CSI 445/660 – Network Science – Fall 2015

Handout 6.1 – Outline of a Simple Algorithm for Computing Betweenness

Notes:

(a) The outline and running time analysis discussed below are based on the material in Slides 6-37 through 6-43.

(b) In the description of the algorithm, we use "BFS subgraph" of a connected graph $G(V, E)$ to mean the following. Recall that doing a breadth-first-search (BFS) on G results in a spanning tree T of G. The BFS subgraph contains all the edges of T along with all the edges of G that join nodes in *successive* levels. Thus, this subgraph contains all the edges of G *except* those that join nodes at the same level in T. (Given the starting node s for a BFS, the BFS subgraph is unique; this subgraph is needed to ensure that the number of shortest paths from s to the other nodes of G are correctly computed.)

(c) It is straightforward to modify the algorithm for BFS so that it produces the BFS subgraph of $G(V, E)$ in $O(|V| + |E|)$ time.

Input: A connected undirected graph $G(V, E)$ without edge weights.

Output: The betweenness centrality value $\beta(v)$ for each node $v \in V$.

Steps of the Algorithm:

1. **for** each node $s \in V$ **do**
   (a) Construct the BFS subgraph of G rooted at s.
   (b) For each node $t \in V - \{s\}$, compute the value σ_{st}, that is, the number of s-t shortest paths.
2. **for** each node $v \in V$ **do**
   (a) Construct graph G_v from G by deleting v and all the edges incident on v.
   (b) **for** each node $s \in V - \{v\}$ **do**
      i. Construct the BFS subgraph of G_v rooted at s.
      ii. For each node $t \in V - \{v, s\}$, compute the value σ_{st} (i.e., the number of s-t shortest paths) in G_v. (Note that this gives the number of s-t shortest paths that *don't* pass through v in G.)
3. Using the values computed in Steps 1 and 2 above, compute the value of $\beta(v)$ for each $v \in V$.

Running Time Analysis:

- As discussed in the slides, the running time for Steps 1 and 2 are respectively $O(|V|(|V|+|E|))$ and $O(|V|^2(|V| + |E|))$.

- In Step 3, for each node v, finding the value of $\beta(v)$ requires the computation of the sum of $O(|V|^2)$ values. So, the time for computing the $\beta(v)$ values for all the nodes is $O(|V|^3)$.
- The overall running time, which is dominated by Step 2, is $O(|V|^2(|V| + |E|))$.
- This running time is $O(|V|^4)$ for **dense** graphs and $O(|V|^3)$ for **sparse** graphs.