On Computing the Diameter of Real World Networks

Andrea Marino


Dipartimento di Informatica, University of Milan

Luchon, June 2014
Given a graph $G = (V, E)$ connected.

- The distance $d(u, v)$ is the number (sum of the weights) of edges along the shortest path from $u$ to $v$.
- The eccentricity of a node $u$, $\text{ecc}(u) = \max_{v \in V} d(u, v)$: in how many hops $u$ can reach any node?

### Definition (Diameter)

$$D = \max_{u, v \in V} d(u, v) = \max_{u \in V} \text{ecc}(u)$$
Why people compute the diameter?
- It gives bounds on information diffusion time, e.g., lower bound on how quickly information reaches EVERY individual.

How they do it?
- Textbook Algorithm ($n = |V|$, $m = |E|$). Too expensive.
  - Perform a Breadth First Search (BFS) for each node and return maximum ecc.
    - A BFS from $x$ returns all the distances from $x$ and takes $O(m)$ time.
  - All other approaches (see [Zwick, 2001]) solves also all pairs shortest path: they have to require at least $\Omega(n^2)$. Still too expensive.
- Otherwise approximation algorithms or heuristics, as for instance empirically finding lower bound $L$ and upper bound $U$.
  - That is, $L \leq D \leq U$
  - $D$ found, when $L = U$
Bound Refinement: iterative fringe upper bound

Recall that

The *textbook* algorithm runs a BFS for any node and return the maximum ecc found.

iFUB is a particular case in which we:

- specify the order in which the BFSes have to be executed
- refine a lower bound,
  - that is the maximum ecc found until that moment.
- upper bound the eccentricities of the remaining nodes.
- stop when the remaining nodes cannot have eccentricity higher than our lower bound.

Good order by analyzing how nodes are placed in the BFS tree of a central node $u$.

- $u$ is the highest degree node or the node returned by some heuristic (2-SWEEP or 4-SWEEP).
For any $1 \leq i < \text{ecc}(u)$ and $1 \leq k < i$, and for any $x \in N_{i-k}(u)$ such that \( \text{ecc}(x) > 2(i - 1) \), there exists $y \in N_j(u)$ such that $d(x, y) = \text{ecc}(x)$ with $j \geq i$.

All the nodes $x$ above the level $i$ in $\text{BFS}(u)$ having $\text{ecc}$ greater than $2(i - 1)$ have a corresponding node $y$, whose $\text{ecc}$ is greater or equal to $\text{ecc}(x)$, below or on the level $i$ in $\text{BFS}(u)$. 
Experiments for undirected graphs

The number of visits seems to be constant.

It has been used to compute the diameter of Facebook (721.1M nodes and 68.7G edges, Diameter 41) with just 17 BFSes.
Experiments for directed graphs

The algorithm has been extended also to directed graphs (SEA 2012): in this case it is called directed iFUB.

The number of visits seems to be constant.

Andrea Marino
On Computing the Diameter of Real World Networks
Bad cases

Cases in which nodes have close eccentricity or the BFS trees are isomorphic.

A cycle

Special regular graphs [such as Moore graphs]
Thanks
Computing lower bound and upper bound easily

A graph

A BFS tree from $v_1$

Lower bound

The eccentricity, $\text{ecc}$ (height of the BFS tree) of a node.

In the example 3: at least a pair is at distance 3.

Upper bound

The double of the eccentricity $\text{ecc}$ of a node.

In the example 6: every node can reach another node going to $v_1$ by $\leq 3$ edges and going to the destination in $\leq 3$ edges.

Bounds by sampling are often not tight: $L < D < U$ (see SNAP experiments)

In the example diameter is 4: $d(v_7, v_8) = 4$. 
2-SWEEP

1. Run a BFS from a random node $r$: let $a$ be the farthest node.
2. Run a BFS from $a$: let $b$ be the farthest node.
3. Return the length of the path from $a$ to $b$.

Return $d(a, b)$. 

Good lower bounds in directed graphs: directed 2-SWEEP

2-dSWEEP

1. Run a forward BFS from a random node $r$: let $a_1$ be the farthest node.
2. Run a backward BFS from $a_1$: let $b_1$ be the farthest node.
3. Run a backward BFS from $r$: let $a_2$ be the farthest node.
4. Run a forward BFS from $a_2$: let $b_2$ be the farthest node.
5. If $\text{ecc}_B(a_1) > \text{ecc}_F(a_2)$, then return the length of the path from $b_1$ to $a_1$. Otherwise return the length of the path from $a_2$ to $b_2$.

Return the maximum between $d(a_2, b_2)$ and $d(b_1, a_1)$.

First time used in directed graph by Broder et al. to study Graph structure in the web.
In this modified grid with \( k \) rows and \( 1 + 3k/2 \) columns. The algorithm returns \( k + 1 \). The diameter of the network is instead \( 3k/2 \).