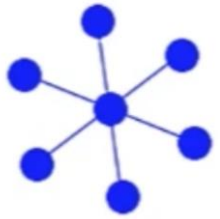
# HOPS: Probabilistic Subtree Mining for Small and Large Graphs

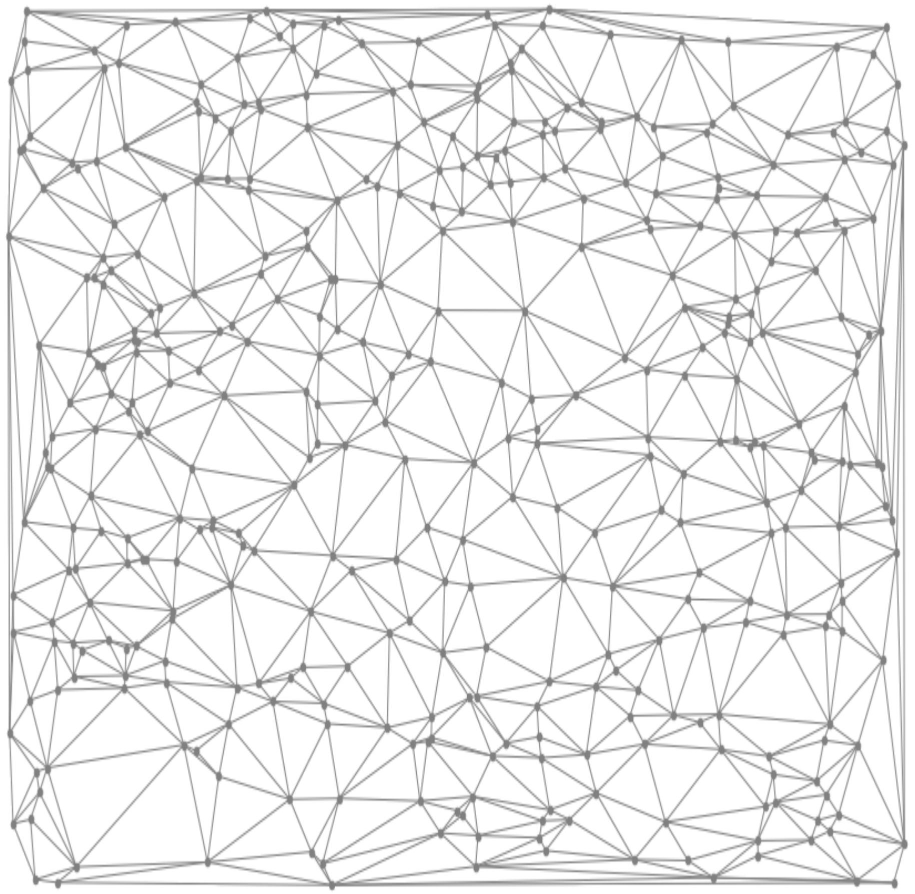
Pascal Welke, Florian Seiffarth, **Michael Kamp**, Stefan Wrobel

## Counting patterns in Large Graphs

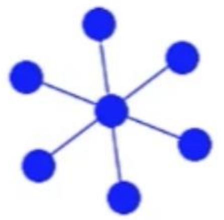
pattern $H$



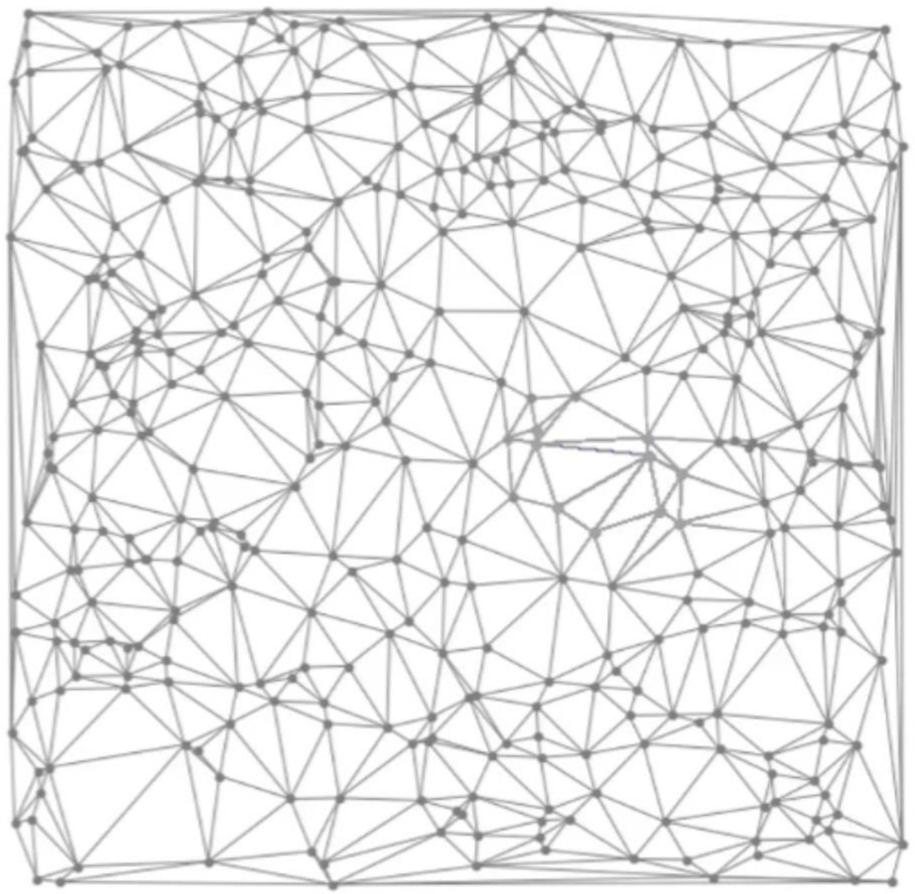
graph $G = (V, E)$



## Counting patterns in Large Graphs



Pattern Count: ?



## Importance Sampling

**Problem:**

- Exact counting is slow
- Only algorithms with exponential runtime known, (e.g. Ullmann, 1976)

**Need estimation algorithms!**

**Idea:**

Importance Sampling to Estimate Pattern Count

- Find embedding and estimate number of possible inclusions

$$\mathbb{E}[Z] = \sum_{i \in C_H(G)} p_i \frac{1}{p_i} = |C_H(G)| \qquad Z = \frac{1}{p_i}$$

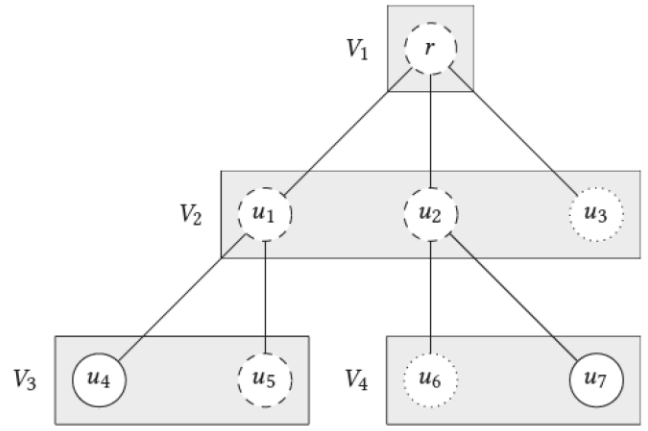
- Requires to know probability of having found a certain embedding $p_i$

## Theoretical problem setting

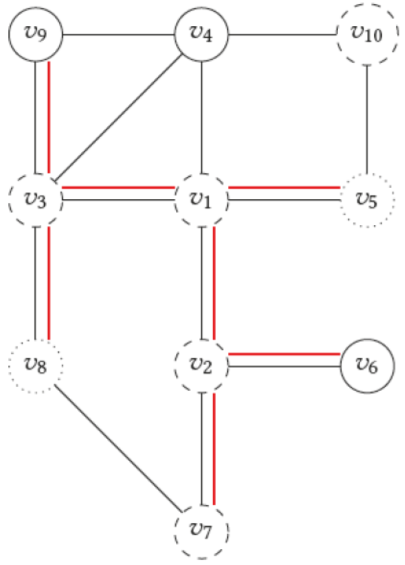
- Labeled (large) graph: $G = (V(G), E(G), l(G))$
- Labeled pattern graph: $H = (V(H), E(H), l(H))$

**Estimate the number of subgraph isomorphisms from H to G!**

H



(a)



(b)

G

Set of embeddings:

$$C_H(G)$$

Number of embeddings:

$$|C_H(G)|$$

## Estimate Pattern Count

- Simple idea: Importance Sampling
  - Find (partial) embeddings and estimate the number of possible inclusions

    $Z = \frac{1}{p_i}$ $\mathbb{E}[Z] = \sum_{i \in C_H(G)} p_i \frac{1}{p_i} = |C_H(G)|$
  - Iterate k times and take averaged estimation $p_i$
  - Requires to know probability of having found a certain embedding

**For arbitrary graphs this is still expensive!**

# Importance Sampling in arbitrary graphs

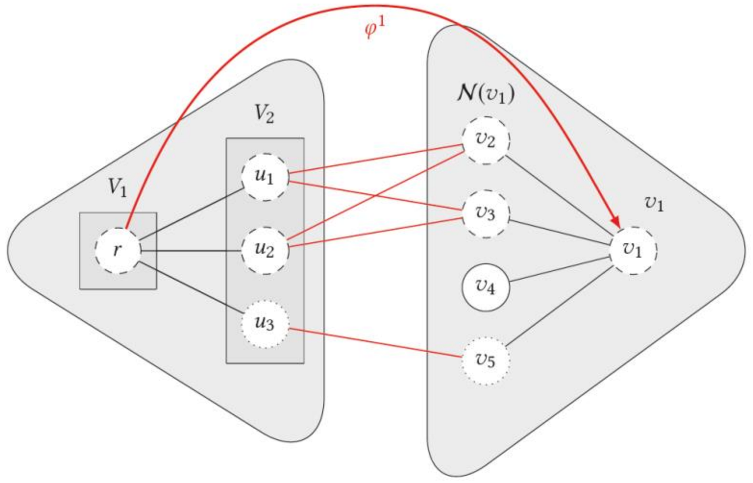
- Pattern graph needs an ordered bipartite decomposition (OBD) for efficient approximation (Fürer, 2014)
- Need to sample maximum matchings almost uniformly at random which is expensive
- Requires to compute ordered bipartite decomposition of pattern (OBD), Runtime for k iterations

$$\mathcal{O}\left(k\frac{s(D(H))\Delta(G)!}{(\Delta(G)-w(D(H)))!}\cdot(w(D(H))+\Delta(G))\right)$$

  - factorial in the degree of G
  - depending on the size of OBD of H

**Our solution: Restrict pattern graphs H to trees, often patterns are already trees**

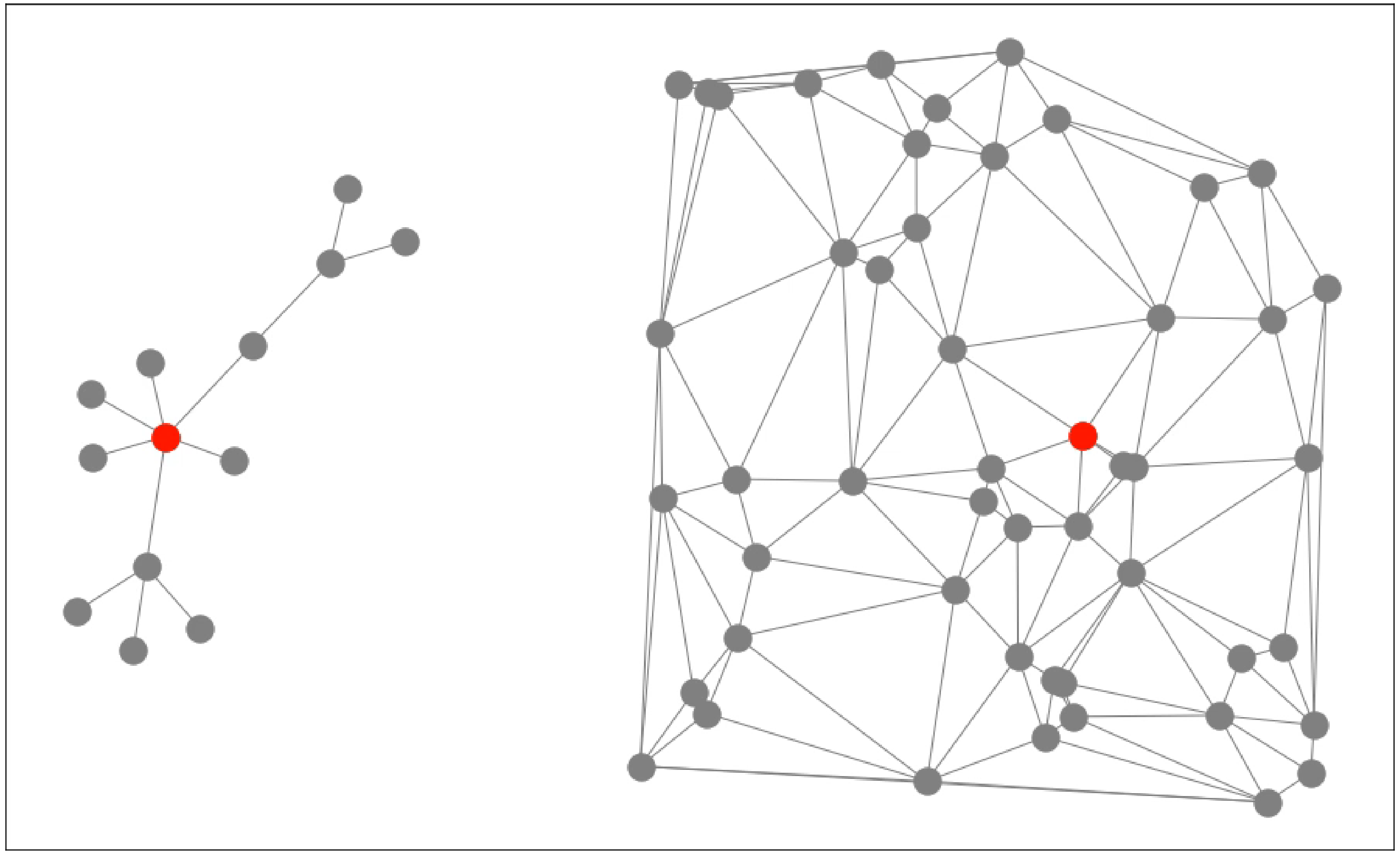
OBD $D(H)$



## Restricting to tree patterns

- Select tree root uniformly at random

## The HOPS algorithm



## Restricting Patterns to Trees

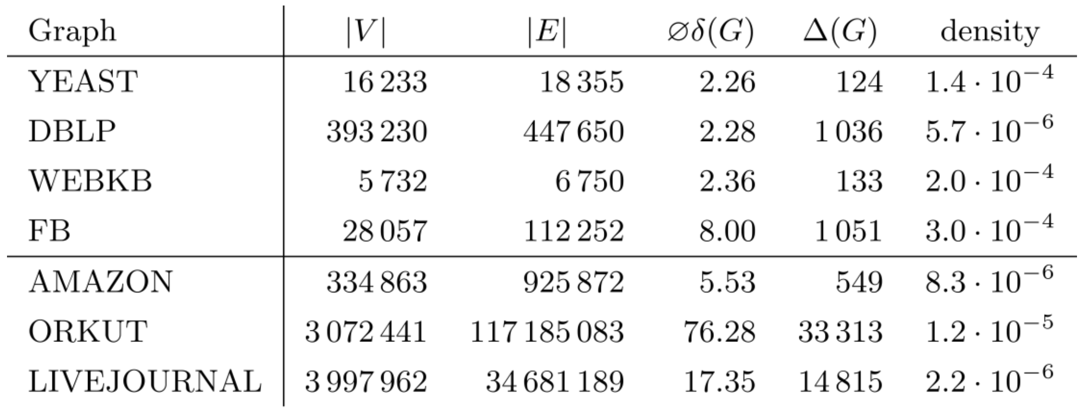
- Advantages:
  - Easy implementation and fast
  - **No** computation of OBD
  - **No** explicit matching algorithm necessary
  - Depends **linearly** on pattern size and large graph degree
    (large graph size does not matter!)
- Disadvantages:
  - Only trees as patterns, often patterns are actually trees

## Estimating Embeddings for Tree Patterns is Accurate and FAST!

- same guarantees hold as for general patterns (FPRAS)
- runtime $\mathcal{O}\left(k\left|V(H)\right|\Delta(G)\right)$
  - depends only linearly on degree of G
  - depends only linearly on number of vertices in pattern

## HOPS Estimation Experiments

Graph Data:



| Graph | $\lvert V\rvert$ | $\lvert E\rvert$ | $\varnothing\delta(G)$ | $\Delta(G)$ | density |
|---|---|---|---|---|---|
| YEAST | 16 233 | 18 355 | 2.26 | 124 | $1.4 \cdot 10^{-4}$ |
| DBLP | 393 230 | 447 650 | 2.28 | 1 036 | $5.7 \cdot 10^{-6}$ |
| WEBKB | 5 732 | 6 750 | 2.36 | 133 | $2.0 \cdot 10^{-4}$ |
| FB | 28 057 | 112 252 | 8.00 | 1 051 | $3.0 \cdot 10^{-4}$ |
| AMAZON | 334 863 | 925 872 | 5.53 | 549 | $8.3 \cdot 10^{-6}$ |
| ORKUT | 3 072 441 | 117 185 083 | 76.28 | 33 313 | $1.2 \cdot 10^{-5}$ |
| LIVEJOURNAL | 3 997 962 | 34 681 189 | 17.35 | 14 815 | $2.2 \cdot 10^{-6}$ |

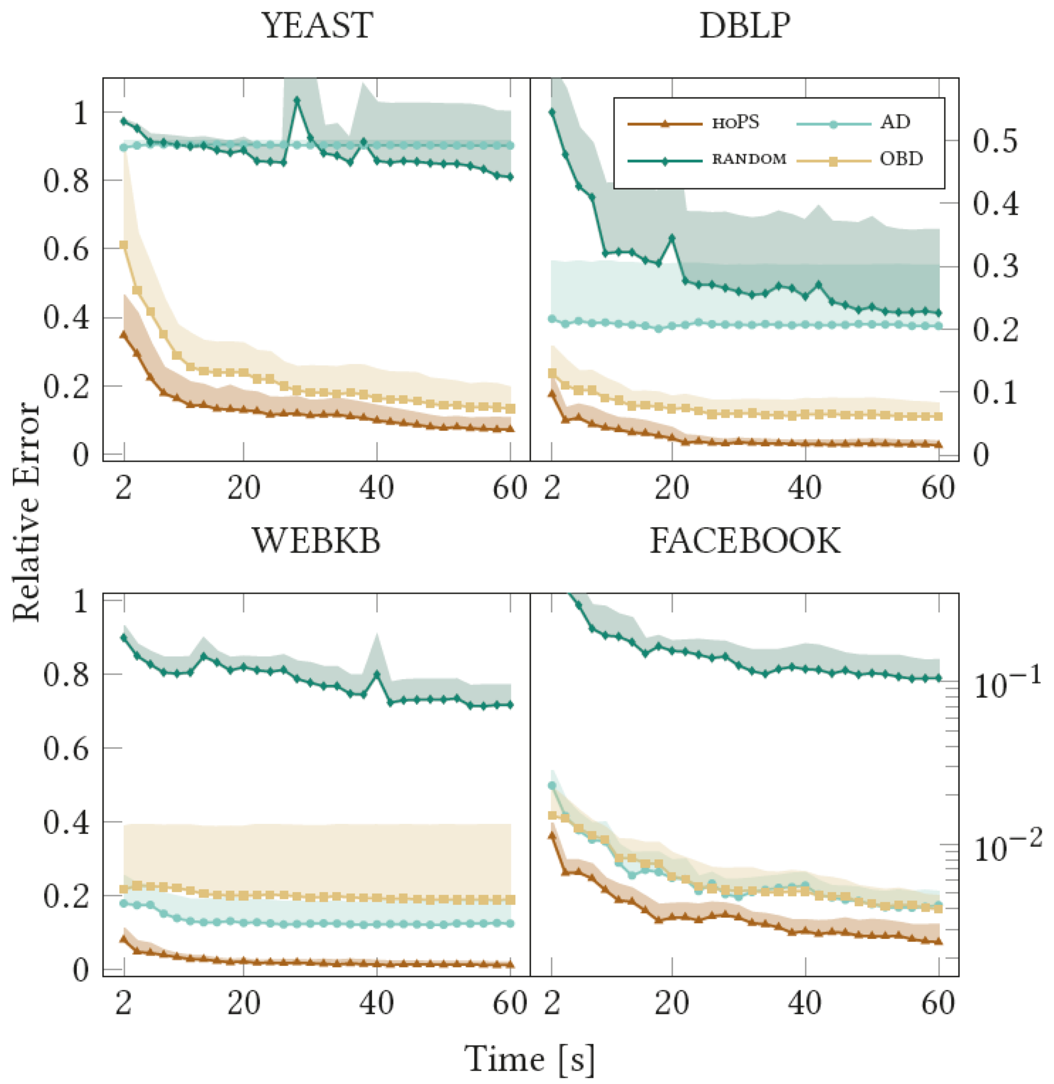
Table 1: Graph datasets

Baselines:

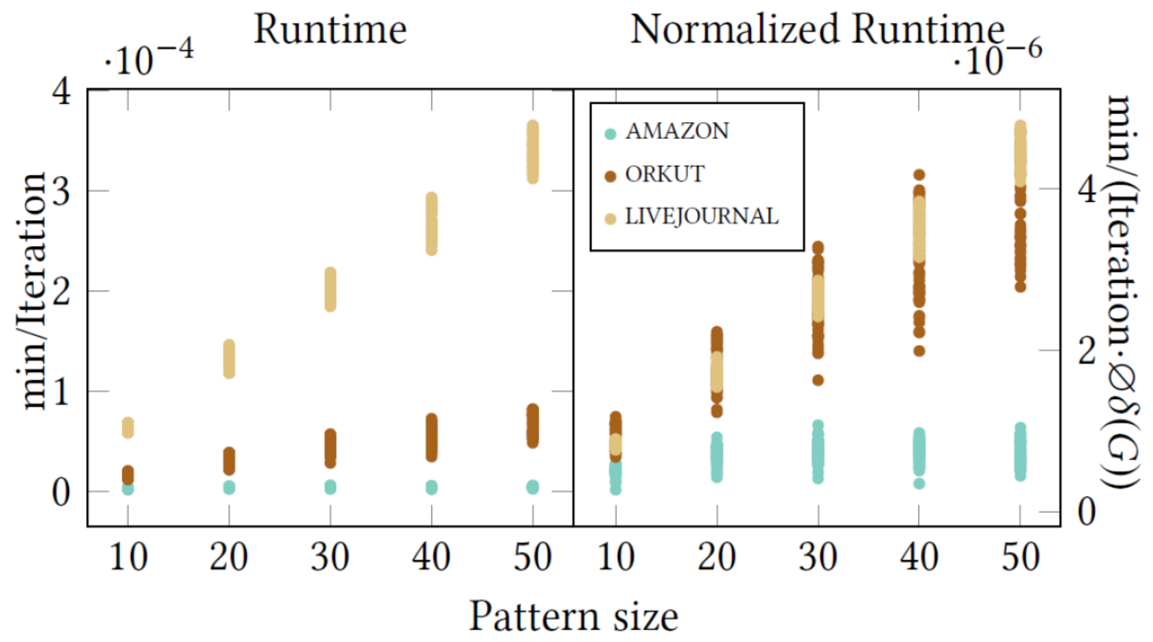
- Exact algorithm (Ullmann, 1976)
- OBD/AD algorithms (Ravkic, 2018)

# Estimation Results

- HOPS outperforms baselines
- Lower relative error
- Lower deviation
- Very Fast compared to exact algorithm



## How fast is HOPS in Practice?



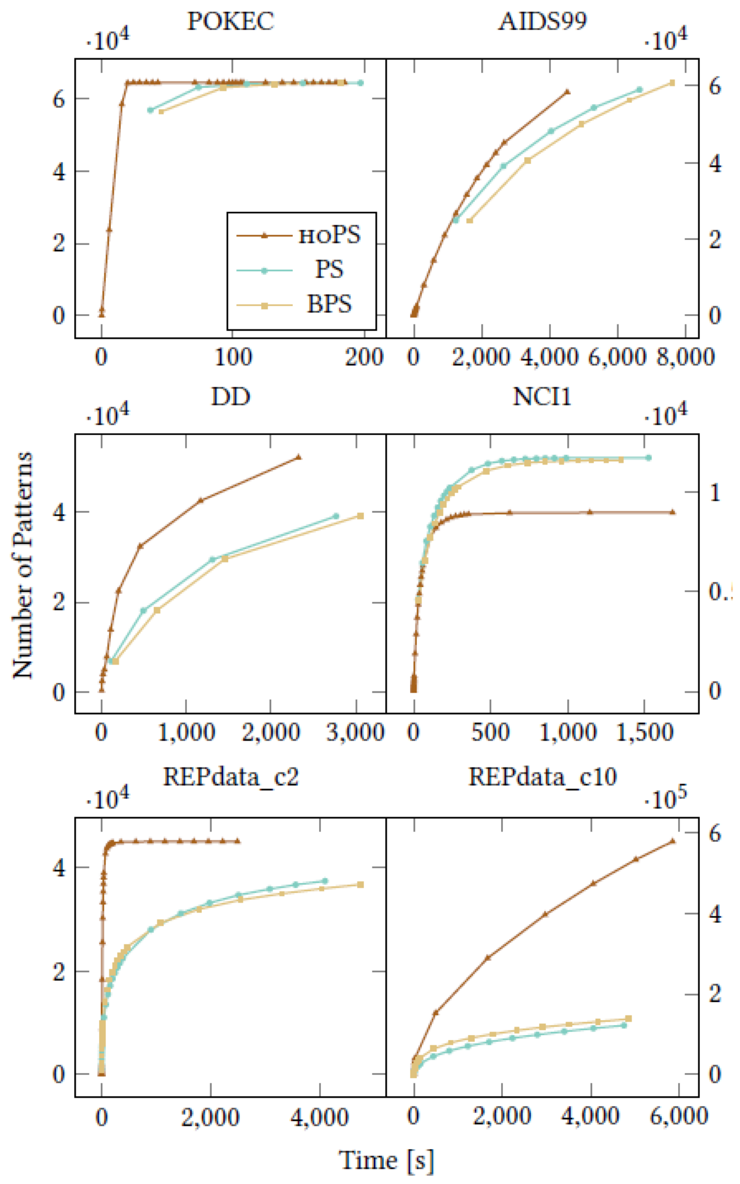
- At least linear in the pattern size (faster for bigger patterns because no embedding found)
- Does not depend on large graph sizes

**Note:** Usually no chance of exact embedding number for patterns bigger than 20 in acceptable time

# HOPS Frequent Subtree Mining Experiment

Use HOPS to decide whether candidate pattern is frequently present in graph database

- Finds substantially more patterns in the same time than state-of-the-art
- In some cases faster by orders of magnitude



## Conclusion

- Very fast, independent of target graph size!
- Accurate, outperforms state of the art estimation algorithms
- easy to implement
- allows to estimate number of trees in graphs too large and patterns too large for state-of-the-art
- Finds frequent subtrees orders of magnitudes faster than state-of-the-art