On Computing Distance Distribution of Real World Graphs

Andrea Marino

ALGORITMI E PROGRAMMAZIONE PER L'ANALISI DEI DATI,
Università di Firenze

Firenze, a.a. 2019-20

In the late 50s, Milgram aimed at answering the following question:

> *given two individuals selected randomly from the population, what is the probability that the minimum number of intermediaries required to link them is* $0, 1, 2, \ldots, k$*?.*

start
finish
North Dakota
Minnesota
South Dakota
Wisconsin
Michigan
Nebraska
Iowa
Illinois
Indiana
Ohio
New York
Pennsylvania
Kansas
Missouri
Kentucky
Virginia
Oklahoma
Arkansas
Tennessee
North Carolina
South Carolina
Mississippi
Alabama
Georgia
Texas
Louisiana
Florida
New Hampshire
Vermont
Massachusetts
Maine
Rhode Island
Connecticut
New Jersey
Delaware
Maryland
West Virginia

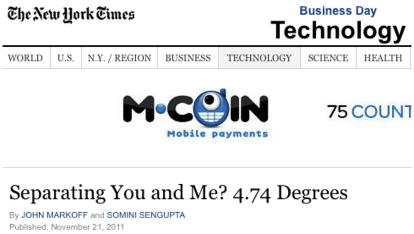
- He selected 296 volunteers (the starting population) and asked them to dispatch a message to a specific individual (the target person), a stockholder living in Sharon, MA, a suburb of Boston, and working in Boston.
- The message could not be sent directly to the target person (unless the sender knew him personally), but could only be mailed to someone known personally who is more likely than the sender to know the target person.
- The starting population:
    - 100 of them were people living in Boston,
    - 100 were Nebraska stockholders (i.e., people living far from the target but sharing with him their profession) and
    - 96 were Nebraska inhabitants chosen at random.

The results obtained from Milgram's experiments were:

- only 64 chains (22%) were completed (i.e., they reached the target);
- the average number of intermediaries in these chains was 5.2,
- there was a marked difference between the Boston group (4.4) and the rest of the starting population, whereas the difference between the two other subpopulations was not statistically significant;
- the random group from Nebraska (far away) needed 5.7 intermediaries on average (i.e., rounding up, "six degrees of separation").

The average path length is small, much smaller than expected, Milgram was measuring the average length of a routing path on a social network, which is an upper bound on the average distance

- indeed the people involved in the experiment were not necessarily sending the postcard to an acquaintance on a shortest path to the destination).


The New York Times

Business Day
Technology

WORLD | U.S. | N.Y. / REGION | BUSINESS | TECHNOLOGY | SCIENCE | HEALTH

Separating You and Me? 4.74 Degrees

By JOHN MARKOFF and SOMINI SENGUPTA
Published: November 21, 2011

First world-scale social-network graph-distance computations, using the entire Facebook network of active users ($\approx$ 721 million users, $\approx$ 69 billion friendship links). The average distance observed is 4.74, corresponding to 3.74 intermediaries or *degrees of separation*.

Lars Backstrom, Paolo Boldi, Marco Rosa, Johan Ugander, Sebastiano Vigna: Four degrees of separation. WebSci 2012: 33-42

Definitions

Given a graph $G = (V, E)$ (strongly) connected.

Definition (Distance)

The distance $d(u, v)$ is the number (sum of the weights) of edges along shortest path from u to v.

Definition (Distance Distribution)

For any h, the fraction of pairs of nodes x, y such that $d(x, y) = h$.

$$N_h = \frac{|\{(u, v) \in V \times V : d(u, v) = h\}|}{n(n-1)}$$

$$N_h(u) = \{v : v \in V, d(u, v) = h\}$$

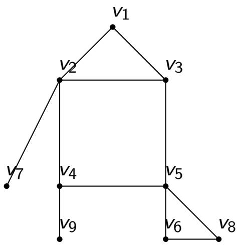
We denote $N_1(u)$ also as $N(u)$.

Definition (Average Distance)

$$\sum_{u,v \in V} \frac{d(u, v)}{n(n-1)} = \sum_h h \cdot N_h$$

Approximating N_h implies approximating average distance.

An Example Undirected Graph



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_1	0	1	1	2	2	3	2	3	3
v_2	1	0	1	1	2	3	1	3	2
v_3	1	1	0	2	1	2	2	2	3
v_4	2	1	2	0	1	2	2	2	1
v_5	2	2	1	1	0	1	3	1	2
v_6	3	3	2	2	1	0	4	1	3
v_7	2	1	2	2	3	4	0	4	3
v_8	3	3	2	2	1	1	4	0	3
v_9	3	2	3	1	2	3	3	3	0

The average distance is the average among all the entries.

- N_1 How many entries are 1 (divided by $n(n-1)$)?
- N_2 How many entries are 2 (divided by $n(n-1)$)?
- ...
- N_D How many entries are D (divided by $n(n-1)$)?

Part I

Classical Sampling

Given a random $U \subseteq V$ approximate

$$N_h = \frac{|\{(u,v) \in V \times V : d(u,v) = h\}|}{n(n-1)}$$

as

$$N_h(U) = \frac{|\{(u,v) \in U \times V : d(u,v) = h\}|}{|U|\,(n-1)}$$

The algorithm

Perform a random sample of k_{EW} vertices from V obtaining the multiset $U = \{u_1, u_2, \ldots, u_{k_{\mathrm{EW}}}\} \subseteq V$.
Run *iteration* $i = 1, 2, \ldots, k_{\mathrm{EW}}$, computing the distances $d(u_i, v)$ for all $v \in V$, by executing a BFS traversal of G starting from vertex u_i.
Return the approximation $N_h(U)$.

The running time of the algorithm is $O(k_{\mathrm{EW}}\, m)$ for unweighted graphs, with a space occupancy of $O(n)$.

The method is unbiased

Since $N_h(\{u_i\}) = \frac{|\{(u_i,v):v\in V\wedge d(u_i,v)=h\}|}{n-1}$

$$N_h(U) = \frac{|\{(u_i,v) : u_i \in U, v \in V \wedge d(u,v) = h\}|}{k_{\mathrm{EW}}\,(n-1)} = \frac{\sum_{i=1}^{k_{\mathrm{EW}}} N_h(\{u_i\})}{k_{\mathrm{EW}}}$$

If vertex u_i is randomly chosen in V, then $E[N_h(\{u_i\})] = N_h$. Indeed,

$$E[N_h(\{u_i\})] = \frac{1}{n}\sum_{v\in V} N_h(\{v\}) = N_h(V) = N_h.$$

Hence, if all elements of U are randomly chosen, we have that, by the linearity of the expectation,

$$E[N_h(U)] = E\left[\frac{\sum_{i=1}^{k_{\mathrm{EW}}} N_h(\{u_i\})}{k_{\mathrm{EW}}}\right] = \frac{\sum_{i=1}^{k_{\mathrm{EW}}} E[N_h(\{u_i\})]}{k_{\mathrm{EW}}} = N_h.$$

Bounding the error

Our goal is to keep k_{EW} as small as possible still ensuring a bounded error.

Theorem (Azuma-Hoeffding bound)

If $x_1, x_2, \ldots, x_k$ are independent random variables such that $\mu = E[\sum x_i/k]$ and for each i there exist a_i and b_i such that $a_i < x_i < b_i$, then, for any $\xi > 0$,

$$Pr\left\{\left|\frac{\sum_{i=1}^{k} x_i}{k} - \mu\right| \geq \epsilon\right\} \leq 2e^{-2k^2\epsilon^2/\sum_{i=1}^{k}(b_i - a_i)^2}.$$

In our case,

- $k = k_{\mathrm{EW}}$,
- $x_i = N_h(\{u_i\})$, and
- $\mu = N_h$ (since we have shown that $\mu = N_h = E[\sum_{i=1}^{k_{\mathrm{EW}}} N_h(\{u_i\})/k_{\mathrm{EW}}] = E[\sum x_i/k])$
- For $1 \leq i \leq k_{\mathrm{EW}}$, $0 \leq N_h(\{u_i\}) \leq 1$ implying $a_i = 0$ and $b_i = 1$.

$$Pr\left\{\left|\frac{\sum_{i=1}^{k_{\mathrm{EW}}} N_h(\{u_i\})}{k_{\mathrm{EW}}} - N_h\right| \geq \epsilon\right\} \leq 2e^{-2\,k_{\mathrm{EW}}\,\epsilon^2}.$$

- In

$$Pr\left\{\left|\frac{\sum_{i=1}^{k_{\mathrm{EW}}} N_h(\{u_i\})}{k_{\mathrm{EW}}} - N_h\right| \geq \epsilon\right\} \leq 2e^{-2\,k_{\mathrm{EW}}\,\epsilon^2}.$$

  if we choose $k_{\mathrm{EW}} = \frac{\alpha}{2}\epsilon^{-2}\ln n$ for any constant $\alpha > 0$, we have that this probability is bounded by $2/n^{\alpha}$.
- This number of iterations (BFSes) guarantees that the absolute error is bounded by ϵ with high probability.

Hence

Theorem

Let G be a (strongly) connected graph with n vertices and m edges. For any arbitrarily small $\epsilon > 0$, the EW *algorithm with $k_{\mathrm{EW}} = \Theta(\epsilon^{-2} \log n)$ computes in time $O(k_{\mathrm{EW}}\, m)$ an approximation of the distance distribution N_h of G whose absolute error is bounded by ϵ, with high probability.*

- The previous analysis can be easily extended to the case of *weighted* (strongly) connected graphs, by making use of the Dijkstra's algorithm: the running time becomes $O(k_{\mathrm{EW}}(m + n\log n)) = O(\epsilon^{-2}(m\log n + n\log^2 n))$.

In a similar way, we can compute

- An approximation of the *average distance* of G, which is defined as $\overline{d} = \sum_{u\in V}\sum_{v\in V,\ v\neq u} d(u,v)/n(n-1)$.
- An approximation of the α-*diameter* of G, which is defined as the minimum h for which $\sum_{i=1}^{h} N_h \geq \alpha$:
  - it suffices to repeat the analysis with respect to $\sum_{i=1}^{h} N_h = \frac{N'_h}{n(n-1)} = \frac{|\{(u,v)\in V\times V: u\neq v,\, d(u,v)\leq h\}|}{n(n-1)}$.

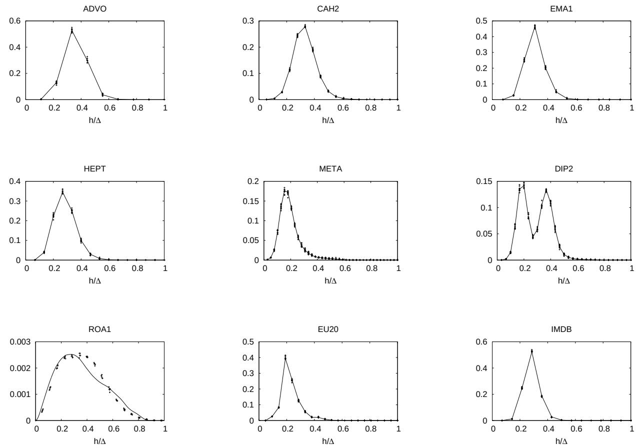


Figure: Approximate distribution (starred points) computed by classical sampling versus actual distribution of N_h (continuous line). The x-axis represents the normalized value h/D.

- David Eppstein, Joseph Wang: Fast Approximation of Centrality. J. Graph Algorithms Appl. 8: 39-45 (2004)
- Pierluigi Crescenzi, Roberto Grossi, Leonardo Lanzi, Andrea Marino: A Comparison of Three Algorithms for Approximating the Distance Distribution in Real-World Graphs. TAPAS 2011: 92-103
- Jure Leskovec, Christos Faloutsos: Sampling from large graphs. KDD 2006: 631-636

Part II

Priority Sampling

These methods study the following.

Definition (Cumulative Distance Distribution)

For any h, the fraction of pairs of nodes x, y such that $d(x, y) \leq h$.

$$B_h = \frac{|\{(u, v) \in V \times V : d(u, v) \leq h\}|}{n^2}$$

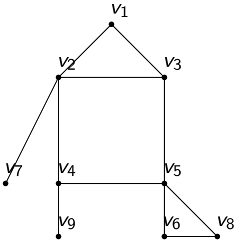
$$B_h(u) = \{v : v \in V : d(u, v) \leq h\}$$

Obviously, $N_h = \frac{n^2(B_h - B_{h-1})}{n(n-1)}$.

For any u:

$$B_1(u) = \{u\} \cup N(u)$$

$$B_h(u) = B_{h-1}(u) \cup \bigcup_{v \in N(u)} B_{h-1}(v)$$



	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9
v_1	0	1	1	2	2	3	2	3	3
v_2	1	0	1	1	2	3	1	3	2
v_3	1	1	0	2	1	2	2	2	3
v_4	2	1	2	0	1	2	2	2	1
v_5	2	2	1	1	0	1	3	1	2
v_6	3	3	2	2	1	0	4	1	3
v_7	2	1	2	2	3	4	0	4	3
v_8	3	3	2	2	1	1	4	0	3
v_9	3	2	3	1	2	3	3	3	0

- $B_1(v_1) = \{v_1, v_2, v_3\}$
- $B_2(v_1) = B_1(v_1) \cup B_1(v_2) \cup B_1(v_3) =$
  $B_1(v_1) \cup \{v_1, v_2, v_3, v_4, v_7\} \cup \{v_1, v_2, v_3, v_5\} = \{v_1, v_2, v_3, v_4, v_5, v_7\}$
- $B_3(v_1) = B_2(v_1) \cup B_2(v_2) \cup B_2(v_3) =$
  $B_2(v_1) \cup \{v_1, v_2, v_3, v_4, v_5, v_7, v_9\} \cup \{v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8\} = V$

This gives an exact algorithm that in D steps computes $|B_h(u)|$ for any u, B_h, and h.

Exact algorithm for cumulative distance distribution

- for any u, $B_1(u) = \{u\} \cup N(u)$ and output $\sum_u |B_1(u)|/n^2 = B_1$
- for $h \in 2 \dots D$:
    - for any u, $B_h(u) = B_{h-1}(u) \cup \bigcup_{v \in N(u)} B_{h-1}(v)$
    - output $\sum_u |B_h(u)|/n^2 = B_h$.

The memory usage is $\Theta(n \cdot n)$, since the size of $B_{h-1}(u)$ for any u can be $\Theta(n)$.

Our Target: for any u,

$$B_h(u) = B_{h-1}(u) \cup \bigcup_{v \in N(u)} B_{h-1}(v)$$

Question: for any u, given some limited information (constant or almost constant) about the sets $B_{h-1}(u)$, and $B_{h-1}(v)$ for any $v \in N(u)$, is there a way to approximate $B_h(u)$?

- If the information stored for $B_{h-1}(u)$ is constant for any u, then we have $\Theta(n)$ instead of $\Theta(n \cdot n)$
- We have to *approximate* a set A in constant space, so that if we want an *approximation* of the union set $A \cup B$ we can combine the approximation we did of A and B.

Probabilistic counting

A sketch $S(A)$ is a compressed form of representation for a given set A providing the following operations:

`INIT` $(S(A))$ How a sketch $S(A)$ for A is initialized.

`UPDATE` $(S(A), u)$ How a sketch $S(A)$ for A modifies when an element u is added to A.

`UNION` $(S(A), S(B))$ Given two sketches for A and B, provide a sketch for $A \cup B$.

`SIZE` $(S(A))$ Estimate the number of distinct elements of A.

Properties

- Given two sketches $S(A)$ and $S(B)$ for any two sets A and B, $S(A \cup B)$ can be computed just by looking at $S(A)$ and $S(B)$, i.e.: C=`UNION` $(S(A), S(B)) \equiv$ (`INIT` (C); for $u \in A \cup B$, `UPDATE` (C, u) ; `RETURN` C)
- If we call `UPDATE` $(S(A), u)$ and we already did `UPDATE` $(S(A), u)$ with the same u the sketch does not modify, e.g. the operation `SIZE` $(S(A))$ return the same value.

Plugging the probabilistic counting into the framework

Exact algorithm for cumulative distance distribution

- for any u, $B_1(u) = \{u\} \cup N(u)$ and output $\sum_u |B_1(u)|/n^2 = B_1$
- for $h \in 2 \dots D$:
    - for any u, $B_h(u) = B_{h-1}(u) \cup \bigcup_{v \in N(u)} B_{h-1}(v)$
    - output $\sum_u |B_h(u)|/n^2 = B_h$.

Approximation algorithm for cumulative distance distribution

```
for any $u$, INIT $S(B_1(u))$
for any $u$, for any $v \in \{u\} \cup N(u)$, UPDATE $(S(B_1(u)), v)$.
Output $\sum_u$SIZE $(S(B_1(u)))/n^2$
for $h \in 2 \dots D$:
    for any $u$,
        $S(B_h(u)) := S(B_{h-1}(u))$
        for any edge $(u, v)$,
            $S(B_h(u)) := $UNION$(S(B_h(u), S(B_{h-1}(v)))$
    Output $\sum_u$SIZE $(S(B_h(u)))/n^2$
```

Memory usage is $\Theta(n)$, since the sizes of $S(B_{h-1}(u))$ are constant.