Scaling Up Distance Labeling on Graphs with Core-Periphery Properties

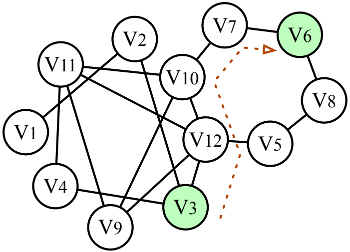
Wentao Li, Miao Qiao, Lu Qin, Ying Zhang,
Lijun Chang, and Xuemin Lin

University of Technology Sydney, University of Auckland,
The University of Sydney, The University of New South Wales

SIGMOD 2020

Problem

Given a graph $G(V,E)$ and two nodes $s,t \in V$, report the length of the shortest path from s to t



Solutions

- Online search such as Breadth-first search (BFS)
- Index-based approaches such as 2-hop labeling
  - Each node v has a label L_v
    - » $L_{v_3} = \{v_1 : 2, v_2 : 1, v_3 : 0\}$
  - Label size is denoted as $|L_v|$, e.g., $|L_{v_3}| = 3$
  - Index size is computed as $\Sigma_{v \in V}|L_v|$

Observations on Index Size

Graphs with larger # nodes/edges do not mean larger index

- DBLP ($n = 1.3M$) with $51G$ index
- INDO ($n = 7.4M$) with $17.7G$ index

Index of social networks are normally larger than road networks

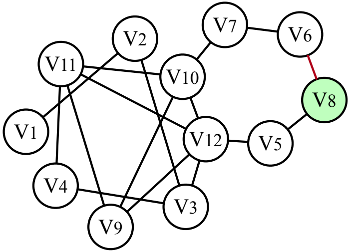
- BELG ($n = 1.4M$) with $4.4G$ index

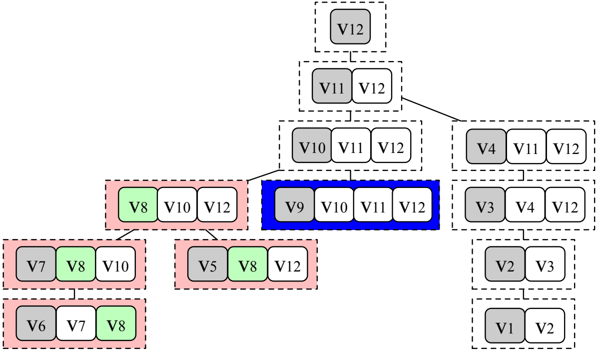
Index Size vs. Treewidth

Explanations from Treewidth

A tree decomposition T of a graph G is to convert G into a tree, where each tree vertex (i.e., bag) contains a subset of nodes in G

- Each node appears in at least one bag
- Each edge is contained in at least one bag
- Each node induces a subtree





- The width of a tree decomposition T is its maximal bag size, e.g. width $= 4$
- The treewidth of G is the minimum width over all tree decompositions of G

Explanations from Treewidth

The treewidth of G is the minimum width over all tree decompositions

Contribution #1

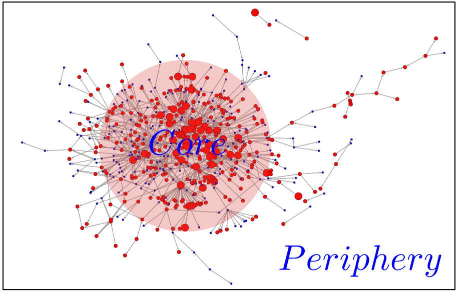
Given a graph with n nodes and treewidth tw, the index size generated by the best 2-hop labeling algorithm is $\tilde{\Theta}(n \cdot tw)$ in the worst-case.

- 2-hop labeling properly handles graphs with relative low treewidth, e.g., road networks
- 2-hop labeling *fails* for the oversized index on graphs with relatively large treewidth, e.g., social networks and web graphs
  - The index size on UK07 exceeds 500G

Core Periphery Structure

2-hop labeling fails for the oversized index on graphs with relatively large treewidth.
The core-periphery structure has been identified in real graphs

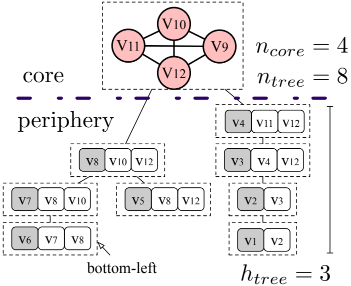
- A densely connected core
- The other nodes, called the periphery, are of limited connectivity



Core Tree Decomposition and Our Contributions

Core Tree Decomposition[1] is a tree decomposition with parameter d

- One big bag for core part (with bag size $> d$)
  - n_{core} be the number of nodes (in G) that appear in the core
- Many small bags for periphery part (with bag size $\leq d$)
  - n_{tree} be the number of bags in the periphery
  - h_{tree} be the maximum height of trees in the periphery
  - w be the width of tree decomposition after decomposing the core



Core Tree Decomposition ($d = 3$)

Contribution #2 (Index Size)

$\tilde{O}((n_{core} + n_{tree}) \cdot w) \rightarrow \tilde{O}(n_{core} \cdot w) + O(n_{tree} \cdot (d + h_{tree}))$

- 2-hop labeling on core
- Tree index on periphery ($w \rightarrow d + h_{tree}$)

Contribution #3 (Tree Index Time)

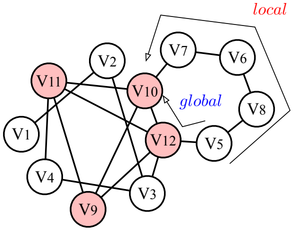
$O(n_{tree} \cdot m) \rightarrow O(n_{tree} \cdot d(d + h_{tree}))$

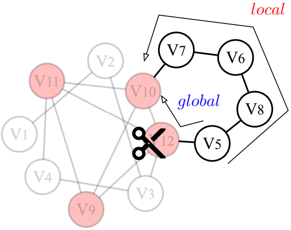
- n_{tree} BFSs, each with cost $O(m)$

[1] Takuya, Akiba, et al. Shortest-path queries for complex networks: exploiting low tree-width outside the core

Local Distances

- Local distance from s to t is the minimized length of paths from s to t not via any node in the core
  - Path # 1, $< v_5, v_{12}, v_{10} >$, contains v_{12} in the core ☹
  - Path # 2, $< v_5, v_8, v_6, v_7, v_{10} >$, does not via nodes in the core ☺
- To *avoid* exploring the whole graph to compute the distances
- Local distances are sufficient for efficiently computing distance queries





Experiments

Algorithms

- PSL^+ and PSL^* (2-hop labeling[2])
- CT-Index, the proposed algorithm ($d = 100$)

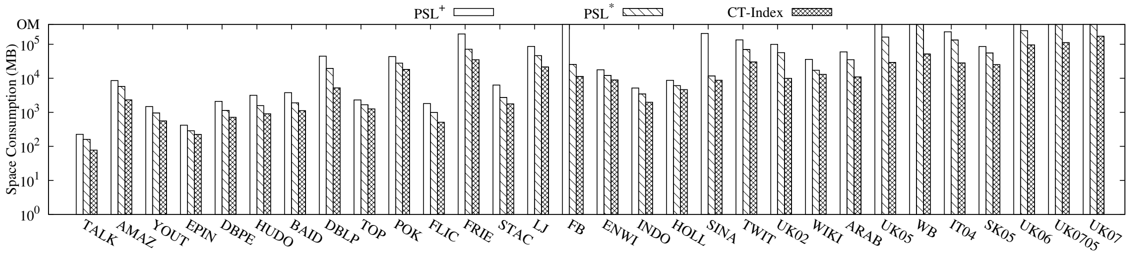
Dataset

- 30 real graphs
  - Including social networks, web graphs, coauthorship graphs, communication networks, and interaction networks
- The largest graph has over 5.5 billion edges

[2] Wentao, Li, et al. Scaling distance labeling on small-world networks

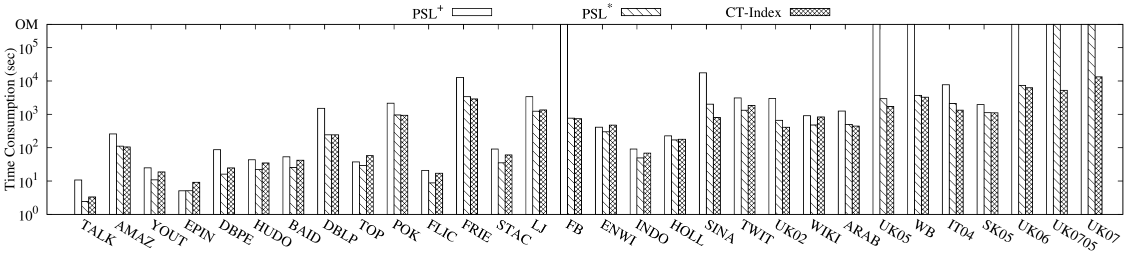
The Comparison of the Index Size

- CT-Index can index massive graphs such as UK0705 and UK07
- CT-Index vs. PSL^+: reduces 4.79 on average, 23.72 at a maximum
- CT-Index vs. PSL^*: reduces 2.31 on average, 5.66 at a maximum

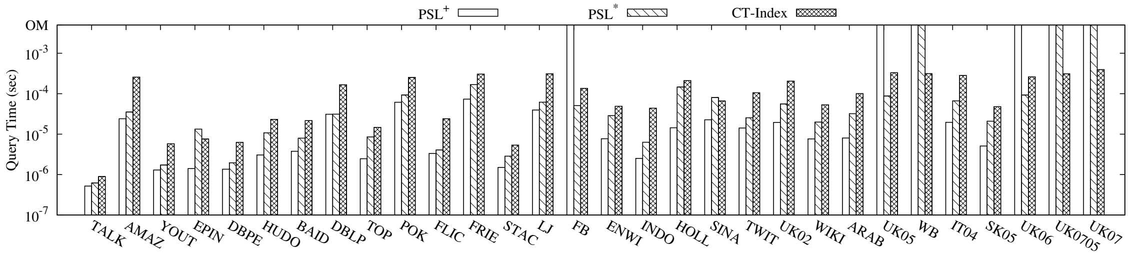


The Comparison of the Index Time

- CT-Index shortens the index time on most graphs
- CT-Index vs. PSL$^+$: reduces 3.26 on average, 21.85 at a maximum
- CT-Index vs. PSL*: reduces 1.68 on average, 4.64 at a maximum



- CT-Index vs. PSL^{+}: 7.55 times slower on average
- CT-Index vs. PSL^{*}: 3.17 times slower on average
- Below 0.4 milliseconds including on UK07 with 5.5 billion edges



Summary

- Limitation of 2-hop labeling on graphs with relative high treewidth
- Core tree decomposition for smaller index size
- Local distances for efficient tree index construction

Thank You