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

Yosuke Yano¹  Takuya Akiba¹
Yoichi Iwata¹  Yuichi Yoshida²,³

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

• Network Analysis
  – Biology, Web

• Source Code Analysis

• XML Query Engines (SPARQL, XQuery)
Related Work

Many methods on reachabilility 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 $1\mu s$ per query)
- **Moderate Index Size** (less than 1GB)
- **High Scalability** (on graphs with 10M-edges)
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 $t$ $\iff$ $L_{out}(s) \cap L_{in}(t) \neq \emptyset$

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

![Diagram](image)

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

![Diagram showing path labeling](image)
## Results: Average Query Time ($\mu s$)

GRAIL (Yildirim+ ‘12), IL (Schaik+ ‘11), PWAH (Schaik+ ‘11)

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

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

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

- **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