Fast Shortest Path Distance Estimation in Large Networks

Michalis Potamias        Francesco Bonchi
Carlos Castillo          Aristides Gionis
Context-aware Search

...use shortest-path distance in wikipedia links-graph!
Social Search

John searches Mary

Ranking:
1. Mary A
2. Mary B
3. Mary C

…use shortest-path distance in friendship graph!
Problem and Solutions

• DB: Graph $G = (V,E)$
• Query: Nodes $s$ and $t$ in $V$
• Goal: Compute fast shortest path $d(s,t)$

• Exact Solution
  – BFS - Dijkstra
  – Bidirectional - Dijkstra with A* (aka ALT methods)
    • [Ikeda, 1994] [Pohl, 1971] [Goldberg and Harrelson, SODA 2005]

• Heuristic Solution
  – Avoid traversals – Use Random Landmarks
    • [Kleinberg et al, FOCS 2004] [Vieira et al, CIKM 2007]
  – Can we choose Better Landmarks ?!!
The Landmarks’ Method

• Offline
  – Precompute distance of all nodes to a small set of nodes (landmarks)
  – Each node is associated with a vector with its SP-distance from each landmark (embedding)

• Query-time
  – \( d(s,t) = ? \)
  – Combine the embeddings of \( s \) and \( t \) to get an estimate of the query
Contribution

1. Proved that covering the network with landmarks is NP-hard.

2. Devised heuristics for good landmarks.

3. Experiments with 5 large real-world networks and more than 30 heuristics. Comparison with state of the art.

4. Application to Social Search.
Algorithmic Framework

• Triangle Inequality

\[ d_G(s, t) \leq d_G(s, u) + d_G(u, t), \]
\[ d_G(s, t) \geq |d_G(s, u) - d_G(u, t)| \]

• Observation: the case of equality

\[ d_G(s, t) = d_G(s, u) + d_G(u, t) \]
\[ d_G(s, t) = |d_G(s, u) - d_G(u, t)| \]
The Landmarks’ Method

1. Selection: Select $k$ landmarks

2. Offline: Run $k$ BFS/Dijkstra and store the embeddings of each node:
   \[
   \Phi(s) = <d(s, u_1), d(s, u_2), \ldots, d(s, u_k)>
   \]
   \[
   = <s_1, s_2, \ldots, s_k>
   \]

3. Query-time: $d(s,t) = \ ?$
   - Fetch $\Phi(s)$ and $\Phi(t)$
   - Compute $\min_i{s_i + t_i}$ (i.e. inf of UB) ... in time $O(k)$
Example query: $d(s, t)$

<table>
<thead>
<tr>
<th></th>
<th>$d(_, u_1)$</th>
<th>$d(_, u_2)$</th>
<th>$d(_, u_3)$</th>
<th>$d(_, u_4)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\Phi(s)$</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>$\Phi(t)$</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>4</td>
</tr>
</tbody>
</table>

UB

<table>
<thead>
<tr>
<th></th>
<th>5</th>
<th>9</th>
<th>6</th>
<th>6</th>
</tr>
</thead>
<tbody>
<tr>
<td>LB</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

$$\max_i |s_i - t_i| \leq d_G(s, t) \leq \min_j \{s_j + t_j\}$$
Coverage Using Upper Bounds

• A landmark \( u \) covers a pair \((s, t)\), if \( u \) lies on a shortest path from \( s \) to \( t \)

• Problem Definition: find a set of \( k \) landmarks that cover as many pairs \((s,t)\) in \( V \times V \) as possible
  - NP-hard
  - \( k = 1 \): node with the highest betweenness centrality
  - \( k > 1 \): greedy set-cover (approximation - too expensive)

...central nodes are a good start for devising heuristics!
Landmarks Selection: Basic Heuristics

• **Random** (baseline)

• Choose central nodes!
  - Degree
  - **Closeness** centrality
    • Closeness of $u$ is the average distance of $u$ to any vertex in $G$

• Caveat: many central nodes may *cover* the same pairs: newly added landmarks should cover different pairs
  
  *spread the landmarks in the graph!*
Constrained Heuristics

• Remove immediate neighborhood

1. Rank all nodes according to Degree or Centrality
2. Iteratively choose the highest ranking nodes. Remove $h$-neighbors of each selected node from candidate set

• Denote as
  – Degree/h
  – Closeness/h
  – Best results for $h = 1$
Partitioning-based Heuristics

• Use graph-partitioning to spread nodes.
• Utilize any partitioning scheme and
  – Degree/P
    • Pick the node with the highest degree in each partition
  – Closeness/P
    • Pick the node with the highest closeness in each partition
  – Border/P
    • Pick the node closer to the border in each partition. Maximize the border-value that is given from the following formula:

\[
b(u) = \sum_{j \in C, u \in C(i), i \neq j} d_j(u) \cdot d_i(u)
\]
Versus Random - error

**Flickr Explicit dataset**

- Rand
- Degree
- Centr
- Degree/1
- Centr/1
- Degree/P
- Centr/P
- Border

**Flickr Implicit dataset**

- Rand
- Degree
- Centr
- Degree/1
- Centr/1
- Degree/P
- Centr/P
- Border
Versus Random - triangulation

random landmarks have theoretical guarantees [FOCS04]
### Versus ALT - efficiency

<table>
<thead>
<tr>
<th></th>
<th>flickr™</th>
<th>flickr™</th>
<th>WIKIPEDIA</th>
<th>dblp.uni-trier.de</th>
<th>YAHOO MESSANGER</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Ours (10%)</strong></td>
<td>20</td>
<td>100</td>
<td>500</td>
<td>50</td>
<td>50</td>
</tr>
<tr>
<td><strong>ALT LB</strong></td>
<td>60K</td>
<td>40K</td>
<td>80K</td>
<td>20K</td>
<td>2K</td>
</tr>
<tr>
<td><strong>Visited Nodes</strong></td>
<td>7K</td>
<td>10K</td>
<td>20K</td>
<td>2K</td>
<td>2K</td>
</tr>
</tbody>
</table>

- **state of the art exact ALT methods** [SODA05]

- ALT LB comparisons:
  - >300x
  - >400x
  - >160x
  - >400x
  - >40x
Social Search Task

random landmarks have been used [CIKM07]
Conclusion

• Novel search paradigms need distance as primitive
  – Approximations should be computed in milliseconds

• Heuristic landmarks yield remarkable tradeoffs for SP-distance estimation in huge graphs
  – Hard to find the optimal landmarks
  – Border and Centrality heuristics:
    • outperform Random even by a factor of 250.
    • are, for a 10% error, many orders of magnitude faster than state of the art exact algorithms (ALT)

• Future Work
  – Provide fast estimation for more graph primitives!
Thank you!