Graph mining
SD212
Betweenness centrality

Thomas Bonald

2017 – 2018

These lecture notes introduce the notion of *betweenness centrality*, that quantifies the propensity of nodes to lie on shortest paths between other nodes [3]. An efficient algorithm is presented for computing it. More details can be found in the original paper [1].

1 Notion of betweenness centrality

Let $G = (V, E)$ be a graph of n nodes and m edges, which we assume undirected, unweighted and connected for simplicity. The extension to more general graphs is discussed in Section 4. For all nodes $s, t \in V$, let σ_{st} be the number of shortest paths between s and t, with $\sigma_{st} = 1$ if $s = t$ by convention. For each $u \in V$ and $\{u, v\} \in E$, we denote by $\sigma_{st}(u)$ and $\sigma_{st}(u, v)$ the numbers of shortest paths between s and t containing node u and edge $\{u, v\}$, respectively.

The *betweenness centrality* of any node u is defined as the fraction of shortest paths going through u, summed over all source-destination pairs, that is

$$C(u) = \sum_{s,t \neq u; s<t} \frac{\sigma_{st}(u)}{\sigma_{st}}. \tag{1}$$

Observe that

$$0 \leq C(u) \leq \binom{n-1}{2}.$$

Both bounds are attained in a star network. When normalized by $\binom{n-1}{2}$, the betweenness centrality u is equal to the probability that a random shortest path between a random pair of nodes different from u goes through u.

2 A first algorithm

For computing the betweenness centrality of node u, it is necessary to count the *number* of shortest paths between any pair of nodes s, t. The number of shortest paths from s to any other node t follows from the following adaptation of BFS (Breadth First Search) from node s:

- let $d_{st} = 0$ for $t = s$ and $d_{st} = +\infty$ otherwise (distance from s to t);
- let $\sigma_{st} = 1$ for $t = s$ and $\sigma_{st} = 0$ otherwise (number of shortest paths from s to t);
- in BFS, when exploring a neighbor v of some node u, set $d_{sv} = d_{su} + 1$ if this is the first visit to node v; then increase σ_{sv} by σ_{su} if $d_{sv} = d_{su} + 1$.

This algorithm runs in $O(m)$ time and $O(n)$ space. When applied to all nodes s, this requires $O(mn)$ time and $O(n^2)$ space.

In view of (1), the betweenness centrality of u depends on $\sigma_{st}(u)$, the number of shortest paths between s and t containing u, for any $s, t \neq u$. Observing that

$$\sigma_{st}(u) = \begin{cases} \sigma_{su}\sigma_{ut} & \text{if } d_{st} = d_{su} + d_{ut}, \\ 0 & \text{otherwise,} \end{cases}$$

this takes $O(n^2)$ time. Thus computing the betweenness centrality of all nodes requires $O(n^3)$ time and $O(n^2)$ space.

3 A more efficient algorithm

A more efficient algorithm, proposed by Brandes [1], is based on the following recursion, proved in the appendix:

Theorem 1 *For any $s \neq u$, define*

$$\delta_s(u) = \sum_{t \neq u} \frac{\sigma_{st}(u)}{\sigma_{st}}. \tag{2}$$

Then,

$$\delta_s(u) = \sum_{v : d_{sv} = d_{su}+1} \frac{\sigma_{su}}{\sigma_{sv}} (1 + \delta_s(v)). \tag{3}$$

Observe that the betweenness centrality of u is

$$C(u) = \frac{1}{2} \sum_{s \neq u} \delta_s(u), \tag{4}$$

because each pair s, t with $s < t$ is counted twice in the sum.

For any fixed s, the computation of $\delta_s(u)$ for all u takes $O(m)$ in time and $O(n)$ in space:

- a BFS for computing d_{st} and σ_{st} for all s, t, and obtaining a list of nodes in reverse order of their distance to s;
- the recursion (3), applied to nodes u in reverse order of their distance to s, with initial value $\delta_s(u) = 0$ for all u.

Computing the betweenness centrality of all nodes thus takes $O(mn)$ in time and $O(n)$ in space. The gain compared to the first algorithm is substantial in both time and space. The time complexity of the algorithm can be further reduced to $O(mk)$ by sampling k source nodes s, which provides an approximate value of the betweenness centrality.

4 Extensions

If the graph is disconnected, then we have $\sigma_{st} = 0$ for some node pairs s, t. By convention, we discard these pairs of nodes so that the betweenness centrality of u becomes:

$$C(u) = \sum_{s,t \neq u; s<t; \sigma_{st}>0} \frac{\sigma_{st}(u)}{\sigma_{st}}.$$

Normalizing by $\binom{n-1}{2}$ as before, we get the probability that a random shortest path goes through u. Observe that nodes belonging to large connected components are more likely to be on (shortest) paths of other nodes and thus have typically larger betweenness centralities.

For directed graphs, the betweenness centrality of node u is simply defined by

$$C(u) = \sum_{s,t \neq u; \sigma_{st} > 0} \frac{\sigma_{st}(u)}{\sigma_{st}},$$

so that

$$0 \leq C(u) \leq (n-1)(n-2).$$

The recursion (3) applies and we get

$$C(u) = \sum_{s \neq u} \delta_s(u).$$

For weighted graphs with positive (additive) weights, BFS must be replaced by Dijkstra's algorithm, whose time complexity is $O(m+n \log n)$ when the associate priority queue is implemented through a Fibonacci heap [2]. The distance in the sum of recursion (3) must be replaced by the number of hops from s in the tree of shortest paths from s. The overall time complexity is $O(mn + n^2 \log n)$.

Appendix

Proof of Theorem 1

For any $s, t \neq u$, we have:

$$\sigma_{st}(u) = \sum_{v: d_{sv} = d_{su}+1} \sigma_{st}(u, v).$$

Now for each node v such that $d_{sv} = d_{su} + 1$, we have

$$\sigma_{st}(u, v) = \frac{\sigma_{su}}{\sigma_{sv}} \sigma_{st}(v).$$

We obtain for any $s \neq u$:

$$\begin{aligned}
\delta_s(u) &= \sum_{t \neq u} \frac{\sigma_{st}(u)}{\sigma_{st}}, \\
&= \sum_{v: d_{sv} = d_{su}+1} \frac{\sigma_{su}}{\sigma_{sv}} \sum_{t \neq u} \frac{\sigma_{st}(v)}{\sigma_{st}}, \\
&= \sum_{v: d_{sv} = d_{su}+1} \frac{\sigma_{su}}{\sigma_{sv}} \left(1 + \sum_{t \neq u, v} \frac{\sigma_{st}(v)}{\sigma_{st}} \right), \\
&= \sum_{v: d_{sv} = d_{su}+1} \frac{\sigma_{su}}{\sigma_{sv}} (1 + \delta_s(v)),
\end{aligned}$$

where we have used the fact that $\sigma_{st}(v) = 0$ for $t = u$ whenever $d_{sv} = d_{su} + 1$.

References

[1] U. Brandes. A faster algorithm for betweenness centrality. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[2] M. L. Fredman and R. E. Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM (JACM)*, 34(3):596–615, 1987.

[3] L. C. Freeman. A set of measures of centrality based on betweenness. *Sociometry*, pages 35–41, 1977.