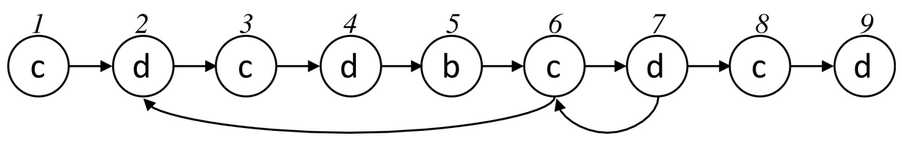
PADERBORN UNIVERSITY

STEFAN BÖTTCHER, RITA HARTEL, SVEN PEETERS

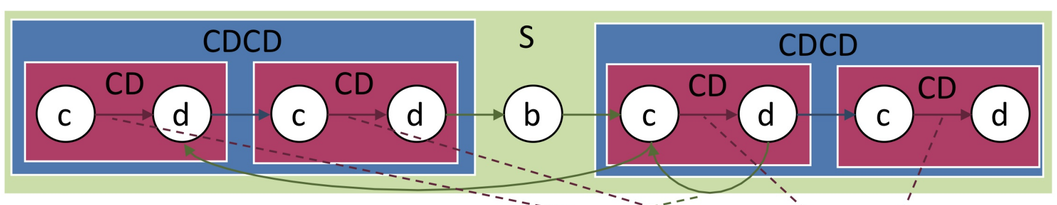
PATTERN SEARCH IN GRAMMAR-COMPRESSED GRAPHS

Example

Original Graph:

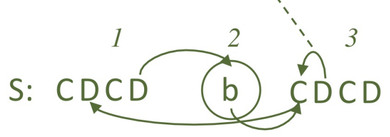


Generation of Graph Grammar from Graph:

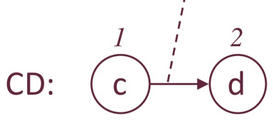


Graph Grammar:

Edge: (S/3:CDCD/1:CD/2:d, S/3:CDCD/1:CD/1:c)



Edge: (CD/1:c, CD/2:d)



Further Edges:

(S/1:CDCD/2:CD/2:d, S/2:b)
(S/2:b, S/3:CDCD/1:CD/1:c)
(S/3:CDCD/1:CD/1:c, S/1:CDCD/1:CD/2:d)
(CDCD/1:CD/2:d, CDCD/2:CD/1:c)

Main Idea

Grammar-based graph compression:

- Repeatedly use a new nonterminal to replace each occurrence of a connected subgraph that occurs multiple times with a node labeled with this nonterminal

- Single nodes of the original graph can be adressed within the grammar by **Grammar Paths**:
  - Node with ID 7 can be adressed by the Grammar Path S/3:CDCD/1:CD/2:d, meaning 3rd node in S, 1st node in CDCD, 2nd node in CD

- Sets of nodes within the original graph that are represented by a single grammar node can be adressed by a **Grammar Path Suffix**:
  - The Grammar Path Suffix CD/2:d adresses the nodes with IDs 2, 4, 7, and 9 of the original graph

Improved Pattern search for Grammar-compressed Graphs:

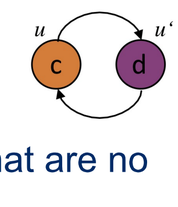
- Predecessor search on large node sets (within the original graph) is replaced by predecessor search on smaller sets of Grammar Path Suffixes (within the compressed grammar).
- Pattern simulation based on predecessor search is significantly accelerated.

Pattern Search

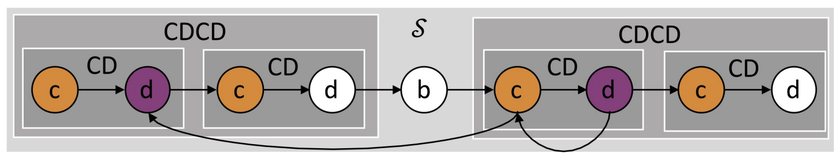
Idea: Iteratively sharpen sets of candidate Grammar Path Suffixes by computing predecessors on sets of Grammar Path Suffixes (small) instead of on sets of nodes (large).

Initially: sim[u] = {c}; sim[u']={d}

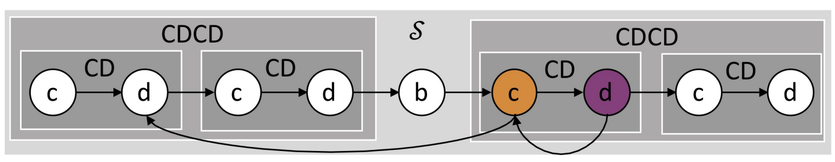
Pattern:



Step: Exclude all Grammar Path Suffixes from sim[u'] that are no predecessors of a Grammar Path Suffix of sim[u].
Whenever necessary, Grammar Path Suffixes have to be split:
new sim[u'] = {CDCD/1:CD/2:d}



Finally (intermediate steps omitted): A fixed-point is reached.
sim[u] = {S/3:CDCD/1:CD/1:c}; sim[u'] = {S/3:CDCD/1:CD/2:d}

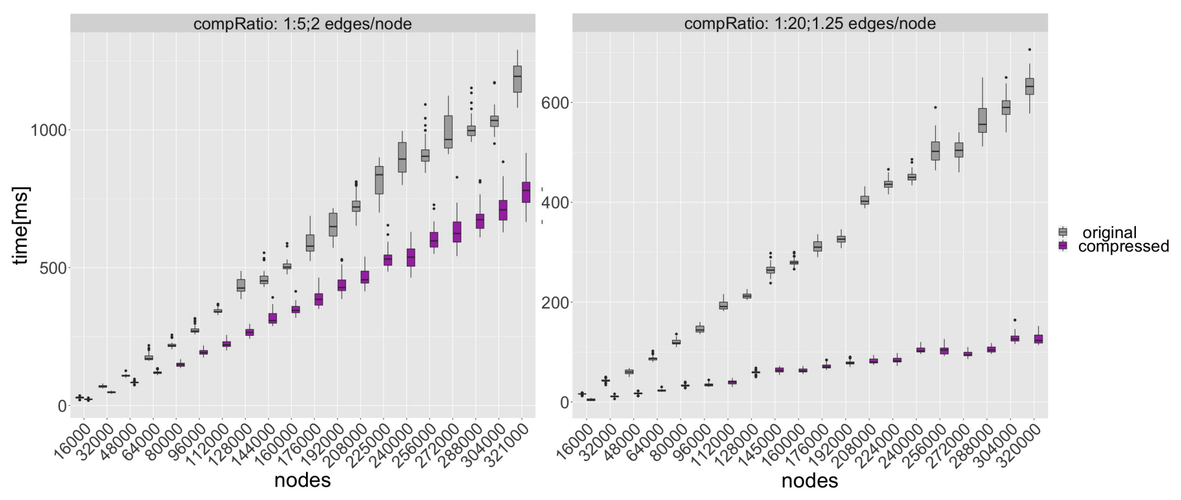


Results

- Search on compressed graphs outperforms search on original graph
- The bigger the graphs, the stronger the benefit

Random graphs:

- The stronger the compression, the bigger the benefit



LDBC Benchmark and RDF graphs (dbpedia):

- The more complex the patterns, the stronger the benefit

Stefan Böttcher
stb@uni-paderborn.de

Rita Hartel
rst@uni-paderborn.de

Sven Peeters
speeters@campus.uni-paderborn.de

Paper:

SCAN ME