Efficient Top-$k$ Shortest-Path Distance Queries on Large Networks by Pruned Landmark Labeling with Application to Network Structure Prediction

Takuya Akiba \textit{U Tokyo}
Takanori Hayashi \textit{U Tokyo}
Nozomi Nori \textit{Kyoto U}
Yoichi Iwata \textit{U Tokyo}
Yuichi Yoshida \textit{NII & PFI}
Proximity Measures on Networks

Strength of relevance between vertex pairs

(B, A) > (A, C)
Distance as a proximity measure

Applications of distance as the proximity measure

- Context-Aware Search [CIKM’08][CIKM’09]
- Socially-Sensitive Search [CIKM’07][VLDB’08][CIKM’13]

Efficient indexing methods

- Landmark [CIKM’09] [WSDM’10] [CIKM’10] [ICDE’12]
- TD [SIGMOD’10] [EDBT’12]
- 2-Hop [SIGMOD’12] [ESA’12] [SIGMOD’13]
Graph Indexing Methods

Given a graph $G = (V, E)$
1. construct a data structure (an index) to
2. answer queries quickly
Distance as a proximity measure

**Advantage:** Scalable and fast

**Drawback:** Poor expressiveness
- Unweighted graphs
- Small-world phenomenon

Distances are small integers
<table>
<thead>
<tr>
<th>Graph</th>
<th>Distance</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td><img src="image" alt="Graph (a)" /></td>
</tr>
<tr>
<td>(b)</td>
<td><img src="image" alt="Graph (b)" /></td>
</tr>
<tr>
<td>(c)</td>
<td><img src="image" alt="Graph (c)" /></td>
</tr>
</tbody>
</table>
As a proximity measure, we propose to use **Top-$k$ distances**

Defined as:

Lengths of $k$ shortest paths

- $k = 1$ → standard distance
- Loops are allowed
<table>
<thead>
<tr>
<th>Graph</th>
<th>Distance</th>
<th>Top-(k) Distances</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a)</td>
<td>4</td>
<td>[4 6 6 6 6 8 8 ...]</td>
</tr>
<tr>
<td>(b)</td>
<td>4</td>
<td>[4 4 4 6 6 6 6 ...]</td>
</tr>
<tr>
<td>(c)</td>
<td>4</td>
<td>[4 4 4 4 4 4 4 ...]</td>
</tr>
<tr>
<td>Graph</td>
<td>Distance</td>
<td>Top-$k$ Distances</td>
</tr>
<tr>
<td>-------</td>
<td>----------</td>
<td>------------------</td>
</tr>
<tr>
<td>(a)</td>
<td>4</td>
<td>[4 6 6 6 6 8 8 ...]</td>
</tr>
<tr>
<td>(b)</td>
<td>4</td>
<td>[4 4 4 6 6 6 6 ...]</td>
</tr>
<tr>
<td>(c)</td>
<td>4</td>
<td>[4 4 4 4 4 4 4 4 ...]</td>
</tr>
</tbody>
</table>
Top-$k$ Distances as a proximity measure?

Top-$k$ distances are vectors of dimension $k$

Not scalar
direct substitution impossible
Top-$k$ Distances as a proximity measure?

Top-$k$ distances are vectors of dimension $k$

**as feature vectors, nice with Machine Learning**

\[
\begin{align*}
[4 6 6 6 6 8 8 \ldots] & \quad \rightarrow \quad \text{SVM} \quad \rightarrow \quad \text{Yes} \\
[4 4 4 6 6 6 6 \ldots] & \quad \rightarrow \quad \text{SVM} \quad \rightarrow \quad \text{No} \\
[4 4 4 4 4 4 4 4 \ldots] & \quad \rightarrow \quad \text{SVM} \quad \rightarrow \quad \text{Yes}
\end{align*}
\]
### Previous Algorithms for Top-\(k\) Distances

**BFS-based naïve algorithm** [folklore]

- \(O((n + m)k)\) time

**Eppstein’s algorithm** [Eppstein, 1998]

- \(O(n + m + k)\) time, but slow in practice for moderate \(k\)

In contrast to many indexing methods for standard distance, no indexing methods have been proposed.

- We are to evaluate many pairs of vertices
- Proportional to graph size \(\rightarrow\) not scalable
Contributions at a glance

Contribution 1
An indexing method for top-$k$ distance queries
- Based on pruned labeling [Akiba+, SIGMOD’13]
- Simple, fast and scalable

Enables

Contribution 2
Top-$k$ distances as a proximity measure
- Suited to machine learning
- Empirical study on link prediction
Indexing Method for Top-$k$ Distance Queries

Part 2: Algorithm
Trivial extension? → No! 😞
from standard distance queries?

Main new challenge

- Carefully avoiding double-counts
  The same path may come differently

Naïve approach may count a path twice or more
2-Hop Cover for distance queries [Cohen+, 2002]

Data Structure: a label for each vertex $v$

**Label $L(v)$**
- $L(v) = [(u_1, \delta_1), (u_2, \delta_2), ...]$  
- $\delta_i = d(v, u_i)$
2-Hop Cover for distance queries [Cohen+, 2002]

Data Structure: a label for each vertex $v$

**Label** $L(v)$
- $L(v) = [(u_1, \delta_1), (u_2, \delta_2), ...]$
- $\delta_i = d(v, u_i)$

**Query: 2-hop paths using labels**

$$\min_{u \in L(s) \cap L(t)} d(s, u) + d_G(u, t)$$
Top-\(k\) 2-Hop Cover (this work)

Data structure

**Distance Label** \(L(v)\)
- \(L(v) = [(u_1, \delta_1), (u_2, \delta_2), ...]\)
- \(\delta_i = d^{> \nu}_{j-th}(v, u_i)\)

**Loop Label** \(C(v)\)
- \(C(v) = [\delta_1, \delta_2, \delta_3, ...]\)
- \(\delta_i = d^{\geq \nu}_{i-th}(v, v)\)

\(d^{> \nu}_{j-th}(v, u_i)\) is restricted distance
Top-$k$ 2-Hop Cover (this work)

Data structure

Distance Label $L(v)$ + Loop Label $C(v)$

Query: 3-hop paths

$L(s)$ $C(u)$ $L(t)$
Indexing Algorithm

**Challenge** on computing labels

- **Correctness** *(Exactness)*
- **Sizes of labels** *(Index Size & Query Time)*
- **Efficiency** *(Scalability)*

Algorithm based on pruned labeling

[Akiba-Iwata-Yoshida, SIGMOD’13]

+ Performance improving techniques
Experimental Evaluation

*Part 3: Experiments*
Indexing Time Experimental Results

< 1 hour on graphs with ten million edges

$k = 8$, Intel Xeon X5670 (2.93 GHz), 48 GB Memory, C++
Index Size Experimental Results

![Graph showing index size vs graph size for social networks and web graphs.](image)

- **Index size [MB]**
- **Graph size |E||

- **Social networks**
- **Web graphs**

\[ \text{Index size} \lt 10 \text{ GB on graphs with ten million edges} \]

\[ k = 8, \text{Intel Xeon X5670 (2.93 GHz), 48 GB Memory, C++} \]
Query Time Experimental Results

Graph size $|E|$ vs. Query time [ms]

- **Proposed**
- **Naive**
- **Eppstein**

**Consistently <0.1 ms**

$k = 8$, Intel Xeon X5670 (2.93 GHz), 48 GB Memory, C++

10^6 X faster
Application to Link Prediction

Part 3: Experiments
Link Prediction Problem [Liben-Nowell+, 2003]

Methods

- 7 Baseline Methods
- Top-$k$ Distances + SVM

Online Social Network
Protein Interaction
### Application to Link Prediction

<table>
<thead>
<tr>
<th>Dataset</th>
<th>CN</th>
<th>Jaccard</th>
<th>Adamic</th>
<th>Pref.</th>
<th>Comb.</th>
<th>SVD</th>
<th>RWR</th>
<th>Top-k</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Facebook-1</td>
<td>0.806</td>
<td>0.812</td>
<td>0.817</td>
<td>0.754</td>
<td>0.89</td>
<td>0.792</td>
<td>0.873</td>
<td>0.901</td>
<td>0.808</td>
</tr>
<tr>
<td>Facebook-2</td>
<td>0.776</td>
<td>0.777</td>
<td>0.777</td>
<td>0.875</td>
<td>0.755</td>
<td>0.823</td>
<td>0.949</td>
<td>0.931</td>
<td>0.931</td>
</tr>
<tr>
<td>Last.fm</td>
<td>0.596</td>
<td>0.597</td>
<td>0.603</td>
<td>0.831</td>
<td>0.861</td>
<td>0.644</td>
<td>0.844</td>
<td>0.876</td>
<td>0.802</td>
</tr>
<tr>
<td>GrQc</td>
<td>0.658</td>
<td>0.658</td>
<td>0.658</td>
<td>0.709</td>
<td>0.793</td>
<td>0.791</td>
<td>0.802</td>
<td>0.824</td>
<td>0.799</td>
</tr>
<tr>
<td>HepTh</td>
<td>0.546</td>
<td>0.546</td>
<td>0.547</td>
<td>0.686</td>
<td>0.714</td>
<td>0.774</td>
<td>0.779</td>
<td>0.817</td>
<td>0.775</td>
</tr>
<tr>
<td>CondMat</td>
<td>0.763</td>
<td>0.763</td>
<td>0.764</td>
<td>0.749</td>
<td>0.877</td>
<td>0.875</td>
<td>0.900</td>
<td>0.929</td>
<td>0.896</td>
</tr>
</tbody>
</table>

Precision: AUC (Area under the ROC curve)
Highlighted: statistically significant winners (by paired t-test with \( p < 0.05 \))
Setting: training (60% edges) \( \rightarrow \) evaluation (40% edges), 10 times
Conclusions

**Contribution 1**
An indexing method for top-\( k \) distance queries

**Contribution 2**
Top-\( k \) distances as a proximity measure

Software available!  [http://git.io/topk_pll](http://git.io/topk_pll)
Hope to see further exploration of top-\( k \) distances in various applications!