges of the same type of e_{uv} , this database graph is not effective.

Finding Candidate Graphs

- We are to find:

- Variability :
  whether there are no less than γ database graphs which contain at least one embedding of g with corresponding vertices u′ and v′. If each embedding in a database graph has exactly β edge difference from g, and u′ and v′ are not connected with label $L(e_{uv})$, this database graph is not effective.

- Tolerance:
  whether there are no less than γ-α database graphs which contain at least one embedding of g with corresponding vertices u′ and v′. If in none of the embeddings contain all the edges of the same type of e_{uv} , this database graph is not effective.