Determining the Diameter of Small World Networks

Frank W. Takes & Walter A. Kosters

Leiden University, The Netherlands

CIKM 2011 — October 27, 2011 — Glasgow, UK

NWO COMPASS project (grant #612.065.92)
Overview

- Introduction
- Preliminaries
- Problem statement
- Related work
- Algorithm
- Results
- Conclusion
- Future work
Introduction

- Small world networks
- Power law degree distribution, giant component, low average pairwise distances
- Examples: social networks, webgraphs, communication networks, collaboration networks, information networks, protein-protein interaction networks, citation networks, etc.
- **Diameter**: length of longest shortest path in a graph
Diameter Example

Figure: Graph with diameter 6. Numbers denote node eccentricity
Preliminaries

- Graph $G = (V, E)$ with $|V| = n$ nodes and $|E| = m$ edges
- Distance $d(u, v)$: length of shortest path between $u, v \in V$
- Undirected: $d(u, v) = d(v, u)$ for all $u, v \in V$
- One connected component: $d(u, v)$ is finite for all $u, v \in V$
- Neighborhood $N(u)$: set of nodes connected to $u$ via an edge
- Degree $\deg(u)$: number of edges connected to node $u$
Problem statement

- Consider a connected undirected graph $G = (V, E)$
- Our aim is to compute in large small-world graphs $(1,000 \leq n \leq 100,000,000, \bar{d}(u, v) \approx 6$ for all $u, v \in V)$:
Problem statement

- Consider a connected undirected graph \( G = (V, E) \)
- Our aim is to compute in large small-world graphs \((1,000 \leq n \leq 100,000,000, \bar{d}(u, v) \approx 6 \text{ for all } u, v \in V)\):
  - **Eccentricity** \( e(v) \): length of a longest shortest path from \( v \):
    \[ e(v) = \max_{w \in V} d(v, w) \]
  - **Diameter** \( D(G) \): maximal distance (longest shortest path length) over all node pairs:
    \[ \max_{v, w \in V} d(v, w) \]
Problem statement

Consider a connected undirected graph $G = (V, E)$

Our aim is to compute in large small-world graphs ($1,000 \leq n \leq 100,000,000$, $d(u, v) \approx 6$ for all $u, v \in V$):

- **Eccentricity** $e(v)$: length of a longest shortest path from $v$: $e(v) = \max_{w \in V} d(v, w)$
- **Diameter** $D(G)$: maximal distance (longest shortest path length) over all node pairs: $\max_{v, w \in V} d(v, w)$
- **Diameter** $D(G)$ (alternative definition): maximal eccentricity over all nodes: $\max_{v \in V} e(v)$
Problem statement

- Consider a connected undirected graph $G = (V, E)$
- Our aim is to compute in large small-world graphs $(1,000 \leq n \leq 100,000,000, \bar{d}(u, v) \approx 6$ for all $u, v \in V$):
  - **Eccentricity** $e(v)$: length of a longest shortest path from $v$: 
    $e(v) = \max_{w \in V} d(v, w)$
  - **Diameter** $D(G)$: maximal distance (longest shortest path length) over all node pairs: 
    $\max_{v, w \in V} d(v, w)$
  - **Diameter** $D(G)$ (alternative definition): maximal eccentricity over all nodes: 
    $\max_{v \in V} e(v)$
  - **Radius** $R(G)$: minimal eccentricity over all nodes: 
    $\min_{v \in V} e(v)$
  - Eccentricity distribution: (relative) frequency $f(x)$ of each eccentricity value $x$
    $$f(x) = \frac{|\{u \in V \mid e(u) = x\}|}{n}$$
Figure: Relative eccentricity distribution of five large graphs
Diameter Applications

- Router networks: what is the worst-case response time between any two machines?
- Social networks: in how many steps does a message released by a single user reach everyone in the network?
- Biological interaction networks: which proteins are likely to not influence each other at all?
- Information networks (i.e., Wikipedia): how do I change the conversation topic to a maximally different subject? ;-)
- Eccentricity has been suggested as a worst-case measure of node centrality: the relative importance of a node based on the graph’s structure
Naive Algorithm

- Diameter is equal to the largest value returned by an All Pairs Shortest Path (APSP) algorithm.
- Brute-force: for each of the $n$ nodes, execute a Breadth First Search (BFS) run in $O(m)$ time to find the eccentricity, and return the largest value found.
- Time complexity $O(mn)$
- Problematic if $n = 8$ million and $m = 1$ billion. Then one BFS takes 6 seconds on a 3.4GHz machine. That results in 1.5 years to compute the diameter . . .
Related work

- Approximation algorithms, for example ANF (Palmer et al.)
- Use a random sample of the set of nodes (Mislove et al.)
- Heuristics, for example repeatedly select the farthest node until there is no more improvement (Leskovec et al.)
- Matrix multiplication for APSP in $O(n^{2.376})$ (Yuster et al.)
- Bounds: diameter upper bound is at most two times the lowest found eccentricity value (Magnien et al.)
Social Network Example (1)

- If I am connected to everyone in at most 6 steps, then
  - My direct friend is connected to everyone in at most 7 steps (he reaches everyone through me)
  - My direct friend is connected to everyone in at least 5 steps (I reach everyone through him)

- If I can reach everyone in the network in 6 steps, then
  - There is nobody who can reach everyone in less than 3 steps (or I could have utilized him)
  - There is nobody who needs more than 12 steps to reach everyone (or he could have utilized me)
If a node $v$ has eccentricity $e(v)$, then
- Nodes $w$ at distance $d(v, w)$ needs at most $e(v) + d(v, w)$ steps
  ($w$ reaches every node via $v$)
- Nodes $w$ at distance $d(v, w)$ needs at least $e(v) - d(v, w)$ steps ($v$ reaches every node via $w$)

We call this the **Eccentricity bounds**

If a node $v$ can reach every other node in $e(v)$ steps, then
- There is no node that can reach everyone in less than $\lceil e(v)/2 \rceil$ steps (or $v$ could have used that node)
- There is no node that needs more than $e(v) \cdot 2$ steps to reach all other nodes (or that node could have used $v$)

We call this the **Diameter bounds**
Eccentricity bounds

Theorem

For nodes \( v, w \in V \) we have

\[
\max(e(v) - d(v, w), d(v, w)) \leq e(w) \leq e(v) + d(v, w).
\]

Proof

- **Upper bound** \( e(v) + d(v, w) \): if node \( w \) is at distance \( d(v, w) \) of node \( v \), it can always employ \( v \) to get to every other node in \( e(v) \) steps. To get to node \( v \), exactly \( d(v, w) \) steps are needed, totalling \( e(v) + d(v, w) \) steps to get to any other node.

- **Lower bound** \( e(v) - d(v, w) \): interchanging \( v \) and \( w \) in the previous statement.

- **Lower bound** \( d(v, w) \): the eccentricity of \( w \) is at least equal to some found distance to \( w \).
Diameter bounds

- Let $e_L(v)$ and $e_U(v)$ denote the lower and upper eccentricity bounds derived using the Eccentricity bounds.
- Then we can derive the following **diameter bounds**:
  $$\max_{v \in V} e_L(v) \leq D(G) \leq \max_{v \in V} e_U(v)$$
- Let $D_L(G)$ and $D_U(G)$ denote these lower and upper diameter bounds. $D_L(G) \leq D(G) \leq D_U(G)$
BoundingDiameters Algorithm

Input: Graph G
Output: Diameter of G

\[ W \leftarrow V \quad D_\ell \leftarrow -\infty \quad D_u \leftarrow +\infty \]

for \( w \in W \) do
    \[ e_\ell[w] \leftarrow -\infty \quad e_u[w] \leftarrow +\infty \]
end for

while \( D_\ell \neq D_u \) and \( W \neq \emptyset \) do
    \( v \leftarrow \text{SELECTFROM}(W) \)
    \( e[v] \leftarrow \text{ECCENTRICITY}(v) \)

    \[ D_\ell \leftarrow \max(D_\ell, e[v]) \]
    \[ D_u \leftarrow \min(D_u, 2 \cdot e[v]) \]

    for \( w \in W \) do
        \[ e_\ell[w] = \max(e_\ell[w], \max(e[v] - d(v, w), d(v, w))) \]
        \[ e_u[w] = \min(e_u[w], e[v] + d(v, w)) \]
        if (\( e_u[w] \leq D_\ell \) and \( e_\ell[w] \geq D_u / 2 \)) or
            (\( e_\ell[w] = e_u[w] \)) then
            \( W \leftarrow W - \{w\} \)
        end if
    end for

    \( D_u \leftarrow \min(D_u, \max_{w \in V}(e_u[w])) \)
end while

return \( D_\ell \);
Bounding Diameters Algorithm

- Initialize candidate set $W$ to $V$
  While $D_L(G) \neq D_U(G)$:
    1. Select a node $v$ from $W$ cf. some Selection strategy
    2. Compute $v$’s eccentricity, and update $e_L(v)$ and $e_U(v)$ for every node $v \in W$ according to the Eccentricity bounds
    3. Update the diameter bounds $D_L(G)$ and $D_U(G)$
    4. Remove nodes $w$ that can no longer contribute to refining the Diameter bounds

- Worst-case: $n$ iterations, best-case: 2 iterations (investigate $v$ and $w$ with $e(v) = 2 \cdot e(w) = D(G)$)

- To compute the complete eccentricity distribution, stop when: $\forall v \in V : e_L(v) = e_U(v)$

- Selection strategy is important (and discussed later)
What is the diameter of this graph?

\[ D_L = -\infty \text{ and } D_U = \infty \]
Example run (1)

Iteration 1: after computing the eccentricity of node F

\[ D_L = 5 \text{ and } D_U = 10 \]
Example run (2)

Iteration 2: after computing the eccentricity of node T
\[ D_L = 7 \text{ and } D_U = 10 \]
Iteration 3: after computing the eccentricity of node L

\[ D_L = 7 \text{ and } D_U = 7 \]
Selection strategy

- Random node ("smart APSP")
- Based on the degree of the node
- Eccentricity bounds difference (1)
- Interchange smallest eccentricity lower bound and largest eccentricity upper bound (2)
- Repeated farthest distance (cf. Leskovec et al.) (3)
Results

1. Eccentricity bounds difference
2. Alternate between smallest eccentricity lower bound and largest upper bound
3. Repeatedly select a node furthest away from the previous node

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Nodes</th>
<th>(D(G))</th>
<th>Strat. 1</th>
<th>Strat. 2</th>
<th>Strat. 3</th>
<th>Pruned</th>
</tr>
</thead>
<tbody>
<tr>
<td>AstroPhys</td>
<td>17,903</td>
<td>14</td>
<td>18</td>
<td>9</td>
<td>63</td>
<td>185</td>
</tr>
<tr>
<td>ENRON</td>
<td>33,696</td>
<td>13</td>
<td>12</td>
<td>11</td>
<td>61</td>
<td>8,715</td>
</tr>
<tr>
<td>Web</td>
<td>855,802</td>
<td>24</td>
<td>20</td>
<td>4</td>
<td>28</td>
<td>91,965</td>
</tr>
<tr>
<td>YouTube</td>
<td>1,134,890</td>
<td>24</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>399,553</td>
</tr>
<tr>
<td>Flickr</td>
<td>1,624,992</td>
<td>24</td>
<td>10</td>
<td>3</td>
<td>7</td>
<td>553,242</td>
</tr>
<tr>
<td>Skitter</td>
<td>1,696,415</td>
<td>31</td>
<td>10</td>
<td>4</td>
<td>19</td>
<td>114,803</td>
</tr>
<tr>
<td>Wikipedia</td>
<td>2,213,236</td>
<td>18</td>
<td>21</td>
<td>3</td>
<td>583</td>
<td>947,582</td>
</tr>
<tr>
<td>Orkut</td>
<td>3,072,441</td>
<td>10</td>
<td>357</td>
<td>106</td>
<td>389</td>
<td>27,429</td>
</tr>
<tr>
<td>LiveJournal</td>
<td>5,189,809</td>
<td>23</td>
<td>6</td>
<td>3</td>
<td>14</td>
<td>318,378</td>
</tr>
<tr>
<td>Hyves</td>
<td>8,057,981</td>
<td>25</td>
<td>40</td>
<td>21</td>
<td>44</td>
<td>446,258</td>
</tr>
</tbody>
</table>

**Table:** Comparison of three node selection strategies
Pruning

Theorem
Assume $n > 2$. For a given $v \in V$, all nodes $w \in N(v)$ with $\text{deg}(w) = 1$ have $e(w) = e(v) + 1$.

Proof
- Node $w$ is only connected to node $v$, and will thus need node $v$ to reach every other node in the graph. If node $v$ can do this in $e(v)$ steps, then node $w$ can do this in exactly $e(v) + 1$ steps.
- The restriction $n > 2$ on the graph size excludes the case in which $w$ realizes the eccentricity of $v$.

(alternative proof is possible, based on graph homomorphism)
Discussion

- **Main result:** in real-world graphs `BOUNDINGDIAMETERS` is much faster than the naive algorithm (a handful vs. $n$ BFSes)
- **Why does it work?** There is always diversity in the eccentricity values of nodes, allowing central nodes to influence the eccentricity of peripheral nodes, and vice versa
- **When does it not work so well?** In graphs with little diversity in the eccentricity values, e.g., circle-shaped graphs
- **Side result:** efficiently computing derived measures such as the radius, center, periphery and even the exact eccentricity distribution is also possible (after some modifications)
Conclusion

- Our algorithm computes the diameter of large real world graphs much faster compared to the naive algorithm.
- Our algorithm improves upon previously suggested techniques, because:
  - we obtain an exact result instead of an approximation
  - it is possible to obtain the actual diameter path
  - information between iterations is not thrown away
  - computation time is very short, even for graphs with millions of nodes
- Future work: optimize the node selection strategy even further and incremental updates as the graph changes over time through the addition and deletion of nodes and edges.
Questions?