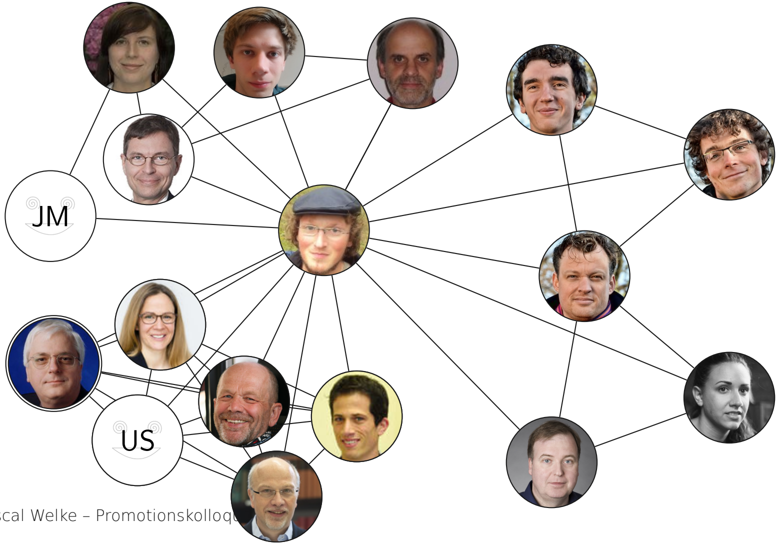
Efficient Frequent Subtree Mining Beyond Forests

Pascal Welke

Promotionskolloquium

UNIVERSITÄT BONN

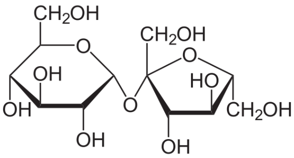
Example: Co-authorship Networks



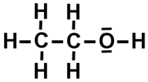
A *graph* $G = (V, E)$ consists of a set of vertices V and a set of edges E connecting pairs of vertices.

(Pictures taken from the home pages of the authors)

Example: Chemical Molecules



Saccharose



Ethanol

(commons.wikimedia.org)

How to Learn From Graphs?

- Similarity based learning methods
  - "close by objects behave similarly"

How to Learn From Graphs?

- Similarity based learning methods
  - "close by objects behave similarly"
- Hence: What does "close by" mean if objects are graphs?

- We would like to learn a suitable similarity function between graphs from a given graph database $\mathcal{D}$

Frequent Subgraph Mining

- We would like to learn a suitable similarity function between graphs from a given graph database $\mathcal{D}$
- *Frequent subgraphs* are a reasonable choice (e.g. Deshpande et al, 2005) to define similarities in a domain of graphs

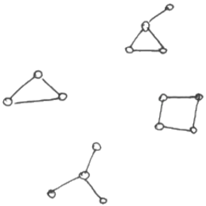
Frequent Subgraph Mining

- We would like to learn a suitable similarity function between graphs from a given graph database $\mathcal{D}$
- *Frequent subgraphs* are a reasonable choice (e.g. Deshpande et al, 2005) to define similarities in a domain of graphs

Frequent Connected Subgraph Mining (FCSM)

Given a dataset of graphs $\mathcal{D} \subseteq \mathcal{G}$ and an integer threshold $t \leq |\mathcal{D}|$

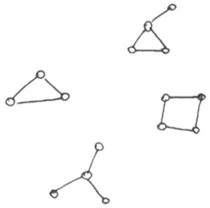
$\mathcal{D}$



Frequent Subgraph Mining

- We would like to learn a suitable similarity function between graphs from a given graph database $\mathcal{D}$
- *Frequent subgraphs* are a reasonable choice (e.g. Deshpande et al, 2005) to define similarities in a domain of graphs

$\mathcal{D}$



Frequent Connected Subgraph Mining (FCSM)

Given a dataset of graphs $\mathcal{D} \subseteq \mathcal{G}$ and an integer threshold $t \leq |\mathcal{D}|$

List all connected graphs $P \in \mathcal{P}$ that are subgraph isomorphic to at least t graphs in $\mathcal{D}$.

2-frequent subgraphs of $\mathcal{D}$

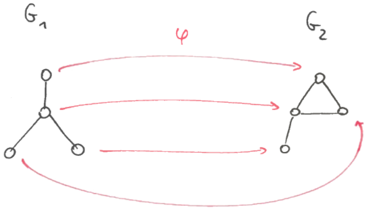
Definition

A *subgraph isomorphism* is an injective mapping

$$\varphi : V(G_1) \to V(G_2)$$

such that

$$(v_1, v_2) \in E(G_1) \Rightarrow (\varphi(v_1), \varphi(v_2)) \in E(G_2)$$



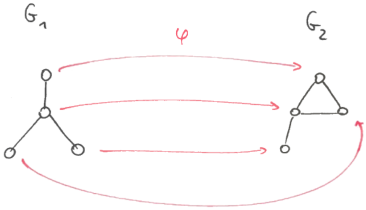
Definition

A *subgraph isomorphism* is an injective mapping

$$\varphi : V(G_1) \to V(G_2)$$

such that

$$(v_1, v_2) \in E(G_1) \Rightarrow (\varphi(v_1), \varphi(v_2)) \in E(G_2)$$



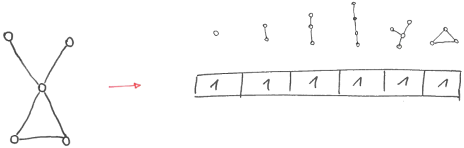
Deciding whether one exists, is *NP-hard*.

FCSM based Similarity Functions

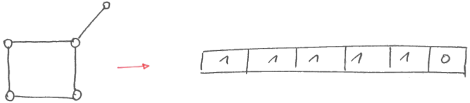
- We *embed* unseen graphs G_1, G_2 into the Hamming space $\{0,1\}^{\mathcal{F}}$ spanned by the frequent patterns $\mathcal{F}$

- We *embed* unseen graphs G_1, G_2 into the Hamming space $\{0,1\}^{\mathcal{F}}$ spanned by the frequent patterns $\mathcal{F}$

G_1

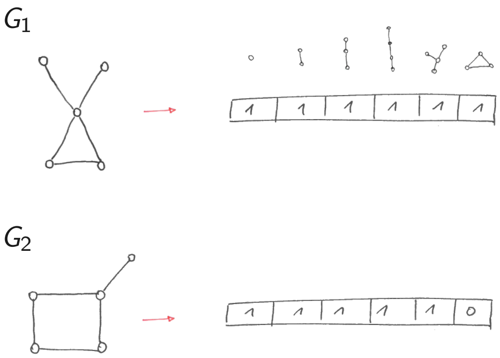


G_2



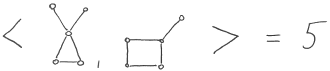
FCSM based Similarity Functions

- We *embed* unseen graphs G_1, G_2 into the Hamming space $\{0,1\}^{\mathcal{F}}$ spanned by the frequent patterns $\mathcal{F}$
- ...and can employ any similarity function for real valued vector spaces

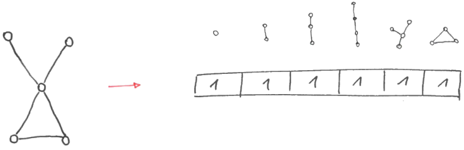


FCSM based Similarity Functions

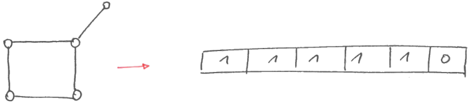
- We *embed* unseen graphs G_1, G_2 into the Hamming space $\{0,1\}^{\mathcal{F}}$ spanned by the frequent patterns $\mathcal{F}$
- ...and can employ any similarity function for real valued vector spaces



G_1



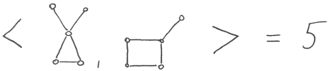
G_2



UNIVERSITÄT BONN

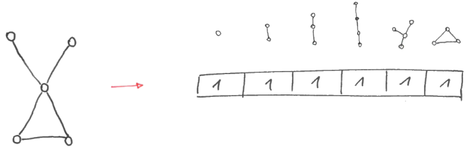
FCSM based Similarity Functions

- We *embed* unseen graphs G_1, G_2 into the Hamming space $\{0,1\}^{\mathcal{F}}$ spanned by the frequent patterns $\mathcal{F}$
- ...and can employ any similarity function for real valued vector spaces

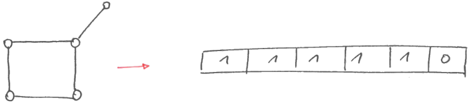


- This allows to apply e.g. all kernel methods to graph databases, like SVMs (Cortes and Vapnik, 1995) etc.

G_1



G_2



Problems with FCSM

- Software exists, e.g.
  - FSG (Kuramochi and Karypis, 2001)
  - gSpan (Yan and Han, 2002)
  - Gaston (Nijssen and Kok, 2005)...

Problems with FCSM

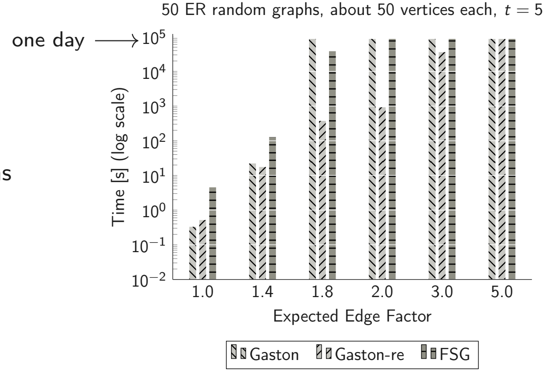
- Software exists, e.g.
  - FSG (Kuramochi and Karypis, 2001)
  - gSpan (Yan and Han, 2002)
  - Gaston (Nijssen and Kok, 2005)...
- *However:*

Problems with FCSM

- Software exists, e.g.
  - FSG (Kuramochi and Karypis, 2001)
  - gSpan (Yan and Han, 2002)
  - Gaston (Nijssen and Kok, 2005)...
- *However:*
  - *only works* for (very simple) chemical graphs

Problems with FCSM

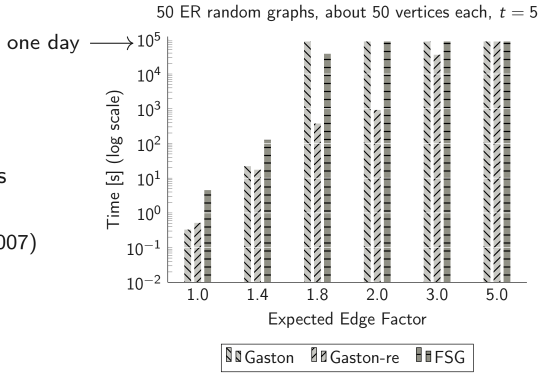
- Software exists, e.g.
  - FSG (Kuramochi and Karypis, 2001)
  - gSpan (Yan and Han, 2002)
  - Gaston (Nijssen and Kok, 2005)...
- *However:*
  - *only works* for (very simple) chemical graphs
  - *explodes* on most other datasets



UNIVERSITÄT BONN

Problems with FCSM

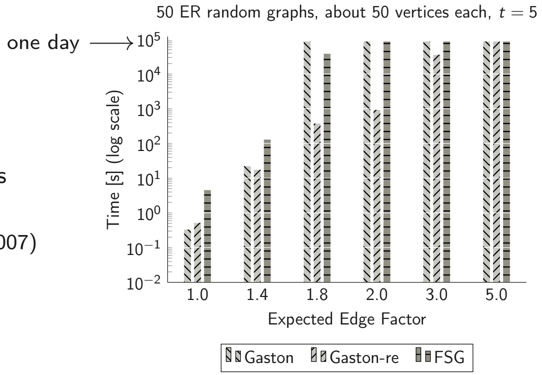
- Software exists, e.g.
  - FSG (Kuramochi and Karypis, 2001)
  - gSpan (Yan and Han, 2002)
  - Gaston (Nijssen and Kok, 2005)...
- *However:*
  - *only works* for (very simple) chemical graphs
  - *explodes* on most other datasets
- *Computationally Intractable* (Horváth et al, 2007)



UNIVERSITÄT BONN

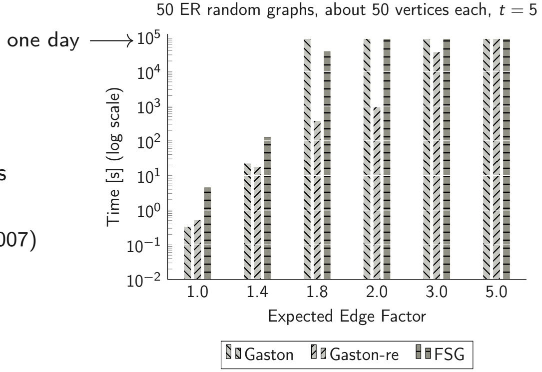
Problems with FCSM

- Software exists, e.g.
  - FSG (Kuramochi and Karypis, 2001)
  - gSpan (Yan and Han, 2002)
  - Gaston (Nijssen and Kok, 2005)...
- *However:*
  - *only works* for (very simple) chemical graphs
  - *explodes* on most other datasets
- *Computationally Intractable* (Horváth et al, 2007)
  - previous work deals with this using some properties of very simple graphs



Problems with FCSM

- Software exists, e.g.
  - FSG (Kuramochi and Karypis, 2001)
  - gSpan (Yan and Han, 2002)
  - Gaston (Nijssen and Kok, 2005)...
- *However:*
  - *only works* for (very simple) chemical graphs
  - *explodes* on most other datasets
- *Computationally Intractable* (Horváth et al, 2007)
  - previous work deals with this using some properties of very simple graphs



⇒ *There is no system that can reliably mine frequent subgraphs for arbitrary graph databases of small to medium sized graphs*

Main Contributions

Main Contributions

1. Introduction of a new pattern mining paradigm: *probabilistic frequent subtrees*
   - *efficient* for *arbitrary* graph databases
   - though *incomplete*, comparable predictive performance to exact frequent subgraphs

Main Contributions

1. Introduction of a new pattern mining paradigm: *probabilistic frequent subtrees*
   - *efficient* for *arbitrary* graph databases
   - though *incomplete*, comparable predictive performance to exact frequent subgraphs
2. Novel results on *subtree isomorphism*
   - *boosted probabilistic frequent subtrees*
   - positive complexity results for exact frequent subtree mining

Main Contributions

1. Introduction of a new pattern mining paradigm: *probabilistic frequent subtrees*
   - *efficient* for *arbitrary* graph databases
   - though *incomplete*, comparable predictive performance to exact frequent subgraphs
2. Novel results on *subtree isomorphism*
   - *boosted probabilistic frequent subtrees*
   - positive complexity results for exact frequent subtree mining
3. Fast computation of frequent subgraph based similarity functions
   - exploitation of partially ordered set structure on frequent patterns

Main Contributions

1. Introduction of a new pattern mining paradigm: *probabilistic frequent subtrees*
   - *efficient* for *arbitrary* graph databases
   - though *incomplete*, comparable predictive performance to exact frequent subgraphs
2. Novel results on *subtree isomorphism*
   - *boosted probabilistic frequent subtrees*
   - positive complexity results for exact frequent subtree mining
3. Fast computation of frequent subgraph based similarity functions
   - exploitation of partially ordered set structure on frequent patterns

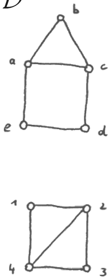
$\Rightarrow$ Resulting in the first *theoretically efficient* and *practically robust* system for frequent subtree mining in arbitrary graph databases

Related Publications

- Pascal Welke, Tamás Horváth, Stefan Wrobel (2019) Probabilistic and exact frequent subtree mining in graphs beyond forests. *Machine Learning Journal*
  - Pascal Welke, Tamás Horváth, Stefan Wrobel (2015) On the complexity of frequent subtree mining in very simple structures. Inductive Logic Programming (ILP), *Springer LNCS*
- Pascal Welke, Tamás Horváth, Stefan Wrobel (2018) Probabilistic frequent subtrees for efficient graph classification and retrieval. *Machine Learning Journal*
  - Pascal Welke, Tamás Horváth, Stefan Wrobel (2016a) Probabilistic frequent subtree kernels. New Frontiers in Mining Complex Patterns (NFMCP), *Springer LNCS*
  - Pascal Welke, Tamás Horváth, Stefan Wrobel (2016b) Min-hashing for probabilistic frequent subtree feature spaces. Discovery Science (DS), *Springer LNCS*
- Pascal Welke (2017) Simple necessary conditions for the existence of a Hamiltonian path with applications to cactus graphs. *CoRR*

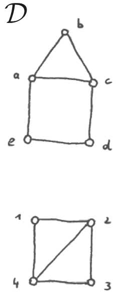
Part 1.
Probabilistic Frequent Subtrees

$\mathcal{D}$



Probabilistic Frequent Subtree Mining

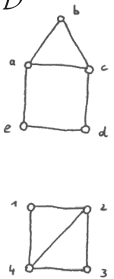
- We simplify our problem by mining only *frequent subtrees* ⟶ $t = 2$
  - thus far, mining and embedding are still computationally intractable



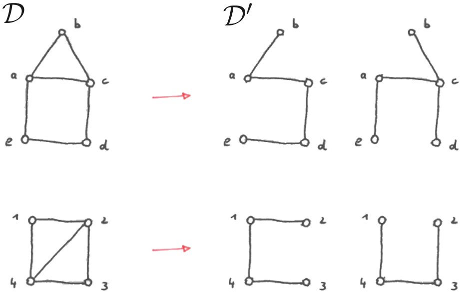
- We simplify our problem by mining only *frequent subtrees*
  - thus far, mining and embedding are still computationally intractable
- We give up the completeness of mining

$t = 2$

$\mathcal{D}$

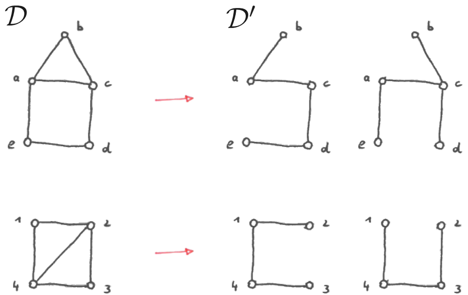


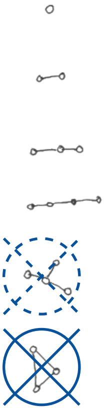
- We simplify our problem by mining only *frequent subtrees* $\longrightarrow$ $t = 2$
  - thus far, mining and embedding are still computationally intractable
- We give up the completeness of mining
  - by *sampling* a fixed number of spanning trees for each graph



UNIVERSITÄT BONN

- We simplify our problem by mining only *frequent subtrees* ⟶ $t = 2$
  - thus far, mining and embedding are still computationally intractable
- We give up the completeness of mining
  - by *sampling* a fixed number of spanning trees for each graph
  - $\Rightarrow$ some frequent subtrees might not be found





- Sampling of spanning trees *maps graphs to forests*

Main Trick!

- Sampling of spanning trees *maps graphs to forests*

Main Trick!

- Frequent *trees in forest databases* can be mined with *polynomial delay* (Horváth and Ramon, 2010)

- Sampling of spanning trees *maps graphs to forests*

Main Trick!

- Frequent *trees in forest databases* can be mined with *polynomial delay* (Horváth and Ramon, 2010)

Theorem

Probabilistic Frequent Subtrees can be mined with polynomial delay.

Main Trick!

- Sampling of spanning trees *maps graphs to forests*
- Frequent *trees in forest databases* can be mined with *polynomial delay* (Horváth and Ramon, 2010)

Theorem

Probabilistic Frequent Subtrees can be mined with polynomial delay.

- Does it work in practice?

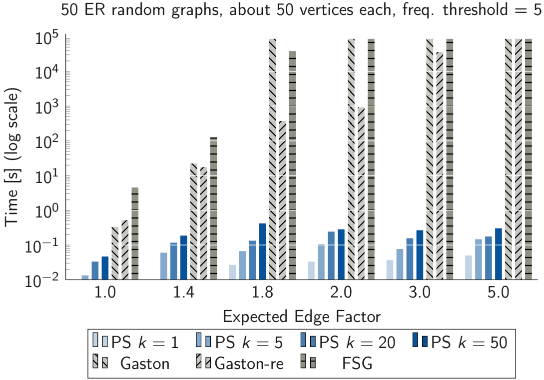
- Sampling of spanning trees *maps graphs to forests* *Main Trick!*
- Frequent *trees in forest databases* can be mined with *polynomial delay* (Horváth and Ramon, 2010)

Theorem

Probabilistic Frequent Subtrees can be mined with polynomial delay.

- Does it work in practice?
  - ...in feasible time?
  - ...as basis for similarity based learning?

Runtime of Probabilistic Subtrees (PS)

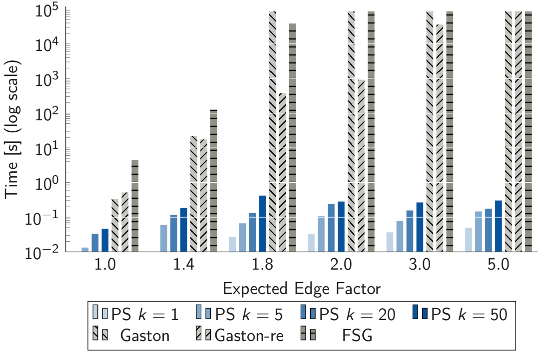


k is the number of sampled spanning trees per graph

Experimental Results

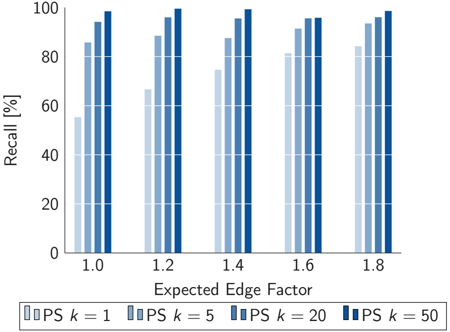
Runtime of Probabilistic Subtrees (PS)

50 ER random graphs, about 50 vertices each, freq. threshold = 5



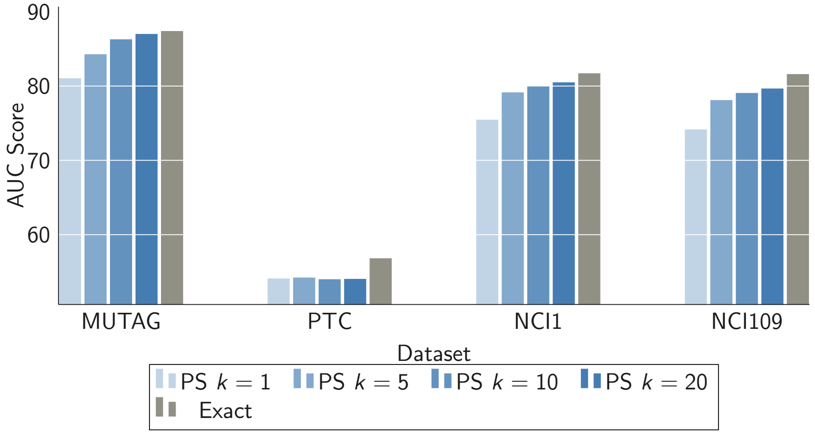
Recall of PS

same dataset as on left



k is the number of sampled spanning trees per graph

Classification Quality of Probabilistic Subtree (PS) based Learners



Wrap Up

- We have presented a system that can *reliably* mine frequent subtrees in *arbitrary graph databases* of small to medium sized graphs
  - we give up completeness
  - ...but it still works well

Wrap Up

- We have presented a system that can *reliably* mine frequent subtrees in *arbitrary graph databases* of small to medium sized graphs
  - we give up completeness
  - ...but it still works well
- *Are we done, yet?*

Wrap Up

- We have presented a system that can *reliably* mine frequent subtrees in *arbitrary graph databases* of small to medium sized graphs
  - we give up completeness
  - ...but it still works well
- *How can we improve probabilistic frequent subtree mining?*

- We have presented a system that can *reliably* mine frequent subtrees in *arbitrary graph databases* of small to medium sized graphs
  - we give up completeness
  - ...but it still works well
- *How can we improve probabilistic frequent subtree mining?*
  - there are up to n^{n-2} spanning trees in a graph with n vertices (Cayley, 1889)

Wrap Up

- We have presented a system that can *reliably* mine frequent subtrees in *arbitrary graph databases* of small to medium sized graphs
  - we give up completeness
  - ...but it still works well
- *How can we improve probabilistic frequent subtree mining?*
  - there are up to n^{n-2} spanning trees in a graph with n vertices (Cayley, 1889)
  - our method can efficiently consider only a sample of polynomial size

Wrap Up

- We have presented a system that can *reliably* mine frequent subtrees in *arbitrary graph databases* of small to medium sized graphs
  - we give up completeness
  - ...but it still works well
- *How can we improve probabilistic frequent subtree mining?*
  - there are up to n^{n-2} spanning trees in a graph with n vertices (Cayley, 1889)
  - our method can efficiently consider only a sample of polynomial size
  - $\Rightarrow$ there is an exponential gap, which might reduce recall

Part 2.
Boosted Probabilistic Frequent Subtrees

Boosted Probabilistic Frequent Subtrees

- Improve the probabilistic subgraph isomorphism algorithm

Boosted Probabilistic Frequent Subtrees

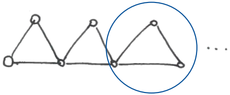
- Improve the probabilistic subgraph isomorphism algorithm
- Implicitly consider *exponentially* many spanning trees *in polynomial time*
  - ...by using additional insights into graphs

Boosted Probabilistic Frequent Subtrees

- Improve the probabilistic subgraph isomorphism algorithm
- Implicitly consider *exponentially* many spanning trees *in polynomial time*
  - ...by using additional insights into graphs

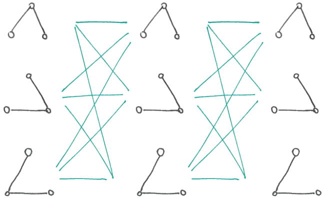
Boosted Probabilistic Frequent Subtrees

- Improve the probabilistic subgraph isomorphism algorithm
- Implicitly consider *exponentially* many spanning trees *in polynomial time*
  - ...by using additional insights into graphs



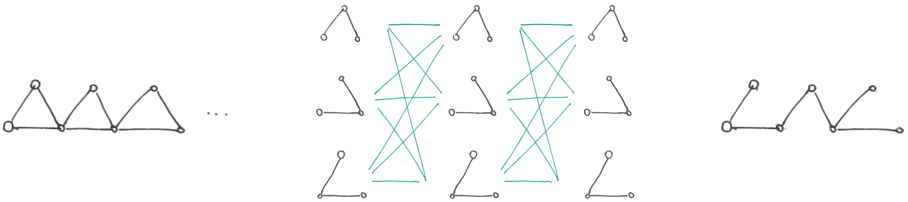
Biconnected Component

- Improve the probabilistic subgraph isomorphism algorithm
- Implicitly consider *exponentially* many spanning trees *in polynomial time*
  - ...by using additional insights into graphs



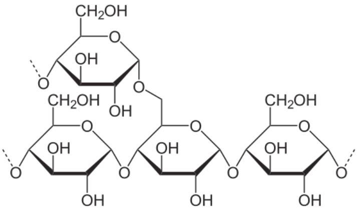
27 spanning trees

- Improve the probabilistic subgraph isomorphism algorithm
- Implicitly consider *exponentially* many spanning trees *in polynomial time*
  - ...by using additional insights into graphs



27 spanning trees

- Improve the probabilistic subgraph isomorphism algorithm
- Implicitly consider *exponentially* many spanning trees *in polynomial time*
  - …by using additional insights into graphs

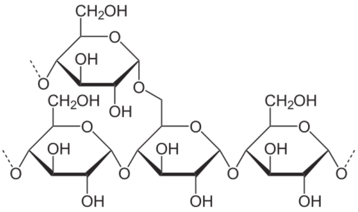


Amylopectin

`commons.wikimedia.org`

Boosted Probabilistic Frequent Subtrees

- Improve the probabilistic subgraph isomorphism algorithm
- Implicitly consider *exponentially* many spanning trees *in polynomial time*
  - ...by using additional insights into graphs



Amylopectin

`commons.wikimedia.org`

...you already see enough graph for $6^4 = 936$ spanning trees

Theorem

Near optimal for trees G

Subtree Isomorphism from a tree H into an arbitrary graph G can be solved in time

$$O\left(f_{\max}^2(G) \cdot |E(G)| \cdot |V(H)|^{1.5}\right)$$

where $f_{\max}(G)$ is the maximum number of spanning trees in any graph induced by the union of biconnected components containing some vertex $v \in V(G)$.

Boosted Subtree Isomorphism

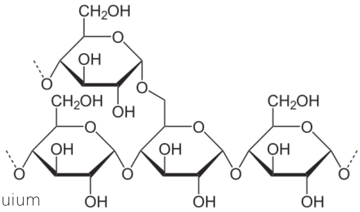
Theorem

Near optimal for trees G

Subtree Isomorphism from a tree H into an arbitrary graph G can be solved in time

$$O\left(f_{\max}^2(G) \cdot |E(G)| \cdot |V(H)|^{1.5}\right)$$

where $f_{\max}(G)$ is the maximum number of spanning trees in any graph induced by the union of biconnected components containing some vertex $v \in V(G)$.



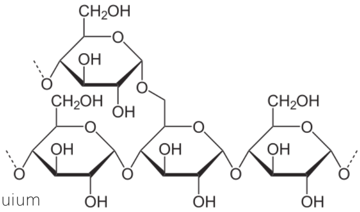
Theorem

Near optimal for trees G

Subtree Isomorphism from a tree H into an arbitrary graph G can be solved in time

$$O\left(f_{\max}^2(G) \cdot |E(G)| \cdot |V(H)|^{1.5}\right)$$

where $f_{\max}(G)$ is the maximum number of spanning trees in any graph induced by the union of biconnected components containing some vertex $v \in V(G)$.



$f_{\max}(G) = 6$

Theorem (Informally)

The Subtree Isomorphism problem can be solved with one-sided error in time

$$O\left(\ell^2 \cdot |E(G)| \cdot |V(H)|^{1.5}\right)$$

where ℓ is the maximum number of local spanning trees drawn in certain neighborhoods of articulation vertices, considering $\ell^{O(n)}$ global spanning trees.

Theorem (Informally)

The Subtree Isomorphism problem can be solved with one-sided error in time

$$O\left(\ell^2 \cdot |E(G)| \cdot |V(H)|^{1.5}\right)$$

where ℓ is the maximum number of local spanning trees drawn in certain neighborhoods of articulation vertices, considering $\ell^{O(n)}$ global spanning trees.

This allows to efficiently mine boosted probabilistic frequent subtrees in arbitrary graph databases.

An Algorithm

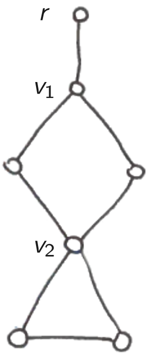
Input : tree H with $|V(H)| > 1$ and an arbitrary connected graph G with $|V(G)| \geq |V(H)|$
Output: True if $H \preccurlyeq G$; o/w False

Main:

set $\mathcal{C} := \emptyset$
pick a vertex $r \in V(G)$ and compute the complete guidance tree $\mathbb{T} = (\mathcal{T}, \mathcal{S})$ of G for the tree skeleton $\mathcal{T}$ rooted at r
for all $v \in V(\mathcal{T})$ in a postorder **do**
  // $\mathcal{S}_v \in \mathcal{S}$ is the bag of v in $\mathbb{T}$
  for all $\tau \in \mathcal{S}_v$ **do**
    for all $w \in V(\tau)$ in a postorder **do**
      $\mathcal{C} := \mathcal{C} \cup$ Characteristics(v, u, τ, w)
      if $(H_u^u, \tau, w) \in \mathcal{C}$ **then return** True
return False

Characteristics(v, u, τ, w):
  $\mathcal{C}_\tau := \emptyset$
  for all $\theta \in \Theta_{vw}(\tau)$ **do**
    for all $u \in V(H)$ **do**
      let τ' be the tree satisfying $\theta = \tau \cup \tau'$
      let C_τ (resp. $C_{\tau'}$) be the set of children of w in τ (resp. τ') and $C_\theta := C_\tau \cup C_{\tau'}$
      let $B = (C_\theta \dot\cup \mathcal{N}(u), E)$ be the bipartite graph with
        $cu' \in E$ if and only if
        $(c \in C_\tau \wedge (H_{u'}^u, \tau, c) \in \mathcal{C}) \vee (c \in C_{\tau'} \wedge (H_{u'}^u, \tau', c) \in \mathcal{C})$
        for all $cu' \in C_\theta \times \mathcal{N}(u)$
      if B has a matching that covers $\mathcal{N}(u)$ **then**
        add (H_u^u, τ, w) to $\mathcal{C}_\tau$
      for all $y \in \mathcal{N}(u)$ **do**
        if B has a matching covering $\mathcal{N}(u) \setminus \{y\}$ **then**
          add (H_u^y, τ, w) to $\mathcal{C}_\tau$
  return $\mathcal{C}_\tau$

Tree H

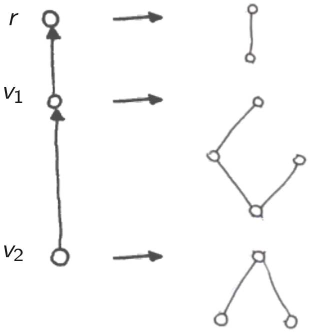


Graph G

Guidance Trees and Bags of Spanning Trees

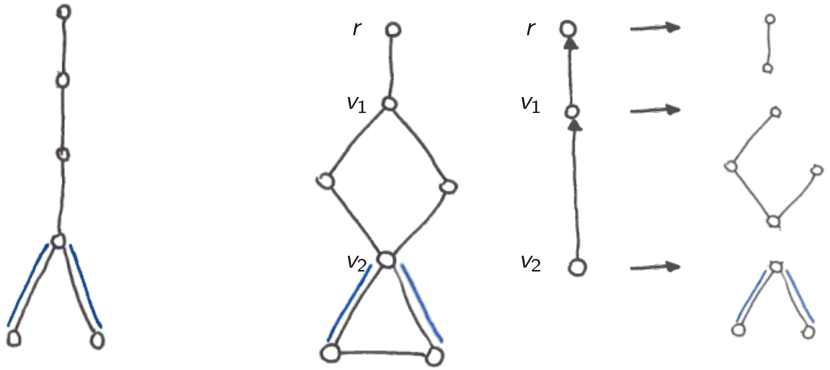
Tree H

Graph G



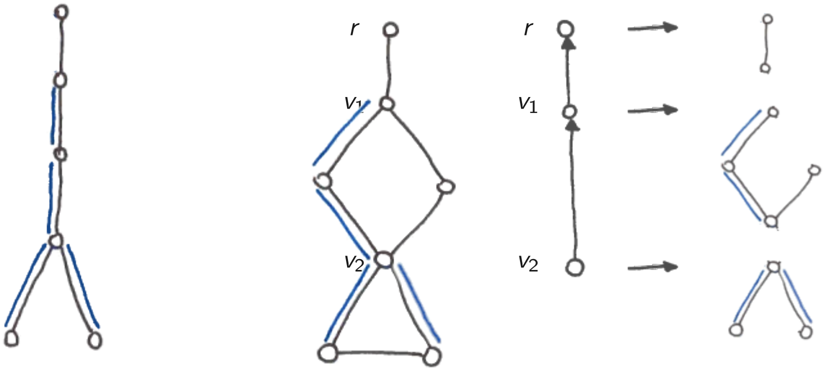
Guidance Tree $\mathbb{T}(G)$

Guidance Trees and Bags of Spanning Trees



Tree H Graph G Guidance Tree $\mathbb{T}(G)$

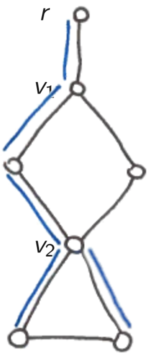
Guidance Trees and Bags of Spanning Trees



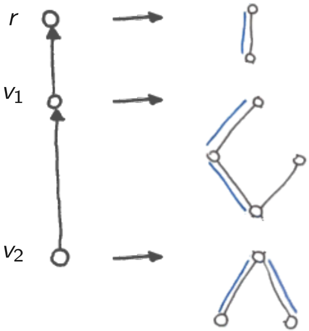
Tree H Graph G Guidance Tree $\mathbb{T}(G)$

Guidance Trees and Bags of Spanning Trees

Tree H



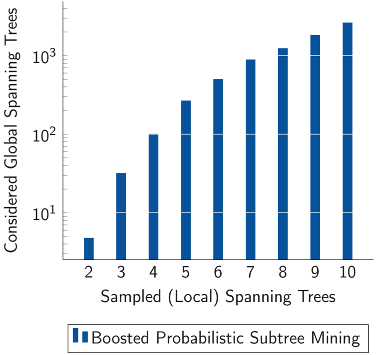
Graph G



Guidance Tree $\mathbb{T}(G)$

Brownian Motion Dataset (200 graphs à 30 vertices)

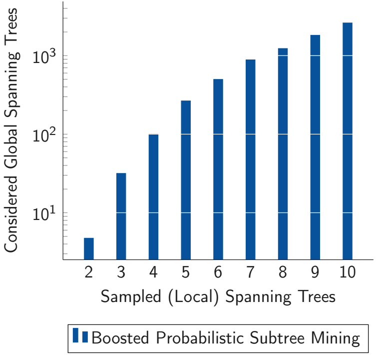
Average Number of Considered Spanning Trees per Graph



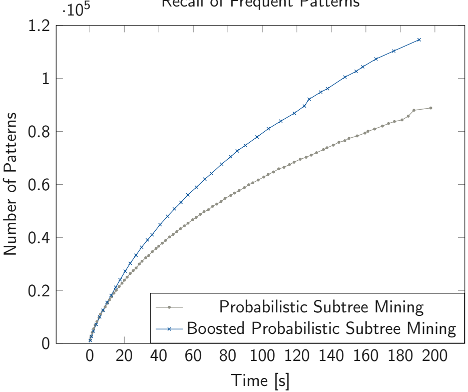
UNIVERSITÄT BONN

Brownian Motion Dataset (200 graphs à 30 vertices)

Average Number of Considered Spanning Trees per Graph



Recall of Frequent Patterns



Wrap Up

- *Probabilistic* frequent subtree mining

Wrap Up

- *Probabilistic* frequent subtree mining
  - we can implicitly sample *exponentially many* spanning trees in *polynomial time*

- *Probabilistic* frequent subtree mining
  - we can implicitly sample *exponentially many* spanning trees in *polynomial time*
  - this *increases the recall per time* for probabilistic frequent subtree mining in practically relevant graph databases

- *Probabilistic* frequent subtree mining
  - we can implicitly sample *exponentially many* spanning trees in *polynomial time*
  - this *increases the recall per time* for probabilistic frequent subtree mining in practically relevant graph databases
- *Exact* frequent subtree mining for *locally easy graphs*

Wrap Up

- *Probabilistic* frequent subtree mining
  - we can implicitly sample *exponentially many* spanning trees in *polynomial time*
  - this *increases the recall per time* for probabilistic frequent subtree mining in practically relevant graph databases
- *Exact* frequent subtree mining for *locally easy graphs*
  - *efficient exact* frequent subtree mining *is possible* for a new graph class

Wrap Up

- *Probabilistic* frequent subtree mining
  - we can implicitly sample *exponentially many* spanning trees in *polynomial time*
  - this *increases the recall per time* for probabilistic frequent subtree mining in practically relevant graph databases
- *Exact* frequent subtree mining for *locally easy graphs*
  - *efficient exact* frequent subtree mining *is possible* for a new graph class
  - *in practice:* relevant, as many chemical graphs are contained

Wrap Up

- *Probabilistic* frequent subtree mining
  - we can implicitly sample *exponentially many* spanning trees in *polynomial time*
  - this *increases the recall per time* for probabilistic frequent subtree mining in practically relevant graph databases
- *Exact* frequent subtree mining for *locally easy graphs*
  - *efficient exact* frequent subtree mining *is possible* for a new graph class
  - *in practice:* relevant, as many chemical graphs are contained
  - *conjecture:* we *cannot mine* frequent trees efficiently *beyond locally easy graphs*

What is still missing?

- We have identified patterns in training database

What is still missing?

- We have identified patterns in training database
- We want to use them as features to describe new data

What is still missing?

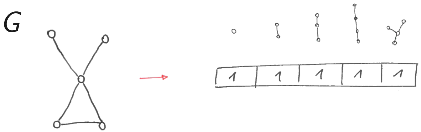
- We have identified patterns in training database
- We want to use them as features to describe new data
- How to compute feature representations / similarities for novel graphs?

Part 3.

Fast Embedding Vector Computation

Embedding Computation Problem

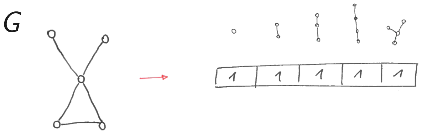
Given a set of tree patterns $\mathcal{F}$ and a (novel) text graph G



Embedding Computation Problem

Given a set of tree patterns $\mathcal{F}$ and a (novel) text graph G

Compute the feature vector for G in $\mathcal{F}$



3. Embedding Vector Computation in (B)PS Feature Spaces

Embedding Computation Problem

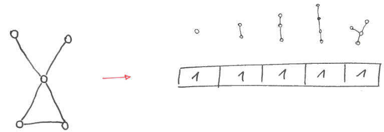
Given a set of tree patterns $\mathcal{F}$ and a (novel) text graph G

Compute the feature vector for G in $\mathcal{F}$

NP-hard problem

(Garey and Johnson, 1979)

G



UNIVERSITÄT BONN

Embedding Computation Problem

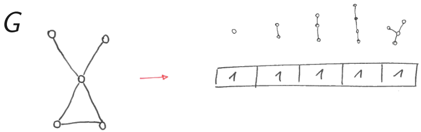
Given a set of tree patterns $\mathcal{F}$ and a (novel) text graph G

Compute the feature vector for G in $\mathcal{F}$

Often ignored in the subgraph mining community

NP-hard problem

(Garey and Johnson, 1979)



Embedding Computation Problem

Given a set of tree patterns $\mathcal{F}$ and a (novel) text graph G

Compute the feature vector for G in $\mathcal{F}$

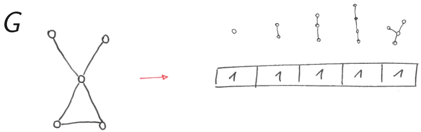
Often ignored in the subgraph mining community

NP-hard problem

(Garey and Johnson, 1979)

Solution Same as before: *probabilistic embedding operator*

$\Rightarrow$ *efficient* algorithm with *one-sided error*



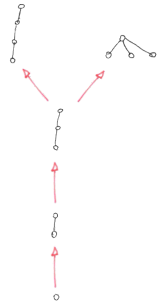
- Naive solution requires $|\mathcal{F}|$ calls to a subgraph isomorphism algorithm

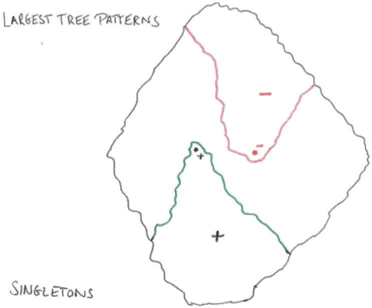
- Naive solution requires $|\mathcal{F}|$ calls to a subgraph isomorphism algorithm
- To *reduce* the number of calls utilize

- Naive solution requires $|\mathcal{F}|$ calls to a subgraph isomorphism algorithm
- To *reduce* the number of calls utilize
  - *partially ordered set* structure on tree patterns

- Naive solution requires $|\mathcal{F}|$ calls to a subgraph isomorphism algorithm
- To *reduce* the number of calls utilize
  - *partially ordered set* structure on tree patterns
  - *monotonicity* of subgraph isomorphism

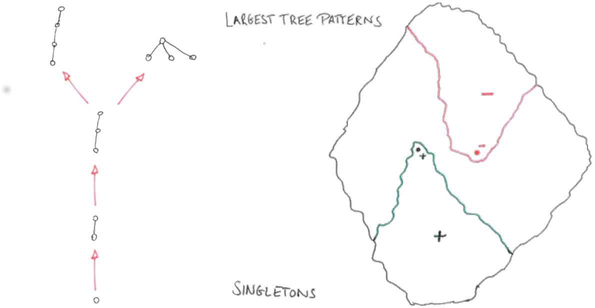
- Naive solution requires $|\mathcal{F}|$ calls to a subgraph isomorphism algorithm
- To *reduce* the number of calls utilize
  - *partially ordered set* structure on tree patterns
  - *monotonicity* of subgraph isomorphism





UNIVERSITÄT BONN

- Naive solution requires $|\mathcal{F}|$ calls to a subgraph isomorphism algorithm
- To *reduce* the number of calls utilize
  - *partially ordered set* structure on tree patterns
  - *monotonicity* of subgraph isomorphism



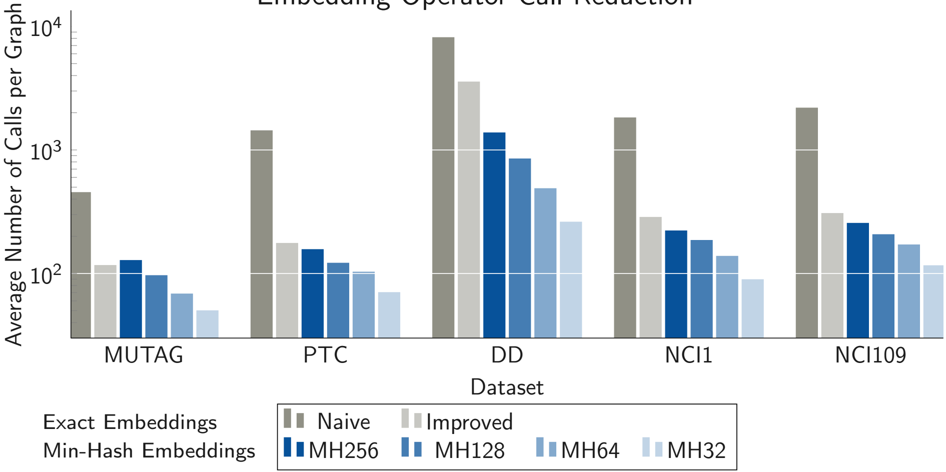
$\Rightarrow$ When computing the embedding of a graph, we do not need to test all patterns for subgraph isomorphism

- Naive solution requires $|\mathcal{F}|$ calls to a subgraph isomorphism algorithm
- To *reduce* the number of calls utilize
  - *partially ordered set* structure on tree patterns
  - *monotonicity* of subgraph isomorphism
  - we can further reduce calls by combining this with *min-hashing*

⇒ When computing the embedding of a graph, we do not need to test all patterns for subgraph isomorphism

UNIVERSITÄT BONN

Embedding Operator Call Reduction



Summary

Summary

Previously

- *Frequent subgraphs are useful* features for learning from graphs, *but*

Summary

Previously

- *Frequent subgraphs are useful* features for learning from graphs, *but*
  - *mining* and *embedding* are computationally *intractable*

UNIVERSITÄT BONN

Summary

Previously

- *Frequent subgraphs are useful* features for learning from graphs, *but*
  - *mining* and *embedding* are computationally *intractable*
  - *other mining software* only works for *very simple graph databases*

Summary

Previously

- *Frequent subgraphs are useful* features for learning from graphs, *but*
  - *mining* and *embedding* are computationally *intractable*
  - *other mining software* only works for *very simple graph databases*

Contributions

- I propose the first *theoretically efficient* and *practically robust* system for frequent subtree mining in *arbitrary* graph databases

Summary

Previously

- *Frequent subgraphs are useful* features for learning from graphs, *but*
  - *mining* and *embedding* are computationally *intractable*
  - *other mining software* only works for *very simple graph databases*

Contributions

- I propose the first *theoretically efficient* and *practically robust* system for frequent subtree mining in *arbitrary* graph databases
  - introduction of (boosted) *probabilistic frequent subtrees*

Summary

Previously

- *Frequent subgraphs are useful* features for learning from graphs, *but*
  - *mining* and *embedding* are computationally *intractable*
  - *other mining software* only works for *very simple graph databases*

Contributions

- I propose the first *theoretically efficient* and *practically robust* system for frequent subtree mining in *arbitrary* graph databases
  - introduction of (boosted) *probabilistic frequent subtrees*
  - *fast computation of embedding vectors* in probabilistic frequent subtree feature spaces

Summary

Previously

- *Frequent subgraphs are useful* features for learning from graphs, *but*
  - *mining* and *embedding* are computationally *intractable*
  - *other mining software* only works for *very simple graph databases*

Contributions

- I propose the first *theoretically efficient* and *practically robust* system for frequent subtree mining in *arbitrary* graph databases
  - introduction of (boosted) *probabilistic frequent subtrees*
  - *fast computation of embedding vectors* in probabilistic frequent subtree feature spaces
- Computational complexity of *exact* frequent subtree mining

Summary

Previously

Contributions

- *Frequent subgraphs are useful* features for learning from graphs, *but*
  - *mining* and *embedding* are computationally *intractable*
  - *other mining software* only works for *very simple graph databases*
- I propose the first *theoretically efficient* and *practically robust* system for frequent subtree mining in *arbitrary* graph databases
  - introduction of (boosted) *probabilistic frequent subtrees*
  - *fast computation of embedding vectors* in probabilistic frequent subtree feature spaces
- Computational complexity of *exact* frequent subtree mining
  - for *locally easy graphs*, frequent subtree *mining and embedding* are *computationally tractable*

Summary

Previously

- *Frequent subgraphs are useful* features for learning from graphs, *but*
  - *mining* and *embedding* are computationally *intractable*
  - *other mining software* only works for *very simple graph databases*

Contributions

- I propose the first *theoretically efficient* and *practically robust* system for frequent subtree mining in *arbitrary* graph databases
  - introduction of (boosted) *probabilistic frequent subtrees*
  - *fast computation of embedding vectors* in probabilistic frequent subtree feature spaces
- Computational complexity of *exact* frequent subtree mining
  - for *locally easy graphs*, frequent subtree *mining and embedding* are *computationally tractable*
  - we conjecture: this result is *close to the border between tractable and intractable*

References I

Arthur Cayley (1889) A theorem on trees. Quarterly Journal of Pure and Applied Mathematics 23:376–378, doi: 10.1017/cbo9780511703799.010

Michael R Garey, David S Johnson (1979) Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman

Corinna Cortes, Vladimir Vapnik (1995) Support-vector networks. Machine Learning 20(3):273–297, doi: 10.1007/bf00994018

Michihiro Kuramochi, George Karypis (2001) Frequent subgraph discovery. In: Nick Cercone, Tsau Young Lin, Xindong Wu (eds) IEEE International Conference on Data Mining (ICDM), Proceedings, IEEE Computer Society, pp 313–320, doi: 10.1109/ICDM.2001.989534

Xifeng Yan, Jiawei Han (2002) gSpan: Graph-based substructure pattern mining. In: Vipin Kumar, Shusaku Tsumoto, Ning Zhong, Philip S Yu, Xindong Wu (eds) IEEE International Conference on Data Mining (ICDM) Proceedings, IEEE Computer Society, pp 721–724, doi: 10.1109/icdm.2002.1184038

Mukund Deshpande, Michihiro Kuramochi, Nikil Wale, George Karypis (2005) Frequent substructure-based approaches for classifying chemical compounds. Transactions on Knowledge and Data Engineering 17(8):1036–1050, doi: 10.1109/tkde.2005.127

Siegfried Nijssen, Joost N Kok (2005) The gaston tool for frequent subgraph mining. Electronic Notes in Theoretical Computer Science 127(1):77–87, doi: 10.1016/j.entcs.2004.12.039

Tamás Horváth, Björn Bringmann, Luc De Raedt (2007) Frequent hypergraph mining. In: Stephen Muggleton, Ramón P Otero, Alireza Tamaddoni-Nezhad (eds) Inductive Logic Programming (ILP) Revised Selected Papers, Springer, Lecture Notes in Computer Science, vol 4455, pp 244–259, doi: 10.1007/978-3-540-73847-3_26

Tamás Horváth, Jan Ramon (2010) Efficient frequent connected subgraph mining in graphs of bounded tree-width. Theoretical Computer Science 411(31–33):2784–2797, doi: 10.1016/j.tcs.2010.03.030

Pascal Welke, Tamás Horváth, Stefan Wrobel (2015) On the complexity of frequent subtree mining in very simple structures. In: Jesse Davis, Jan Ramon (eds) Inductive Logic Programming (ILP) Revised Selected Papers, Springer, Lecture Notes in Computer Science, vol 9046, pp 194–209, doi: 10.1007/978-3-319-23708-4_14

References II

Pascal Welke, Tamás Horváth, Stefan Wrobel (2016b) Min-hashing for probabilistic frequent subtree feature spaces. In: Toon Calders, Michelangelo Ceci, Donato Malerba (eds) Discovery Science (DS) Proceedings, Lecture Notes in Computer Science, vol 9956, pp 67–82, doi: 10.1007/978-3-319-46307-0_5

Pascal Welke, Tamás Horváth, Stefan Wrobel (2016a) Probabilistic frequent subtree kernels. In: Michelangelo Ceci, Corrado Loglisci, Giuseppe Manco, Elio Masciari, Zbigniew Ras (eds) New Frontiers in Mining Complex Patterns (NFMCP) Revised Selected Papers, Springer, Lecture Notes in Computer Science, vol 9607, pp 179–193, doi: 10.1007/978-3-319-39315-5_12

Pascal Welke (2017) Simple necessary conditions for the existence of a Hamiltonian path with applications to cactus graphs. CoRR abs/1709.01367, URL http://arxiv.org/abs/1709.01367

Pascal Welke, Tamás Horváth, Stefan Wrobel (2018) Probabilistic frequent subtrees for efficient graph classification and retrieval. Machine Learning 107(11):1847–1873, doi: 10.1007/s10994-017-5688-7

Pascal Welke, Tamás Horváth, Stefan Wrobel (2019) Probabilistic and exact frequent subtree mining in graphs beyond forests. Machine Learning doi: 10.1007/s10994-019-05779-1, (online)