Computer Communications 34 (2011) 619–628

Contents lists available at ScienceDirect

Computer Communications

journal homepage: www.elsevier.com/locate/comcom

Second order centrality: Distributed assessment of nodes criticity in complex networks

Anne-Marie Kermarrec [a], Erwan Le Merrer [b], Bruno Sericola [a], Gilles Trédan [c,*]

[a] INRIA Rennes – Bretagne Atlantique – Campus de Beaulieu, 263 avenue du Général Leclerc, 35042 Rennes, France
[b] Technicolor - 1, avenue Belle Fontaine, F-35576 Cesson Sevigne, France
[c] Deutsche Telekom/TU-Berlin – FG INET, Research Group Anja Feldmann Sekr. TEL 16, Ernst-Reuter-Platz 7, D - 10587 Berlin

ARTICLE INFO

Article history:
Available online 12 June 2010

Keywords:
Centralities
Random walk
Topologies

ABSTRACT

A complex network can be modeled as a graph representing the "who knows who" relationship. In the context of graph theory for social networks, the notion of centrality is used to assess the relative importance of nodes in a given network topology. For example, in a network composed of large dense clusters connected through only a few links, the nodes involved in those links are particularly critical as far as the network survivability is concerned. This may also impact any application running on top of it. Such information can be exploited for various topological maintenance issues to prevent congestion and disruption. This can also be used offline to identify the most important actors in large social interaction graphs. Several forms of centrality have been proposed so far. Yet, they suffer from imperfections: initially designed for small social graphs, they are either of limited use (degree centrality), either incompatible in a distributed setting (*e.g.* random walk betweenness centrality).

In this paper we introduce a novel form of centrality: the second order centrality which can be computed in a distributed manner. This provides locally each node with a value reflecting its relative criticity and relies on a random walk visiting the network in an unbiased fashion. To this end, each node records the time elapsed between visits of that random walk (called return time in the sequel) and computes the standard deviation (or second order moment) of such return times. The key point is that central nodes see regularly the random walk compared to other topology nodes. Both through theoretical analysis and simulation, we show that the standard deviation can be used to accurately identify critical nodes as well as to globally characterize graphs topology in a distributed way. We finally compare our proposal to well-known centralities to assess its competitivity.

1. Introduction

1.1. Context

Large scale networks, as organizational/social contacts, peer-to-peer, grids or wireless sensors networks often exhibit complex and huge interaction graph structure. The scale of these graphs is such that it usually prevents to compute any global characteristic aggregated from individual nodes [3,25]. Consequently, designing distributed solutions (in which network components participate only based on local or close neighborhood information) is of the utmost importance.

The offline analysis of complex social networks has been addressed by physicians and sociologists since the 1950s [34]. These works provide metrics to extract characteristics on the interactions captured and the importance or role of individuals in these graphs. The notion of *centrality* [7,9,10,18,20,30] typically provides such importance information, *w.r.t.* graph structure. Each centrality defines a particular importance, be it on central nodes, high degree nodes or on nodes part of redundant paths for example. Existing algorithms computing node centrality exhibit a high complexity, since they are used offline and imply costly operations on adjacency matrices (generally in $O(n^3)$ time, with n being the number of nodes). Along with time complexity, space required to store those matrices for computation is also a major issue, considering the growing size of distributed systems, or the size of today's social graphs (Facebook counts over 400 million users). One way of avoiding such a need for storage on a single computer is to use the distributed solution. In such a paradigm, all nodes of the studied network participate in order to produce the final result (here importance value computation on each node). Unfortunately, distribution of centrality computation is yet to be deepen. If centrality based on nodes' degree is straightforward to distribute [34] (considering that a node is important if it has a high connectivity), it remains of a limited interest. Nanda et al. [29] propose a heuristic

* Corresponding author. Tel.: +49 30 8353 58551.
E-mail address: gilles@net.t-labs.tu-berlin.de (G. Trédan).

0140-3664/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.comcom.2010.06.007

that only detects nodes bridging highly connected regions. Abiteboul et al. [2] propose an online solution to compute *eigenvector centrality*, which does not need the matrix storage; this centrality gives importance to nodes that are neighbors of also important nodes. All other classic centralities (*e.g. closeness, eccentricity, betweenness*) are yet to be distributed.

To overcome these issues, we introduce a novel notion of centrality, called *second order centrality*, which is distributed by design. The second order centrality provides each individual node with its relative importance in a given topology, *w.r.t.* its betweenness; the gathering of all nodes' centrality allows global characterization of a complex network.

1.2. Contributions

In this paper we make the following contributions; (i) we define a novel notion of centrality, called the second order centrality and show that it represents a meaningful metric to characterize both the criticity of individual nodes in a given topology as well as the global *health* of a complex network with respect to connections. (ii) We provide a lightweight algorithm to compute in a distributed way the second order centrality of each node. The strength of this algorithm lies in its simplicity: it relies on a single random walk visiting the network. Nodes compute their centrality by simply recording the return times of that walk. The standard deviation of those return times is at the core of our approach. We show through analysis at a global scale (assuming the knowledge of the full graph) that what is expected at a local scale (on nodes) is correct. (iii) We show that this algorithm can be used to provide graph signatures and therefore information about the global characteristics of a graph. (iv) We provide some simulation results that not only accurately match the analysis but also provides some evidence of the relevance of that metric in a practical setting. (v) We finally compare our technique to other well know centralities, to highlight its benefits.

1.3. Roadmap

The rest of this paper is organized as follows. Section 2 provides the design rationale of the second order centrality. Section 3 reviews various forms of centrality and emphasizes their characteristics and limitations. Section 4 describes the second order centrality algorithm. The analysis is provided in Section 5. Simulation results are then provided in Section 6, followed in Section 7 by head to head comparison with well-known forms of centralities. Section 8 concludes this paper.

2. Design rationale

We consider an arbitrary network, represented as an undirected graph $\mathcal{G} = (V, E)$, with n vertices and m edges. For a node $i \in V$, Γ_i denotes its set of neighbors in $\mathcal{G}$ (vertices with an adjacent edge to i), and d_i its degree, namely the size of Γ_i. Note that the resulting graph may represent any peer-to-peer, grid, social, or physical network. We assume a connected graph; this is crucial to avoid a result only on the connected component where the algorithm is started.

We use a *random walk* on the graph $\mathcal{G}$, *i.e.* a process progressing in the network from a node i to another node chosen uniformly in i's neighborhood Γ_i. We consider this walk to be *permanent* for it has no stop condition; this is one of the main differences with most of random walk based algorithms (*e.g.* sampling [19] or search [27]). The random walk is initiated by an arbitrary node of the network. We also assume, for the sake of comprehension, that the network is static *i.e.* the topology does not change during the process execution. Finally, the random walk is never lost.

Our approach relies on the fact that the relative importance of a given node can be inferred from the regularity at which the random walk visits the node. Typically, the algorithm exploits the time elapsed between two consecutive visits of such random walk on a given node (called return time hereafter). These return times can be either absolute time, thus implying a clock on every system node, or simply the number of steps proceeded by the random walk (which thus carries that information). Note that in the first case, nodes' clock do not need to be synchronized, as we are interested in local standard deviation of return times.

Note that our model only requires one random walk for the whole system to provide the algorithm result, as opposed *e.g.* to one random walk launched by each node for its own purposes. On the design side, our approach aims at replacing storage space on a single server by time complexity; our solution with second order centrality is then computable on a distributed setting. It is also computable in a standard fashion on a central server, using graph adjacency matrix (with provided formula), when used *e.g.* for small graphs or for networks where participants cannot collaborate (offline analysis).

3. Related work

3.1. Centralities: criticity of individuals in the graph

We are in this paper interested in the properties of individual nodes and their impact on overall graph connectivity. This impact is reflected by *centrality* indices.

The *betweenness centrality* [18,7,9,20] is considered the most relevant in that context. It consists in computing on each node the fraction of shortest paths that pass through it. Formally, the betweenness centrality for a node v is $C_b(v) = \sum_{s,t \in V} \frac{\sigma_{s,t}(v)}{\sigma_{s,t}}$, where $\sigma_{s,t}(v)$ is the number of shortest paths from node s to node t passing through v, and $\sigma_{s,t}$, the total number of shortest paths from s to t. The original algorithm requires $\Omega(n^3)$ time steps to complete; a first approximation approach completes in $O(nm)$ steps [9]. Finally, recent experimental studies [10,20] propose some approximations for practical use in large networks.

The other most current centralities are precisely defined in Section 7, for comparison purposes with our proposal.

3.2. Random walk based betweenness centrality

Despite a great interest towards the betweenness metric, Newman showed [30] that the notion of betweenness centrality suffers from some imperfections as it considers only nodes involved in shortest paths. A typical example is presented in Fig. 1, where two clusters are linked by a few nodes only. Here, node c, which is outside the shortest paths linking left and right clusters is given a very low score of betweenness centrality despite its clear importance for alternative paths.

Such left out nodes can also be of vital importance for the network resilience, for load balancing or facing failures of shortest path nodes. Another metric, known as *flow betweenness*, also suffers from a sim-

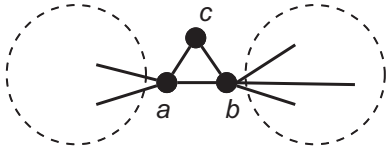

Fig. 1. An example where betweenness centrality give node c a low score despite its importance.

ilar drawback [30]. *Random walk betweenness* has been introduced to fix this issue: the idea is that a random process also takes into account non-optimal paths. Assuming the knowledge of the whole graph, the proposed method uses its adjacency matrix and completes in $O((m+n)n^2)$ steps. It consists in launching a random walk from each node s to every other node t. The random walk betweenness of a node i is equal to the number of times that a random walk starting at s and ending at t passes through i along the way, averaged by all possible (s,t) pairs. A heuristic is also added to avoid counting back and forth random walk passages on nodes.

Despite a proper definition of the random walk based centrality, the Newman's approach requires a global knowledge of the graph and this prevents its application in a distributed setting.

Instead, we define a new form of centrality, called the *second order*, as opposed to Newman's approach interested in the first order (expected number of visits to a given node for all source-target pairs [30]). This centrality is computed through the use of a single random walk, in a distributed manner. Each node is required to be aware of its direct neighborhood Γ only; there is no need for any global information anymore.

Note that the random walk tool is also used for other purposes in the context of complex networks, such as *e.g.* community detection [32].

4. Second order centrality

In this section, we first describe the intuition behind our approach. We then present the distributed algorithm which outputs on each node its centrality, the value of which reflects the relative importance of that node in the network.

4.1. The high clustering intuition

To illustrate our purpose, we consider an extreme setting, known as the *barbell graph*. The following barbell graph is composed of two fully connected components (called bells) of m_1 nodes each, connected by a path of m_2 nodes. Fig. 2 depicts such a graph with $m_1 = 5$ and $m_2 = 2$.

Consider a random walk visiting the network from node to node. Firstly, let us consider node v_L in Fig. 2: if a random walk is running in the left bell, v_L has the same probability $1/m_1$, than any other node in this bell to be visited by the random walk at each step. Yet, once the random walk has passed to the right bell, v_L is the mandatory passage point for the walk to get back to the left bell. Therefore, such bridge nodes, v_L and v_R, are visited more *regularly* than other nodes by a random walk continuously running. This is turned into a reduced standard deviation of the number of steps needed for a random walk to return to them, after an initial passage (called *return time*). Our claim is then that different roles of nodes in the topology can be inferred from the return times.

Secondly, return times can also be used to discover topology issues. It can easily be shown (see *e.g.* [5]) that a random walk starting at any node in the left bell, say v_l, takes as mean time $m_1^2 m_2$ steps to reach another node, v_r, in the right bell (and conversely). On the other hand, nodes from a given bell are visited by the random walk often as long as the random walk remains in the same bell. Once it has crossed the path to the other bell, the trends reverses. By simply comparing the standard deviation of return times of the random walk, every graph node can then locally detect the presence of critical paths, or traps, in the topology.

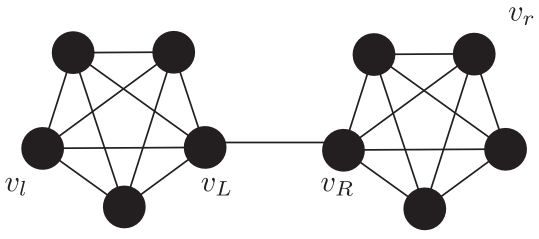

Fig. 2. Example of a Barbell graph.

4.2. Distributed second order algorithm

Based on those observations, we propose in this section the second order centrality, along with a distributed algorithm to compute it, in which each node simply computes the standard deviation of the return times of a permanent unbiased random walk running on the topology.

4.2.1. Unbiased random walk

The paper by Newman considered *simple* random walk s over the graph's transition probability matrix. This represents the classic process where at each node, a random walk is directed to a neighbor of the node, picked uniformly at random. Such a simple forwarding process obviously favors high degree nodes, as its *stationary distribution* $\pi_i = d_i/2m$ (see *e.g.* [28]). The more a node is connected the more often it is visited. Newman points out this issue and shows experimentally that it results in a correlation between a node's degree and its random walk based betweenness centrality, even for nodes that are not central in the topology.

To overcome this issue and cope with heterogeneous distribution of degree, the proposed distributed second order algorithm, relies on an *unbiased* random walk, where π_i is $1/n$ for all $i \in V$. Informally, this means that after a sufficient number of steps (called *mixing time* in literature), the random walk has an equal probability to be on any graph node.

Unbiasing the random walk ensures that the only cause of the variations of return times on nodes is due to their relative importance in the topology, and does not depend on local factors such as node degrees. In addition, the fact that the random walk eventually visits all nodes an equal number of times, speeds up the algorithm's convergence by evenly providing return times to all nodes. The Metropolis–Hastings technique [23,33] is used to unbias the random walk. The node hosting the random walk selects a neighbor uniformly at random: the random walk is forwarded to the chosen neighbor with a probability depending on the degree of both nodes. This process is described from line 1 to line 9 in the algorithm description (Algorithm 1).

Algorithm 1. Second order centrality algorithm

1: **Upon reception of random walk on node** i:
2: /* *Metropolis–Hastings random walk* */
3: Choose a neighbor j from Γ_i uniformly at random
4: Query j for d_j
5: Generate a random number $p \in [0,1]$ uniformly
6: **if** $p \leqslant d_i/d_j$ **then**
7: forward the random walk to j
8: **else**
9: random walk remains at i
10: /* *Standard deviation* */
11: **if** first visit of the random walk on i **then**
12: Create array Ξ_i
13: **else**
14: Compute return time r since last visit
15: Add r to Ξ_i
16: **If** $|\Xi_i| \geqslant 3$ **then**
17: Compute standard deviation $\sigma_i(N)$:
18: $$\sqrt{\frac{1}{N-1}\sum_{k=1}^{N}\Xi_i(k)^2 - \left[\frac{1}{N-1}\sum_{k=1}^{N}\Xi_i(k)\right]^2}$$

4.2.2. Standard deviation of return times

The key point of the algorithm is the variation of the frequency at which a random walk visits nodes. Every node i in $\mathcal{G}$ joins the process on the first visit of the random walk, by creating an array Ξ_i that logs every return time. Recall that the return time to node i is defined by the time, for the walk starting at i, to return to i. We denote by $\Xi_i(k)$ the kth return time of the random walk to node i. A simple solution to capture an irregularity of visits on nodes is to proceed as follows: after the third recorded return time, a node i computes the standard deviation of the N values in Ξ_i. These return times being independent, we have from the strong law of large numbers: $\lim_{N\to\infty}\sigma_i(N) = \sigma_i$.

Once the random walk has run for a sufficiently long time on the graph, σ values represent the relative importance of nodes in the graph: the lower the value, the higher the importance of a node. The description of the algorithm is provided in Algorithm 1.

4.2.3. Algorithm convergence time

4.2.3.1. Theoretical convergence. Each node needs to be visited a few times by the random walk to compute meaningful deviation results. Therefore the algorithm convergence time is related to the *cover time* of graph $\mathcal{G}$. Cover time is defined as the number of steps needed by a random walk to visit each vertex of $\mathcal{G}$. Feige [15] showed that cover time, for a simple random walk, ranges from $(1 + o(1))n\ln n$ steps for a complete undirected graph to at most at $\frac{4}{27}n^3 + o(n^3)$ [16] for the *lollipop* graph (a fully connected graph of $\frac{n}{2}$ nodes, linked to a line of the remaining nodes). This lower bound result holds for an unbiased random walk as all nodes in a complete graph have the same degree. We are not aware of an upper bound for an unbiased random walk. Therefore, our algorithm requires the number of steps to cover the graph, times a constant, so that each node is visited several times ($O(n^3)$).

Other random walk based algorithms (*e.g.* for graph connectivity assessment [17]) exhibit running time of $O(n^3)$ in worst case of input graph irregularity; we expect reduced running times for application-realistic networks. Cover time of the Barabási–Albert graph (often used to model the WWW) for example drops to $\frac{2m}{m-1}n\log n$ [12].

4.2.3.2. Trivial parallelization. For the sake of clarity, we described our algorithm as a purely sequential process, since it uses only one random walk at a time. A simple way of making it parallel is to make all nodes run multiple random walks. The number k of return times needed by each node to get precise approximations of the standard deviation is obtained from the central limit theorem; it can be used, since the successive return times to a node form a sequence of independent and identically distributed random variables. In practice, it is well-known that for $k = 30$, we generally obtain a good approximation in the use of the central limit theorem. Actually, the convergence speed of the empirical distribution towards the normal distribution can be obtained by the Berry-Essen theorem. In that light, if each node launches 30 random walks, algorithm convergence time is then trivially bounded only by one cover time, then removing the constant previously needed. The analysis of a tighter bound on convergence, for the case of the algorithm using multiple random walks (as *e.g.* studied in [6] for particular bias of the walks), is subject to future work.

In the remaining of the paper, we analyse our algorithm by still considering one single random walk in the system.

4.2.3.3. The termination question. We discuss here a limitation of our approach as well as some other random walk based protocols. The fully distributed nature of our model implies that we do not make assumptions on the knowledge of global parameters such as n, the size of the graph, or its conductance (related to random walk mixing time). As a consequence, a single node cannot decide on its own when the algorithm has converged, which means that it does not know when the random walk has run a long enough period of time to have visited a few times the whole graph. Such a problem is related to the general problem of distributed termination detection. An extreme case is the Barbell graph considered in Fig. 2. Assume the random walk starts in the left bell; then node v_l has a high probability to be visited several times before the random walk passes in the right bell. If the graph was less severely degenerated, the few return times computed by v_l would have been sufficient to get a representative σ value; in this particular case, we would like the node to take a value as a result of the algorithm when the random walk has went several times back and forth in both bells. Convergence thus depends on network size and conductance. This problem is actually related to the *mixing time* of the random walk process [5], introduced in the related work Section. Many random walk based algorithms also suffer from this factor [19,33].

In this light, our approach *eventually* converges to a satisfying standard deviation value, meaning a convergence after a constant factor (number of visits needed on each node) times the upper bound on cover time (that handles worst cases of topology wiring). A practical approach to this decision problem on each node is to use periodic comparison of values between nodes. Another one is to relax the assumption on the non-knowledge of n, by having a rough estimation of its order of magnitude (*e.g.* for offline social graph analysis, researchers have an idea of the data-set size, or in peer-to-peer systems, designers have an idea of the popularity of their application), then worst case time to wait is directly derivable from cover time bounds.

5. Return times in Markov chains

This section provides some theoretical analysis of the second order algorithm. More specifically we provide a formula to compute the theoretical standard deviation of return times for any node, given an input graph as a transition probability matrix. In addition, it is used as the baseline centralized theoretical prediction against which the simulation results of the distributed algorithm are compared.

This formula is also useful for (i) a system administrator who wants to predict the behavior of random walks over a particular graph structure before deploying it, or for (ii) graph nodes to derive an expected algorithm completion time when basic properties of the graph are known.

5.1. Standard deviation of return times

We look for a formula that provides standard deviation of return times on a particular node, given by the graph transition probability matrix. This return time is function of the position of this node in the graph. This idea is at the core of our approach, and is generally forgotten due to the fact that literature often provides bounds to return times that hide the local variations nodes may experience (big O notation). Those potentially small variations suffice to differentiate nodes with respect to their position in the graph.

We use a classical discrete time Markov chain model to represent the random walk running on the input graph. States of the Markov chain are nodes, or vertices V of $\mathcal{G}$. The general case of a biased walk is presented first, as an unbiased walk is simply a subcase of it. Finally, for the purpose of the demonstration and to give theoretical results on return times, we consider the transition probability matrix of the considered graph; as precised in Section 2. This global knowledge of the graph is obviously not assumed

in our distributed algorithm proposal. Proofs are deferred in the Appendix.

Let $X = \{X_n,\ n \in \mathbb{N}\}$ be a homogeneous and irreducible discrete time Markov chain on the finite state space S. We denote by $P = (P(i,j))_{i,j\in S}$ its transition probability matrix and we are interested in the computation of the return times for every state of S. For every state $j \in S$, we denote by $\tau(j)$ the number of transitions (random walk steps) needed to reach state j, i.e.

$$\tau(j) = \inf\{n \geqslant 1 | X_n = j\}.$$

The state space S being finite and X being irreducible, X is recurrent which means that $\tau(j)$ is finite a.s. We denote by $f_j^{(n)}(i)$ the distribution of $\tau(j)$ when the initial state of X is i, that is, for every $n \geqslant 1$,

$$f_j^{(n)}(i) = \mathbb{P}\{\tau(j) = n | X_0 = i\}.$$

$f_i^{(n)}(i)$ represents the probability, starting from state i, that the first return to state i occurs at instant n and, for $i \neq j$, $f_j^{(n)}(i)$ represents the probability, starting from state i, that the first visit to state j occurs at instant n. These probabilities are given by the following theorem [11]. For the sake of completeness, the proof of this theorem is also provided in the Appendix.

Theorem 1. *For every $i,j \in S$ and $n \geqslant 1$, we have*

$$f_j^{(n)}(i) = \begin{cases} P(i,j) & \text{if } n = 1, \\ \sum_{\ell \in S - \{j\}} P(i,\ell) f_j^{(n-1)}(\ell) & \text{if } n \geqslant 2. \end{cases} \quad (1)$$

For every $j \in S$ and $n \geqslant 1$, we denote by $f_j^{(n)}$ the column vector containing the values $f_j^{(n)}(i)$ for every $i \in S$. For every $j \in S$, we introduce the matrix Q_j obtained from matrix P by replacing the jth column by zeros, that is

$$Q_j(i,\ell) = \begin{cases} P(i,\ell) & \text{if } \ell \neq j, \\ 0 & \text{if } \ell = j. \end{cases}$$

We also introduce the column vector P_j containing the jth column of matrix P, i.e. $P_j(i) = P(i,j)$. Eq. (1) can then be written in matrix notation as

$$f_j^{(n)} = \begin{cases} P_j & \text{if } n = 1, \\ Q_j f^{(n-1)_j} & \text{if } n \geqslant 2, \end{cases} \quad (2)$$

which leads to an easy computation of the vectors $f_j^{(n)}$. We now define the matrix $M = (M(i,j))_{i,j\in S}$ by $M(i,j) = \mathbb{E}\{\tau(j) | X_0 = i\}$. $M(i,i)$ represents the expected time between two successive visits of X to state i, and, for $i \neq j$, $M(i,j)$ represents the expected time, starting from state i, to reach state j for the first time. The Markov chain X being irreducible, we have $M(i,j) < \infty$ for every $i,j \in S$ and

$$M(i,i) = \frac{1}{\pi_i},$$

where π_i is the ith entry of the probability distribution π, which is the unique solution to the system $\pi = \pi P$.

To compute all the entries of matrix M, we introduce the column vector M_j containing the jth column of matrix M, i.e. $M_j(i) = M(i,j)$ and the column vector of ones denoted by $\mathbb{1}$. These expected values are given by the following result.

Corollary 2. *For every $j \in S$, we have*

$$M_j = (I - Q_j)^{-1}\mathbb{1}. \quad (3)$$

In practice, the column vector M_j is obtained for every $j \in S$ by solving the linear system $(I - Q_j)M_j = \mathbb{1}$.

Let us consider now the second moment of $\tau(j)$. We define the matrix $H = (H(i,j))_{i,j\in S}$ by $H(i,j) = \mathbb{E}\{\tau(j)^2 | X_0 = i\}$. $H(i,i)$ represents the second moment of the time between two successive visits of X to state i, and, for $i \neq j$, $H(i,j)$ represents the second moment of the time, starting from state i, to reach state j for the first time. We introduce the column vector H_j containing the jth column of matrix H, i.e. $H_j(i) = H(i,j)$. These values are given by the following result.

Corollary 3. *For every $j \in S$, we have*

$$H_j = (I - Q_j)^{-1}(I + Q_j)M_j.$$

In practice, the column vector H_j is obtained for every $j \in S$ by solving the linear system $(I - Q_j)H_j = (I + Q_j)M_j$.

The standard deviation $\sigma(i)$ of the return time to state i on a target graph is thus given by

$$\sigma(i) = \sqrt{H(i,i) - [M(i,i)]^2}. \quad (4)$$

5.1.1. Unbiased random walks

When the random walk is unbiased, all the nodes in the graph have the same degree d (Metropolis–Hastings method virtually adds self-loops to poorly connected nodes to adjust their degree) and $P(i,j) = 1/d$ if nodes i and j are connected in the graph and 0 otherwise. This means in particular that matrix P is symmetric and thus bistochastic, i.e. $\mathbb{1}^t P = \mathbb{1}^t$, where t denotes the transpose operator. We then have $\pi_i = 1/|S|$ and $M(i,i) = |S|$. For what concerns the second order moments $H(i,i)$ of return times to state i, we have from Corollary 3,

$$(I - Q_j)H_j = (I + Q_j)M_j.$$

Premultiplying by 1^t, we get

$$\mathbb{1}^t H_j - \mathbb{1}^t Q_j H_j = \mathbb{1}^t M_j + \mathbb{1}^t Q_j M_j. \quad (5)$$

By definition of Q_j, we have $\mathbb{1}^t Q_j = \mathbb{1}^t - e_j$, where e_j is the jth unit row vector, i.e. $e_j(i) = 1$ if $i = j$ and 0 otherwise. So Eq. (5) simplifies as

$$\mathbb{1}^t H_j - (\mathbb{1}^t - e_j)H_j = \mathbb{1}^t M_j + (\mathbb{1}^t - e_j)M_j$$

and thus

$$H(j,j) = 2\sum_{i\in S} M(i,j) - M(j,j) = 2\sum_{i\in S} M(i,j) - |S|.$$

The standard deviation $\sigma(j)$ then writes, from relation (4), as

$$\sigma(j) = \sqrt{2\sum_{i\in S} M(i,j) - |S|(|S|+1)}. \quad (6)$$

5.2. Results for 3 classes of particular graphs

We instantiate this formula for three extreme graph diameter settings: a complete graph (diameter 1), a ring (diameter $\lfloor \frac{n}{2} \rfloor$) and a line (diameter n). On all graphs, we consider an unbiased random walk.

Fig. 3 summarizes our results for which proofs can be found in Appendix. Because of the regularity of the Ring graph and the Complete graph, all nodes of these structures have the same standard deviation. Note that for the line $\sigma(j) = \sigma(n - j - 1)$ because of the symmetry of the structure, and that $\sigma(j)$ is minimal for nodes $\lfloor \frac{n-1}{2} \rfloor$ and $\lceil \frac{n-1}{2} \rceil$. This last remark highlights the fact that our algorithm produces a centrality result, as on a line, critical nodes (*w.r.t.* centrality) are in the middle of it.

Moreover, in the context of symmetric graphs, the ring and the complete graph are extreme cases of resiliency: the ring is the weakest (it is only 2-connected), whereas the complete graph is the most robust structure (it is $(n-1)$-connected). We believe it is reasonable to expect the standard deviation of symmetric graphs

Graph	Transition probability ($i \in S = \{0, \ldots, n-1\}$)	Standard deviation
Ring	P(i,i+1 (mod n)) = P(i,i-1 (mod n)) = 1/2.	$\sigma(j) = \sigma = \sqrt{\frac{n(n-1)(n-2)}{3}}$.
Complete	$P(i,j) = 1/(n-1)$ if $i \neq j$ and $P(i,i) = 0$	$\sigma(j) = \sigma = \sqrt{(n-1)(n-2)}$.
Line	$P(i,i+1) = P(i,i-1) = 1/2$ if $i \in \{1, \ldots, n-2\}$ $P(0,0) = P(0,1) = P(n-1,n-2) = P(n-1,n-1) = \frac{1}{2}$.	$\sigma(j) = \sqrt{\frac{n(n-1)(4n-5)}{3} - 4nj(n-j-1)}$.

Fig. 3. Second order centrality for three particular graphs.

to evolve between the corresponding values (*i.e.* between $o(n)$ for the complete graph, and $o(n^{3/2})$ for the ring).

5.3. Application to specific graphs

Another contribution of this paper is, through the previous formula (6), to be able to provide *signatures*[1] of graphs. We now expose different signatures, that can help to sort graphs according to their *health*: we consider that a graph is healthy if it is robust to multiple targeted attacks, *i.e.* it has a low vulnerability in the sense of [4]. A targeting strategy that consists in aiming central nodes is described in Section 7.2.

Note that this definition of robustness also captures to some extend other interesting robustness aspects of the graph, such as it's average shortest path length, or it's resilience to congestion. Indeed, the former definition captures the average number of intermediary nodes (and thus, points of failure) linking any two nodes of the network, and the latter definition closely relates to the average amount of alternative paths linking them. Intuitively, a robust graph has most of it's nodes close to each other, and linked by many redundant paths. Note also that a healthy graph (*i.e.* resilient to targeted attacks) is also resilient to random attacks (since targeting should be more efficient than random).

The σ value is computed for each node, based on the transition probability matrix (using a random walk unbiased with the Metropolis–Hastings method). We considered the following graphs for theoretical computation: (i) a *random graph*, constructed on the Erdös–Rényi model [14]. The probability that two edges are connected is given by $p = \frac{\varrho \ln n}{n}$. Probability $p > \frac{\ln n}{n}$ insures connectivity of $\mathcal{G}$, *i.e.* that no vertex is isolated. (ii) a *clusterized graph* composed of two equal size random graphs linked by a single edge. The resulting graph is close to a Barbell graph presented earlier in the paper, with the difference that left and right bells are not fully connected (*i.e.* not complete graphs). (iii) a *ring lattice*, where a node i is connected to nodes $i - k/2, i - k/2 + 1, \ldots, i + k/2$ (values mod$n$, and k being an input parameter), excluding itself. Finally (iv) a *scale-free* graph, based on the Barabási/Albert model [3]. We believe that those graphs are representative of classical graph families, *i.e.* graphs that are both widely studied and often targeted in today network designs, or parts of actual social graphs [35]. All four graphs have a size of 10^3 nodes, and all input parameters have been set so that their average degree is 20, insuring a fair comparison.

Results are presented as histograms in Fig. 4. A particular point, say ($x = 2500, y = 3$), expresses that three network nodes have a resulting σ of 2500. We first observe a clear difference in the distribution of results for the tested graphs. A thin distribution of values for a particular graph basically means that all nodes have a similar role or importance in the structure. Contrarily, a significant scattering in the values is to be interpreted as an important irregularity of roles. Furthermore, in graphs of equal sizes and degree

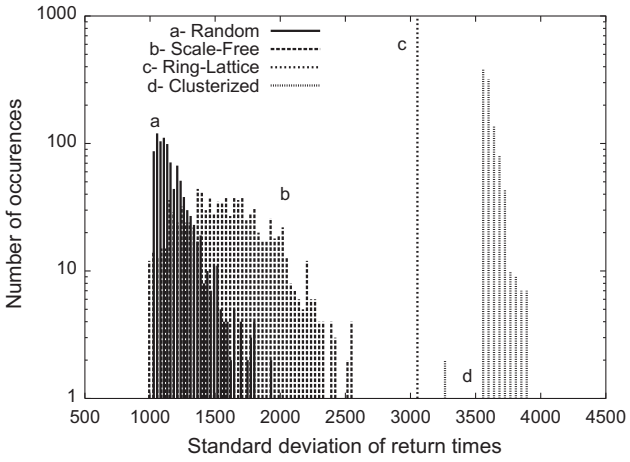

Fig. 4. Histogram of theoretical standard deviation of return times, for four particular graphs.

average, differences in tendencies of mean σ value, reflect discrepancies in the navigation properties of those graphs (related to diameter or presence of bottlenecks).

The graph exhibiting the lowest σ values is the random graph (marked a), as the gathered values on nodes lie approximately in a [1000:1500] step interval. This matches the consensus about the attractive properties (low diameter and low clustering coefficient) of such graphs. It is followed by the Barabási/Albert graph (b), which has a larger repartition of values (1000-2500). This is due to the fact that hubs (highest degree nodes) are part of a lot of shortest paths, and that most of other nodes have a far inferior global importance. In the ring lattice (c), a line-shaped distribution (note the logscale on the y-axis) is observed, due to the perfect regularity of the lattice structure. Finally, the clusterized graph (d) exhibits an interesting distribution for two reasons. First, this structure, composed of two random graphs linked by an edge, produces a σ value, three time the one of a (single) random graph. This is obviously due to the difficulty of the random walk progression in the structure, as exposed in Section 4.1. The second observation concerns the detached line around 3250: the two nodes responsible of this line are the bridge nodes of the structure (as nodes v_L and v_R in Fig. 2). This confirms the intuition given in Section 4.1 stating that those bridges play an important role in the topology and thus see the random walk more regularly than other nodes.

To summarize, this formal method can also be leveraged to assign particular signatures to class of graphs, allowing to sort them by order of healthiness or usability in practice.

6. Evaluation

We now evaluate our distributed algorithm through simulations on various graph topologies, including the previously introduced ones. The distributed second order centrality algorithm is

[1] A graph signature could be seen as a footprint constituted by the distribution of centrality values of its nodes.

evaluated along the following metrics: (i) the ability to produce unbiased results in the presence of heterogeneous degree distributions; (ii) the matching between the theoretical expectations and the experimental results with respect to convergence time, and finally (iii) the practicality of the approach: typically we show that for graphs used in practice, convergence time is, as expected, far less than the upper bounds given by theory.

Theoretical values computed and reported in previous section are used as baseline for comparison. We experimentally show that our algorithm results on nodes converge (*i.e.* tends to accurate results asymptotically), thus validating our proposal.

Experiments have been obtained using the PeerSim discrete event simulator [1].

6.1. Degree bias removal

We now provide a simple algorithm run example, aiming to assess the effectiveness of the Metropolis–Hastings method used jointly to our algorithm, on our typical clusterized graph. Fig. 5 depicts simulation results of the proposed algorithm, over the previously introduced clusterized graph ($n = 10^3$, $p = \frac{1.5 \ln n}{n}$). We run a simple random walk instance, and an unbiased version; results are plotted after 2×10^6 random walk steps.

We observe that for the simple random walk case, there is a clear correlation between resulting standard deviations and nodes' degree. Recall that low σ means a high importance in the topology; high degree nodes then all get a high value, despite a non necessarily real importance. Contrariwise, no effect is measured on the unbiased case, as the σ are concentrated in a tighter range, that does not decrease when nodes' degree is increasing. This example confirms that our algorithm removes the fact of considering a node with a high degree as more important than it really is, regarding the global topology structure.

Note that using an unbiased random walk preserves the importance of high degree nodes, if this high degree is correlated to its importance (for example a node may be some kind of hub and is therefore lying on many shortest paths [30]).

6.2. Speed of convergence

This section studies how fast simulations match the theoretical expectations provided by Formula (6). Fig. 6 plots, for the four previously introduced graphs, the error ratio of algorithm results as the random walk proceeds. The y-axis represents the computation error, that is the algorithm result value minus the theoretical one, normalized by the theoretical value ($y = 0$ then exhibit a perfect result). Presented curves are the average result of all graph nodes' value.

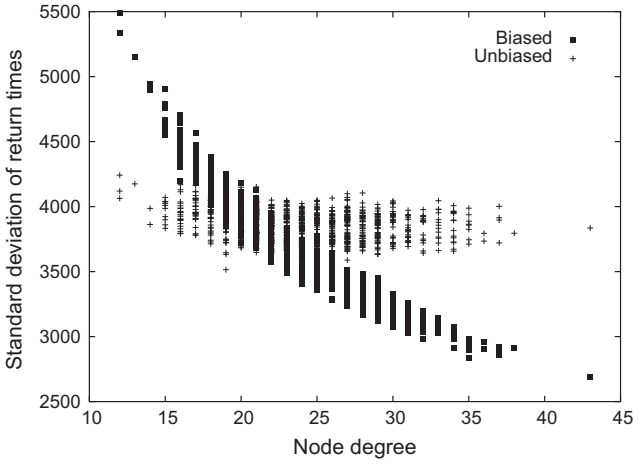

Fig. 5. Simple VS unbiased random walk centrality estimation.

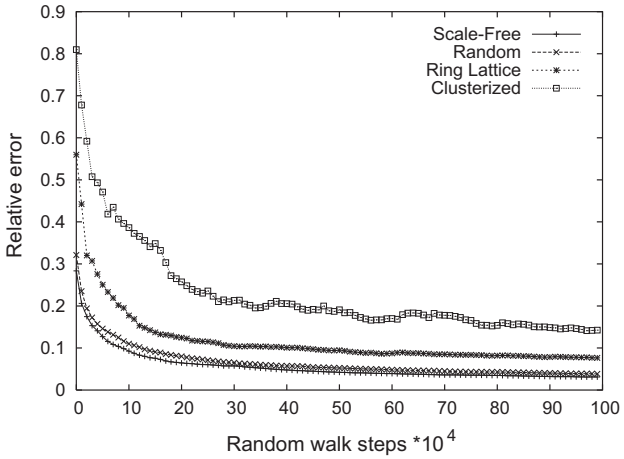

Fig. 6. Convergence of algorithm runs towards theoretical prediction, for four particular graphs.

As the random walk progresses in the graph, the number of visits on nodes strictly increases, and thus gives a larger set of return time values in their Ξ array. The computation of standard deviations then provides continuously improved estimations of importance (we assume here unlimited memory as we believe storing simple integers is not an issue in practice). We observe that this algorithm quickly converges to a small error window, validating its behavior against theory prediction and showing a relatively fast behavior (compared to the worst case $O(n^3)$ theoretical prediction). This reflects the theoretical analysis provided in Section 5.3: the largest the diameter and the more clusterized the graphs, the longer the convergence of our algorithm toward an acceptable value. In high diameter/clusterized settings, the random walk process may get "trapped" in specific zones. Escaping from such zones may take time, thus slowing down the computation of an acceptable standard deviation.

7. Second order centrality compared to other centralities

In addition to Betweenness and Random Walk Betweenness centralities introduced in related work, we briefly define in next subsection other well-known centralities, before comparing them all on a node-removal scenario.

7.1. Other forms of centralities

7.1.1. Degree centrality

The simplest form of centrality, degree centrality assesses the importance of a node according to its degree in the interaction graph. We note $C_d(i) = d_i$, the degree centrality of a node i.

7.1.2. Closeness centrality

Here important nodes are nodes close to all others in the graph. Practically, this is computable for a node i by averaging the distance between i and all other nodes v in G; we note $C_c(i) = \frac{1}{\sum_{j \in V} d(i,j)}$, where $d(i,j)$ is defined as the distance (shortest path) between nodes i and j in the current graph.

7.1.3. Eccentricity

Eccentricity [22] now takes the notion of maximal distance between pairs of nodes, to compute their importance: $C_e(i) = \frac{1}{max_{j \in V} d(i,j)}$

for a node i. The intuition is that a node is central if no node is far from it.

7.1.4. Eigenvector centrality

Another method, proposed by Bonacich [8], is to consider the importance of neighbors of a node; in other words, an important node has important neighbors in the graph topology. Considering a node i, we then have $C_\lambda(i) = \sum_{j \in \Gamma_i} C_\lambda(j)$. Google's *pagerank* algorithm is currently using a variant of eigenvector centrality [31].

7.2. Efficiency of centralities on a relationship network

We now present experiments we ran to compare our approach to existing centralities. The experimental network is a 191 nodes network modeling the largest connected component of jazz players collaborations [21]. In this graph, 10% of nodes have more than 40 neighbors and 10% of nodes have less than 7 neighbors; such imbalance is representative of common social graphs [3].

One way of assessing the absolute importance of nodes given by centralities *w.r.t.* the topology is by removing highest ranked nodes, and observing the resulting graph. Since each centrality allows to rank the nodes according to their importance, each centrality defines a strategy for a targeted attack of the network. We thus compare the effectiveness of the targeting strategy associated with each centrality, considering that a centrality is efficient if it induces an efficient node targeting strategy. A similar approach can be found in [13].

For a given centrality, we sequentially removed nodes starting from the most important remaining one, and then computed on the resulting graph (i) the relative size of the biggest connected component (Fig. 7), and (ii) the average path length between all nodes belonging to this component (Fig. 8). For example, considering ($x = 60$) for the degree centrality, we learn that the original graph minus the 60 highest degree nodes still connects 90% of the remaining 131 nodes, and that the average distance between two nodes belonging that connected part is around 3.8 hops.

As awaited [34], betweenness measures succeed to give importance to critical nodes for graph connectivity, by providing a very similar effect on resulting structure. A drop is observed for both centralities around 45 nodes removed (Fig. 7); it represents the last removal before a first relatively large part of the graph is disconnected (here around 25% of nodes). This expresses their efficiency for identifying critical nodes for structural disruption. An advantage goes to random walk betweenness, which does not only consider optimal paths; the removal of the nodes it identifies damages alternative but yet centrality-important paths. At the same time,

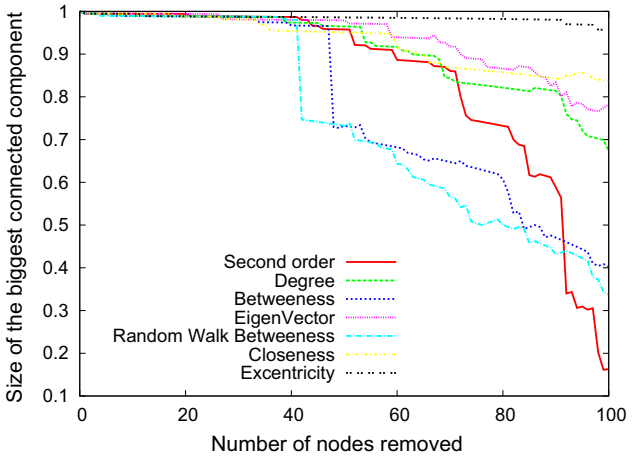

Fig. 7. Impact of removal on component size.

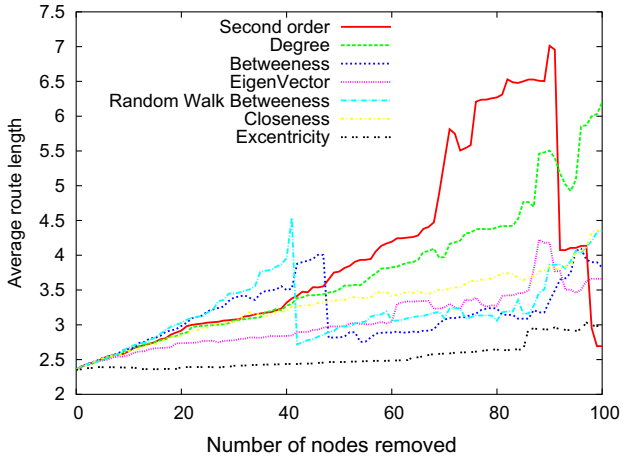

Fig. 8. Impact of removal on route length.

average route length drops, as main component size is reduced due to partition.

Second order centrality does not quickly disconnect large graph parts, but considerably stretches route lengths in the main component. It is somehow related to the random walk betweenness efficiency for non-optimal paths, but by being less critical, it smoothly targets relatively important nodes and then outputs a hardly navigable structure. Note that it ends up with the smallest components: after removing 100 nodes, the largest connected component of the remaining graph is composed by approximately 15 nodes.

We can observe that eccentricity, degree, eigenvector and closeness do not significantly affect graph connectivity; meanwhile, node removal have different effects on route lengths. Degree centrality indeed stretches paths more importantly; it succeeds in sorting important nodes for route lengths, as this simulation actually removes topology hubs that play shortcut roles. Eccentricity behaves poorly in both scenarios, confirming its very particular applicability.

We then conclude that focusing on identification of critical nodes (*w.r.t.* topology disruption), results of our approach sit between the two betweenness techniques we aim to mimic on a distributed way, and the other centralities from state of the art. A more complete panorama, considering applicability and tradeoff of those different centralities, can be found in paper [26].

8. Conclusion

While evaluating the global characteristic of complex networks with respect to connectivity, it is also of the utmost importance to clearly identify the criticity of individual nodes. Such nodes may be at the origin of bottlenecks for example that can significantly hamper the performance of any application running on the network. In an attempt to overcome the drawback of current approaches, which are either not addressing all individual nodes or need to store the complete matrix on a single point (considering contemporary network sizes), we present a new centrality, called the second order centrality. Its preserves the advantages highlighted by Newman over previously introduced centralities, while being simple, lightweight and able to be computed in a decentralized way. Based on our claim that regularity of visits on nodes reflects their relative importance, we showed that a single random walk, running permanently in the system, can distributedly provide values that allow a ranking of nodes in the topology. Theoretical analysis of the second order centrality has been provided. Simulation

results match the analysis and highlight the fact that such an algorithm can be considered in practice to assess the relative importance of nodes in large scale networks, and that results compete with state of the art centralities.

In the light of recent work [6] on the use of multiple parallel random walk s to lower cover time (k times linear speed up for large classes of graphs, for $k \leqslant \log n$ walks), or on the fact [24] that a small extra neighborhood knowledge can suffice to bias a random walk in order to speed up cover time ($O(n^2 \log n)$ instead of $O(n^3)$), forthcoming studies may show that those applications can lower convergence time of our approach, without producing significant side effects on real importance of nodes.

Appendix A. Proofs

A.1. Proof of Theorem 1

By definition of $f_j^{(n)}(i)$ we have, for $n = 1$, $f_j^{(1)}(i) = P(i,j)$. For $n \geqslant 2$, we have

$$\begin{aligned} f_j^{(n)}(i) &= \mathbb{P}\{\tau(j) = n | X_0 = i\} = \mathbb{P}\{X_n = j, X_k \neq j, 1 \leqslant k \leqslant n-1 | X_0 = i\} \\ &= \sum_{\ell \in S - \{j\}} \mathbb{P}\{X_n = j, X_k \neq j, 2 \leqslant k \leqslant n-1, X_1 = \ell | X_0 = i\} \\ &= \sum_{\ell \in S - \{j\}} P(i,\ell)\mathbb{P}\{X_n = j, X_k \neq j, 2 \leqslant k \leqslant n-1 | X_1 = \ell\} \\ &= \sum_{\ell \in S - \{j\}} P(i,\ell)\mathbb{P}\{X_{n-1} = j, X_k \neq j, 1 \leqslant k \leqslant n-2 | X_0 = \ell\} \\ &= \sum_{\ell \in S - \{j\}} P(i,\ell) f_j^{(n-1)}(\ell), \end{aligned}$$

where the last but one and the antepenultimate equalities come respectively from the Markov property and the homogeneity of the Markov chain X. □

A.2. Proof of Corollary 2

Using Relation (2), we obtain

$$\begin{aligned} M_j &= \sum_{n=1}^{\infty} n f_j^{(n)} = P_j + Q_j \sum_{n=2}^{\infty} n f_j^{(n-1)} = P_j + Q_j \left(\sum_{n=1}^{\infty} n f_j^{(n)} + \sum_{n=1}^{\infty} f_j^{(n)} \right) \\ &= P_j + Q_j (M_j + \mathbb{1}), \end{aligned}$$

and, since $P_j + Q_j \mathbb{1} = \mathbb{1}$, we get

$$M_j = Q_j M_j + \mathbb{1}.$$

Matrix Q_j is the submatrix of the transition probability matrix of an absorbing Markov chain with $|S|$ transient states and one absorbing state, thus the matrix $I - Q_j$ is invertible. This leads to

$$M_j = (I - Q_j)^{-1} \mathbb{1}. \qquad \square$$

A.3. Proof of Corollary 3

Using again Relation (2), we obtain

$$\begin{aligned} H_j &= \sum_{n=1}^{\infty} n^2 f_j^{(n)} = P_j + Q_j \sum_{n=2}^{\infty} n^2 f_j^{(n-1)} \\ &= P_j + Q_j \left(\sum_{n=1}^{\infty} n^2 f_j^{(n)} + 2 \sum_{n=1}^{\infty} n f_j^{(n)} + \sum_{n=1}^{\infty} n f_j^{(n)} \right) \\ &= P_j + Q_j (H_j + 2M_j + \mathbb{1}) = Q_j H_j + 2 Q_j M_j + \mathbb{1} \\ &= Q_j H_j + Q_j M_j + M_j, \end{aligned}$$

since, from Corollary 2, we have $Q_j M_j + \mathbb{1} = M_j$. This leads to

$$H_j = (I - Q_j)^{-1} (I + Q_j) M_j. \qquad \square$$

A.4. Proof of Theorem 4 – standard deviation on a Ring graph

On the ring we have $d = 2$ and the non zero transition probabilities are given, f or every $i \in S = \{0, \ldots, n-1\}$, by

$$P(i, i+1 (mod\, n)) = P(i, i-1 (mod\, n)) = 1/2.$$

The standard deviations of the return times are given by the following theorem.

Theorem 4. *For the unbiased random walk on a n nodes ring, we have $\sigma(j) = \sigma$ for every j where*

$$\sigma = \sqrt{\frac{n(n-1)(n-2)}{3}}.$$

Proof. It is easily checked that the solution to Eq. (3) is given, for $i \neq j$, by

$$M(i,j) = (n - |i-j|)(|i-j|)$$

and as mentioned above, we have $M(i,i) = n$. We then have

$$\sum_{i=0}^{n-1} M(i,j) = n + \frac{(n-1)n(n+1)}{6}$$

Using Eq. (6), we obtain the desired result. □

Note that the fact all the $\sigma(j)$'s are equal is due to the regularity of the structure.

A.5. Proof of Theorem 5 – standard deviation on a Complete graph

On the complete graph, we have $d = n - 1$ and thus, the transition probabilities are given, for every $i,j \in S$, by $P(i,j) = 1/(n-1)$ if $i \neq j$ and $P(i,i) = 0$. The standard deviations of the return times are given by the following theorem.

Theorem 5. *For an unbiased random walk on a n nodes complete graph, we have $\sigma(j) = \sigma$ for every j where*

$$\sigma = \sqrt{(n-1)(n-2)}.$$

Proof. It is easily checked that the solution to Eq. (3) is given, for $i \neq j$, by

$$M(i,j) = n - 1$$

and as mentioned above, we have $M(i,i) = n$. We then have

$$\sum_{i=0}^{n-1} M(i,j) = n + (n-1)^2$$

Using Eq. (6), we obtain the desired result. □

Again the fact all the $\sigma(j)$'s are equal is due to the regularity of the structure.

A.6. Proof of Theorem 6 – standard deviation on a Line graph

On a line we have $d = 2$ and the non zero transition probabilities are given, for every $i \in \{1, \ldots, n-2\}$, by

$$P(i, i+1) = P(i, i-1) = 1/2$$

and $P(0,0) = P(0,1) = P(n-1, n-2) = P(n-1, n-1) = 1/2$. The stan-

dard deviations of the return times are given by the following theorem.

Theorem 6. *For the unbiased random walk on a n nodes line, we have for every* $j \in \{0, 1, \ldots, n-1\}$,

$$\sigma(j) = \sqrt{\frac{n(n-1)(4n-5)}{3} - 4nj(n-j-1)}.$$

Proof. It is easily checked that the solution to Eq. (3) is given, for $i \neq j$, by

$$M(i,j) = (j-i)(i+j+1) \quad \text{for } i < j,$$
$$M(i,j) = (i-j)(2n-(i+j+1)) \quad \text{for } i > j$$

and as mentioned above, we have $M(i,i) = n$. We then have

$$\sum_{i=0}^{n-1} M(i,j) = \frac{n(2n^2-3n+4)}{3} - 2nj(n-j-1).$$

Using Eq. (6), we obtain the desired result. □

References

[1] <http://peersim.sourceforge.net>.
[2] Serge Abiteboul, Mihai Preda, Gregory Cobena, Adaptive on-line page importance computation, in: WWW '03: Proceedings of the 12th International Conference on World Wide Web, New York, NY, USA, 2003, pp. 280–290.
[3] R. Albert, A.-L. Barabási, Statistical mechanics of complex networks, Rev. Mod. Phys. 74 (2002) 47–97.
[4] R. Albert, H. Jeong, A.L. Barabási, Error and attack tolerance of complex networks, Nature 406 (6794) (2000) 378–382.
[5] David Aldous, James Allen Fill, Reversible Markov Chains and Random Walks on Graphs, 1995. <http://stat-www.berkeley.edu/users/aldous>.
[6] Noga Alon, Chen Avin, Michal Koucky, Gady Kozma, Zvi Lotker, Mark R. Tuttle, Many random walks are faster than one, in: SPAA '08: Proceedings of the 20th Annual Symposium on Parallelism in Algorithms and Architectures, ACM, New York, NY, USA, 2008, pp. 119–128.
[7] Marc Barthelemy, Betweenness centrality in large complex networks, Eur. Phys. J. B 38 (2004) 163.
[8] P. Bonacich, Factoring and weighting approaches to status scores and clique identification, J. Math. Sociol. 2 (1972) 113–120.
[9] Ulrik Brandes, A faster algorithm for betweenness centrality, J. Math. Sociol. 25 (2001) 163–177.
[10] Ulrik Brandes, Christian Pich, Centrality estimation in large networks, Int. J. Bifurcat. Chaos (2007) 2303–2318.
[11] E. Cinlar, Introduction to Stochastic Processes, Prentice Hall, New Jersey, 1975.
[12] Colin Cooper, Alan Frieze, The cover time of the preferential attachment graph, J. Comb. Theory Ser. B 97 (2) (2007) 269–290.
[13] L. DallAsta, A. Barrat, M. Barthelemy, A. Vespignani, Vulnerability of weighted networks, J. Stat. Mech.: Theory Exp. 2006 (2006) P04006.
[14] P. Erdös, A. Renyi, On random graphs, in: Publicationes Mathematicae, vol. 6, 1959, pp. 290–297.
[15] Uriel Feige, A tight lower bound for the cover time of random walks on graphs, random structures and algorithms 6, in: Random Structures and Algorithms, 1995, pp. 433–438.
[16] Uriel Feige, A tight upper bound on the cover time for random walks on graphs, RSA: Random Struct. Algor. 6 (1995).
[17] Uriel Feige, A fast randomized logspace algorithm for graph connectivity, Theor. Comput. Sci. 169 (2) (1996) 147–160.
[18] Linton C. Freeman, A set of measures of centrality based on betweenness, Sociometry 40 (1) (1977) 35–41.
[19] Ayalvadi Ganesh, Anne-Marie Kermarrec, Erwan Le Merrer, Laurent Massoulié, Peer counting and sampling in overlay networks based on random walks, Distrib. Comput. 20 (4) (2007) 267–278.
[20] Robert Geisberger, Peter Sanders, Dominik Schultes, Better approximation of betweenness centrality, in: Proceedings of the Ninth Workshop on Algorithm Engineering and Experiments (ALENEX), 2008.
[21] P. Gleiser, L. Danon, Community structure in jazz, Adv. Complex Syst. 6 (2003) 565.
[22] P. Hage, F. Harary, Eccentricity and centrality in networks, Social Netw. 17 (1) (1995) 57–63.
[23] W.K. Hastings, Monte Carlo sampling methods using Markov chains and their applications, Biometrika 57 (1) (1970) 97–109.
[24] Satoshi Ikeda, Izumi Kubo, Norihiro Okumoto, Masafumi Yamashita, Impact of local topological information on random walks on finite graphs, in: ICALP, 2003, pp. 1054–1067.
[25] Matthieu Latapy, Clémence Magnien, Measuring fundamental properties of real-world complex networks, CoRR, abs/cs/0609115, 2006.
[26] Erwan Le Merrer, Gilles Trédan, Centralities: capturing the fuzzy notion of importance in social graphs, in: SNS '09: ACM EuroSys Workshop on Social Network Systems, 2009, pp. 33–38.
[27] Qin Lv, Pei Cao, Edith Cohen, Kai Li, Scott Shenker, Search and replication in unstructured peer-to-peer networks, in: ICS '02: Proceedings of the 16th International Conference on Supercomputing, ACM, New York, NY, USA, 2002, pp. 84–95.
[28] Rajeev Motwani, Prabhakar Raghavan, Randomized Algorithms, Cambridge University Press, New York, NY, USA, 1995.
[29] Soumendra Nanda, David Kotz, Localized bridging centrality for distributed network analysis, in: Proceedings of the 17th International Conference on Computer Communications and Networks (ICCCN), August 2008, pp. 1–6.
[30] Newman, A measure of betweenness centrality based on random walks, Social Netw. 27 (1) (2005) 39–54.
[31] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, The pagerank citation ranking: bringing order to the web, Technical Report, Stanford Digital Library Technologies Project, 1998.
[32] Pascal Pons, Matthieu Latapy, Computing communities in large networks using random walks, in: Computer and Information Sciences – ISCIS 2005, vol. 3733, 2005, pp. 284–293.
[33] Daniel Stutzbach, Reza Rejaie, Nick Duffield, Subhabrata Sen, Walter Willinger, On unbiased sampling for unstructured peer-to-peer networks, in: IMC '06: Proceedings of the Sixth ACM SIGCOMM Conference on Internet Measurement, ACM, New York, NY, USA, 2006, pp. 27–40.
[34] Stanley Wasserman, Katherine Faust, Dawn Iacobucci, Social Network Analysis : Methods and Applications (Structural Analysis in the Social Sciences), Cambridge University Press, 1994. November.
[35] Jennifer Xu, Hsinchun Chen, The topology of dark networks, Commun. ACM 51 (10) (2008) 58–65.