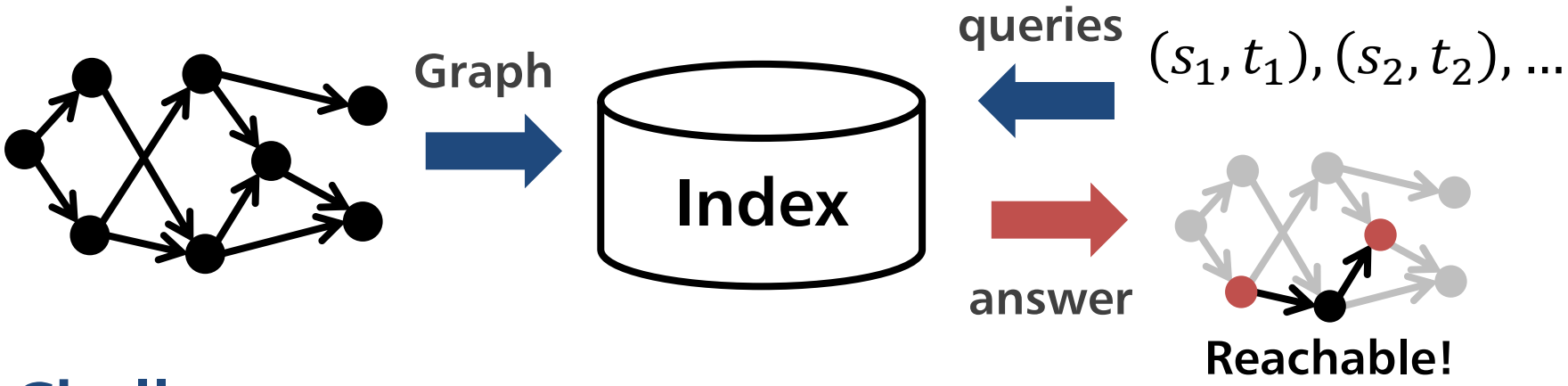
Fast and Scalable Reachability Queries on Graphs by Pruned Labeling with Landmarks and Paths (CIKM 2013)

Yosuke Yano[1] Takuya Akiba[1]
Yoichi Iwata[1] Yuichi Yoshida[2,3]

1. The University of Tokyo
2. National Institute of Informatics 3. Preferred Infrastructure Inc.

Problem Definition

Given a directed graph G, construct an index to answer **reachability queries** asking if there is a path between two vertices



Challenge

Trade-off between **Query Time** and **Scalability**

- Aim to achieve **fast query time** on **large graphs**

Application

Used as a building block for more complex task

- Network Analysis
  - Biology, Web
- Source Code Analysis
- XML Query Engines (SPARQL, XQuery)

Related Work

Many methods on reachability queries

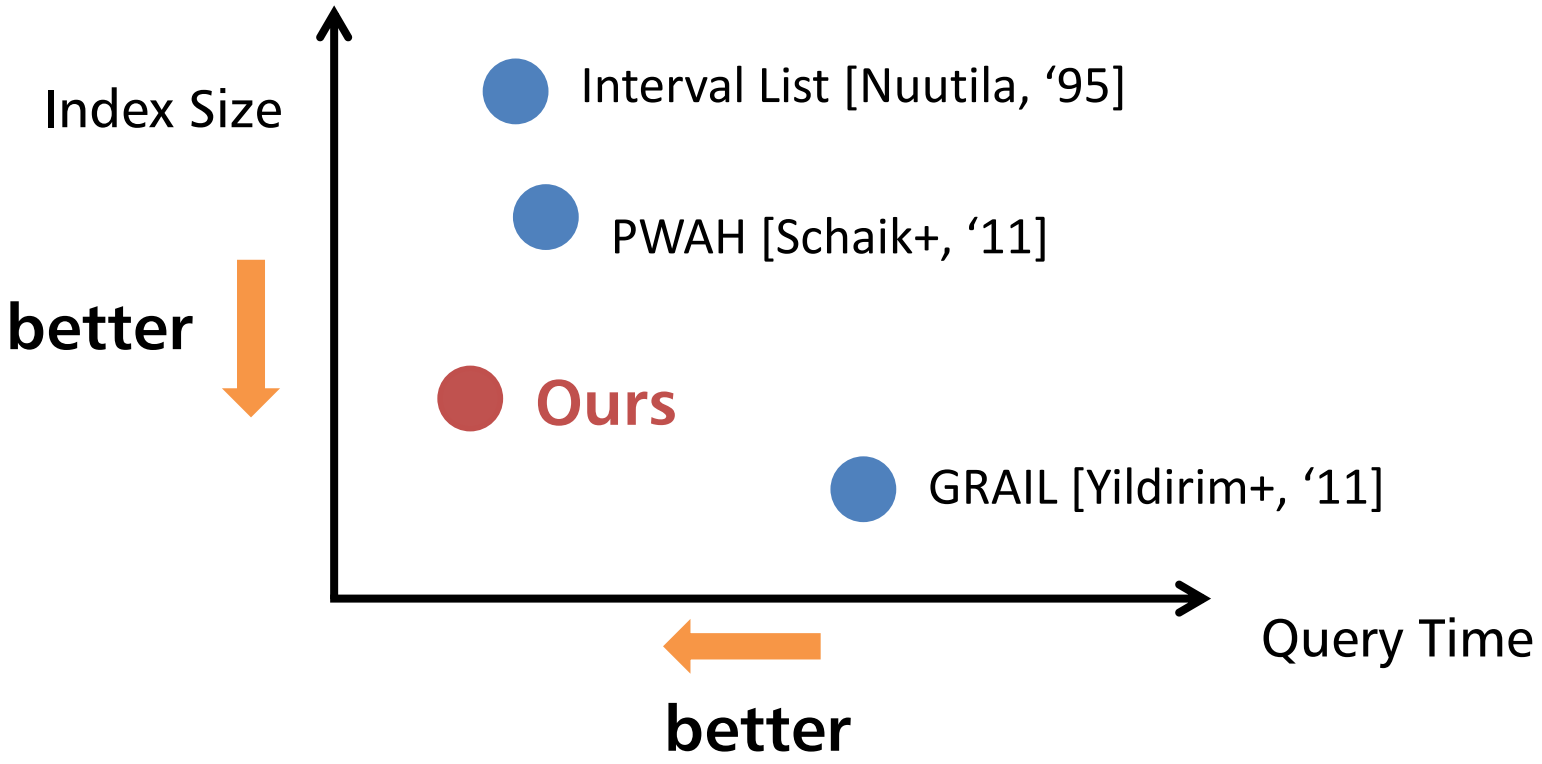
- GRAIL [Yildirim+ ’11]
- Interval List & PWAH [Schaik+ ’11]
- SCARAB [Jin+ ’12]
- TF-Label [Cheng+ ’13]
- …
- SCISSOR [Mullangi+, CIKM’13]

However, creating an index for **large-scale graphs** is still a challenging task

Our Contributions

Propose two new methods for reachability

- **Fast Query Time** (within 1μs per query)
- **Moderate Index Size** (less than 1GB)
- **High Scalability** (on graphs with 10M- edges)

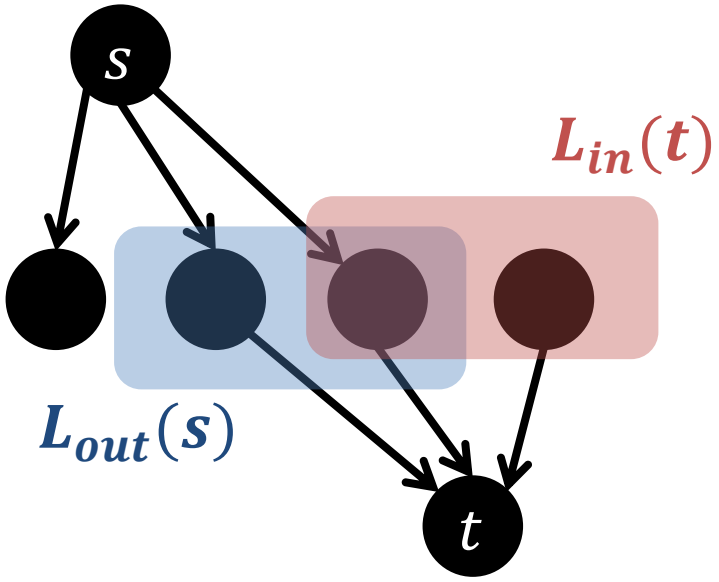


Data Structure & Querying: 2-hop [Cohen+ '02]

Each vertex keeps two sets L_{in} and L_{out}

$L_{in}(v)$: part of vertices that can reach v

$L_{out}(v)$: part of vertices that can be reached from v



Querying

s can reach $t \Leftrightarrow$
$L_{out}(s) \cap L_{in}(t) \neq \emptyset$

Create L_{in} and L_{out} which can answer **any query correctly**

Pruned Landmark Labeling (PLL)

PLL for
Shortest Path Queries
[Akiba, Iwata, Yoshida, '13]

- The **first** practical method for making a 2-hop index on large-scale graphs

Target Network:
Undirected Complex Network

PLL for
Reachability Queries
[This paper]

- Specialized for reachability queries on directed graphs

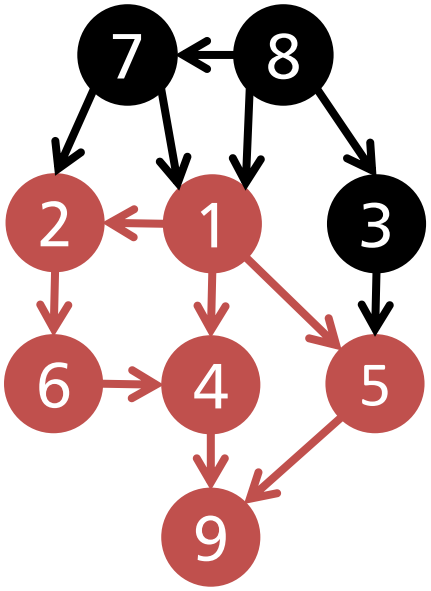
Target Network:
not specified

Different Queries in a **Unified Approach**

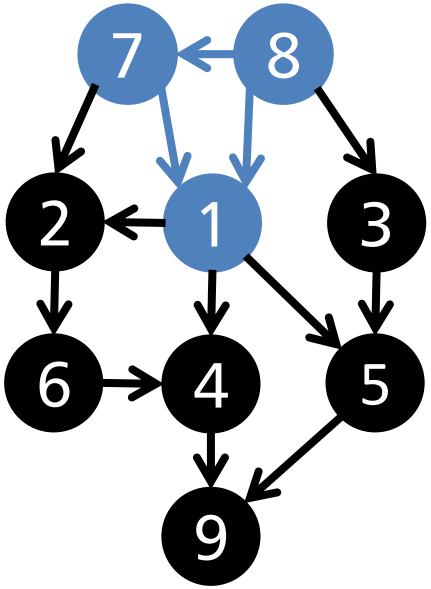
Pruned Landmark Labeling (PLL)

Indexing Algorithm

- Give a certain heuristic order to the vertices
- Conduct **pruned** BFSs from vertex 1



Red vertices can be reached from v1

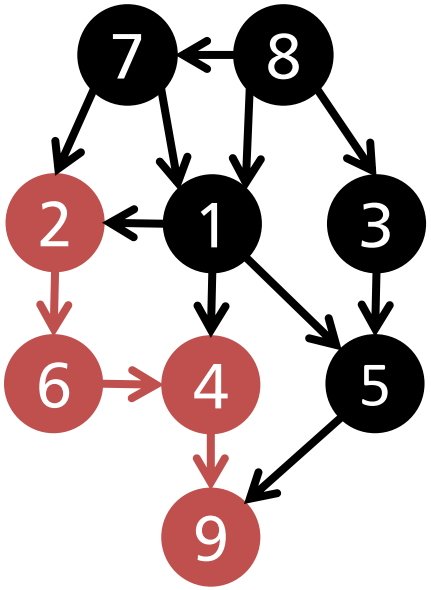


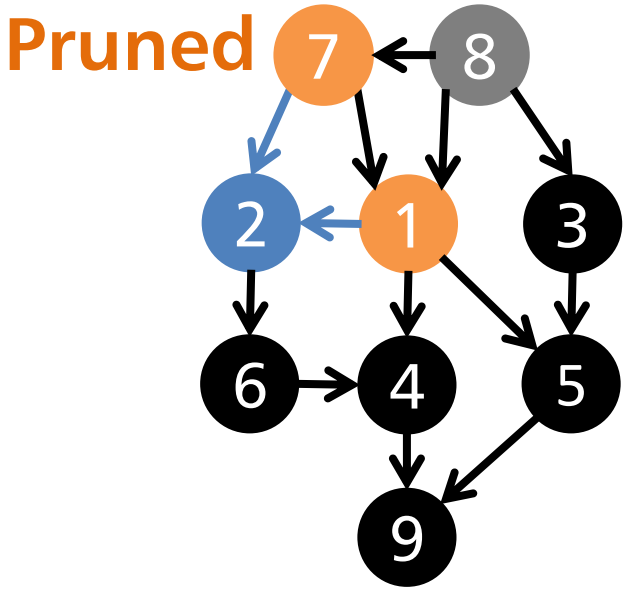
Blue vertices can reach v1

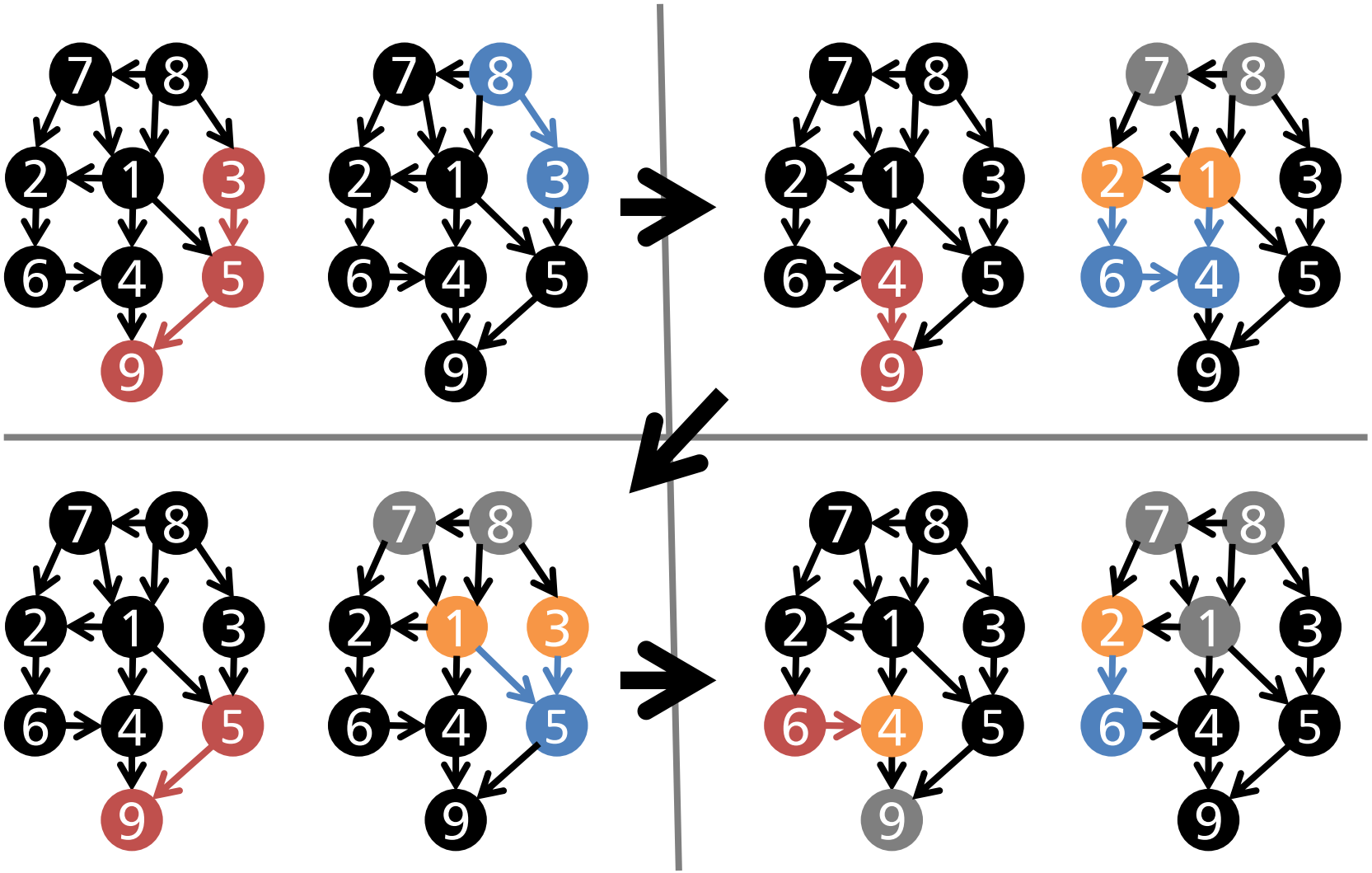
Pruned Landmark Labeling (PLL)

Indexing Algorithm

- Similarly conduct pruned BFSs from vertex 2
- We can prune vertex 1 and 7 in the example
  - Since we know that 7 can reach 2 from index added in BFSs from vertex 1







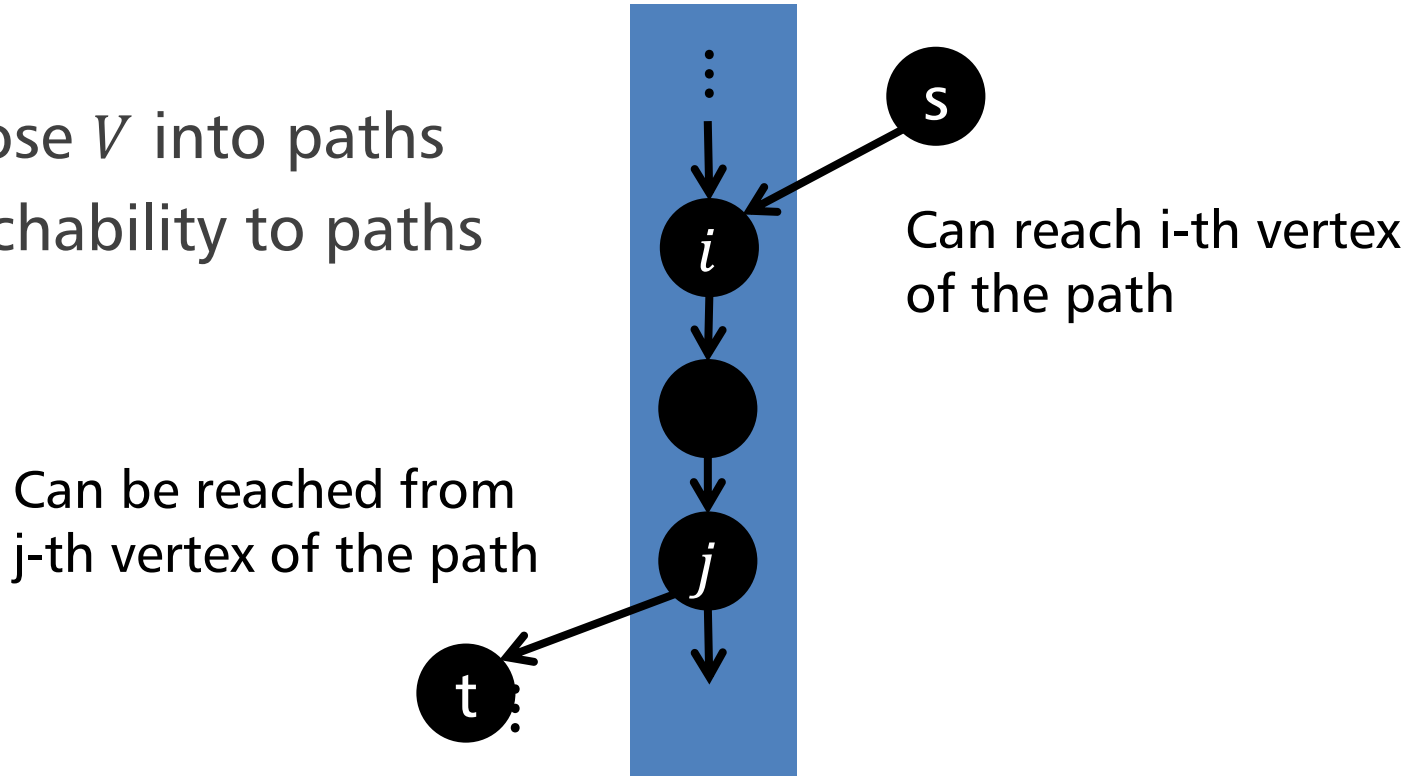
Intuitively speaking, search space **quickly gets smaller** **by pruning** as the algorithm progresses

Pruned Path Labeling (PPL)

On **3-hop cover** framework [Jin+ '09] similar to 2-hop cover, we can also construct an index by using pruned BFS

Idea

- Decompose V into paths
- Keep reachability to paths

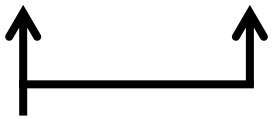


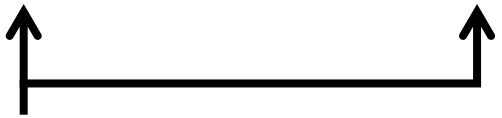
Results: Average Query Time (μs)

GRAIL (Yildirim+ '12), **IL** (Schaik+ '11), **PWAH** (Schaik+ '11)

Dataset	PLL	PPL	GRAIL	IL	PWAH
ff/successors	**0.085**	0.133	0.279	0.154	0.202
uniprot22m	**0.083**	0.122	0.403	0.173	0.243
uniprot100m	**0.133**	0.197	0.743	0.292	0.361
uniprot150m	**0.153**	0.223	0.776	0.248	0.351
citeseerx	0.124	0.164	**27.946**	**0.103**	0.214
cit-patents	**0.253**	0.296	**11.591**	0.292	**15.451**
go-uniprot	**0.156**	0.194	0.520	0.233	0.521

Graphs with **10M- edges**





1st and 2nd in query time on almost all datasets

Very slow on some datasets

Results: Index Size (MB)

Better than PLL on one dataset

Dataset	PLL	PPL	GRAIL	IL	PWAH
ff/successors	122.3	91.6	29.7	40.0	34.1
uniprot22m	19.4	19.4	25.5	19.6	19.5
uniprot100m	206.8	206.8	257.4	223.0	218.8
uniprot150m	334.0	334.0	400.6	373.8	366.2
citeseerx	122.0	126.7	104.6	441.3	156.0
cit-patents	664.6	691.2	60.4	22444	5593
go-uniprot	263.1	273.5	111.5	792.7	255.9

Moderate index size

Index size exploded on one large dataset

Conclusions

- Proposed new simple methods for constructing reachability index on 2-hop & 3-hop frameworks
  - Conducting BFSs with **pruning** by index created so far
- Showed that our methods attained **good trade-offs** between query time and scalability
  - **Fast query time**
  - **In moderate index size**
  - **On large-scale graphs**
- In our paper, we also analyzed our algorithms using graph minor theory