A faster Algorithm for Betweeness Centrality

6.886

Joana M. F. da Trindade

Feb 16th 2018

Motivation

Centrality indicates importance of the nodes

- Indicates that a vertex can reach others on relatively short paths

However it may be expensive to compute for larger networks

- As previous 2 presentations showed, requires BFS -> O(m + n)

How can we make it more efficient?

- Can we break it into multiple single source BFS passes?
- Key intuition: count the number of shortest paths that pass by a given node

Types of centrality

$$C_C(v) = \frac{1}{\sum_{t \in V} d_G(v,t)}$$ *closeness centrality* (Sabidussi, 1966)

$$C_G(v) = \frac{1}{\max_{t \in V} d_G(v,t)}$$ *graph centrality* (Hage and Harary, 1995)

$$C_S(v) = \sum_{s \neq v \neq t \in V} \sigma_{st}(v)$$ *stress centrality* (Shimbel, 1953)

$$C_B(v) = \sum_{s \neq v \neq t \in V} \frac{\sigma_{st}(v)}{\sigma_{st}}$$ *betweenness centrality* (Freeman, 1977; Anthonisse, 1971)

Key Contribution

“We introduce more efficient algorithms based on a new accumulation technique that integrates well with traversal algorithms solving the single-source shortest-paths problem, and thus exploiting the sparsity of typical instances”

Key Intuition

To eliminate the need for explicit summation of all pair-dependencies, we introduce first the notion of the *dependency* of a vertex $s \in V$ on a single vertex $v \in V$, defined as

$$\delta_{s\bullet}(v) = \sum_{t \in V} \delta_{st}(v).$$

The crucial observation is that these partial sums obey a recursive relation. In the following special case, this relation is particularly easy to recognize.

Counting number of shortest paths per node

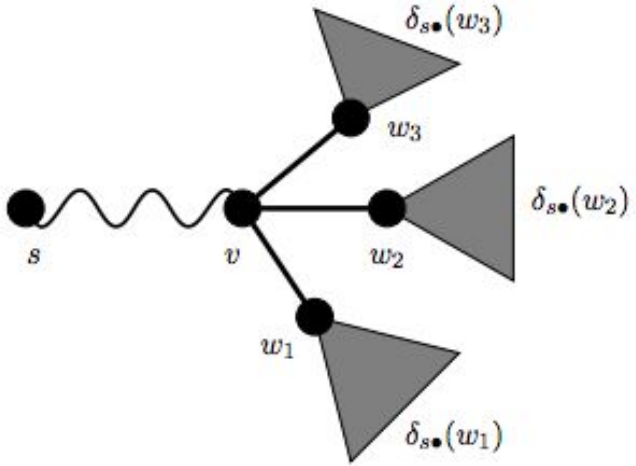

Figure 1: With the assumption of Lemma 5, a vertex lies on all shortest paths to its successors in the tree of shortest paths from the source

Counting number of shortest paths per node

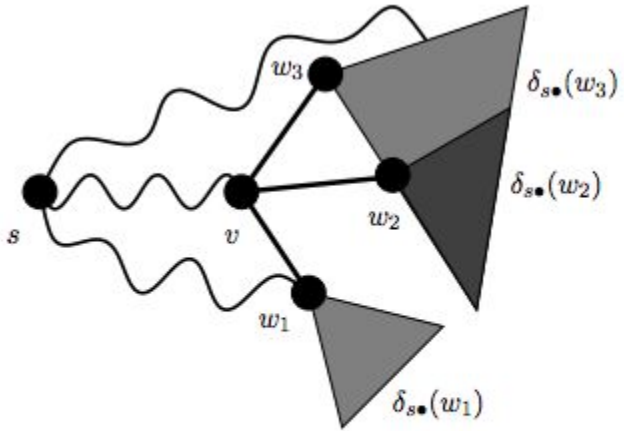

Figure 2: In the general case of Theorem 6, fractions of the dependencies on successors are propagated up along the edges of the directed acyclic graph of shortest paths from the source

Algorithm

Algorithm 1: Betweenness centrality in unweighted graphs

```
C_B[v] ← 0, v ∈ V;
for s ∈ V do
    S ← empty stack;
    P[w] ← empty list, w ∈ V;
    σ[t] ← 0, t ∈ V;   σ[s] ← 1;
    d[t] ← −1, t ∈ V;   d[s] ← 0;
    Q ← empty queue;
    enqueue s → Q;
    while Q not empty do
        dequeue v ← Q;
        push v → S;
        foreach neighbor w of v do
            // w found for the first time?
            if d[w] < 0 then
                enqueue w → Q;
                d[w] ← d[v] + 1;
            end
            // shortest path to w via v?
            if d[w] = d[v] + 1 then
                σ[w] ← σ[w] + σ[v];
                append v → P[w];
            end
        end
    end
    δ[v] ← 0, v ∈ V;
    // S returns vertices in order of non-increasing distance from s
    while S not empty do
        pop w ← S;
        for v ∈ P[w] do δ[v] ← δ[v] + (σ[v]/σ[w]) · (1 + δ[w]);
        if w ≠ s then C_B[w] ← C_B[w] + δ[w];
    end
end
```

Theoretical Results

Theorem 8 *Betweenness centrality can be computed in* $\mathcal{O}(nm + n^2 \log n)$ *time and* $\mathcal{O}(n + m)$ *space for weighted graphs. For unweighted graphs, running time reduces to* $\mathcal{O}(nm)$.

The other shortest-path based centrality measures defined in Section 2 are easily computed during the execution of single-source shortest-paths traversals. The same holds for a recently introduced index called *radiality* (Valente and Foreman, 1998)

$$C_R(v) = \frac{\sum_{t \in V} (D(G) + 1 - d_G(v,t))}{(n-1) \cdot D(G)},$$

Performance

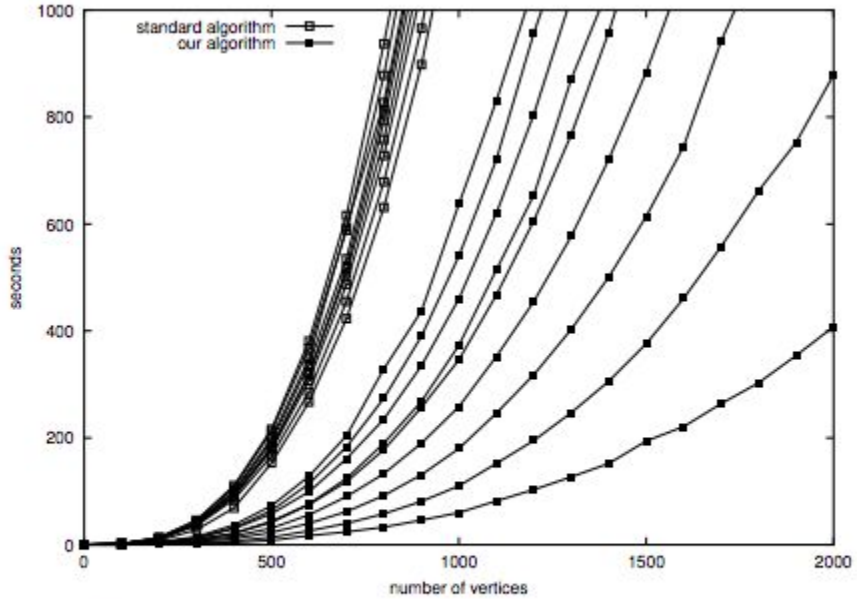

Figure 3: Seconds needed to the compute betweenness centrality index for random undirected, unweighted graphs with 100 to 2000 vertices and densities ranging from 10% to 90%

Performance

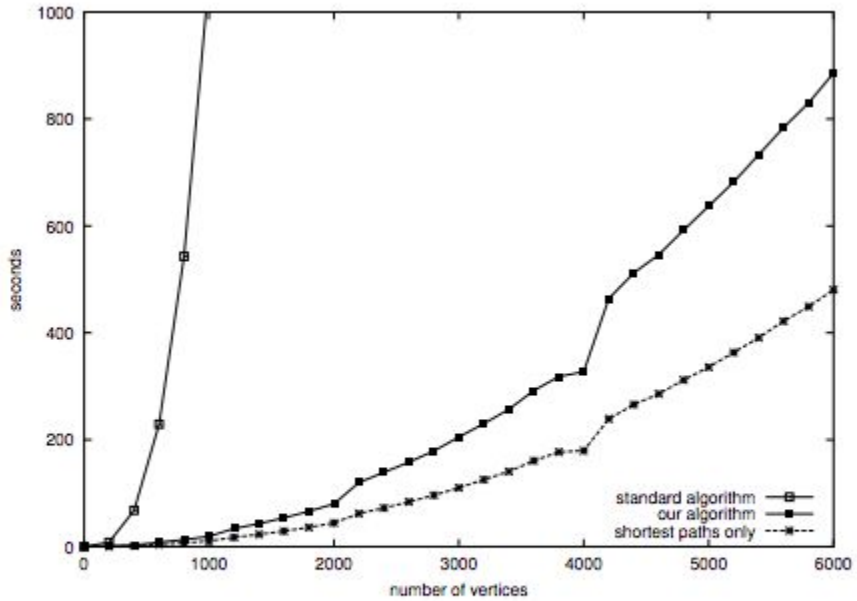

Figure 4: Seconds needed to the compute betweenness centrality index for random undirected, unweighted graphs with constant average degree 20. The funny jumps are attributed to LEDA internals

Conclusion

Nice theoretical results on how to make centrality scale for large networks

Authors were hopeful that it could be used for networks with at least 10,000 nodes