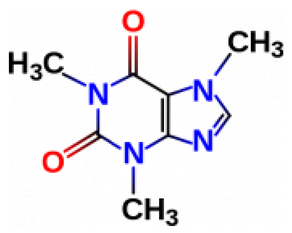
EmbAssi: Embedding Assignment Costs for Similarity Search in Large Graph Databases

Franka Bause[1], Erich Schubert[2], Nils M. Kriege[1]

[1]University of Vienna [2]TU Dortmund University

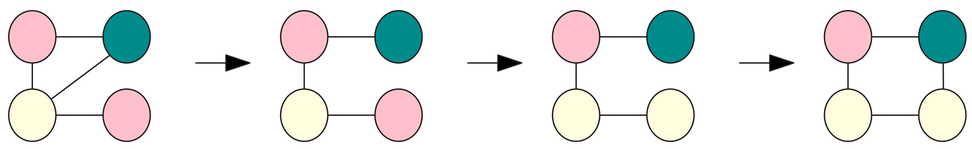
Searching in Large Graph Databases

Goal: Find graphs similar to a query graph efficiently



Graph Edit Distance (GED)

- **Distance measure** for **graphs**
- Transform one graph into the other (**edit path**)
- Cost of transformation $\hat{=}$ GED



- **NP-hard** problem

Notation

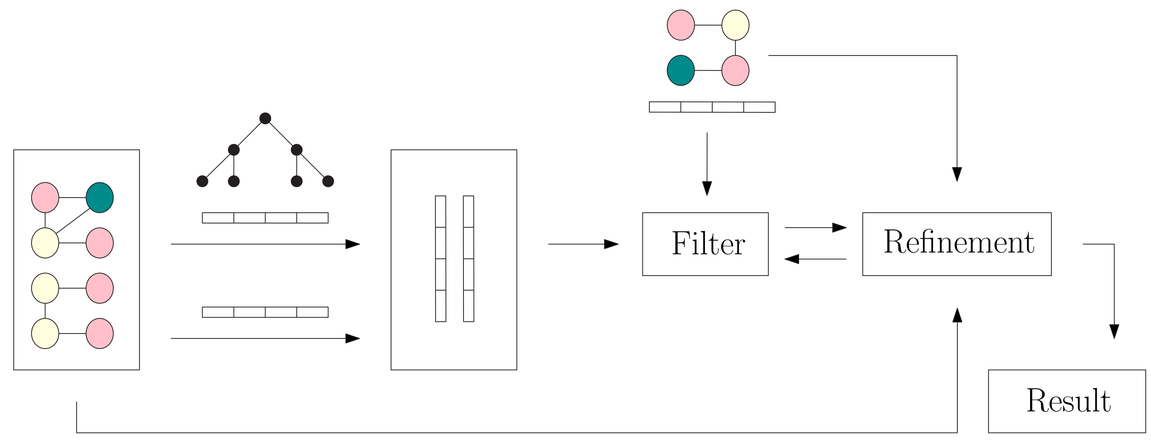
Cost of Edit Operations:

c_v : delete/insert vertex
c_e : delete/insert edge
c_{vl} : change label of vertex
c_{el} : change label of edge

Our Contribution

Tight lower bounds for efficient filtering

- **Efficient** lower bounds for the graph edit distance based on **tree metrics**
- Embeddable to $\ell_1 \rightarrow$ **index for vectors** can be used
- Suitable for **similarity search**
- Lower bound for generating **initial candidates**

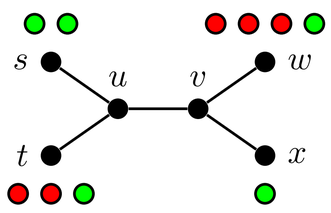


Lower Bounds Based on Optimal Assignments

- Compute **optimal assignment** between vertices
- Cost of assignment $\hat{=}$ lower bound for GED
- Embeddable to ℓ_1 if cost function is **tree metric**

Embedding Assignment Costs

- Cost function has to be a **tree metric**
- Representable as **tree** T with cost $\hat{=}$ length of path

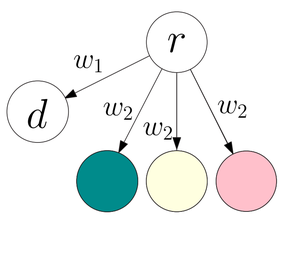


	A	B
$\overleftarrow{su}$	0	2
$\overleftarrow{tu}$	2	1
$\overleftarrow{uv}$	2	3
$\overleftarrow{vw}$	2	4
$\overleftarrow{vx}$	5	4

- $S_{\overleftarrow{uv}}$: $\#s \in S$ associated with nodes in subtree containing u when deleting uv
- Cost of optimal assignment: $\sum_{uv \in E(T)} \mid A_{\overleftarrow{uv}} - B_{\overleftarrow{uv}} \mid \cdot w(uv)$
- $\phi(S) = [S_{\overleftarrow{uv}} \cdot w(uv)]_{uv \in E(T)}$

Label Lower Bound (LLB)

- Minimum **vertex relabelings/deletions**
- Edge weights:
  $w_1 = c_v - 0.5 \cdot c_{vl} \quad w_2 = 0.5 \cdot c_{vl}$

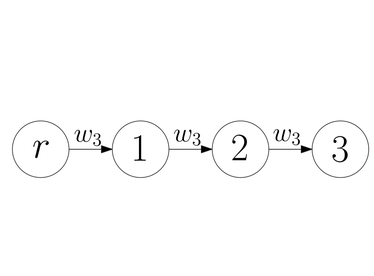


$\phi(G_1)$
$0 \cdot w_1$
$1 \cdot w_2$
$1 \cdot w_2$
$2 \cdot w_2$

$\phi(G_2)$
$0 \cdot w_1$
$1 \cdot w_2$
$2 \cdot w_2$
$1 \cdot w_2$

Degree Lower Bound (DLB)

- Minimum **edge insertions/deletions** based on vertex degrees
- Edge weights: $w_3 = 0.5 \cdot c_e$
- Each operation influences degree of **two** vertices



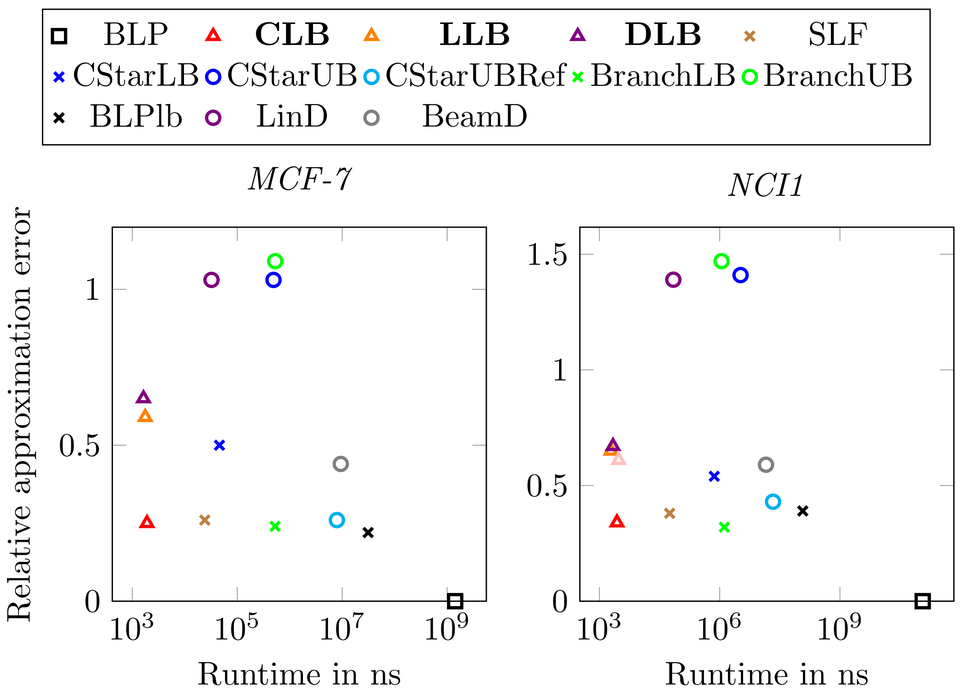
$\phi(G_1)$
$4 \cdot w_3$
$3 \cdot w_3$
$1 \cdot w_3$

$\phi(G_2)$
$4 \cdot w_3$
$4 \cdot w_3$
$0 \cdot w_3$

Combined Lower Bound (CLB)

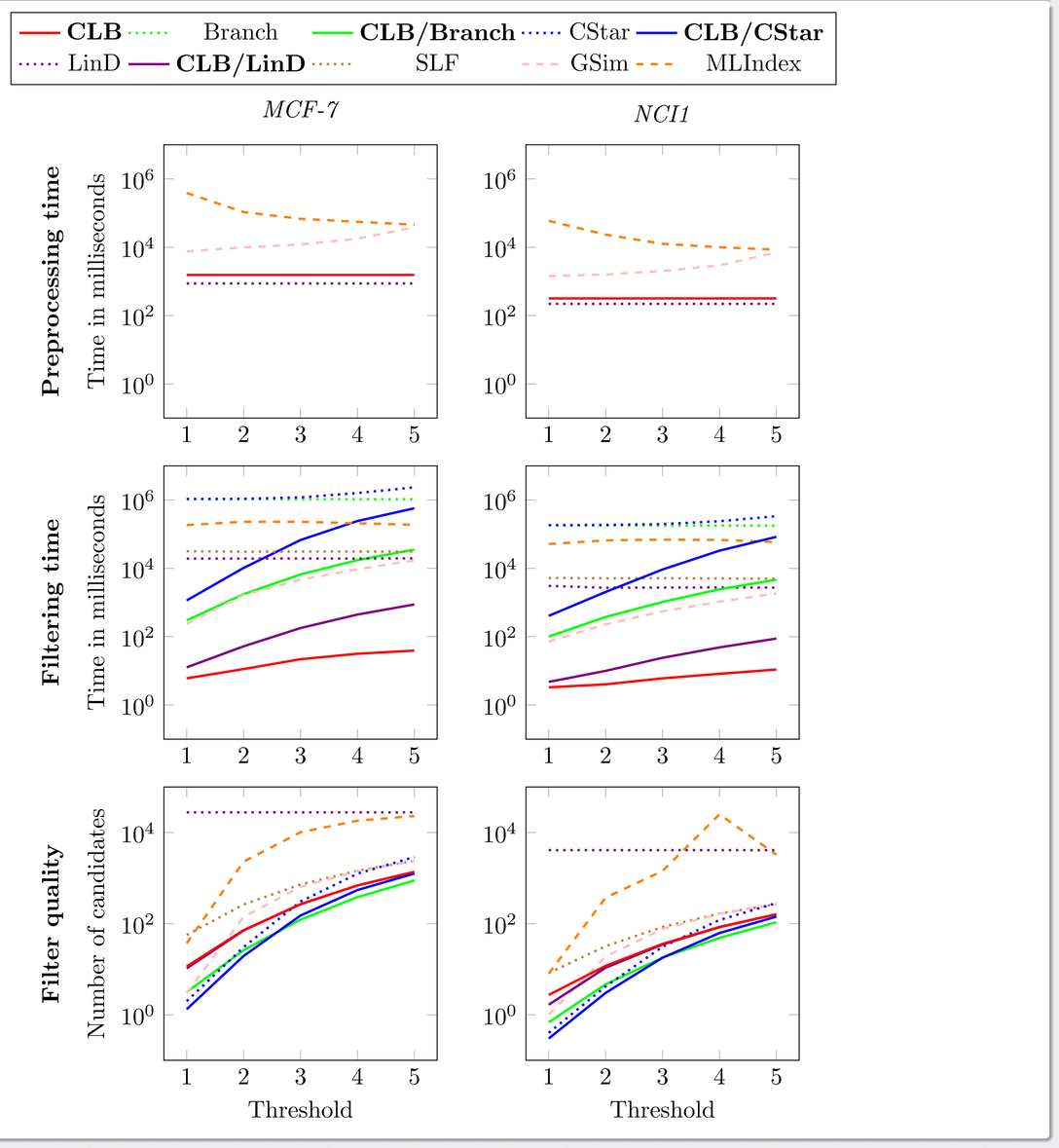
- **Combines** *LLB* and *DLB*
- $d_{CLB}(G_1, G_2) = d_{LLB}(G_1, G_2) + d_{DLB}(G_1, G_2)$

Comparison of Approximation Quality



△: our approximation x: other lower bound ○: upper bound

Comparison to State-of-the-Art



Conclusion

- **Efficiently** computable **tight** lower bounds
- **Scalable** to databases with millions of graphs
- Optimization of runtime (choice of edge direction)
- kNN and range queries both supported
- **Speed up state-of-the-art** approaches through pre-filtering