Node Betweenness Centrality

Description

We use Brandes' algorithm to calculate the 'betweenness centrality' for vertices.

The first definition of betweenness centrality came from Anthonisse and Freeman:

The computation of betweenness centrality is computationally very expensive. Networks of up to several hundred nodes can be analyzed, but the original algorithm does not scale to networks with thousands of nodes. In 2001, Ulrik Brandes proposed a more efficient algorithm for betweenness that exploits the extreme sparseness of typical networks to reduce running time from O(n3) to O(n2 + nm) and memory consumption from O(n2) to O(n+m) [ALGDOC: 3] (n is the number of nodes and m is the number of edges). Moreover, other shortest-path based indices, like closeness or radiality, can be computed simultaneously within the same bounds.

A breadth-first search algorithm for unweighted graphs (and Dijkstra's algorithm for weighted graphs) was implemented to calculate the number of shortest paths between any two nodes and to get the set of all predecessor nodes. Both pieces of information are required to calculate the betweenness centrality according to Brandes' formula.

The breadth-first search explores a graph G=(V,E) across the breadth of the frontier between discovered and undiscovered nodes. A node is examined more, the farther its distance from the starting node s. That is, the algorithm discovers all nodes at distance k from the starting node s before discovering any nodes at distance k+1. As a result, one gets the shortest paths to each node v of V that is reachable from s. A shortest path from s to v is the path which consists of the lowest number of edges. To describe the current state of the breadth-first search every node can be colored white, gray or black. At the beginning all nodes are white. When a node gets visited for the first time it is colored non-white. That is, all gray and black nodes have been visited at least once. An already visited node gets the color black if all its child nodes have already been visited (which means that it is colored non-white). On the other hand, an already visited node gets the color gray if it has at least one child node which is not visited yet (which means that it is white). Therefore, the gray nodes describe the frontier between the visited and not yet visited nodes.

The pseudo code for Dijkstra's Algorithm is as follows:

```
define dijkstra(Graph g, Node s)
    initializeSingleSource(g, s)
    S := {}
    Q := getNodes(g)
    while not empty(Q)
        Node u := extractMin( Q )
        AddNode( S, u )
        for each node v in neighbors( u )
            relax( u, v, weight(u,v) )

define initializeSingleSource(Graph g, Node s)
    for each node v in getNodes(g)
        d(v) = INFINITY
        pi(v) = NIL
    d(s) = 0
    pi(s) = NIL
```

Pros & Cons

Applicable to network data only. The algorithm does not scale to more than 5000 nodes.

See Also