On Variants of Shortest-Path Betweenness Centrality and their Generic Computation [1]

Ulrik Brandes

Department of Computer & Information Science, University of Konstanz

Abstract

Betweenness centrality based on shortest paths is a standard measure of control utilized in numerous studies and implemented in all relevant software tools for network analysis. In this paper, a number of variants are reviewed, placed into context, and shown to be computable with simple variants of the algorithm commonly used for the standard case.

Key words: Betweenness centrality, algorithms, valued networks, load centrality

1 Introduction

Centrality is a core concept for the analysis of social networks, and betweenness is one of the most prominent measures of centrality. It was introduced independently by Anthonisse (1971) and Freeman (1977), and measures the degree to which a vertex is in a position of brokerage by summing up the fractions of shortest paths between other pairs of vertices that pass through it. Betweenness is therefore classified as a measure of mediation in Borgatti and Everett (2006).

While some centrality indices impose requirements on the network data such as undirectedness and connectedness, Anthonisse (1971) defines betweenness on directed networks from the very beginning, and Freeman (1977) stresses applicability to disconnected networks. In this paper, we review and propose variations of the definition of betweenness that either generalize it to even more types of network data, or modify slightly the embodied notion of brokerage.

Email address: Ulrik.Brandes@uni-konstanz.de (Ulrik Brandes).

[1] Research partially supported by DFG under grant Br 2158/2-3.

All of the variants treated in this paper, however, maintain the original model of geodesic trajectories (Borgatti, 2005), i.e., only *shortest* paths are considered in their definition. This excludes alternative models of transportation based on, e.g., network-flow (Freeman et al., 1991) or current-flow (Newman, 2005; Brandes and Fleischer, 2005), and is due to the original motivation for writing this paper, namely to document that none of these variants poses additional algorithmic challenges, as all of them can be calculated with minor modifications of the efficient betweenness algorithm introduced in Brandes (2001).

The contribution of this paper is thus two-fold:

(1) A compilation of shortest-path betweenness variants.
(2) Algorithms for computing them efficiently.

As part of the first contribution we also clarify relations with other measures like stress (Shimbel, 1953) and load (Goh et al., 2001), but there will be no discussion in favor or against substantive applicability of these variant measures. The second contribution is especially valuable for software tools offering betweenness computation, since existing implementations can be adapted with little effort. In fact, most of the modifications discussed in the sequel can be combined in a single implementation. Let me stress again, however, that an evaluation of the appropriateness and usefulness of any of these variants, or their combination, for substantive network research is beyond the scope of this paper. For recent overviews of centrality measures and associated algorithms see Brandes and Erlebach (2005, Chapters 3–5) and Borgatti and Everett (2006).

In the next section, we recall the definition of shortest-path betweenness centrality and outline its efficient computation. The variations are presented in Section 3 together with corresponding modifications to the basic algorithm, and we conclude with a brief discussion.

2 Shortest-Path Betweenness

We will use graph-theoretic terminology neutral to interpretation. See Bollobás (1998) or Diestel (2000) for modern textbooks on graph theory. For introductions to algorithms and their analysis, see Cormen et al. (2001), Goodrich and Tamassia (2002) or Kleinberg and Tardos (2006).

A *directed graph* $G = (V, E)$ consists of a set V of *vertices* (also called nodes, actors) and a set $E \subseteq V \times V$ of directed *edges* (also called links, ties). For a directed edge $e = (u, v) \in E$, v is called the *head* of e, u is called its *tail*, and v is said to be *adjacent* to u.

For our purposes, undirected graphs are equivalently replaced by symmetric directed graphs containing two oppositely directed edges for each undirected edge. Unless explicitly stated otherwise, we will therefore refer to graphs, edges, etc. when we mean directed graphs, directed edges, etc. Also, we may safely assume that there are no loops, i.e. edges connecting a vertex with itself, because they have no influence on betweenness. Multiple and weighted edges will be considered in Section 3.8.

A *path* from a *source* $s \in V$ to a *target* $t \in V$, or (s, t)*-path* for short, is an alternating sequence of vertices and edges $s, (s, v_1), v_1, (v_1, v_2), v_2, \ldots, (v_k, t), t$ starting with s and ending with t, such that the vertices before and after an edge are its tail and head, respectively. The *length* of an (s, t)-path is the number of edges it contains, and the *distance*, $dist(s, t)$, from s to t is defined as the minimum length of any (s, t)-path if one exists, and undefined otherwise. Note that $s = t$ implies $dist(s, t) = 0$.

Denote by $\sigma(s, t)$ the number of *shortest* (s, t)-paths (sometimes called geodesics), and let $\sigma(s, t|v)$ be the number of shortest (s, t)-paths passing through some vertex v other than s, t. If $s = t$, let $\sigma(s, t) = 1$, and if $v \in \{s, t\}$, let $\sigma(s, t|v) = 0$. Then, the *(shortest-path) betweenness* $c_B(v)$ of a vertex $v \in V$ is defined to be

$$c_B(v) = \sum_{s,t \in V} \frac{\sigma(s, t|v)}{\sigma(s, t)}$$

where $\frac{0}{0} = 0$ by convention. The measure is therefore usually interpreted as the degree to which a vertex has control over pair-wise connections between other vertices, based on the assumption that the importance of connections is equally divided among all shortest paths for each pair.

As pointed out already in Freeman (1977), the definition of betweenness applies to disconnected graphs without modification. Though the distance between two vertices not connected by a directed path is undefined, this also means that the number of shortest paths between them is zero, so that the resulting zero contribution is exactly what is desired.

2.2 Basic Algorithm

Efficient computation of betweenness is based on the fact that the cubic number of *pair-wise dependencies* $\delta(s,t|v) = \sigma(s,t|v)/\sigma(s,t)$ can be aggregated without computing all of them explicitly. Defining one-sided dependencies

$$\delta(s|v) = \sum_{t \in V} \delta(s,t|v) \ ,$$

for all $s, v \in V$, we can exploit that

$$\delta(s|v) = \sum_{\substack{w\,:\,(v,w)\in E \text{ and} \\ dist(s,w)=dist(s,v)+1}} \frac{\sigma(s,v)}{\sigma(s,w)} \cdot (1 + \delta(s,w)) \ . \tag{1}$$

as shown in Brandes (2001). This recursive relation asserts that the dependency of a vertex s on some v can be compiled from dependencies on vertices one edge farther away.

Since $c_B(v) = \sum_{s \in V} \delta(s|v)$, Algorithm 1 computes betweenness by iterating over all vertices $s \in V$, each time computing $\delta(s|v)$ for all $v \in V$ in two phases. The first phase is a breadth-first search, in which distances and shortest-path counts from s are determined. The second phase visits all vertices in reverse order of their discovery, i.e. those farthest from s first, to accumulate dependencies according to Equation (1).

Note that this algorithm applies to directed and undirected graphs alike, although it yields doubled scores for undirected graphs because each pair is considered twice, once in either direction. This is of course easily accounted for by halving the scores. More generally, the centrality index computed by Algorithm 1 can be normalized in any of the usual ways, e.g. by dividing by the maximum possible score in a network of the same order (number of vertices) as suggested by Freeman (1979), or by the sum of all scores to analyze the distribution of values. Since we focus on algorithms and normalization involves only multiplication of all scores with a constant, we ignore it from this point on. Note also that the centrality definition in Anthonisse (1971) is already for directed graphs. It was re-stated in Gould (1987) and White and Borgatti (1994) where normalization and centralization issues are discussed in addition.

We will refer to Algorithm 1 as the *basic algorithm*, and modify it to compute variants of betweenness. The notation used for the basic algorithm serves to avoid repeating much of the same pseudo-code over and over again. Instructions that make up the main steps are grouped and labeled starting with a ▼. In the next section, we will only spell out the main steps that actually need

Algorithm 1: Shortest-path vertex betweenness (Brandes, 2001)

input: directed graph $G = (V, E)$
data: queue Q, stack S (both initially empty) and for all $v \in V$:
$dist[v]$: distance from source
$Pred[v]$: list of predecessors on shortest paths from source
$\sigma[v]$: number of shortest paths from source to $v \in V$
$\delta[v]$: dependency of source on $v \in V$
output: betweenness $c_B[v]$ for all $v \in V$ (initialized to 0)

```
for s ∈ V do
    ▼ single-source shortest-paths problem
        ▼ initialization
            for w ∈ V do Pred[w] ← empty list
            for t ∈ V do dist[t] ← ∞;   σ[t] ← 0
            dist[s] ← 0;   σ[s] ← 1
            enqueue s → Q
        while Q not empty do
            dequeue v ← Q;   push v → S
            foreach vertex w such that (v, w) ∈ E do
                ▼ path discovery // — w found for the first time?
                    if dist[w] = ∞ then
                        dist[w] ← dist[v] + 1
                        enqueue w → Q
                ▼ path counting // — edge (v, w) on a shortest path?
                    if dist[w] = dist[v] + 1 then
                        σ[w] ← σ[w] + σ[v]
                        append v → Pred[w]
    ▼ accumulation // — back-propagation of dependencies
        for v ∈ V do δ[v] ← 0
        while S not empty do
            pop w ← S
            for v ∈ Pred[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]
```

to be modified, and ▶ collapse inner steps that do not.

3 Variants and their Computation

We have already argued that the basic algorithm is suitable for vertex betweenness in both directed and undirected graphs. In this section, we discuss modifications that require adjustment of the basic algorithm.

3.1 Endpoints

Depending on the structure modeled, it may be inappropriate to have pairs of vertices depend on intermediaries, but not on themselves (or at least one of them as the source or target). In an information exchange network, for instance, one might argue that the source of information has just as much control over its content as anyone passing it on.

If we adjust the definition of $\delta(s,t|v)$ to equal one rather than zero in case $v \in \{s,t\}$, then the betweenness score of every $v \in V$ will go up by the number of vertices that can be reached from v plus the number of vertices that can reach v.

In a graph in which every vertex can reach every other vertex on a directed path, the increase is uniformly $2n-2$ for all vertices, if we exclude the trivial paths starting and ending at the same vertex to maintain that isolated vertices have zero centrality. Such constant increase is easily accounted for in the computation and does not change the rank order of centralities. If, however, some vertex can not reach every other vertex, the increase caused by including endpoints may be non-uniform and thus requires modifying the basic algorithm to actually determine the number of times a vertex is a source or target of a connection pair.

Algorithm 2: Including endpoints

▼ **accumulation**

$c_B[s] \leftarrow c_B[s] + (|S| - 1)$ // – number of times s is a source

for $v \in V$ **do** $\delta[v] \leftarrow 0$

while S *not empty* **do**

pop $w \leftarrow S$

for $v \in Pred[w]$ **do** $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + \delta[w])$

if $w \neq s$ **then** $c_B[w] \leftarrow c_B[w] + \delta[w] + 1$ // – w is target of s once

It is easily seen from Algorithm 2 that the betweenness increase due including dependencies on sources and targets can be determined separately. Since exactly those vertices that can be reached from a source s (including s itself) are placed on stack S, the size of the stack after solving the single-source shortest-

paths problem minus one equals the increase of $c_B(s)$ due to including s as a source. The increase due to including targets can be added one by one, since there is a connected pair s, t involving t as a target if and only if t is popped from stack S during the accumulation for source s.

If desired, we are therefore in a position to restrict the inclusion of endpoints to either sources or targets, e.g., because we are in a scenario in which only the source has control over a connection, but not the target.

3.2 Proxies

Above we allowed for the possibility that sources and/or targets have an influence. In extreme cases, one might even argue that only the source or the target have control over their pair-wise connection. Betweenness would then reduce to determining the number of actors that can be reached from a source or that can reach a target. In Algorithm 2, this corresponds to simply excluding the dependencies δ from the betweenness summation.

Similarly, we might attribute a special role to yet another position, namely the first or last actor other than source and target. Borgatti argues [2] that the final intermediate can be special, because it is with this actor that the target interacts directly and may thus depend on entirely. Penultimate actors in a shortest-path connection are thus seen as proxies having full control over the connection using this path.

Algorithm 3: Proximal betweenness

output: proximal betweenness $c_{B_{ps}}[v]$, $c_{B_{pt}}[v]$ for all $v \in V$ (initialized to 0)

▼ **accumulation**

```
for v ∈ V do δ[v] ← 0
while S not empty do
    pop w ← S
    for v ∈ Pred[w] do
        δ[v] ← δ[v] + (σ[v]/σ[w]) · (1 + δ[w])
        if v ≠ s then
            c_Bps[v] ← c_Bps[v] + σ[v]/σ[w]    // – v is proximal source for w
        else
            c_Bpt[w] ← c_Bpt[w] + δ[w]    // – w is proximal target for v = s
```

[2] Stephen P. Borgatti, University of Kentucky (personal communication, 2007).

Borgatti therefore defines *proximal source betweenness* as

$$c_{B_{ps}}(v) = \sum_{\substack{s \in V \\ t\,:\,(v,t) \in E}} \frac{\sigma(s,t|v)}{\sigma(s,t)} \ .$$

and a *proximal target betweenness*, $c_{B_{pt}}$, can be defined symmetrically. Both notions of agency are combined into one index, e.g., by taking the sum. Algorithm 3 shows how to accumulate betweenness of both source and target proxies by adding the accumulated dependencies only to neighbors of the source (because they are proximal targets) and by adding the fraction of shortest paths via a predecessors to its betweenness (because it is a proximal source). Analogous to the case of source and target endpoints, the computation can be restricted to either proximal sources or targets; the double counting of vertices neighboring both source and target can be avoided if desired.

Note that, in the spirit of the original betweenness, we have opted to leave out endpoints in Algorithm 3, i.e., a source or target is not counted as a proximal source or target in a path without intermediate vertices.

3.3 Bounded-distance Betweenness

The standard measure of betweenness considers shortest paths irrespective of their length. Since very long connections may not be realistic for network relations such as friendship, Borgatti and Everett (2006) define the k-betweenness of a vertex as the sum of dependencies of pairs at most k apart, i.e. only contributions from shortest paths of length bounded by a constant k are included. Let

$$c_{B(k)}(v) = \sum_{s,t \in V\,:\,dist(s,t) \leq k} \frac{\sigma(s,t|v)}{\sigma(s,t)}$$

and note that this equals standard shortest-path betweenness for $k = n - 1$. For $k = 2$, it is similar to ego network betweenness (Everett and Borgatti, 2005), but differs in that shortest paths of length two with a non-neighbor as intermediate are considered as well.

To exclude paths longer than k, we only have to stop the breadth-first search of the basic algorithm when a vertex of distance k is reached (cf. Algorithm 4).

3.4 Distance-scaled Betweenness

Another, more sophisticated variant also mentioned in Borgatti and Everett (2006) does not filter out paths of excessive length, but weighs all shortest

Algorithm 4: k-Betweenness

input: directed graph $G = (V, E)$, length bound k
output: k-betweenness for all $v \in V$ (initialized to 0)

```
for s ∈ V do
    ▼ single-source shortest-paths problem
        ▶ initialization
        while Q not empty do
            dequeue v ← Q;   push v → S
            foreach vertex w such that (v, w) ∈ E do
                ▼ path discovery  // — w found for the first time?
                    if dist[w] = ∞ then
                        dist[w] ← dist[v] + 1
                        if dist[w] ≤ k then enqueue w → Q
                ▶ path counting
```

paths inversely proportional to their length as in

$$c_{B_{dist}}(v) = \sum_{s \neq t \in V} \frac{1}{dist(s,t)} \cdot \frac{\sigma(s,t|v)}{\sigma(s,t)} \ .$$

The rationale here is that, the longer a path, the less valuable it may be to control it. We refer to this measure as *length-scaled betweenness*, because the dependencies are scaled by a factor that depends only on the length of a shortest path and is thus the same for all its inner vertices. This should be compared to the next measure introduced in this subsection. Since each new target enters the accumulation by adding one to the dependencies accumulated so far, we replace this value by its weighted equivalent, $1/dist(s,t)$, as shown in Algorithm 5.

Algorithm 5: Length-scaled betweenness

```
▼ accumulation
    for v ∈ V do δ[v] ← 0
    while S not empty do
        pop w ← S
        for v ∈ Pred[w] do
            δ[v] ← δ[v] + (σ[v]/σ[w]) · (1/dist(s,t) + δ[w])
        if w ≠ s then c_{B_dist}[w] ← c_{B_dist}[w] + δ[w]
```

A complementary variant is implicitly introduced in Geisberger et al. (2008), where the actual goal is the efficient approximation of standard betweenness centrality. Although designed to be used in a betweenness estimator, it appears to be a plausible variant of betweenness by itself, because the relative position

Algorithm 6: Linearly scaled betweenness (Geisberger et al., 2008)

▼ **accumulation**
  for $v \in V$ **do** $\delta[v] \leftarrow 0$
  while S *not empty* **do**
    pop $w \leftarrow S$
    for $v \in Pred[w]$ **do**
      $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (\frac{1}{dist(s,t)} + \delta[w])$
    if $w \neq s$ **then** $c_{B_{lin}}[w] \leftarrow c_{B_{lin}}[w] + dist(s,v) \cdot \delta[w]$

of vertices in a shortest path are taken into account. We define *linearly scaled betweenness*, $c_{B_{lin}}$, by

$$c_{B_{lin}}(v) = \sum_{s \neq t \in V} \frac{dist(s,v)}{dist(s,t)} \cdot \frac{\sigma(s,t|v)}{\sigma(s,t)} \ .$$

Different from the previous variant, it is not the length of a path that modifies the dependency on v, but v's relative distance from the source. The farther away from the source, and hence the closer to the target, the more influential a vertex is. This measure thus generalizes proximal source betweenness, and can be used to generalize proximal target betweenness by replacing $dist(s,v)$ with $dist(v,t)$. Observe, however, that linearly scaled betweenness equals the usual betweenness in undirected graphs, since the relative distances on (s,t)- and (t,s)-paths sum to one.

The additional factor of $dist(s,v)$ is not propagated in Algorithm 6, because it must not contribute to vertices closer to the root (which have their own, smaller factor).

3.5 Edge Betweenness

A natural extension of betweenness to edges is obtained by replacing $\sigma(s,t|v)$ in the definition of vertex betweenness by $\sigma(s,t|e)$, the number of shortest (s,t)-paths containing the edge e. This version of edge betweenness is already discussed by Anthonisse (1971), and a prominent application is the heuristic clustering approach of Newman and Girvan (2004), where edges of maximum betweenness are removed iteratively to decompose a graph into relatively dense subgraphs.

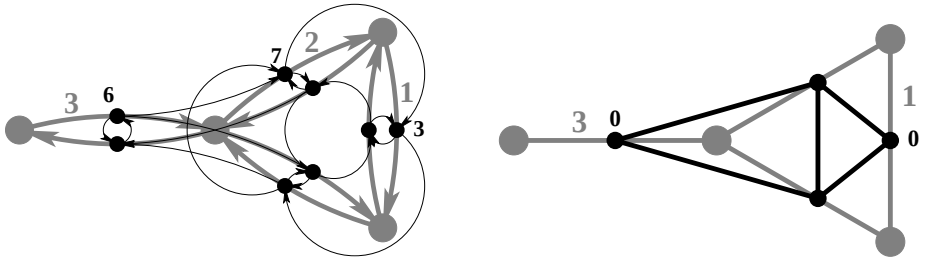
The basic algorithm is based on the back-propagation of dependencies from a vertex to its predecessors, and the value propagated along an edge is exactly its betweenness. This immediately follows from the proof of correctness for Equation (1) given in Brandes (2001), and leads to the simple modification

Algorithm 7: Edge betweenness

output: betweenness $c_B[q]$ for $q \in V \cup E$ (initialized to 0)

▼ **accumulation**

```
for v ∈ V do δ[v] ← 0
while S not empty do
    pop w ← S
    for v ∈ Pred[w] do
        c ← (σ[v] / σ[w]) · (1 + δ[w])
        c_B[(v, w)] ← c_B[(v, w)] + c
        δ[v] ← δ[v] + c
    if w ≠ s then c_B[w] ← c_B[w] + δ[w]
```



(a) directed example: three edges with different rank order

(b) undirected example: two edges with betweenness scores that no line graph centrality can match

Fig. 1. Edge betweenness in a graph (gray) is different from vertex betweenness in its line graph (black).

shown in Algorithm 7.

An alternative and more general approach for defining edge centralities is based on the following construction. The *line graph* $\mathcal{L}(G)$ of a graph G consists of a vertex uv for each edge (u, v) of G, and there is an edge between uv and wx, if and only if $v = w$. If G is undirected, the undirected line graph is obtained by identifying the two vertices created for two directed edges corresponding to one undirected edge. In other words, the line graph of an undirected graph $G = (V, E)$ is the graph with vertex set E in which two vertices are adjacent, if and only if the corresponding edges share a vertex.

The following theorem indicates that the approach via vertex betweenness in the line graph does not yield the same results as the direct definition of edge betweenness, and that the difference between the two approaches cannot be overcome. Koschützki et al. (2005) also give substantive reasons why the line graph approach, though elegant, may be inappropriate for many application scenarios.

Theorem 1 *Vertex betweenness in a line graph $\mathcal{L}(G)$ differs from edge betweenness in the original graph G. In fact, if G is undirected, edge betweenness in G may not equal any vertex centrality that can be defined on $\mathcal{L}(G)$.*

PROOF. Figure 1 shows an example of a graph for which edge betweenness in G and vertex betweenness in $\mathcal{L}(G)$ yield different rank orders. In its undirected version the difference is even more significant, since G contains two edges with different edge betweenness that correspond to structurally equivalent vertices (i.e., vertices with identical neighborhood) in $\mathcal{L}(G)$. Note that any centrality that takes only the graph structure into account must be invariant under automorphisms of the graph, i.e. must in particular assign the same value to structurally equivalent vertices.

3.6 Group Betweenness

Everett and Borgatti (1999) define the *group betweenness*, $c_B(C)$, of a subset $C \subseteq V$ as

$$c_B(C) = \sum_{s,t \in V} \frac{\sigma(s,t|C)}{\sigma(s,t)}$$

where the numerator counts the number of shortest (s,t)-paths containing any vertex of C as an inner vertex.

Algorithm 8: Group betweenness

input: directed graph $G = (V, E)$, group $C \subseteq V$
output: group betweenness $c_B(C)$ (initialized to 0)

▼ **accumulation**
  for $v \in V$ **do** $\delta[v] \leftarrow 0$
  while S *not empty* **do**
    pop $w \leftarrow S$
    for $v \in Pred[w]$ **do**
      if $w \in C$ **then** $i \leftarrow 0$ **else** $i \leftarrow \delta[w]$
      $\delta[v] \leftarrow \delta[v] + \frac{\sigma[v]}{\sigma[w]} \cdot (1 + i)$
    if $w \in C \setminus \{s\}$ **then** $c_B(C) \leftarrow c_B(C) + \delta[w]$

The centrality of a given subset $C \subseteq V$ can be determined during accumulation by disregarding dependencies that have already been accounted for because of another member of C further away from the source. If a vertex w is in C, the dependencies accumulated so far are added to the betweenness of C, but not propagated further to avoid double-counting. In effect, the dependency of the source s on C is built up from the dependencies of s on the last vertex

$w \in C$ before the respective target. The necessary modifications of the basic algorithm are given in Algorithm 8.

While this algorithm is useful to determine the betweenness centrality of a given group, it is not useful for finding a central group. Typically, the interest would be in *small* groups with *high* betweenness, but little is known about the computational tractability of the various optimization problem formulations resulting from the combination of these two criteria.

Observe that we need to do the equivalent of a full vertex betweenness calculation to determine the centrality of a single group. Puzis et al. (to appear) describe an algorithm that, after some $\Theta(n^3)$ preprocessing including one vertex betweenness calculation, requires for each of any number of groups time cubic in its size. Provided there is enough memory to store the $\Theta(n^2)$ preprocessing data, this approach is much more efficient if several small to medium sized groups need to be evaluated.

3.7 Actors of Sorts

The actors in a network need not be of the same kind, or may differ with respect to attributes of interest. Such scenarios are frequently modeled by networks that are referred to as two-mode or, more generally, k-mode networks. However, two-mode networks are also often equated with bipartite graphs and represented in rectangular matrices, in other words restricted to cases where there are no ties between actors of the same mode.

To avoid the confusion with k-partite graphs, we here refer to networks with a partitioned set of actors, who may nevertheless be connected independent of the partition, as *multi-sort networks*, and the partition classes are called *sorts*.

Flom et al. (2004) introduce variants of betweenness to determine actors that bridge between actors of different sorts, defined, e.g., by persons having an infectious disease and persons that do not. Given a graph $G = (V, E)$ with a partition of the vertices into sorts A and B, they define two measures Q_1, Q_2 in which dependencies are counted only for pairs of vertices s, t of opposite sorts as follows. Let σ_1 and δ_1 be defined like σ and δ, but restricted to paths that consist of a sequence of vertices of one sort possibly followed by a sequence of vertices of the other sort, i.e. along which group membership changes at most once.

$$c_{Q_1}(v) = \sum_{s \in A,\, t \in B} \delta_1(s, t|v) + \sum_{s \in B,\, t \in A} \delta_1(s, t|v)$$

$$c_{Q_2}(v) = \sum_{s \in A,\, t \in B} \delta(s,t|v) + \sum_{s \in B,\, t \in A} \delta(s,t|v)$$

In the undirected graphs considered in Flom et al. (2004), s and t are chosen from A and B symmetrically. For directed graphs, it is however relevant whether paths are restricted to run from A to B or vice versa. The above definitions are easily adjusted by leaving out one of the two sums. Normalization is disregarded again, because it does not pose algorithmic challenges.

Algorithm 9: Q-measures

input: directed graph $G = (V, E)$ with two sorts of vertices $V = A \uplus B$
output: Q-measures $c_{Q_1}[v], c_{Q_2}[v]$ for all $v \in V$ (initialized to 0)

```
for s ∈ V do
    ▼ single-source shortest-paths problem
        ▶ initialization
        for t ∈ V do σ1[t] ← 0
        σ1[s] ← 1
        while Q not empty do
            dequeue v ← Q;   push v → S
            foreach vertex w such that (v, w) ∈ E do
                ▶ path discovery
                ▼ path counting
                    if dist[w] = dist[v] + 1 then
                        if s, v or v, w of same sort then σ1[w] ← σ1[w] + σ1[v]
                        σ[w] ← σ[w] + σ[v]
                    append v → Pred[w]
    ▼ accumulation
        for v ∈ V do δ[v] ← 0;   δ1[v] ← 0
        while S not empty do
            pop w ← S
            for v ∈ Pred[w] do
                if s, w of different sort then i ← 1 else i ← 0
                δ1[v] ← δ1[v] + (σ1[v] / σ1[w]) · (i + δ1[w])
                δ[v] ← δ[v] + (σ[v] / σ[w]) · (i + δ[w])
            if w ≠ s then
                cQ1[w] ← cQ1[w] + δ1[w]
                cQ2[w] ← cQ2[w] + δ[w]
```

An efficient algorithm to calculate the above measures is stated as an open problem in Flom et al. (2004), but can in fact be obtained by modifying the basic algorithm as shown in Algorithm 9.

The see why this algorithm works, let us first consider Q_2. Similar to the dependency filtering used for group betweenness above, we modify the propagation of dependencies. Instead of blocking the propagation of dependencies from vertices farther from the source, however, we here propagate dependencies only for targets that are of a sort different from the source s.

Since the same pairs of sources and target are considered, the same strategy works for Q_1, but in addition we have to restrict the shortest paths counted in σ_1 to those that change sorts at most once. This is done during breadth-first search from a source s by passing on path counts from a predecessor v to some w only if at least one of the following two statements is true: s and v are of the same sort, or v and w are of the same sort. This is because otherwise s and w are of the same sort and v is an intermediate vertex of the other sort, so that sorts are changed twice along the (s, w)-path via v.

In principle, both definition and computation strategy carry over to k-sort networks, but the generalization is not straightforward. This is because of the exponentially growing numbers of combinations of sorts for source-target pairs and admissible sort changes along paths.

3.8 Valued Networks and Multigraphs

Dichotomous ties are either present or absent, and therefore frequently yield only coarse approximations to the actual relationships being modeled. If valued ties are measured because they represent a situation more accurately, also the definition of betweenness centrality should be based on these values.

Alternative measures of betweenness for valued networks have been proposed (Freeman et al., 1991; Newman, 2005), but are not based on shortest paths. In fact, different models of transportation (Borgatti, 2005) are assumed, so that these measures can not be seen as proper generalizations of shortest-path betweenness.

Although shortest-path betweenness centrality can indeed be generalized to valued networks while preserving the underlying assumptions, it is important to be clear about the semantics of those values, so that we will actually end up with two different generalizations.

This is in resonance with the two essential factors that determine shortest-path betweenness centrality, namely the length of paths (default: number of edges) and their relative weight (default: uniform). Both these factors can be adapted in the presence of edge values.

The common generalization of shortest paths to graphs with edge values con-

Algorithm 10: Betweenness in valued networks (edge lengths)

input: directed graph $G = (V, E)$ with edge lengths $\lambda\colon E \to \mathbb{R}_{>0}$
data: priority queue Q with keys $dist[\cdot]$

```
▼ single-source shortest-paths problem
    ▶ initialization
    while Q not empty do
        extract v ← Q with minimum dist[v];   push v → S
        foreach vertex w such that (v, w) ∈ E do
            ▼ path discovery  // — shorter path to w?
                if dist[w] > dist[v] + λ(v, w) then
                    dist[w] ← dist[v] + λ(v, w)
                    insert/update w → Q with new key;   σ[w] ← 0;
                    Pred[w] ← empty list
            ▼ path counting
                if dist[w] = dist[v] + λ(v, w) then
                    σ[w] ← σ[w] + σ[v]
                    append v → Pred[w]
```

sists in a re-definition of the length of a path as the sum of values on its edges. This is a proper generalization, since assigning uniform values of one to the edges of a non-valued graph yields a path length that equals the number of edges. The values are referred to as *edge lengths*, and shortest paths according to this generalization can be computed in much the same way by replacing the breadth-first search of the basic algorithm by a more general algorithm for the single-source shortest-path problem.

The generalization interpreting values as lengths is referred to as weighted betweenness in Brandes (2001), where also the use of Dijkstra's algorithm (see any of the above cited textbooks on algorithms) is proposed. The necessary modifications are shown in Algorithm 10 and basically consist of replacing the breadth-first search queue by a priority queue, which worsens the running time bound from $\mathcal{O}(nm)$ to $\mathcal{O}(nm + n^2 \log n)$, though.

Interpreting edge values as lengths implies, however, that higher values reflect weaker ties. If higher values indicate stronger ties, for instance in a communication network with frequency-of-contact values on the edges, one might instead consider reversing the order of all edge values x, e.g., by subtracting from an upper bound like the maximum plus one, or by taking the inverse $\frac{1}{x}$ or a negative exponential e^{-x}. With these transformed values, the previous algorithm can still be applied, but its use can be problematic, since a sum of such rescaled values might not reflect the intuition of distance between two vertices.

Algorithm 11: Betweenness in valued networks (edge weights/multiplicities)

input: directed graph $G = (V, E)$ with edge weights $\omega\colon E \to \mathbb{R}_{>0}$

▼ **path counting**

```
if dist[w] = dist[v] + 1 then
    σ[w] ← σ[w] + ω(v, w) · σ[v]
    append v → Pred[w]
```

An alternative generalization therefore uses such values in the definition of the relative weight rather than the length of a path. Because of the following often-used correspondence between multigraphs and integer edge-valued graphs, a reasonable definition is given by the product of a path's edge values.

If betweenness centrality is computed on a multigraph, i.e. a graph in which loops (which can be ignored) and multiple edges connecting the same pair of vertices are allowed, the number of shortest paths connecting two vertices depends on the multiplicity of their edges: tripling an edge of a path results in three different paths of the same length, because either copy of the tripled edge can be used. If more than one edge has multiplicity larger than one, then any instance of one edge combined with any instance of another edge yields a different path, so that the total number of paths obtained from a generic path is the product of the multiplicities of its edges.

The above edge multiplicities are integer, so that a multigraph can be replaced by a valued simple graph for computing betweenness centralities. In reverse, we can define the betweenness of an integer valued graph via the interpretation as a multigraph. This definition of path weights straightforwardly generalizes to positive real values on the edges. Higher values thus correspond to stronger ties, and Algorithm 11 highlights the simple modifications necessary to adapt the basic algorithm.

It should be noted that this generalization is different from Newman (2004), where unweighted edge betweenness scores are divided by the weight of their corresponding edge. This transformation is purely local, because for the score of an edge it does not matter whether and how the remaining graph is weighted, and it has no influence on vertex betweenness.

Finally, our modifications can be combined for networks with both lengths and weights on the edges.

Fig. 2. Contribution of a single pair (s, t) to load (*left*) and betweenness (*right*).

3.9 Stress and Load

It can be difficult to recognize the definition of betweenness from descriptions given in the literature (e.g., De et al. 2004) and betweenness has actually been considered equal to other measures that we discuss next.

Algorithm 12: Stress centrality

output: stress $c_S[v]$ for all $v \in V$ (initialized to 0)

▼ **accumulation**

```
for v ∈ V do δ[v] ← 0
while S not empty do
    pop w ← S
    for v ∈ Pred[w] do δ[v] ← δ[v] + (1 + δ[w])
    if w ≠ s then c_S[w] ← c_S[w] + σ[w] · δ[w]
```

Betweenness centrality is occasionally described imprecisely as the number of shortest paths passing through a vertex (e.g., Perer and Shneiderman 2006; Lämmer et al. 2006). This description translates into a measure that not only yields different values, but also different rankings. It does yield a measure that can be regarded a centrality, though, and has in fact been referred to as *stress* long before the introduction of betweenness (Shimbel, 1953). To compute stress centrality with the basic betweenness algorithm we simply drop the weighting factor during accumulation. This is a slight abuse of the use of $\delta[w]$, since it no longer stores the full dependency, but only the number of shortest-path suffixes beginning at w. When a vertex is finished, this is corrected by multiplication with the number $\sigma[w]$ of matching prefixes (see Algorithm 12).

Another frequent confusion is with a measure called *load*, introduced in Goh et al. (2001) where it is said to equal betweenness. Load centrality is specified via the following process: each node sends a unit amount of some commodity to each other node. Starting from the respective source, the commodity is always passed to the adjacent vertices closest to the target, and in case there is more than one such vertex the commodity is divided equally among them. The total amount of the commodity passing through a vertex during all these exchanges defines its load. Figure 2 gives an example for the contribution of

a single ordered pair.

This confusion may be related to the (incorrect) betweenness algorithm given in Newman (2001), which essentially proceeds like Algorithm 1 but propagates equal shares of dependency during accumulation (see also Newman 2006). Thus, this algorithm actually calculates load centrality when applied to undirected graphs. This is because their symmetry renders the following consideration irrelevant. For directed graphs, it is important to take into account that the commodity volume is split at branching points and aggregated at meeting points on the way from the source to the target. If equal shares are propagated backwards during accumulation, the actions of splitting and aggregating are swapped.

Algorithm 13: Load centrality

output: load $c_L[v]$ for all $v \in V$ (initialized to 0)

for $s \in V$ **do**
- ▼ **single-source shortest-paths problem**
  - ▶ **initialization**
  - **while** Q *not empty* **do**
    - dequeue $v \leftarrow Q$; push $v \rightarrow S$
    - **foreach** *vertex* w *such that* $(w, v) \in E$ **do**
      - ▶ **path discovery**
      - ▶ **path counting**
- ▼ **accumulation**
  - **for** $v \in V$ **do** $\delta[v] \leftarrow 1$
  - **while** S *not empty* **do**
    - pop $w \leftarrow S$
    - **for** $v \in Pred[w]$ **do** $\delta[v] \leftarrow \delta[v] + \frac{\delta[w]}{|Pred[w]|}$
    - $c_L[w] \leftarrow c_L[w] + \delta[w]$

Algorithm 13 therefore determines load centrality in directed graphs by solving the single-source shortest-paths problems on the reverse graph, i.e. the graph obtained by replacing every edge (u, v) by an edge (v, u), so that splits and aggregations happen in the right places. Note that our formulation of the algorithm includes commodity at endpoints in load scores, which we believe was intended in Goh et al. (2001).

The variants of betweenness discussed previously also apply to stress and load.

4 Discussion

We have discussed several variants of betweenness centrality, in which either the interest is shifted, e.g., to edges, or the range of applicability is extended, e.g., to valued networks. Unlike related measures such as network-flow betweenness (Freeman et al., 1991), current-flow betweenness (Newman, 2005; Brandes and Fleischer, 2005), or load (Goh et al., 2001), these do not alter the underlying model of transportation along geodesic trajectories (Borgatti, 2005).

For all these variants, small modifications of the commonly used algorithm for the standard case resulted in algorithms with the same asymptotic time complexity (except for length-valued edges), and in fact with the same structure of computation. The latter facilitates transfer of these modifications to, e.g., betweenness approximation in large networks based on pivot sampling (Brandes and Pich, 2007; Geisberger et al., 2008).

Remaining challenges include proper normalization in valued networks on the modeling side and efficient re-computation in dynamically changing networks on the algorithmic side. A restricted version of the latter problem is the bottleneck in an edge-betweenness based clustering algorithm (Newman and Girvan, 2004).

Another family of variations beyond the scope of this paper consists of betweenness and other centralities on time-valued networks. Research on social networks in which ties exist over time intervals is in its infancy (see, e.g., Moody 2006), and while many notions of time-dependent paths and distances have been introduced in other branches of research, few of them have actually been translated and studied in the context of social networks.

Acknowledgment. I am grateful to Steve Borgatti, who shared ideas on distance-aware variants of betweenness and made other useful suggestions, and to Peter Sanders and two anonymous reviewers, whose comments were helpful in improving the presentation.

References

Anthonisse, J. M., October 1971. The rush in a directed graph. Tech. Rep. BN 9/71, Stichting Mathematisch Centrum, 2e Boerhaavestraat 49 Amsterdam.

Bollobás, B., 1998. Modern Graph Theory. Vol. 184 of Graduate Texts in Mathematics. Springer-Verlag.

Borgatti, S. P., 2005. Centrality and network flow. Social Networks 27, 55–71.

Borgatti, S. P., Everett, M. G., 2006. A graph-theoretic perspective on centrality. Social Networks 28, 466–484.

Brandes, U., 2001. A faster algorithm for betweenness centrality. Journal of Mathematical Sociology 25 (2), 163–177.

Brandes, U., Erlebach, T. (Eds.), 2005. Network Analysis. Vol. 3418 of Lecture Notes in Computer Science. Springer-Verlag.

Brandes, U., Fleischer, D., 2005. Centrality measures based on current flow. In: Proceedings of the 22nd International Symposium on Theoretical Aspects of Computer Science (STACS'05). Vol. 3404 of Lecture Notes in Computer Science. pp. 533–544.

Brandes, U., Pich, C., 2007. Centrality estimation in large networks. International Journal of Bifurcation and Chaos 17 (7), 2303–2318.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., Stein, C., 2001. Introduction to Algorithms, 2nd Edition. MIT Press.

De, P., Singh, A., Wong, T., Yacoub, W., Jolly, A., 2004. Sexual network analysis of a gonorrhoea outbreak. Sexually Transmitted Infections 80, 280–285.

Diestel, R., 2000. Graph Theory, 2nd Edition. Graduate Texts in Mathematics. Springer-Verlag.

Everett, M. G., Borgatti, S. P., 1999. The centrality of groups and classes. Journal of Mathematical Sociology 23 (3), 181–201.

Everett, M. G., Borgatti, S. P., 2005. Ego network betweenness. Social Networks 27, 31–38.

Flom, P. L., Friedman, S. R., Strauss, S., Neaigus, A., 2004. A new measure of linkage between two sub-networks. Connections 26 (1), 62–70.

Freeman, L. C., 1977. A set of measures of centrality based upon betweenness. Sociometry 40, 35–41.

Freeman, L. C., 1979. Centrality in social networks: Conceptual clarification I. Social Networks 1, 215–239.

Freeman, L. C., Borgatti, S. P., White, D. R., 1991. Centrality in valued graphs: A measure of betweenness based on network flow. Social Networks 13 (2), 141–154.

Geisberger, R., Sanders, P., Schultes, D., 2008. Better approximation of betweenness centrality. In Proceedings of the 10th Workshop on Algorithm Engineering and Experimentation (ALENEX'08). To appear.

Goh, K.-I., Kahng, B., Kim, D., 2001. Universal behavior of load distribution in scale-free networks. Physical Review Letters 87 (27), 1–4.

Goodrich, M. T., Tamassia, R., 2002. Algorithm Design. Wiley.

Gould, R. V., 1987. Measures of betweenness in non-symmetric networks. Social Networks 9, 277–282.

Kleinberg, J., Tardos, É., 2006. Algorithm Design. Addison-Wesley.

Koschützki, D., Lehmann, K. A., Peeters, L., Richter, S., Tenfelde-Podehl, D., Zlotowski, O., 2005. Centrality indices. In Brandes and Erlebach (Eds.)

Lämmer, S., Gehlsen, B., Helbing, D., 2006. Scaling laws in the spatial structure of urban road networks. Physica A 363 (1), 89–95.

Moody, J. W., 2006. Diffusion and visualization in dynamic networks. Paper presented at the Sunbelt XXVI Social Network Conference (Vancouver, Canada).

Newman, M. E. J., 2001. Scientific collaboration networks II: Shortest paths, weighted networks, and centrality. Physical Review E 64, 016132.

Newman, M. E. J., 2004. Analysis of weighted networks. Physical Review E 70, 056131.

Newman, M. E. J., 2005. A measure of betweenness centrality based on random walks. Social Networks 27, 39–54.

Newman, M. E. J., 2006. Erratum: Scientific collaboration networks II: Shortest paths, weighted networks, and centrality. Physical Review E 73, 039906(E).

Newman, M. E. J., Girvan, M., 2004. Finding and evaluating community structure in networks. Physical Review E 69, 026113.

Perer, A., Shneiderman, B., 2006. Balancing systematic and flexible exploration of social networks. IEEE Transactions on Visualization and Computer Graphics 12 (5), 693–700.

Puzis, R., Elovici, Y., Dolev, S., to appear. Fast algorithm for successive computation of group betweenness centrality. Physical Review E.

Shimbel, A., 1953. Structural parameters of communication networks. Bulletin of Mathematical Biophysics 15, 501–507.

Valente, T. W., 2006. Bridges and potential bridges: Changing links to find critical paths and nodes in a network. Paper presented at the Sunbelt XXVI Social Network Conference (Vancouver, Canada).

White, D. R., Borgatti, S. P., 1994. Betweenness centrality measures for directed graphs. Social Networks 16, 335–346.