Fast algorithm for successive computation of group betweenness centrality

Article *in* Physical Review E · December 2007

DOI: 10.1103/PhysRevE.76.056709 · Source: PubMed

Fast algorithm for successive computation of group betweenness centrality

Rami Puzis,* Yuval Elovici, and Shlomi Dolev
Department of Computer Science at Ben-Gurion University, Beer-Sheva, Israel

(Received 10 December 2006; revised manuscript received 17 August 2007; published 29 November 2007)

In this paper, we propose a method for rapid computation of group betweenness centrality whose running time (after preprocessing) does not depend on network size. The calculation of group betweenness centrality is computationally demanding and, therefore, it is not suitable for applications that compute the centrality of many groups in order to identify new properties. Our method is based on the concept of path betweenness centrality defined in this paper. We demonstrate how the method can be used to find the most prominent group. Then, we apply the method for epidemic control in communication networks. We also show how the method can be used to evaluate distributions of group betweenness centrality and its correlation with group degree. The method may assist in finding further properties of complex networks and may open a wide range of research opportunities.

DOI: 10.1103/PhysRevE.76.056709 PACS number(s): 89.75.Hc

I. INTRODUCTION

Complex networks are used to study the structure and dynamics of complex systems in various disciplines [1]. For example, social networks [2,3], protein interactions networks [4], and computer networks such as the internet [5,6] are all classified as complex networks. In social networks, vertices are usually individuals, and edges characterize the relations between them; in computer networks, vertices might be routers connected to each other through communication lines.

Evaluation of the importance of vertices and edges is widely used in analysis of complex networks. To evaluate the importance, various centrality measures such as degree, closeness, and betweenness have been suggested [2,7]. Betweenness centrality (BC or B) [8,9] is considered to be a good approximation for the quantity of information passing through a vertex in communication networks [10]. In fact, the BC of vertices was used in [11] to define a congestion-free routing strategy. The BC of edges was used in [12,13] to identify communities in social and biological networks.

Everett and Borgatti [14] defined group betweenness centrality (GBC or $\ddot{B}$) as a natural extension of the betweenness measure. The GBC can be used to estimate the influence of a group of vertices over the information flow in the network. Freeman [8] has defined the group betweenness centralization index as a measure of the homogeneity of the members' betweenness. In this paper we use the GBC definition of Everett and Borgatti.

Scale-free networks are characterized by heterogeneity of vertices in terms of connectivity degree and BC [15–17]. Such networks withstand random damage but may be easily disrupted when the most central nodes are compromised [18]. Heavy-tailed distribution of BC in these networks suggests that there are vertices that control a significant portion of the information flow in the network. To the best of our knowledge, distribution of GBC has not yet been studied. A heavy right tail of the GBC distribution indicates that there are groups that control nearly all information flow in the network, while the vast majority of the groups will control only a small part of the information flow. A small right tail of the GBC distribution indicates that many groups can have a high degree of influence on the information flow in the network. While the former warrants spending resources to search for the group with high GBC, the latter suggests that such a group can be obtained from a few random samples. Thus, it is important to study the GBC distribution in a variety of complex networks.

In Sec. V we give several examples that illustrate the efficiency of the computational method presented in the analysis of complex networks. We show in the first example that the distribution of GBC for small groups in scale-free networks has a long right tail, although it is not a power-law distribution. For larger groups, the GBC distribution converges to a normal distribution in both random [19] and scale-free networks [15]. We also show that the correlation between GBC and group degree [14] is higher in scale-free networks than in random networks.

Identification of the most prominent group of vertices in a network is an important issue from the theoretical and practical points of view. For example, Ballester *et al.* state in [20] the importance of finding the key group in a criminal network. Borgatti elaborates in [21] on a key player problem that is strongly related to the cohesion of a network. Proactive immunization strategies based on connectivity of the individual vertices [22,23] can be improved by locating the most prominent group. A group of routers with the highest GBC in a computer communication network can be used for compromising the anonymity of the network users [24]. In Sec. V we show that the GBC can be used to optimize deployment of distributed intrusion detection systems in communication networks [25–28].

The problem of finding the group with maximal GBC is proven to belong to a class of problems with unknown efficient solution [28]. hard. Thus, combinatorial optimization techniques (such as heuristic search, genetic algorithms, etc. [29]) should be used to cope with this problem. The time spent on a single GBC computation is of critical importance since such techniques require many GBC computations. In this paper, we propose a method for fast successive computation of GBCs. We also show how the same computational

*Corresponding author. puzis@bgu.ac.il

method can be used to find a group of vertices of particular size whose GBC is negligibly close to the maximum.

We claim that researchers will increasingly use the GBC once they realize that it can be computed faster. We show that it is computationally feasible to compute the GBC of many groups or even to find a group of particular size whose GBC is negligibly below maximal. At the end of the paper, we demonstrate how the GBC can be used for epidemic control in communication networks. Up to now the time required to compute the GBC of many groups in a network was too high to allow extensive use. We believe that our algorithm will facilitate the use of the GBC in further studies of complex networks.

The rest of the paper is structured as follows. In Sec. II we present the definitions and notations that are used in the paper. In Sec. III we describe our algorithm for fast successive GBC computation. In Sec. IV we show how to find a group of vertices whose GBC is negligibly below maximal. In Sec. V we illustrate a few applications of our computational methods. Section IV concludes the paper with a summary and suggestions for future work.

II. DEFINITIONS AND NOTATIONS

In order to simplify the discussion, in this paper we assume that a subject network $G=(V,E)$ where $|V|=n$ and $|E|=m$ is an unweighted, undirected, and connected graph. Computational methods presented in this paper are applicable also to graphs that do not obey the above constraints.

We use small latin letters to indicate vertices $v,u,x,y \in V$. We use capital latin letters to indicate unordered sets or ordered tuples of vertices $C,M,S,P \subseteq V$. We use braces or greek letters to indicate various functions.

A. Betweenness and group betweenness centrality

In the following paragraphs, we will describe the basic definitions we adopted from previous research. Let s and t be two vertices in a graph. Let $\sigma_{s,t}$ be the total number of shortest paths between s and t. Let v be a vertex that lies on a shortest path between s and t. From all paths between s and t, several may pass through v and others may pass an alternative way. We denote the number of shortest paths from s to t that pass through v by $\sigma_{s,t}(v)$. The number of shortest paths from s to t that pass through v is

$$\sigma_{s,t}(v) = \begin{cases} \sigma_{s,v}\sigma_{v,t}, & d(s,t) = d(s,v) + d(v,t), \\ 0 & \text{otherwise}, \end{cases} \tag{1}$$

where $d(x,y)$ is the distance between vertices x and y.

Let $\delta_{s,\bullet}(v)$ be the total fraction of shortest paths that start at s and traverse v [8,30]. Roughly speaking, $\delta_{s,\bullet}(v)$ represents the influence of vertex v on communications originating at s and can be defined by the following equation:

$$\delta_{s,\bullet}(v) = \sum_{t \in V | s \neq t} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}. \tag{2}$$

The betweenness centrality of vertex $v \in V$ represents the total influence that v has on communications between all pairs of vertices. The betweenness centrality of node v is

$$B(v) = \sum_{s,t \in V | s \neq v \neq t} \left(\frac{\sigma_{s,t}(v)}{\sigma_{s,t}} \right) = \sum_{s \in V | s \neq v} \delta_{s,\bullet}(v). \tag{3}$$

Existing algorithms compute $\delta_{s,\bullet}(v)$ for all $s,v \in V$ in $O(mn)$ time and therefore the BC of all vertices can be computed in the same asymptotic running time [30–32].

The BC of individual vertices can be naturally extended to betweenness centrality of groups of vertices [14]. Let $C \subseteq V$ be a group of vertices. $\ddot{B}(C)$ stands for the total fraction of shortest paths between all pairs of vertices that pass through at least one member of the group C. Let $\ddot{\sigma}_{s,t}(C)$ be the number of shortest paths between s and t that traverse at least one member of the group C. The group betweenness centrality of group C is

$$\ddot{B}(C) = \sum_{s,t \in V \setminus C | s \neq t} \left(\frac{\ddot{\sigma}_{s,t}(C)}{\sigma_{s,t}} \right). \tag{4}$$

Note that, in the original definition of betweenness, shortest paths that start or end at the evaluated vertices are not included in the computation. It is more convenient to think that information originating at some vertex is seen by this vertex and hence should be accounted for. In this paper, we include shortest paths that start or end at the evaluated vertices in computation of their betweenness. Inclusion of the end vertices results in addition of a constant term $(2n-2)$ to a single-vertex BC and $[|C|(2n-|C|-1)]$ to $\ddot{B}(C)$. Similar variation of BC was previously mentioned in [17]. In this paper, we use a variant of the algorithm presented in [30] (which accounts for shortest paths that start or end at v) in order to compute $\sigma_{s,t}$ and $\delta_{s,\bullet}(v)$.

B. Path betweenness centrality

In the following paragraphs, we define path betweenness centrality (PB or $\tilde{B}$). We generalize the concept of a single-vertex betweenness to $\tilde{B}(S)$ where $S=(v_1,v_2,\ldots,v_k)$ is an ordered group of vertices. $\tilde{B}(S)$ stands for the total fraction of shortest paths between all pairs of vertices that traverse *all* vertices in S (first v_1, then v_2, etc.). Edge betweenness [9,12] is a special case of PB where $S=(u,v)$ is an edge in the graph. Let $\tilde{\sigma}_{s,t}(S)$ be the number of shortest paths between s and t that traverse all vertices in S (first v_1, then v_2, etc.). The path betweenness of an ordered group S is

$$\tilde{B}(S) = \sum_{s,t \in V | s \neq t} \left(\frac{\tilde{\sigma}_{s,t}(S)}{\sigma_{s,t}} \right). \tag{5}$$

The PB of a pair of vertices can be computed using the following equation:

$$\tilde{B}(x,y) = \sum_{s \in V} \delta_{s,\bullet}(y) \frac{\sigma_{s,y}(x)}{\sigma_{s,y}}. \tag{6}$$

The multiplication in Eq. (6) represents the fraction of shortest paths that start at s and traverse x and then y. Note that, for $x=y$, $\tilde{B}(x,y)=B(x)=B(y)$.

III. EFFICIENT COMPUTATION OF THE GBC

A. Preprocessing

The algorithm described in this paper requires preprocessing. The following three matrices are calculated during the preprocessing.

(1) d—an $n \times n$ matrix whose elements $d(s,t)$ store the distance between vertices s and t.

(2) σ—an $n \times n$ matrix whose elements $\sigma_{s,t}$ store the number of shortest paths between vertices s and t.

(3) $\widetilde{B}$—an $n \times n$ matrix whose elements $\widetilde{B}(x,y)$ store the path betweenness of pair (x,y).

The values of the d and σ matrices can be calculated by algebraic path counting [33] or using breadth first search at $O(nm)$. We use Eq. (6) to compute $\widetilde{B}(x,y)$ for every pair of vertices in the graph. The value of $\delta_{s,\bullet}(y)$ used in Eq. (6) can be computed using existing algorithms at $O(nm)$ for all $s,y \in V$ [30]. Every entry in the $\widetilde{B}$ matrix can be computed at $O(n)$ and, therefore, the worst case asymptotic running time of the full preprocessing stage is $O(n^3)$. In Sec. III C we will demonstrate how the preprocessing time can be reduced if we compute $\widetilde{B}(x,y)$ only for pairs of vertices that are involved in computation of the GBC.

B. GBC computation

In the following paragraphs, we will describe a fast algorithm for successive GBC computation. Assume that we want to compute the GBC of the group C of size k. Let M be a subset of C that includes vertices whose joint contribution to $\ddot{B}(C)$ was already computed. In the course of the algorithm's execution, M grows k times until it is equal to C, and hence at the end of the algorithm $\ddot{B}(M)$ is equal to $\ddot{B}(C)$. Each time we add a vertex to M, we spend $O(k^2)$ time to update the following data structure.

(1) $\ddot{B}(M)$—a variable that stores the current value of $\ddot{B}(M)$. $\ddot{B}(M)$ is initialized to zero.

(2) σ^M—a $k \times k$ matrix whose elements $\sigma^M_{s,t}$: $(s,t \in C)$ store the number of shortest paths between s and t that do not traverse any vertex in M. The initial values of $\sigma^M_{s,t}$ (for $M = \emptyset$) are taken from the precomputed matrix $\sigma_{s,t}$.

(3) $\widetilde{B}^M$—a $k \times k$ matrix whose elements $\widetilde{B}^M(x,y)$: $(x,y \in C)$ store the path betweenness of pair (x,y), disregarding the shortest paths that traverse at least one vertex in M. Initial values of $\widetilde{B}^M(x,y)$ (for $M=\emptyset$) are taken from the precomputed matrix $\widetilde{B}(x,y)$.

Given the above definitions, we can show that $\widetilde{B}^M(v,v)$ represents the GBC that can be gained by adding v to M. On one hand, $\ddot{B}(M)$ is the total fraction of the shortest paths that traverse at least one vertex in M. On the other hand, $\widetilde{B}^M(v,v)$ is the total fraction of shortest paths that traverse v, excluding shortest paths that traverse at least one vertex in M, and therefore $\ddot{B}(M)+\widetilde{B}^M(v,v)=\ddot{B}(M \cup \{v\})$.

We will describe now how to compute the path betweenness of a sequence of three vertices with respect to a set M.

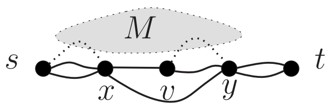


FIG. 1. The solid lines represent paths that are included in σ^M calculations while dotted lines represent paths that are disregarded. $\widetilde{\sigma}^M_{s,t}(x,y)=8$ and $\widetilde{\sigma}^M_{s,t}(x,v,y)=8 \cdot \frac{1}{2}=4$.

We will start the discussion with σ^M. Similar to Eq. (1), $\sigma^M_{s,t}(v)=\sigma^M_{s,v}\sigma^M_{v,t}$, assuming that v lies on a shortest path between s and t. Let x and y be two vertices on a shortest path from s to t. There are three segments ($s \rightsquigarrow x$, $x \rightsquigarrow y$, and $y \rightsquigarrow t$) on the way from s to t. The number of shortest paths from s to t that traverse both x and y but do not traverse M can be found by multiplying the numbers of shortest paths in all three segments, $\widetilde{\sigma}^M_{s,t}(x,y)=\sigma^M_{s,x}\sigma^M_{x,y}\sigma^M_{y,t}$. Similarly, the number of shortest paths between s and t that traverse x, v, and then y is a product of the number of shortest paths in four respective segments $[\widetilde{\sigma}^M_{s,t}(x,v,y)=\sigma^M_{s,x}\sigma^M_{x,v}\sigma^M_{v,y}\sigma^M_{y,t}]$ (see Fig. 1). It is easy to see that

$$\widetilde{\sigma}^M_{s,t}(x,v,y) = \sigma^M_{s,x}\sigma^M_{x,v}\sigma^M_{v,y}\sigma^M_{y,t} = \widetilde{\sigma}^M_{s,t}(x,y)\frac{\sigma^M_{x,y}(v)}{\sigma^M_{x,y}}.$$

We use the above observation to calculate the path betweenness of three vertices. Let $\widetilde{B}^M(x,v,y)$ be the total fraction of the shortest paths between all pairs of vertices that traverse x, then v, and then y without traversing any vertex in M. $\widetilde{B}^M(x,v,y)$ can be computed from $\widetilde{B}^M(x,y)$ as follows:

$$\widetilde{B}^M(x,v,y) = \sum_{s,t \in V} \frac{\widetilde{\sigma}^M_{s,t}(x,v,y)}{\sigma^M_{s,t}} = \frac{\sigma^M_{x,y}(v)}{\sigma^M_{x,y}} \sum_{s,t \in V} \frac{\widetilde{\sigma}^M_{s,t}(x,y)}{\sigma^M_{s,t}} = \frac{\sigma^M_{x,y}(v)}{\sigma^M_{x,y}}\widetilde{B}^M(x,y). \tag{7}$$

An ordered group of three vertices can be constructed by adding one vertex to a pair of vertices. Let $\circ$ be an operator for adding vertex z to an ordered pair of vertices (x,y), such that there is a shortest path that traverses x, y, and z:

$$(x,y) \circ z = \begin{cases} (z,x,y), & d(z,y)=d(z,x)+d(x,y), \\ (x,z,y), & d(x,y)=d(x,z)+d(z,y), \\ (x,y,z), & d(x,z)=d(x,y)+d(y,z). \end{cases} \tag{8}$$

If two or more conditions are satisfied then this operator returns one of the options. For example, if $x=z$ then $(x,y) \circ z=(z,x,y)=(x,z,y)=(x,y)$. If no condition is satisfied (there is no shortest path that traverses x, y, and z) then $(x,y)\circ z$ is not defined and $\widetilde{B}[(x,y)\circ z]=0$.

The implementation of the algorithm can be found in [34] while the pseudocode is presented in Algorithm 1. Lines 1–4 in Algorithm 1 initialize its data structures by copying the relevant information from the precomputed data. The loop in line 5 goes through all the vertices in group C, accumulating their contributions to the GBC in line 6. The contribution of each vertex to GBC is $\widetilde{B}^M(v,v)$. The contribution of the first vertex is its betweenness value $[\widetilde{B}^{\emptyset}(v_1,v_1)=\widetilde{B}(v_1,v_1)$

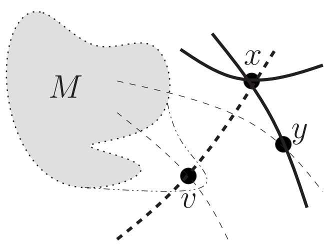


FIG. 2. The gray area represents the vertices whose contribution has already been added to the GBC. The dashed and dotted lines represent the step of the algorithm in which the contribution of v is added to the GBC. Thin dashed lines are shortest paths that were already accounted for. The bold dashed line represents the contribution of v to the GBC.

$=B(v_1)]$. The second vertex can contribute only those shortest paths that do not pass through the first vertex $[\widetilde{B}^{\{v_1\}}(v_2,v_2)]$, and so on. The contribution of vertex v when it is added to M is shown in Fig. 2.

Lines 7–11 update the matrices σ^M and $\widetilde{B}^M$ for the new M by removing shortest paths that traverse v. $\sigma^M_{x,y}$ in line 8 is decreased by $\sigma^M_{x,y}(v)$, removing paths that traverse v $[\sigma^M_{x,y}(v)$ is computed using Eq. (1)]. In the special case, in which $x=y$ (line 9), we subtract from the current contribution of x [given by $\widetilde{B}^M(x,x)$] paths that also traverse v in both directions $x \rightsquigarrow v$ and $v \rightsquigarrow x$. In a general case (line 10), we use σ^M and d to calculate the value of $\widetilde{B}^M[(x,y)\circ v]$ according to Eqs. (7) and (8). We then decrease $\widetilde{B}^M(x,y)$ by this value, removing paths that traverse v. Line 11 adds v to M. At this point, a possible implementation can copy the temporal matrices $\sigma^{M\cup\{v\}}$ and $\widetilde{B}^{M\cup\{v\}}$ into σ^M and $\widetilde{B}^M$, respectively.

Sketch of proof. The correctness of the algorithm can be proved by induction on the size of M.

Invariants. Before each execution of line 5 of the algorithm, the following invariants are always true.

(a) $\forall\ x,y \in C,\ \sigma^M_{x,y}$ is the number of shortest paths from x to y, excluding shortest paths that traverse at least one vertex in M.

(b) $\forall\ x,y \in C,\ \widetilde{B}^M(x,y)$ is the path betweenness of (x,y) excluding shortest paths that traverse at least one vertex in M.

Base case. The number of shortest paths between x and y, $\sigma_{x,y}$, and the total fraction of shortest paths that traverse x and then y, $\widetilde{B}(x,y)$, are calculated in the preprocessing stage. These numbers are assigned in lines 2 and 3 of Algorithm 1 to $\sigma^M_{x,y}$ and $\widetilde{B}^M(x,y)$, respectively. Since M is empty, no shortest paths should be disregarded and, therefore, the above assignments prior to executing line 5 for the first time respect the above invariants.

Induction step. Assume that both invariants are true for M. We show that the invariants are true for $M\cup\{v\}$ after the next execution of lines 6–13 (and immediately before the next execution of line 5). By the induction assumption, after the last time line 5 is executed the value in $\sigma^M_{x,y}$ is the number of shortest paths from x to y excluding the shortest paths that traverse at least one vertex in M. In addition, it holds in this stage that the value in $\widetilde{B}^M(x,y)$ is the total fraction of shortest paths that traverse x and then y excluding shortest paths that traverse at least one vertex in M.

Algorithm 1. $\ddot{B}(C)$.

```
Input: $C$, $d^{n\times n}$, $\sigma^{n\times n}$, $\widetilde{B}^{n\times n}$
Output: $\ddot{B}(C)$
(1) $M \leftarrow \emptyset$
(2) $\forall\ x,y \in C,\ \sigma^M_{x,y} \leftarrow \sigma_{x,y}$
(3) $\forall\ x,y \in C,\ \widetilde{B}^M(x,y) \leftarrow \widetilde{B}(x,y)$
(4) $\ddot{B} \leftarrow 0$
(5) for each $v \in C$
(6)   $\ddot{B} \leftarrow \ddot{B} + \widetilde{B}^M(v,v)$
(7)   for each $x,y \in C$
(8)     $\sigma^{M\cup\{v\}}_{x,y} \leftarrow \sigma^M_{x,y} - \sigma^M_{x,y}(v)$
(9)     if $x=y\neq v$
          $\widetilde{B}^{M\cup\{v\}}(x,x) \leftarrow \widetilde{B}^M(x,x) - \widetilde{B}^M(v,x) - \widetilde{B}^M(x,v)$
(10)    else
          $\widetilde{B}^{M\cup\{v\}}(x,y) \leftarrow \widetilde{B}^M(x,y) - \widetilde{B}^M[(x,y)\circ v]$
(11)  $M \leftarrow M\cup\{v\}$
```

In lines 7 and 8 of Algorithm 1 the value $\sigma^M_{x,y}(v)$ is subtracted from every entry $x,y\in C$ of σ^M. To do so, we calculate the number of shortest paths that start in x and arrive at v and multiply this number by the number of shortest paths that start in v and end in y (only when v is on a shortest path between x and y: the distance between x and y is equal to the distance between x and v plus the distance between v and y). That is, the value obtained following the subtraction is the total number of shortest paths between x and y that do not traverse any vertex in $M\cup v$.

In lines 7 and 9–12 of Algorithm 1, the path betweenness of three vertices x, y, and v is subtracted from every entry $x,y\in C$ of $\widetilde{B}^M$. To do so for the case in which v is on a shortest path between x and y, we use the value in $\sigma^M_{x,y}$ and the calculated $\sigma^M_{x,y}(v)$ (for which we described the calculation above) to compute the fraction of the number of shortest paths that traverse v on the way from x to y, and the total number of shortest paths between x and y. The obtained fraction is used to update $\widetilde{B}^M(x,y)$ by a subtraction of the part of $\widetilde{B}^M(x,y)$ that is related to paths that traverse v. The other cases, in which either x is on a shortest path between v and y or y is on a shortest path between x and v, are treated analogously. The obtained value is the path betweenness of (x,y) excluding paths that traverse any vertex in M or v.

The above two invariants and the fact that $\widetilde{B}^M(v,v)$ in line 6 represents the contribution of v to $\ddot{B}(M)$ imply that the accumulation in line 6 of the algorithm results in the GBC of C. This concludes the proof sketch.

During the execution of Algorithm 1, contributions of all vertices in C are constantly updated. Therefore, an efficient greedy maximization of the GBC can be constructed by starting the algorithm with $C=V$ and each time choosing the next vertex with maximal contribution in line 5 (see Sec. IV). The updated contribution of vertices can also accelerate the search for the group with maximal GBC using a heuristic search [28].

C. Performance evaluation

We will now demonstrate the efficiency of the proposed algorithm by comparing its running time with the running time of state of the art algorithms for GBC computation [14,24]. The state of the art algorithm is an immediate extension of the algorithm proposed by Brandes [30] for faster computation of single-vertex betweenness indices. Our algorithm requires preprocessing to be executed. Therefore, it is supposed to be less efficient for tasks that require a small number of GBC computations per network. However, as will be explained below, even for computing the GBC of two groups, our approach is competitive with the state of the art GBC computation.

The running time of Algorithm 1 without preprocessing scales as k^3. In Sec. III A the preprocessing stage of Algorithm 1 was described. Preprocessing time can be reduced from $O(n^3)$ to $O(mn)$ if we postpone the computation of $\widetilde{B}(x,y)$ until it is needed in line 3 of Algorithm 1. In order to compute the GBC of a group of size k, only k^2 entries of the $\widetilde{B}$ matrix are required. Every entry in the $\widetilde{B}$ matrix can be computed at $O(n)$ using Eq. (6). Therefore, all required $\widetilde{B}$ values can be computed at $O(nk^2)$. When computing the GBC of a single group with $k \le \sqrt{m}$ vertices, the time required to compute σ, d, and $\delta_{\bullet}$ matrices $[O(nm)]$ dominates over the time required to compute relevant $\widetilde{B}$ values $[O(nk^2)]$ and the GBC of the group $[O(k^3)]$. Therefore, the total time spent on computation of the GBC (including preprocessing) of a single group using the method proposed in this paper is $O(nm)$ when $k \le \sqrt{m}$. State of the art algorithms compute the GBC in the same asymptotic running times.

Algorithm 1 proposed in this paper is efficient in computing the GBC of several groups on the same network. State of the art algorithms compute the GBC of l groups of size k on a network with n vertices and m edges in $O(nml)$. With the full preprocessing stage, the total running time required to complete the above task using Algorithm 1 is proportional to $n^3 + lk^3$. The total running time spent on computation of $\widetilde{B}$ values cannot exceed $O(n^3)$ since $\widetilde{B}$ is computed only once for every pair of vertices. Total running time required to compute the GBC of l groups of size k on a network with n vertices and m edges with a reduced preprocessing stage is proportional to $\min\{n^3 + lk^3, nm + lnk^2\}$ in the worst case. Thus, state of the art algorithms for GBC computation may perform better only for *sparse networks* when there are *few large groups* of vertices to evaluate.

In order to demonstrate the effectiveness of our algorithm, we generated ten random networks of size 100, ten of size 200, and so on to 500 vertices. All networks were sparse with twice as many edges as vertices. We computed the GBC for random groups of size 20 using the state of the art algorithm (A1), Algorithm 1 with full preprocessing (A2), and Algorithm 1 with reduced preprocessing (A3). The algorithms were implemented in PYTHON using NUMPY and NETWORKX packages and executed on a PC P.Duo-3 GHz with 2 Gbytes memory [34]. The running time of the algorithms is presented in Fig. 3. We can see in the figure that with the reduced preprocessing the proposed algorithm outperforms the state of the art algorithm when computing the GBC of two or more groups.

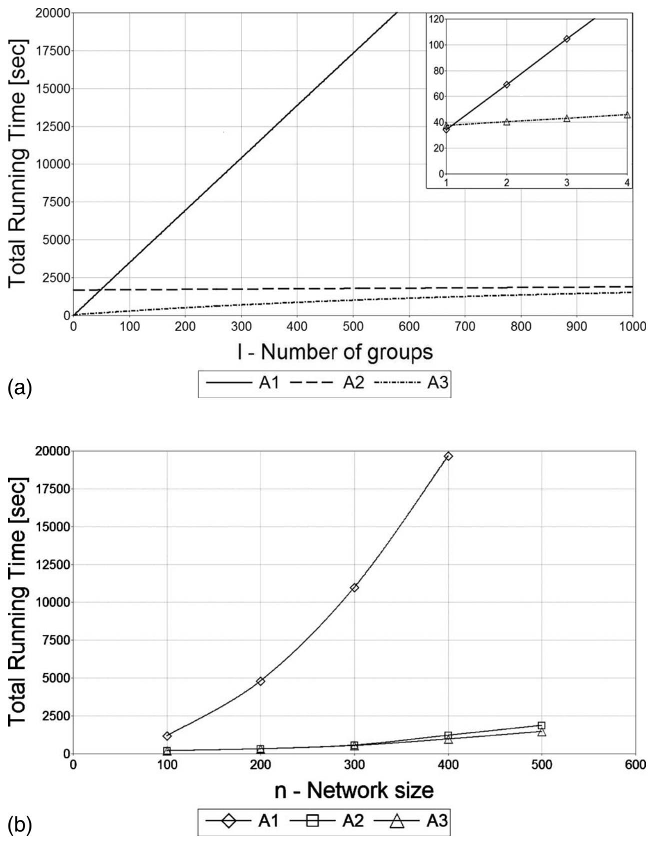


FIG. 3. (a) Running time of algorithms for GBC computation as a function of the number of evaluated groups of size $k=20$ for network size $n=500$. (b) Running time of algorithms for GBC computation as a function of the network size for $l=900$ evaluated groups of size $k=20$. A1: state of the art algorithm. A2: Algorithm 1 with full preprocessing. A3: Algorithm 1 with reduced preprocessing.

IV. FINDING THE PROMINENT GROUP

In this section, we present another algorithm that uses Algorithm 1 to find a prominent group in terms of the GBC. In Sec. V we will apply this algorithm for epidemic control in communication networks.

Ideally we would like to find the group of a given size with the maximal GBC. Unfortunately, it can be proved by reduction from the minimal vertex cover problem [35] that finding a group with maximal GBC is a difficult problem. One way to cope with the complexity of this problem is to employ heuristic search [28,29]. Algorithm 1 helps to find the group with the highest GBC in an efficient manner since it is optimized for computing the GBC of a large number of groups; therefore, it plays a significant role in reducing the time it may take to search for the most central group [28]. There is also evidence that the time it takes to search for the group with maximal GBC is smaller in scale-free [15,18] than in binomial random networks [19].

Another option is to use a polynomial time algorithm to find a group whose GBC is high but not guaranteed to be the maximal. An important property of the computational method described in this paper is that it computes the contribution of vertices to the GBC of a group during its execution. This property is used in Algorithm 2 to find a group of vertices whose GBC is negligibly below maximal. The quality of results produced by this iterative algorithm is higher in scale-free networks than in binomial random networks [28].

Algorithm 2 is an iterative algorithm that, in every step, chooses the vertex with highest contribution to the GBC of vertices already accounted for. Algorithm 2 receives as input three matrices computed during the preprocessing stage (see Sec. III A), the set of candidate vertices from which it will choose the best group, and the size of that group. The time required to execute lines 6–10 of Algorithm 2 is proportional to the square of the number of candidate vertices. Algorithm 2 computes the contribution of all candidate vertices k times. Therefore the running time of Algorithm 2 is $O(k|C|^2)$. The Algorithm 2 running time including preprocessing is proportional to $\min\{nm+n|C|^2, n^3+k|C|^2\}$ (see Sec. III).

Algorithm 2. *Find the group with high GBC.*

```
Input: $C$, $k$, $d^{n\times n}$, $\sigma^{n\times n}$, $\widetilde{B}^{n\times n}$
Output: group $M\subseteq C$ of size $k$
(1) $M\leftarrow\emptyset$
(2) $\forall\ x,y\in C,\ \ \sigma^{M}_{x,y}\leftarrow\sigma_{x,y}$
(3) $\forall\ x,y\in C,\ \ \widetilde{B}^{M}(x,y)\leftarrow\widetilde{B}(x,y)$
(4) for $i=1$ to $k$
(5)   find $v\in C$ with maximal $\widetilde{B}^{M}(v,v)$
(6)   for each $x,y\in C$
(7)     $\sigma^{M\cup\{v\}}_{x,y}\leftarrow\sigma^{M}_{x,y}-\sigma^{M}_{x,y}(v)$
(8)     if $x=y\neq v$
          $\widetilde{B}^{M\cup\{v\}}(x,x)\leftarrow\widetilde{B}^{M}(x,x)-\widetilde{B}^{M}(v,x)-\widetilde{B}^{M}(x,v)$
(9)     else
          $\widetilde{B}^{M\cup\{v\}}(x,y)\leftarrow\widetilde{B}^{M}(x,y)-\widetilde{B}^{M}[(x,y)\circ v]$
(10)  $M\leftarrow M\cup\{v\}$
```

V. APPLICATIONS

In this section, we briefly describe several examples that illustrate the effectiveness of our algorithms in the analysis of complex networks. The first example is an evaluation of GBC distribution in preferential attachment and random networks. In the second example we evaluate the correlation of group betweenness and group degree [14] centrality indices. We show here that the correlation between these two measures decreases as the size of the group grows. The last example describes the application of computational methods proposed in this paper in the field of epidemic control in communication networks. All algorithms used in this section were implemented in PYTHON using NUMPY and NETWORKX packages and executed on a PC P.Duo-3 GHz with 2 Gbytes memory.

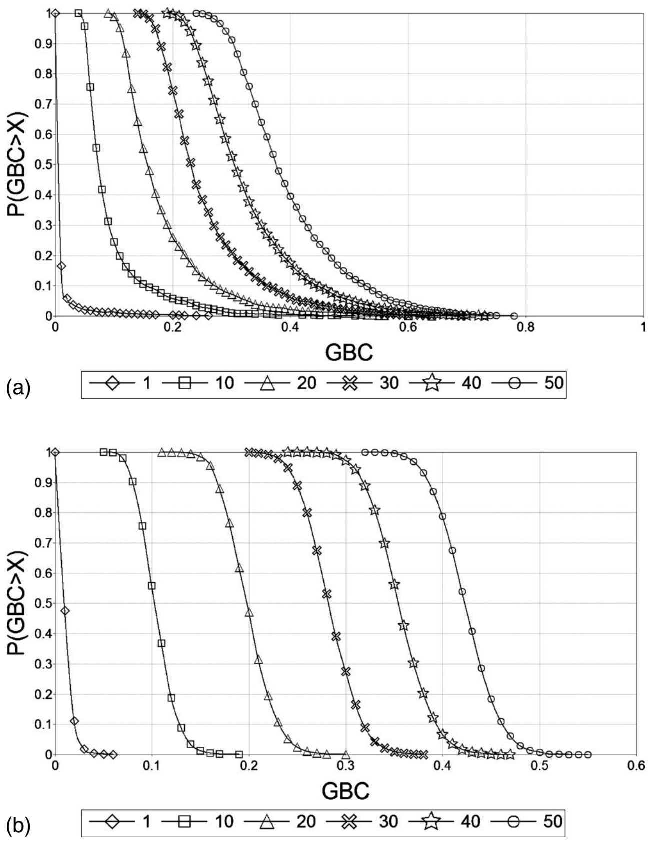
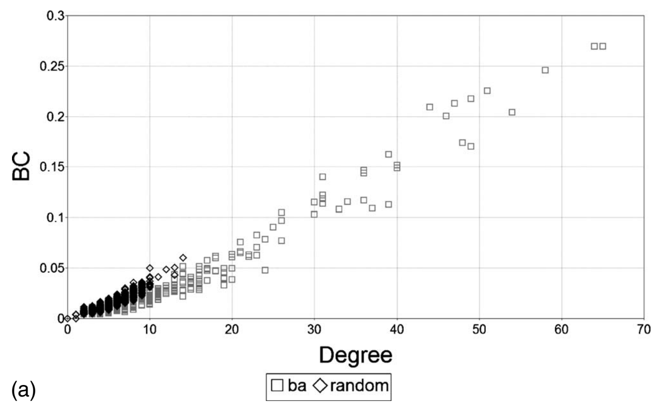


FIG. 4. Cumulative distribution of GBC on BA (a) and random (b) networks of size 500.

A. GBC distribution

It is well known that the betweenness centrality distribution follows a power law in many scale-free networks [15,17]. However, to the best of our knowledge, the distribution of the GBC has not yet been studied. We have evaluated the GBC distribution by analyzing networks of sizes 100,200,…,500 vertices and average degree of 2. Ten networks of each size were generated using the Barabasi-Albert (BA) model [15] and ten were binomial random networks [19]. The GBC distribution was computed for groups of sizes 1, 10, 20, and 30. We have computed the GBC of 1000 random groups for every network.

For singletons and small groups, the GBC distribution is exponential in random graphs and heavy tailed in preferential attachment networks as described in the literature. For large groups, the GBC distribution converges to the normal distribution in random binomial and preferential attachment networks. These results, in uncorrelated networks, can be explained by the central limit theorem. We can see in Fig. 4 that in scale-free networks the convergence to the normal distribution is much slower than in random binomial networks, as the right tail dominates even for relatively large



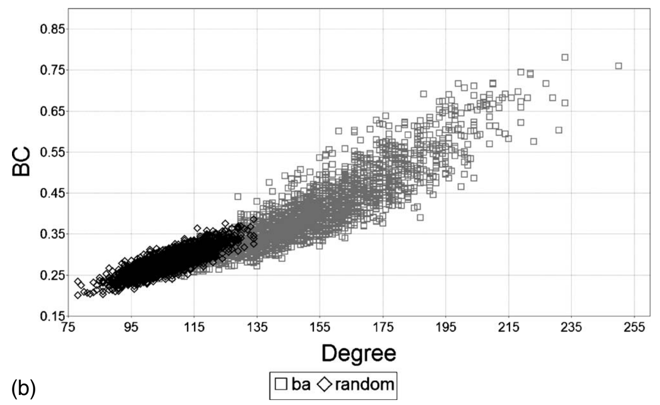


FIG. 5. (a) Degree and BC of random vertices in networks of size 500. (b) Group degree and GBC of random groups of 30 vertices on networks of size 500.

groups. Such behavior of the GBC distribution was observed for all tested network sizes.

B. Correlation between the GBC and group degree

The BC and degree of single vertices are known to be correlated [17,36]. In this example we quantify the correlation between the GBC and group degree for different group sizes in BA and random networks. For this purpose we derived the group degree of every group whose GBC was computed in the previous section. We can see from Figs. 5 and 6 that there is a correlation between the GBC and group degree; however, when the size of the group increases the correlation decreases. The correlation between group degree and GBC as a function of the group size is presented in Fig. 6. The correlation between group degree and GBC is higher in BA networks than in random binomial networks. The high correlation between betweenness and degree suggests that in some cases when computational time is an expensive resource the group degree can be used as a good heuristic for locating a group of vertices with high GBC.

C. Epidemic control in communication networks

Theoretical models of epidemic propagation have long been studied by scientists in the physics community [37–40]. Targeted immunization strategies that prefer vertices with

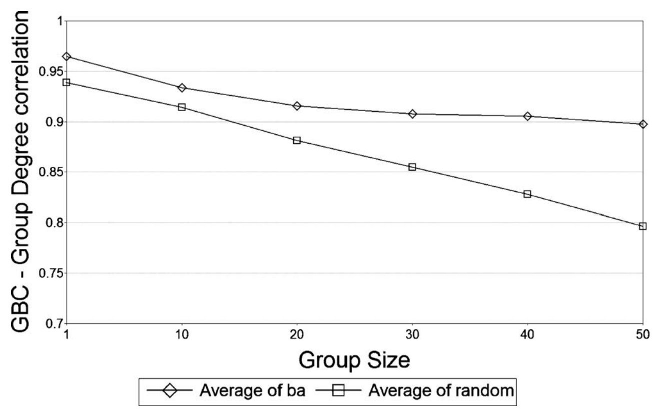
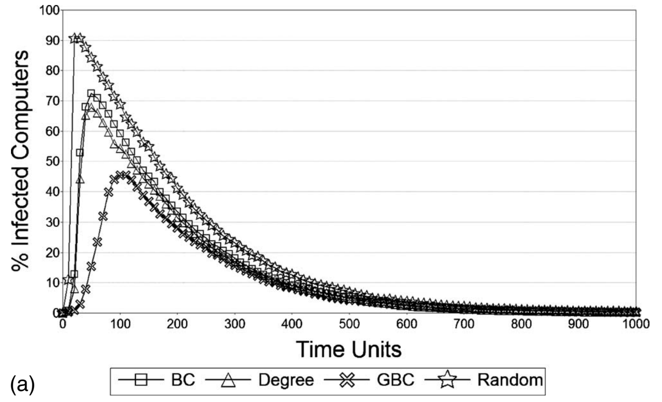


FIG. 6. Correlation between group degree and GBC as a function of the group size on networks with 500 vertices.

high connectivity were proposed in order to control the epidemic propagation in a variety of networks [22,23]. Betweenness is suitable for epidemic control in networks with natural community structure [12] or in sparse networks where cutoff vertices exist. Betweenness is also suitable for epidemic control in cases where the infection is communicated over a public medium. One example of such disease in biology is the dog worm *Spirocercalupi*. This parasite is



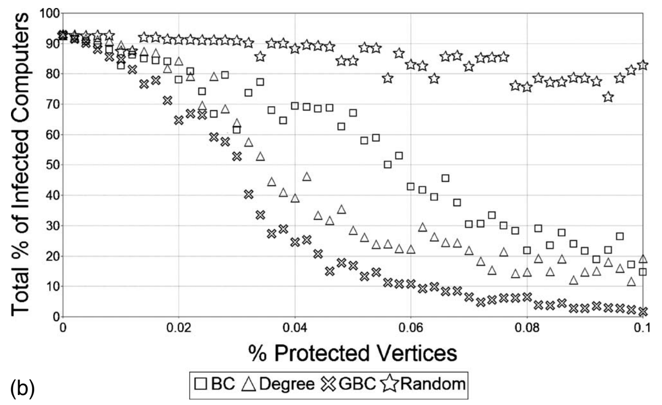


FIG. 7. Dynamics of epidemic propagation on communication network of size 500 protected using random selection of vertices, selection based on the BC, GBC, and degree. (a) The infected mass as a function of time (with 15 protected vertices). (b) The infected mass as a function of the fraction of protected vertices.

transferred from one dog to another through excrement and dung beetles. In this case we can consider dogs as communicating parties and all the areas where dogs are taken out for a walk as the public media. Special attention should be given to cleanliness of the areas with high 2-betweenness [41]. Next we will demonstrate the effectiveness of epidemic control based on the GBC in communication networks.

Computer viruses and worms are able to spread over computer communication networks such as the internet. In computer communication networks, vertices might be routers connected to each other through communication lines. Computers are usually connected to a portion of the routers via internet service providers. When one computer sends contagious data to another, the data pass through several routers in the network. One possible solution that may reduce the number of computers being infected by a virus or worm is to filter out the contagious communications in the network fabric before it has reached computers. Several solutions (devices) that are able to perform this task even on high-speed internet connections have been proposed [42]. Since the price of such devices is high, there is a need to find an optimal deployment strategy where not all the vertices are cleaned.

We model propagation of computer viruses using the susceptible, infective, removed model of epidemics with equal infectivity [43,44]. In the susceptible state, a computer is vulnerable to the particular virus but not yet infected. In the infective state, the computer is infected and facilitates spreading of the virus by infecting one random peer in every time unit. And, finally, in the removed state, the computer has been patched or an antivirus update installed. Dealing with computers, it is natural to assume that both susceptible and infective computers can be patched in every time unit with a constant probability β. This scenario is typical when software vendors release the patch soon after the virus was first observed. It takes time for users to adopt this patch, but finally all users do and the virus prevalence is halted.

During an epidemic each computer can potentially infect every other computer in the network with equal probability. The simulations were performed on five Watts-Strogatz small-world networks [45] of size 500 with an average degree of 6 and rewiring probability of 0.015. We have compared the effectiveness of deployment strategy based on Algorithm 2 to deployment strategies based on the BC, degree, and random deployment strategies. We simulated the virus propagation with $\beta=6.7\times10^{-3}$. We stopped the simulation after 1000 time units. We measured the fraction of computers that were infected during the simulation. The fraction of infected computers as a function of the deployment size averaged over five random starting positions is presented in Fig. 7. We can see from Fig. 7 the superiority of deployment strategy based on the maximized GBC over the random deployment strategy, a deployment strategy based on degree of connectivity, and a deployment strategy based on the BC of vertices.

VI. SUMMARY

In this paper we presented a method for rapid computation of the GBC whose running time (after preprocessing) does not depend on network size. The method is based on the concept of path betweenness centrality defined in this paper. We demonstrated the method's superiority over the state of the art computation method. In addition, we demonstrate how a variation of the method can be used to find the most prominent group of vertices of a particular size.

We presented three applications of the proposed method. In the first we evaluated the GBC distribution in preferential attachment and random networks. In the second, we evaluated the correlation of group betweenness and group degree centrality indices. We show that the correlation between these two measures decreases as the size of the group increases. In the third, we illustrated how the computational method can be used in the field of epidemic control in communication networks. We illustrate through simulations the superiority of a deployment strategy based on the GBC over deployment based on other centrality measures.

The main principles of the method for GBC computation described in this paper may also be applicable to other betweenness centrality measures. Newman [46] proposes a betweenness centrality index that is based on a series of random walks in a network rather than counting shortest paths. Random walk group betweenness centrality of edges could be used to improve the identification of the community structure in networks [12].

We hope that the presented method will help in finding new properties of complex networks and will open a wide range of new research opportunities.

[1] S. H. Strogatz, Nature (London) **410**, 268 (2001).
[2] S. Wasserman and K. Faust, *Social Network Analysis: Methods and Applications* (Cambridge University Press, Cambridge, England, 1994).
[3] J. Scott, *Social Network Analysis: A Handbook* (Sage Publications, London, 2000).
[4] P. Bork, L. J. Jensen, C. von Mering, A. K. Ramani, I. Lee, and E. M. Marcotte, Curr. Opin. Struct. Biol. **14**, 292 (2004).
[5] M. Faloutsos, P. Faloutsos, and C. Faloutsos, Comput. Commun. Rev. **29**, 251 (1999).
[6] S.-H. Yook, H. Jeong, and A.-L. Barabasi, Proc. Natl. Acad. Sci. U.S.A. **99**, 13382 (2002); e-print arXiv:cond-mat/0107417.
[7] L. C. Freeman, Soc. Networks **1**, 215 (1979).
[8] L. C. Freeman, Sociometry **40**, 35 (1977).
[9] J. M. Anthonisse, Stichting Mathematisch Centrum, Amsterdam, Technical Report No. BN 9/71, 1971 (unpublished).
[10] P. Holme, Adv. Complex Syst. **6**, 163 (2003).
[11] G. Yan, T. Zhou, B. Hu, Z.-Q. Fu, and B.-H. Wang, Phys. Rev. E **73**, 046108 (2006).
[12] M. E. J. Newman and M. Girvan, Phys. Rev. E **69**, 026113 (2004).

[13] M. E. J. Newman, Phys. Rev. E **69**, 066133 (2004).
[14] M. G. Everett and S. P. Borgatti, J. Math. Sociol. **23**, 181 (1999).
[15] A.-L. Barabasi and R. Albert, Science **286**, 509 (1999).
[16] A.-L. Barabasi, R. Albert, and H. Jeong, Physica A **281**, 69 (2000).
[17] M. Barthélemy, Eur. Phys. J. B **38**, 163 (2004).
[18] B. Bollobas and O. Riordan, Internet Math. **1**, 1 (2003).
[19] E. N. Gilbert, Ann. Math. Stat. **30**, 1141 (1959).
[20] C. Ballester, A. C. Armengol, and Y. Zenou, http://ideas.repec.org/p/cpr/ceprdp/5329.html
[21] S. P. Borgatti, Comput. Math. Org. Theory **12**, 21 (2006).
[22] D. H. Zanette and M. Kuperman, e-print arXiv:cond-mat/0109273.
[23] R. Pastor-Satorras and A. Vespignani, Phys. Rev. E **65**, 036104 (2002).
[24] D. Yagil, M.S. thesis, Information Systems Engineering, Ben-Gurion University of the Negev, 2005.
[25] V. Yegneswaran, P. Barford, and S. Jha, in *Proceedings of Network and Distributed System Security Symposium (NDSS)* (The Internet Society, San Diego, 2004).
[26] M. Cai, K. Hwang, Y.-K. Kwok, S. Song, and Y. Chen, IEEE Security Privacy **3**, 24 (2005).
[27] P. Blackburn, http://giac.org/certified_professionals/practicals/gsec/3472.php
[28] R. Puzis, Y. Elovici, and S. Dolev, AI Commun. (to be published).
[29] H. Eriksson, Y. Shahar, S. W. Tu, A. R. Puerta, and M. A. Musen, Artif. Intell. **79**, 293 (1995).
[30] U. Brandes, J. Math. Sociol. **25**, 163 (2001).
[31] M. E. J. Newman, Phys. Rev. E **64**, 016132 (2001).
[32] M. E. J. Newman, Phys. Rev. E **73**, 039906(E) (2006).
[33] F. Harary, R. Norman, and D. Cartwright, *Structural Models. An Introduction to the Theory of Directed Graphs* (John Wiley and Sons, New York, 1965).
[34] See EPAPS Document No. E-PLEEE8-76-064711 for PYTHON implementation of fast algorithm for successive GBC computation and code used for evaluating GBC distribution in random and BA networks. For more information on EPAPS, see http://www.aip.org/pubservs/epaps.html
[35] R. G. Downey and M. R. Fellows, Feasible Math. **2**, 219 (1995).
[36] K.-I. Goh, B. Kahng, and D. Kim, Phys. Rev. Lett. **87**, 278701 (2001).
[37] L. K. Gallos and P. Argyrakis, Physica A **330**, 117 (2003).
[38] D. Volchenkov, L. Volchenkova, and P. Blanchard, Phys. Rev. E **66**, 046137 (2002).
[39] Y. Moreno, J. B. Gomez, and A. F. Pacheco, Phys. Rev. E **68**, 035103(R) (2003).
[40] M. Boguna and R. Pastor-Satorras, Phys. Rev. E **66**, 047104 (2002).
[41] N. E. Friedkin, AJS **96**, 1478 (1991).
[42] C. Kruegel, F. Valeur, G. Vigna, and R. Kemmerer, http://citeseer.ist.psu.edu/kruegel02stateful.html
[43] R. M. Anderson and R. M. May, *Infectious Diseases of Humans: Dynamics and Control* (Oxford University Press, Oxford, 1992).
[44] T. Zhou, J.-G. Liu, W.-J. Bai, G. Chen, and B.-H. Wang, Phys. Rev. E **74**, 056109 (2006).
[45] D. J. Watts and S. H. Strogatz, Nature (London) **393**, 440 (1998).
[46] M. E. J. Newman, e-print arXiv:cond-mat/0309045.