Nearly Exact Mining of Frequent Trees in Large Networks

Ashraf M. Kibriya
25th Sep 2012

DTAI
Declarative Languages & Artificial Intelligence
Katholieke Universiteit Leuven

erc

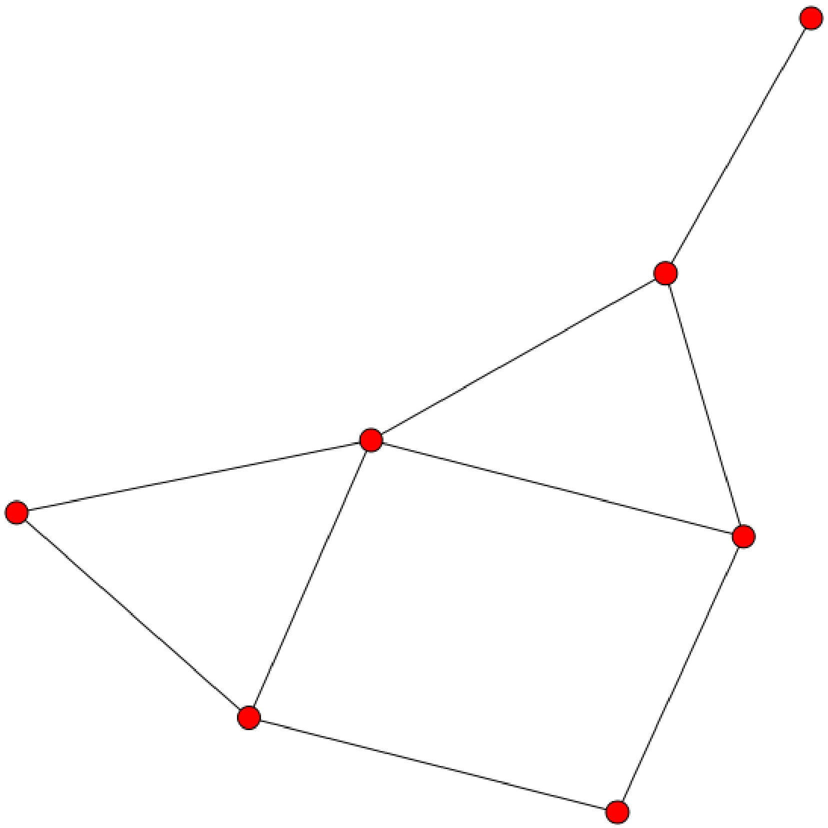
KATHOLIEKE UNIVERSITEIT LEUVEN

Overview …

- Intuition
- Motivation / Existing work
- Main contribution
- Problem definition
- Our Miner
- Experimental results / Conclusion

KATHOLIEKE UNIVERSITEIT LEUVEN

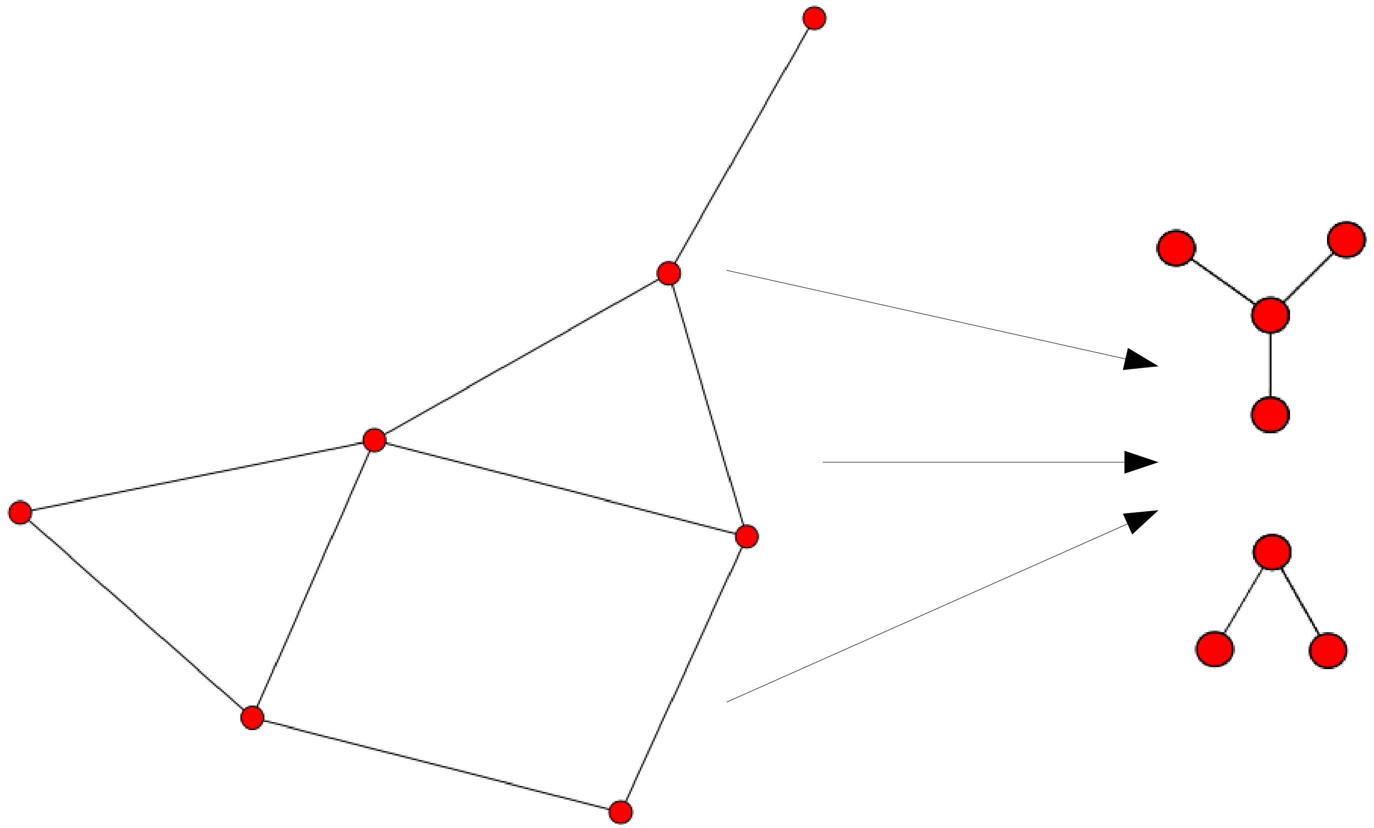
Intuition



- Given a network (sets of nodes and edges)

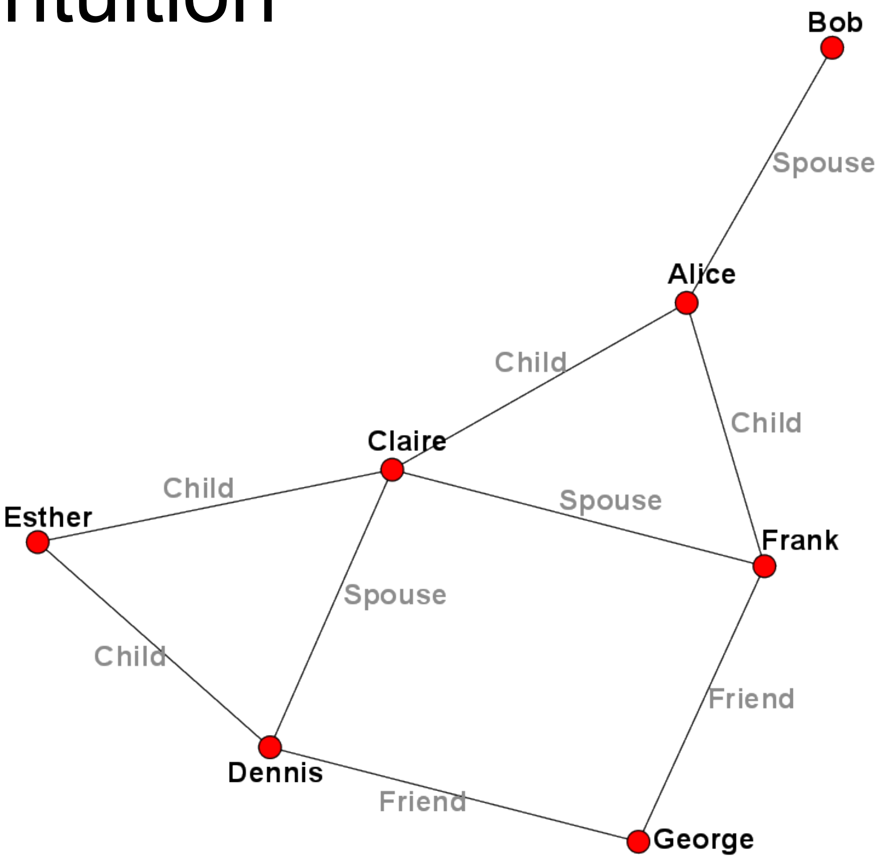
KATHOLIEKE UNIVERSITEIT LEUVEN

Intuition



- Given a network (sets of nodes and edges)
- We want to find frequent substructures

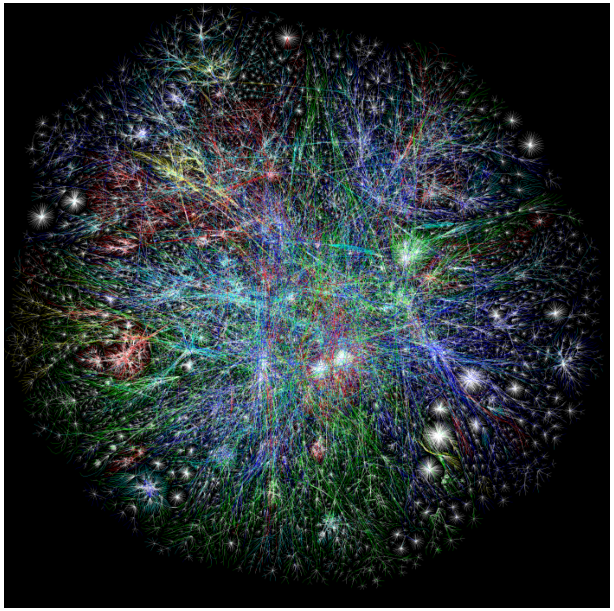
Intuition



- Nodes/edges can be labeled or have other properties attached to them

Motivation

- Large networks are ubiquitous in real-world: computer networks,



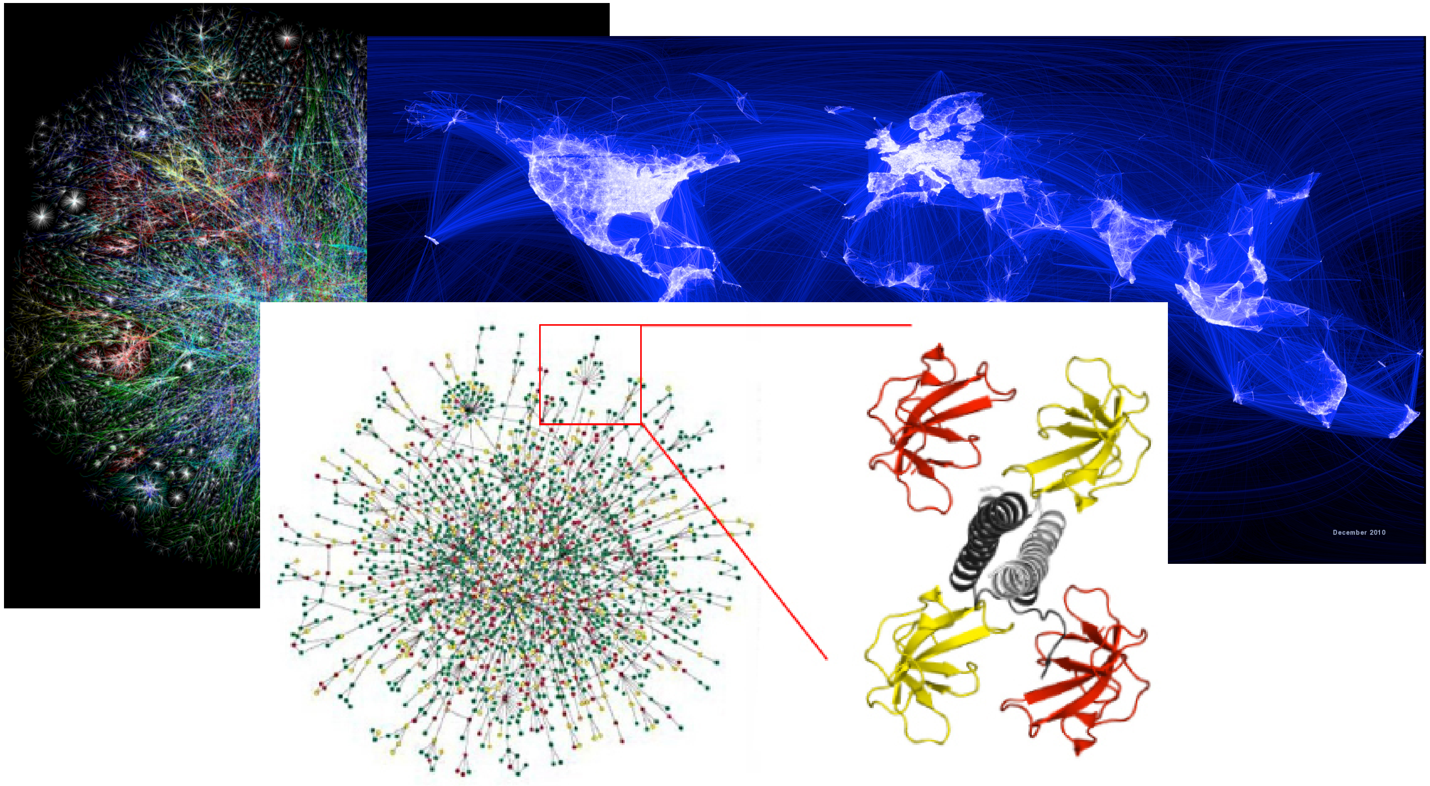
Motivation

- Large networks are ubiquitous in real-world: computer networks, people networks,

KATHOLIEKE UNIVERSITEIT
LEUVEN

Motivation

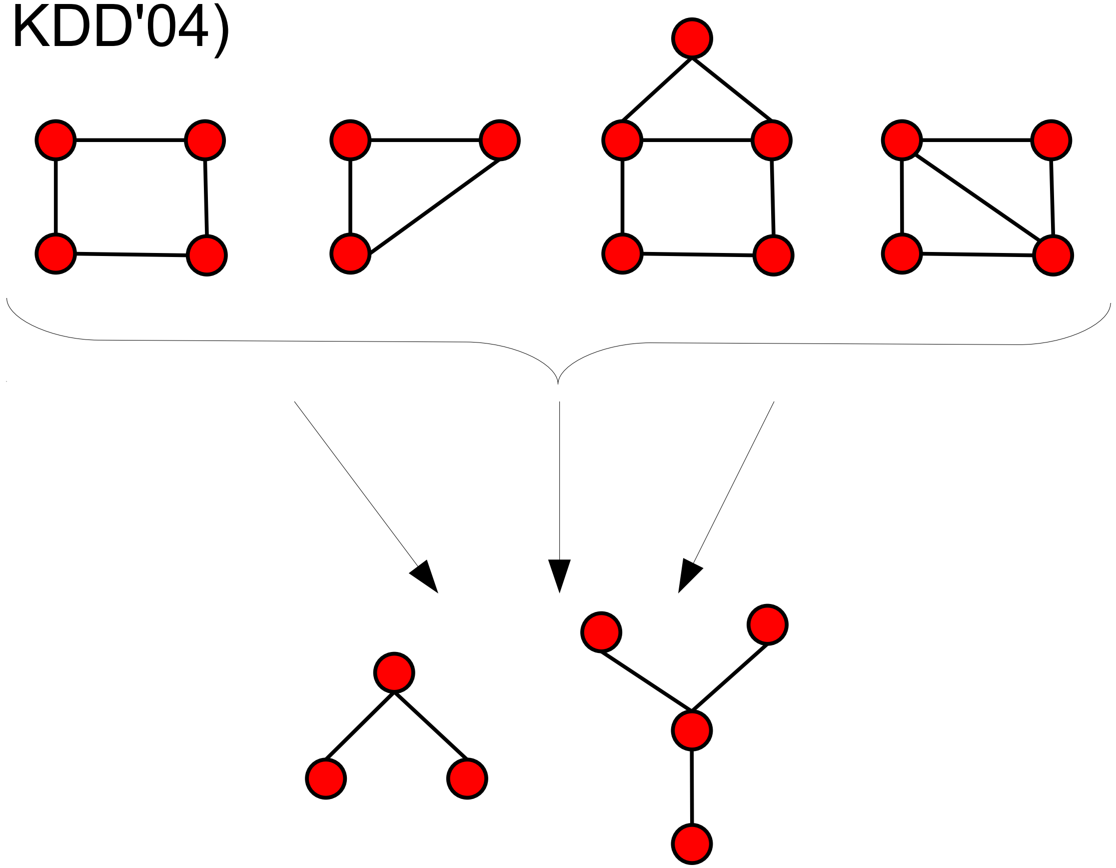
- Large networks are ubiquitous in real-world: computer networks, people networks, natural structures



KATHOLIEKE UNIVERSITEIT LEUVEN

Existing work

- Most graph miners are for transactional setting
  - E.g. gSpan (Yan and Han, ICDM'02), Gaston (Nijssen and Kok, KDD'04)



Existing work

- Most graph miners for transactional setting
  - E.g. gSpan (Yan and Han, ICDM'02), Gaston (Nijssen and Kok, KDD'04)

- Typically graphs are not large: 10-200 nodes
- Trivial modifications to these miners, to mine in single graphs, renders them computationally infeasible
- For *single large* graphs miner only with homomorphism as matching operator, not with isomorphism

KATHOLIEKE UNIVERSITEIT
LEUVEN

Main contribution

- Novel miner for mining labeled rooted trees in large networks under subgraph isomorphism
  - Main focus on tractability since subgraph isomorphism matching is NP-complete
  - Realised theoretical results of Koutis&Williams' [ICALP'09]
- Show practical applicability of our miner by evaluating on synthetic and real-world data
  - Compare with state-of-the-art matching strategy employed in transactional miner

KATHOLIEKE UNIVERSITEIT LEUVEN

Problem definition

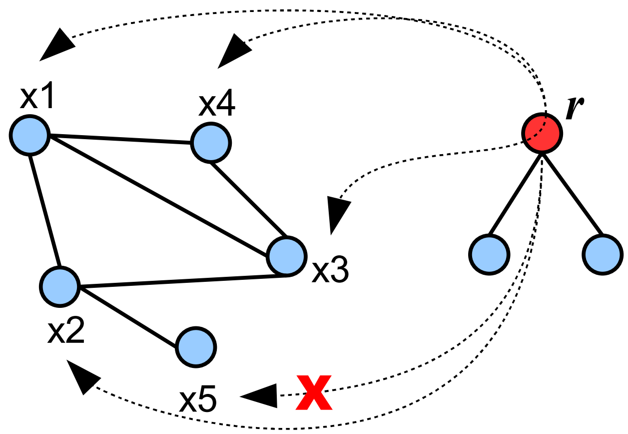
- Let $\boldsymbol{T_r}$ be the set of all rooted Trees
- Given a network (any arbitrary graph) ***G****,* frequency (support) measure f, and a threshold θ find the set of all rooted trees $T \subseteq T_r$ such that:

$$T = \{t \in T_r \mid f(G, t) \geq \theta\}$$

- Finding good frequency measure challenging! Other work only on this problem exits, but not focus of this research
  - Can't count all isomorphic embeddings
    - Not anti-monotone w.r.t pattern size, and #P-hard

Frequency Measure

- Count number of ***root images*** **-** number of vertices x_i to which the root can be mapped under isomorphism
- Gives frequency at most ***n***
- Anti-monotone w.r.t increasing pattern size



frequency: 4

KATHOLIEKE UNIVERSITEIT LEUVEN

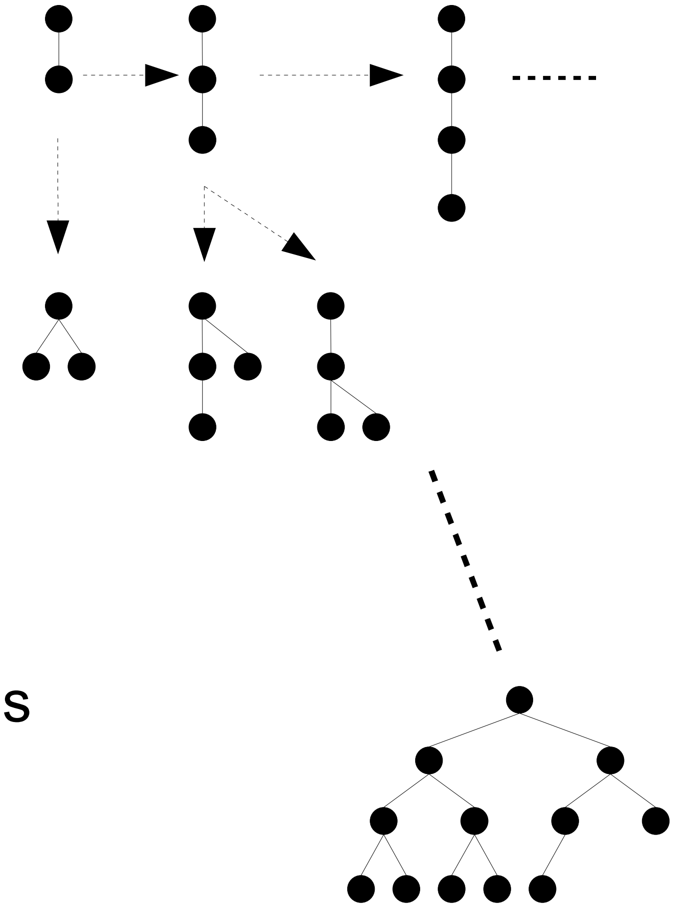
Overview …

- Intuition to what we're trying to achieve
- Motivation / Existing work
- Main contribution
- Problem definition
- ***Our Miner***
  - ***Background***
    - ***Candidate Generation***
    - ***Subgraph Isomorphism - Koutis&Williams***
  - ***Optimizations/Implementation Details***
- Experimental results / Conclusion

KATHOLIEKE UNIVERSITEIT LEUVEN

1. Candidate generation

- Same technique as Nijssen and Kok [KDD'04] and Nakano and Uno [IPL'02]
  - Rightmost extension
  - Canonical form
    - pre-order depth sequence ensures trees are left heavy



- Technique ensures each candidate is generated and tested only once

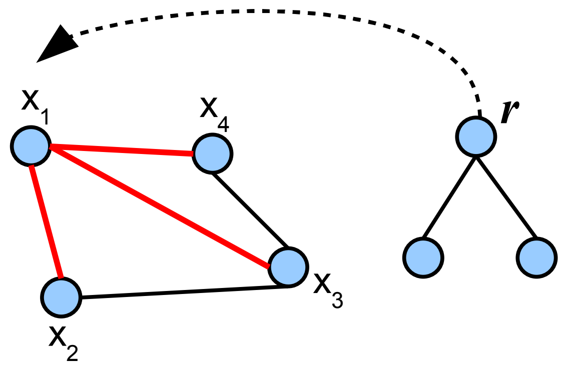
KATHOLIEKE UNIVERSITEIT LEUVEN

2. Evaluating Matching Operator / Frequency Counting

- Use exact/deterministic VF2 [IEEE TPAMI'04] method as baseline in our experiments
  - State-of-the-art for subgraph isomorphism between general/arbitrary graphs
  - Used by some earlier transactional miners
  - In best case O(k), in worst case $O(n^k)$
- Recent theoretical results by Koutis&Williams [ICALP'09] on subgraph isomorphism of rooted trees in graphs
  - Current state-of-the-art for rooted trees (2^k exponential factor)
  - Our method of choice

KATHOLIEKE UNIVERSITEIT LEUVEN

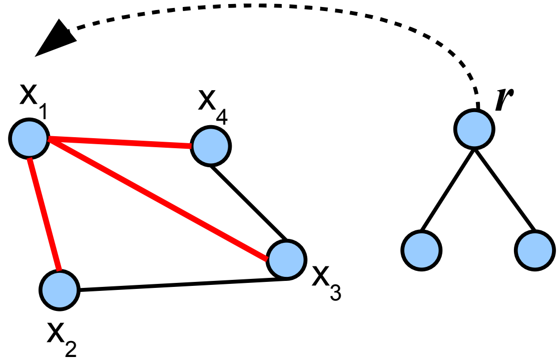
2. Subgraph Isomorphism – Koutis&Williams



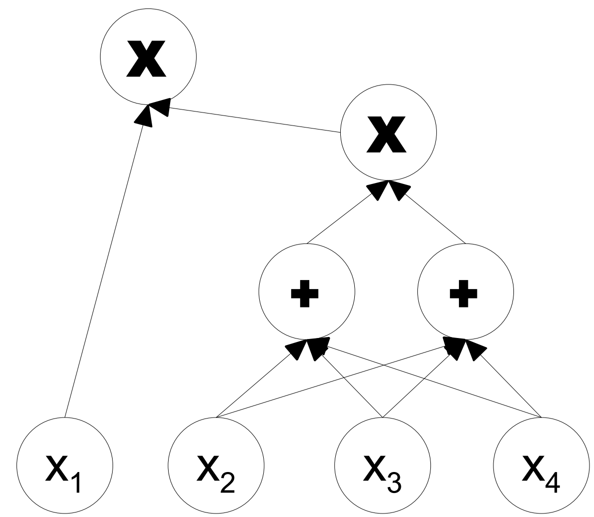
$$x_1 x_2^2 + x_1 x_2 x_3 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_2 x_4 + x_1 x_3^2 + x_1 x_3 x_4 + x_1 x_3 x_4 + x_1 x_4^2$$

- Based on enumeration of all homomorphisms of a (rooted) tree in a network

2. Subgraph Isomorphism – Koutis&Williams



$$x_1 x_2^2 + x_1 x_2 x_3 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_2 x_4 + x_1 x_3^2 + x_1 x_3 x_4 + x_1 x_3 x_4 + x_1 x_4^2$$

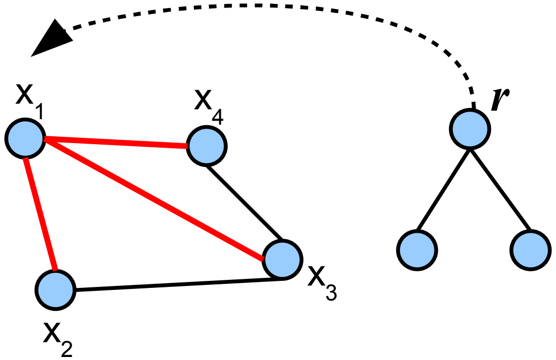


$$x_1 \{ (x_2 + x_3 + x_4)(x_2 + x_3 + x_4) \}$$

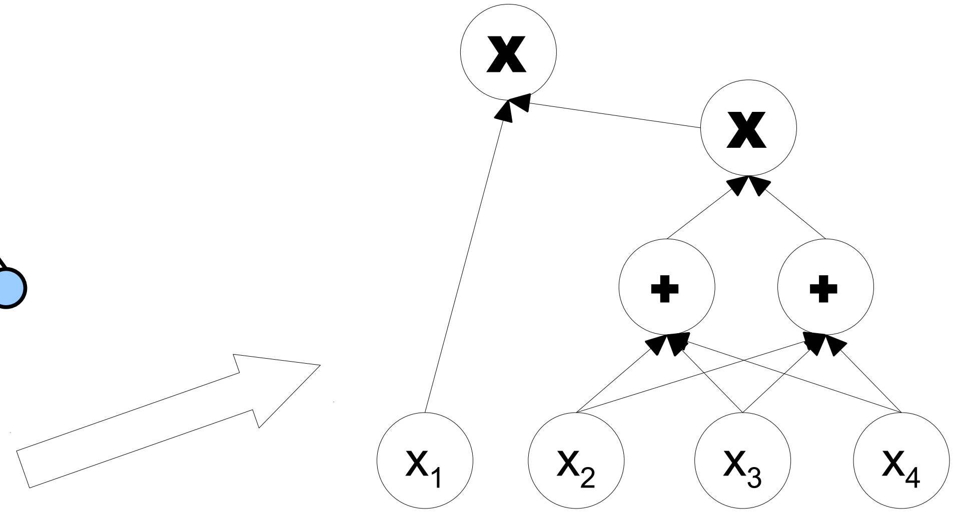
- Encodes $O(n^k)$ size polynomial as an arithmetic circuit of size O(km)
  - For patterns of size k, and network with n nodes, m edges

KATHOLIEKE UNIVERSITEIT LEUVEN

2. Subgraph Isomorphism – Koutis&Williams



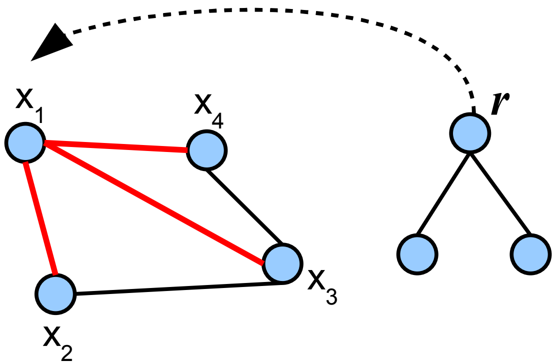
$$x_1 x_2^2 + x_1 x_2 x_3 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_2 x_4 + x_1 x_3^2 + x_1 x_3 x_4 + x_1 x_3 x_4 + x_1 x_4^2$$



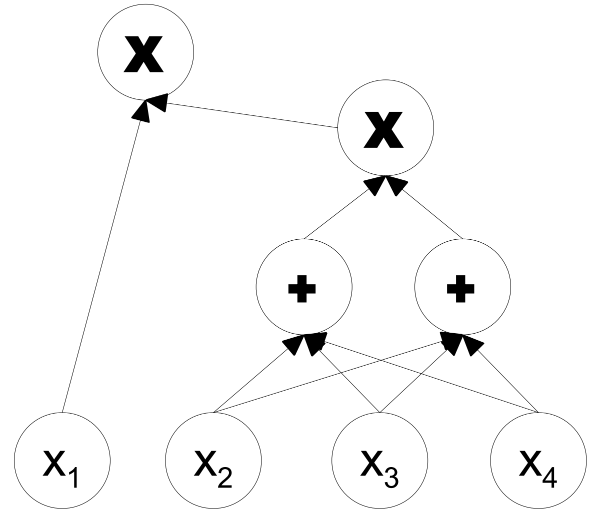
$$x_1 \{ (x_2 + x_3 + x_4)(x_2 + x_3 + x_4) \}$$

- Next step is to evaluate this circuit
- Evaluate on random elements from subset of $GF(2^{(3+\log k)})[\mathbb{Z}_2^k]$

2. Subgraph Isomorphism – Koutis&Williams



$$x_1 x_2^2 + x_1 x_2 x_3 + x_1 x_2 x_3 + x_1 x_2 x_4 + x_1 x_2 x_4 + x_1 x_3^2 + x_1 x_3 x_4 + x_1 x_3 x_4 + x_1 x_4^2$$



$$x_1\{(x_2 + x_3 + x_4)(x_2 + x_3 + x_4)\}$$

- All <u>non-multilinear</u> terms in the circuit evaluate to zero
- Circuit may evalute to <u>non-zero</u>, if there is at least one <u>multilinear term</u> representing an isomorphism

2. Subgraph Isomorphism – Koutis&Williams

- Properties of the method

	Present	*Not Present*
Yes	0.2	**0**
No	0.8	**1**

- Success rate can be boosted to an arbitrary p' by repeating it:

$$\left\lceil \frac{\log(1-p')}{\log(1-0.2)} \right\rceil \text{ times}$$

KATHOLIEKE UNIVERSITEIT LEUVEN

2. Subgraph Isomorphism – Koutis&Williams

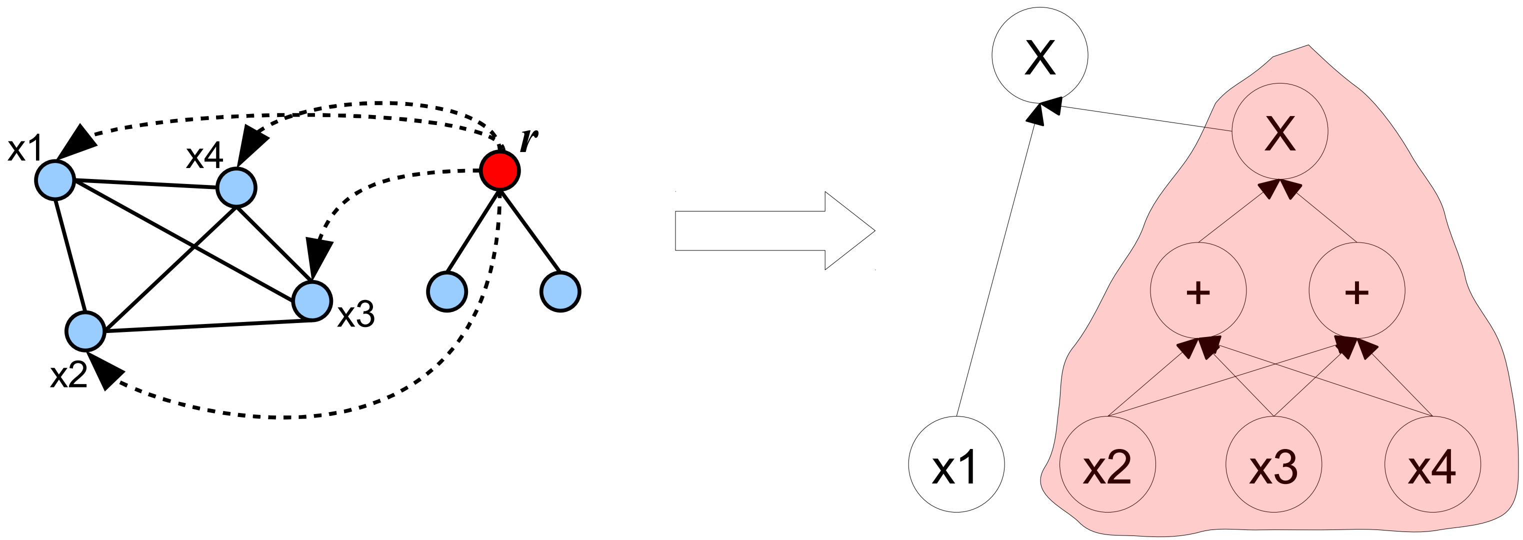
- For e.g.

Repeats	*Success Prob.*
1	0.2
10	0.893
20	0.9885
50	0.999986
100	0.9999999998

- Worst case complexity for counting frequency $O(\log^2(k)k^2m2^k)$
  - Much less than $O(n^k)$ of classical methods such as VF2 or Ullman's

Optimizations

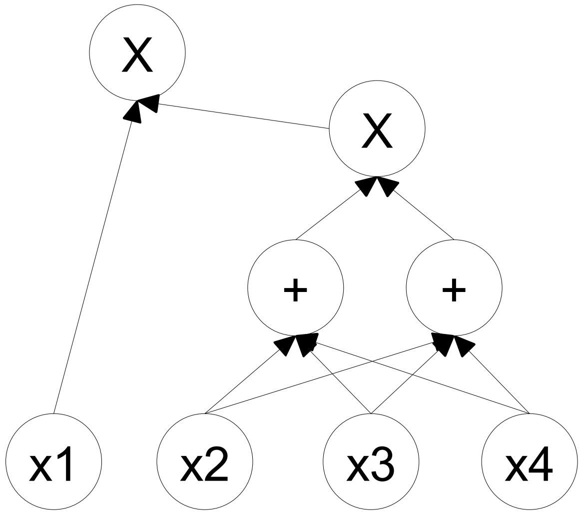
- Build once and use the same circuit
- Can share circuit/results for different root images



$$x1\{(x2+x3+x4)(x2+x3+x4)\}$$
$$x2\cdots$$
$$x3\cdots$$
$$x4\cdots$$

Optimizations

- Can perform homomorphic test first, and cache its results
  - Takes O(n+km) instead of O(n+log^2(k)k^{2}m2^k)
    - Can be done over $\mathbb{Z}$ instead of $GF\left(2^l\right)\left[\mathbb{Z}_2^k\right]$

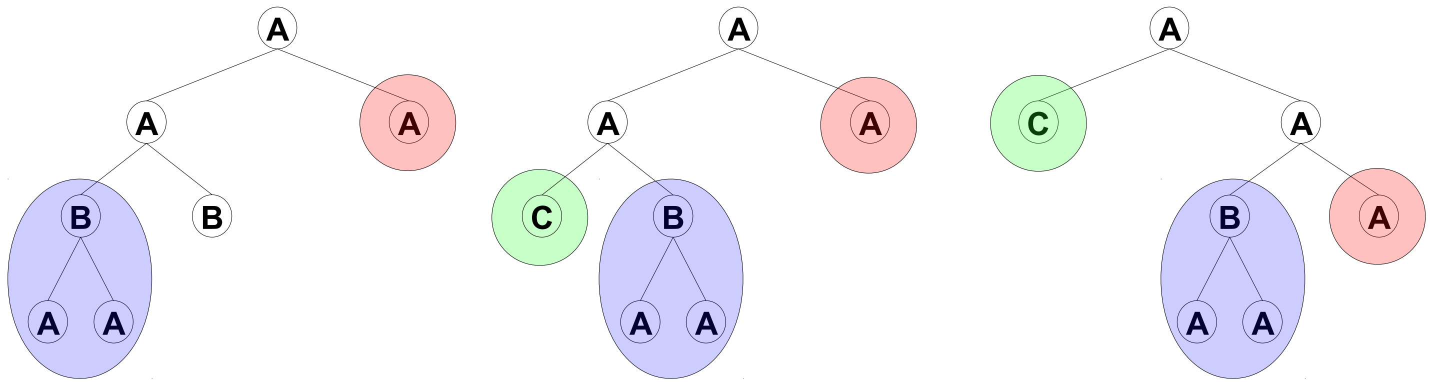


$$x1\{(x2+x3+x4)(x2+x3+x4)\}$$

KATHOLIEKE UNIVERSITEIT LEUVEN

Optimizations

- Can share circuit/results for different candidate trees!
  - Decompose upto **batch-size** candidate trees into subtree
  - Build/Evaluate a circuit for each unique common subtree once



KATHOLIEKE UNIVERSITEIT LEUVEN

Overview …

- Intuition
- Motivation / Existing work
- Main contribution
- Formal problem definition
- Our Miner
  - Background
    - Candidate Generation
    - Subgraph Isomorphism - Koutis&Williams
  - Optimizations/Implementation Details
- ***Experimental results / Conclusion***

KATHOLIEKE UNIVERSITEIT LEUVEN

Experiments - Data

- Synthetic data
  - Power-law graphs (similar to Barabási-Albert [Science'99]), with degree distribution $P(d_i) \propto d_i^{-4}$
- Real data
  - Facebook friendship data (uniform & random walk sampling)
  - Dblp citation network
  - IMDB movie-actor network

Experiments - Setup

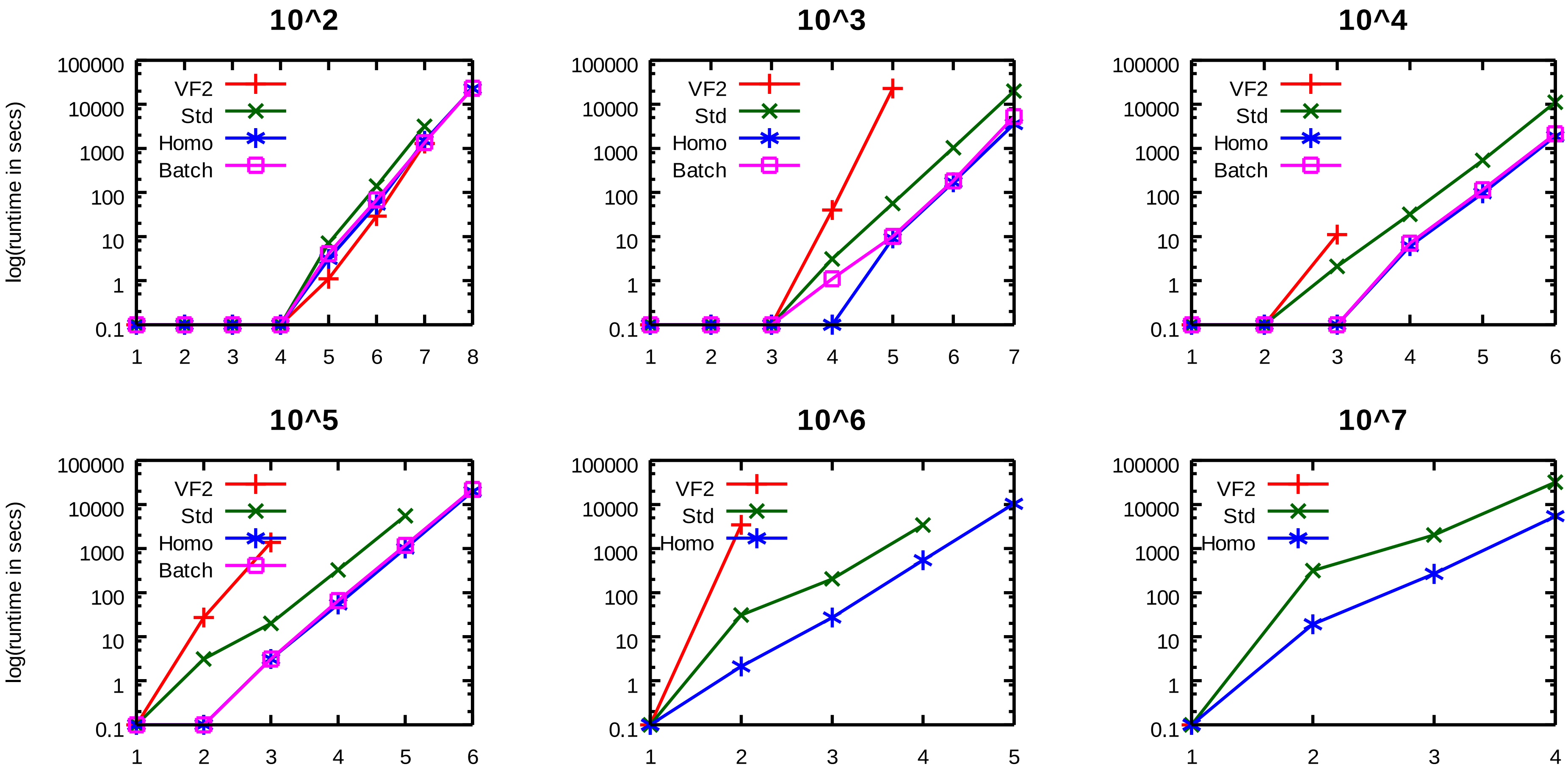
- Randomized method was repeated to get $(1\text{-}10^{-6})$ accuracy
- Performed two sets of experiments:
  - Measure runtime w.r.t to increasing pattern size
    - Used a high enough frequency threshold to find as large trees as possible
  - Used sampling to measure time per pattern
    - To evaluate asymptotic behaviour of Koutis&Williams

KATHOLIEKE UNIVERSITEIT LEUVEN

Experiments - Questions

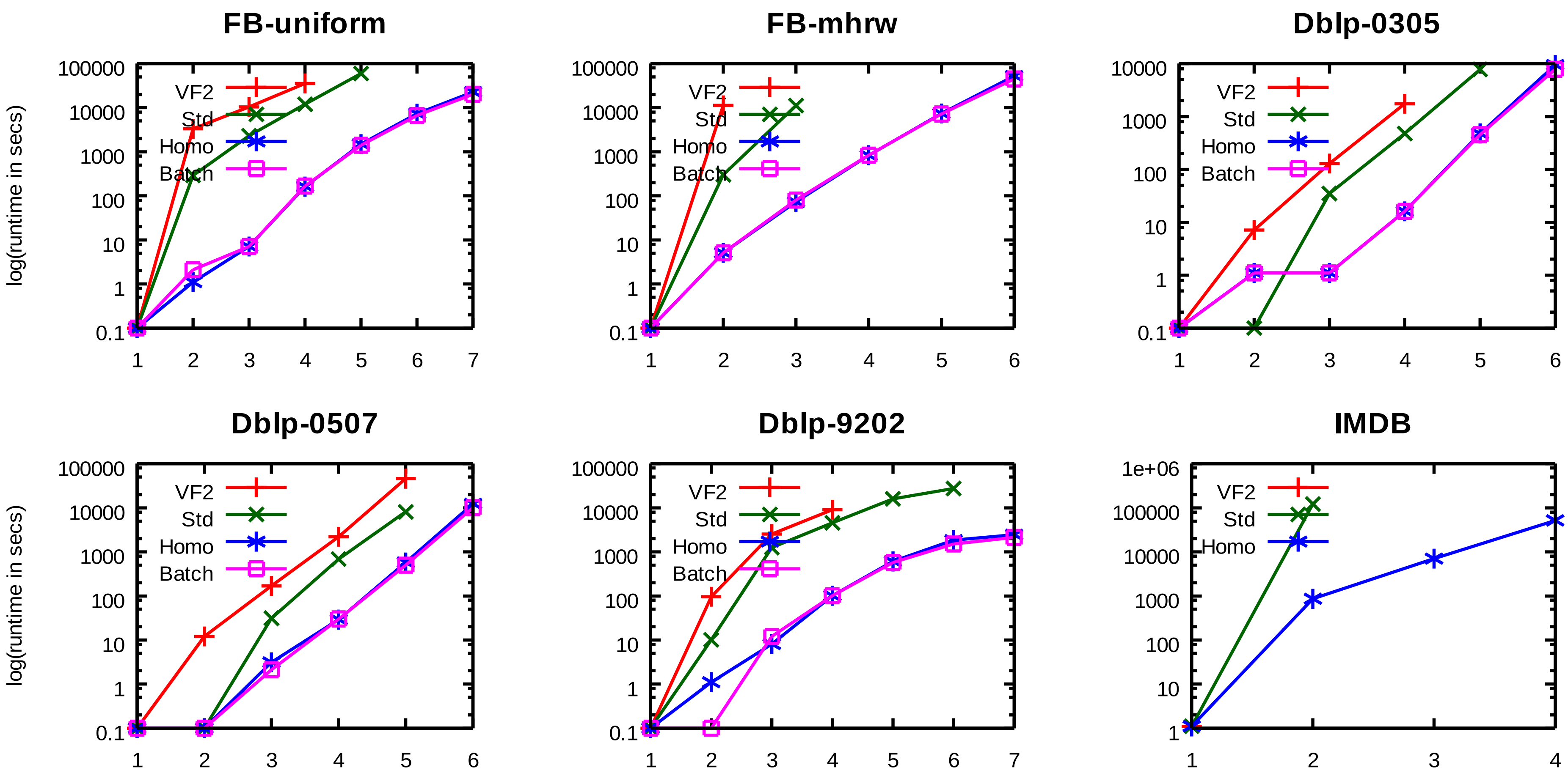
1) What pattern and network sizes can mined in reasonable time?
2) How does our pattern matching strategy compare to state of the art, i.e. VF2 (used by transactional miners)?
3) Does our miner scale as well as the theoretical bounds of Koutis-William's?
4) What is the influence of pattern mining parameters and optimizations?

Results – Synthetic

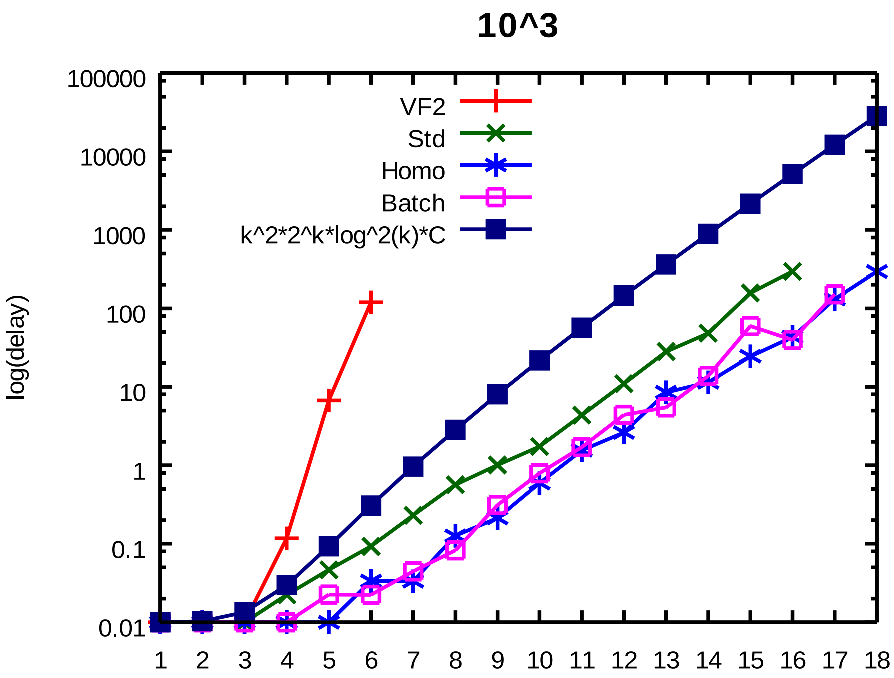


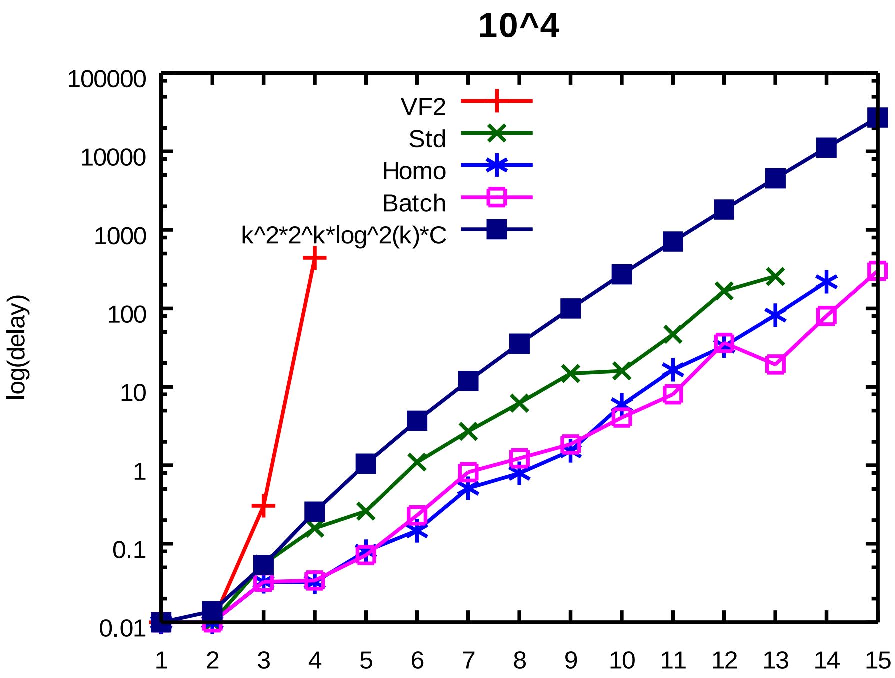
KATHOLIEKE UNIVERSITEIT
LEUVEN

Results – Real data

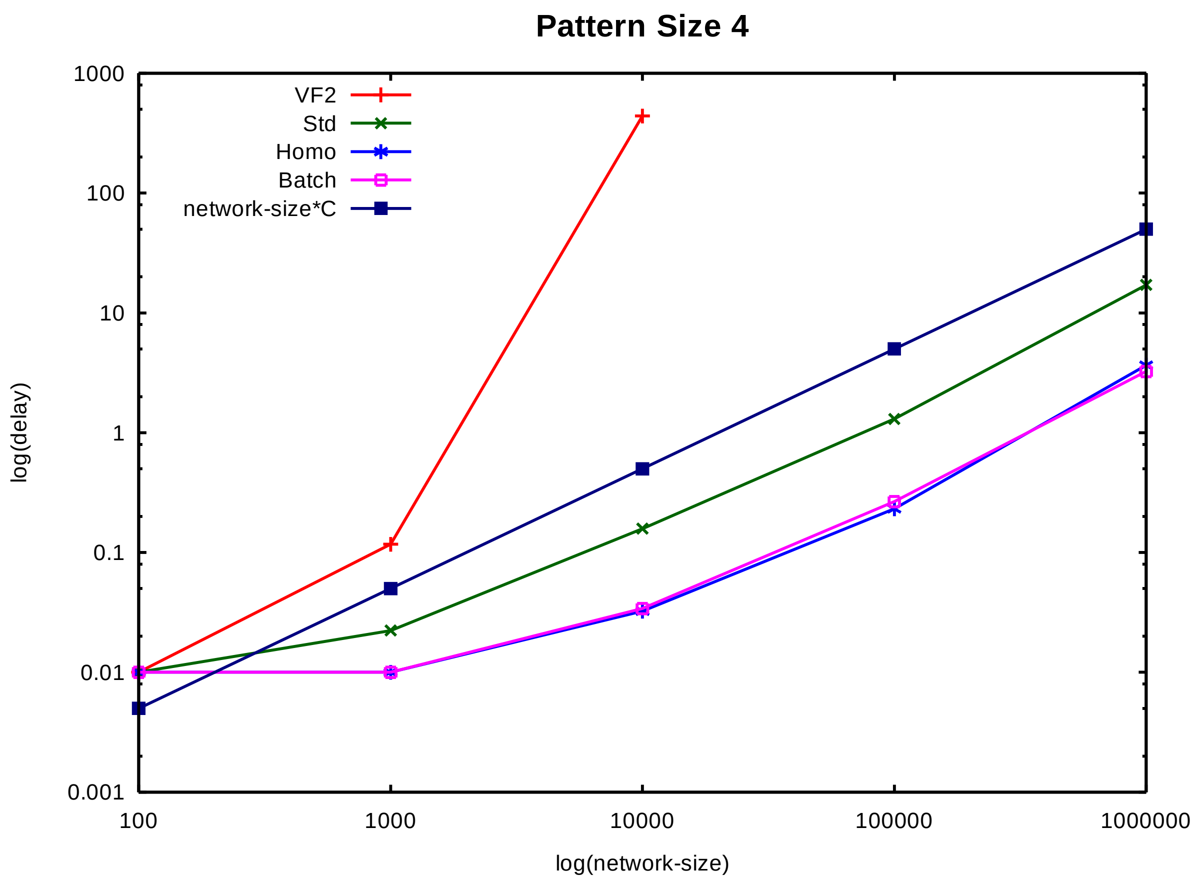


Results – Asymptotic





Results – Asymptotic (2)



Result Summary

1) What pattern and network sizes can mined in reasonable time?
   - *Main bottleneck is exponential growth of freq. patterns, otherwise we can mine upto size 15 in reasonable time*
2) Our pattern matching strategy compared to state-of-the-art?
   - *Ours is clearly orders of magnitude better*
3) Does our miner scale as well as the theoretical bounds?
   - *It does appear to, also as opposed to VF2, it is linear in network size*
4) What is influence of parameters and optimizations?
   - *Homomorphism test adds significant improvement, especially on real-world data, batch optimization adds little significant advantage*

Conclusion & future work

- We have presented a novel pattern miner
  - It works reasonably well in practice, without any restriction to pattern (size / degree etc), or to the network
  - It is ***linear*** in network size
  - Allows for nearly exact mining of frequent trees
  - Orders of magnitude better than existing state-of-the-art
- Further heuristics/optimizations may improve performance on real-world datasets
- Would like to extend the method to other graph classes (other than trees)

Thank you for your attention

The End

4. Subgraph Isomorphism – Koutis&Willams

- Squares (or higher degree terms) in the polynomial evaluate to:

$$
\begin{aligned}
&(W_0 + v_i)^2 \\
&= W_0^2 + 2W_0 v_i + v_i^2 \\
&= W_0 + 2v_i + W_0 \\
&= 2W_0 + 2v_i
\end{aligned}
$$

E.g.

$$\left(\begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \right)^2$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} = 2 \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} + 2 \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}$$

4. Subgraph Isomorphism – Koutis&Willams

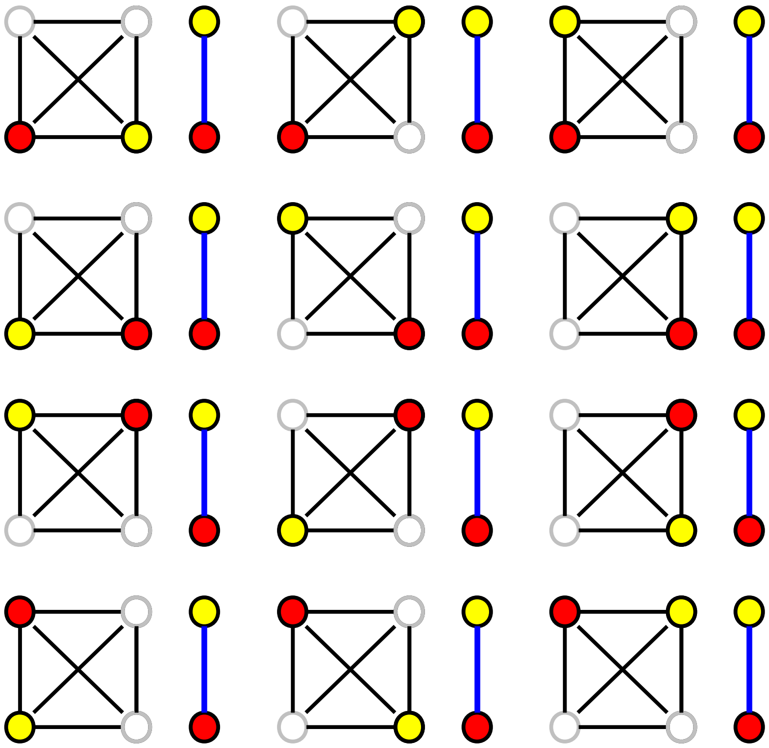
- Now combine group $\mathbb{Z}_2^k$ with coefficients from $GF(2^l)$ and perform the algebra on $GF(2^l)[\mathbb{Z}_2^k]$

$$
\begin{aligned}
&(W_0 + v_i)^2 \\
&= W_0^2 + 2W_0 v_i + v_i^2 \\
&= W_0 + 2v_i + W_0 \\
&= 2W_0 + 2v_i \\
&= 0 + 0 \quad \text{[in field with characteristic 2, } a_x + a_x = 0, \forall a_x \in F\text{]} \\
&= 0
\end{aligned}
$$

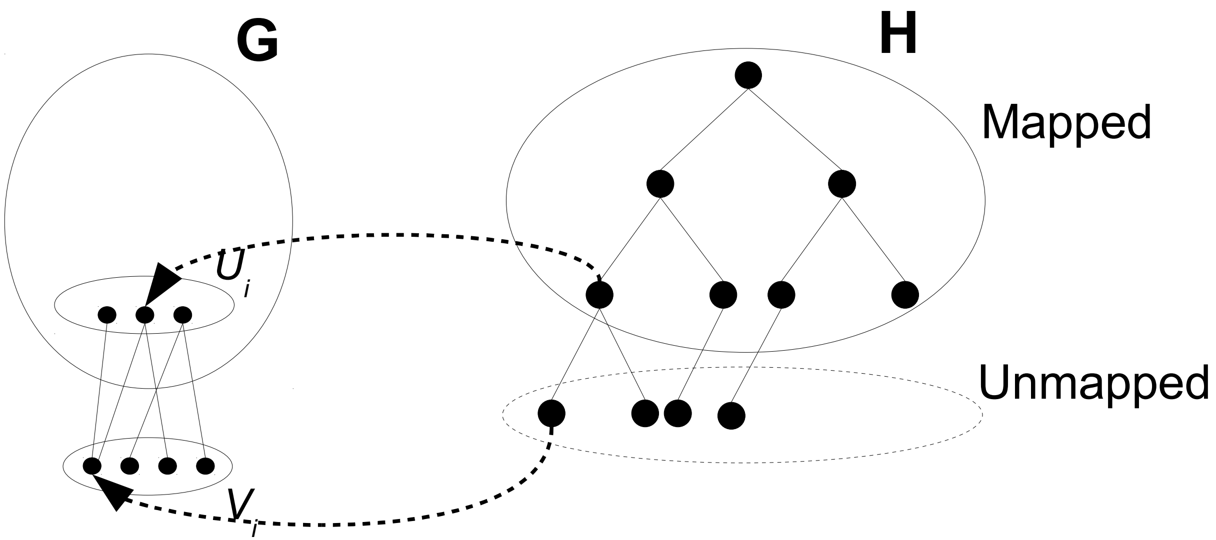
KATHOLIEKE UNIVERSITEIT LEUVEN

3. Isomorphism – VF2 method

- Exhaustive search
  - Considers all possible combinations, with pruning
  - In best case O(k), in worst case $O(n^k)$



3. Isomorphism – VF2 method



- Performs a number of lookahead checks to ensure any new node we map is consistent with previous mappings
- In best case O(k), worst case $O(n^k)$

KATHOLIEKE UNIVERSITEIT LEUVEN

Optimizations to VF2

- Smarter use of core data structures for larger networks
  - On large networks repeated initializations can have a high computational toll

The complete miner

Algorithm 1 PatternMiner

```
function findPatterns(G, k, t):
S_{1...k} := ∅ {array of frequent trees of size 1...k}
C_{1...k} := ∅ {array of candidate trees of size 1...k}
T := ∅ {set of all frequent trees}
for i = 1 to k do
  if i = 1 then
    C_1 := {initialise to just one single vertex graph}
  else
    C_i := generateCandidates(S_{i-1})
  end if
  S_i := {c_j : c_j ∈ C_i ∧ countFreq(c_j) ≥ t}
  T := T ∪ S_i
end for
return T

function generateCandidates(S):
C := ∅
for all t ∈ S do
  for all v ∈ V(t) do
    t' := (V(t) ∪ v', E(t) ∪ (v, v'))
    C := C ∪ {t'}
  end for
end for
return C
```

Koutis & Williams method ...

- We can define all homomorphisms for a tree *T*, with root mapped to x_i, by $\varphi(T, r, x_i \in G)$:
  - *If label(r)=label(x_j)*

    $$return\ \ 0$$
  - *If |V(T)|=1*

    $$return\ \ x_j$$
  - *Else*

    $$T' = T \setminus \{r\}$$

    $$\underbrace{return\ \ x_j \cdot \left(\prod_{(r,r') \in E(T)} \left(\sum_{(j,j') \in G} \varphi(T', r', j') \right) \right)}_{x1\{(x2+x3+x4)(x2+x3+x4)\}}$$