BIROn - Birkbeck Institutional Research Online

De Meo, Pasquale and Levene, Mark and Provetti, Alessandro (2023) Branching processes reveal influential nodes in social networks. Information Sciences 644 (119201), pp. 1-19. ISSN 0020-0255.

Downloaded from: https://eprints.bbk.ac.uk/id/eprint/51392/

Usage Guidelines:
Please refer to usage guidelines at https://eprints.bbk.ac.uk/policies.html or alternatively contact lib-eprints@bbk.ac.uk.

Information Sciences 644 (2023) 119201

Contents lists available at ScienceDirect

Information Sciences

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

Branching processes reveal influential nodes in social networks

Pasquale De Meo [a], Mark Levene [b], Alessandro Provetti [c,*]

[a] Department of Modern and Ancient Civilizations (DICAM), University of Messina, Messina, I-98166, Italy
[b] National Physical Laboratory, Teddington, Middlesex, TW11 0LW, UK
[c] Department of Computer Science and Information Systems, Birkbeck - University of London, London WC1E 7HX, UK

ARTICLE INFO

Keywords:
Network centrality
Branching processes
Web navigation

ABSTRACT

Branching processes are discrete-time stochastic processes which have been largely employed to model and simulate information diffusion processes over large online social networks such as Twitter and Reddit. Here we show that a variant of the branching process model enables the prediction of the popularity of user-generated content and thus can serve as a method for ranking search results or suggestions displayed to users. The proposed branching-process variant is able to evaluate the importance of an agent in a social network and, thus we propose a novel centrality index, called the Stochastic Potential Gain (SPG). The SPG is the first centrality index which combines the knowledge of the network topology with a dynamic process taking place on it which we call *a graph-driven branching process.* SPG generalises a range of popular network centrality metrics such as Katz' and Subgraph. We formulate a Monte Carlo algorithm (called MCPG) to compute the SPG and prove that it is convergent and correct. Experiments on two real datasets drawn from Facebook and GitHub demonstrate that MCPG traverses only a small fraction of nodes to produce its result, thus making the Stochastic Potential Gain an appealing option to compute node centrality measure for Online social networks.

1. Introduction

Branching processes are discrete-time stochastic processes used to describe how a population evolves over time [22] and, more recently, they have gained wide popularity as a tool to study how information spreads in large online social networks such as Twitter [3,44,50,29,18] or Reddit [40]. Understanding the factors triggering an information cascade has important implications both at the scientific level and the commercial one. From a scientific standpoint the viral re-sharing of content on platforms like Twitter is the visible byproduct of complex collective behaviour of uncoordinated and vast masses. An in-depth analysis of the spread of information helps us discriminate between the information that catches user attention and information that leaves them indifferent. From a practical standpoint, members of a Web platform are often overwhelmed by (mostly unwanted) contents while relevant and timely information could be lost. Thus, the ability of correctly estimating which content will be popular is crucial to provide better ranking (and personalisation) of the contents.

In the literature many generative models have been proposed, their common purpose is to simulate diffusion processes and make predictions. They mostly derive either from the standard branching process, often called the *Galton-Watson process,* or from its extensions, e.g., the so-called *Hawkes process* [23,50,29]. We believe that the potential of branching processes as a tool to investigate

* Corresponding author.
E-mail addresses: pdemeo@unime.it (P. De Meo), m.levene@npl.co.uk (M. Levene), a.provetti@bbk.ac.uk (A. Provetti).

https://doi.org/10.1016/j.ins.2023.119201
Received 4 December 2022; Received in revised form 17 May 2023; Accepted 20 May 2023

Available online 26 May 2023
0020-0255/© 2023 The Author(s). Published by Elsevier Inc. This is an open access article under the CC BY license (http://creativecommons.org/licenses/by/4.0/).

the structure of large networks has not yet been fully realised: most previous research focuses on (user-produced) content but the agent who created the content is hardly taken into account. In our opinion, branching processes can serve as a tool to identify *influential users (agents)* in online social networks: intuitively, we may classify as influential (or 'important') those users who create contents which are at the root of an information cascade.

The importance of a user in a social network has been extensively studied recently and *centrality metrics* have been proposed [37,10,47,19] but, to the best of our knowledge, branching processes have not yet been applied to formulate and compute centrality in networks. Here we propose a novel centrality index called the *Stochastic Potential Gain* (SPG in short), which combines structural information regarding the network/graph topology with the theory of branching processes.

Some standard notation to be used throughout the article follows. Let $G = \langle N, E \rangle$ denote an undirected and connected graph where N is its set of nodes and E its set of edges; d_i will be the degree of node i. We will assume that a strictly monotonically decreasing function (called *discount* function) $f : \mathbb{N} \to [0, 1]$ is available. The discount function $f(k)$ returns the probability that a node in G will have one of its neighbours as a child at the k-th generation in a branching process. Thus $f(\cdot)$ quantifies how the 'fertility' of a node varies across generations.

The computation of the SPG associated with an arbitrary node $i \in N$ proceeds as follows: we start from i and set the SPG of i equal to 1. For each neighbour, say j of i, a coin is tossed with success probability equal to $f(1)$. If the coin toss comes up with a success then we pick (i.e., retain) j, otherwise we discard it. This process is repeated for each of the neighbours of i that has been picked and, thus, it is equivalent to drawing a sample of the neighbours of i whose size follows a binomial distribution $\mathcal{B}(d_i, f(1))$. Here, the number of trials is equal to the degree d_i of i and the probability of success is $f(1)$. The process now restarts from the picked nodes: if j has been selected then with (decreased) probability $f(2) < f(1)$ we will continue the exploration towards j's neighbourhood. We will show that with a proper choice of $f(k)$ the process describe above converges.

The SPG of i is thus defined as the sum of all coefficients $f(k)$ associated with nodes in G which have been selected in the network traversal process outlined above. We call the stochastic processes outlined above a *graph-driven branching process.* The main contributions of this research are now summarised.

The graph-driven branching process is a novel stochastic process In standard branching processes the probability of extinction, that is the probability that no individuals exist after a particular generation, *depends only on the expected number of children* an individual can have. In our case, the extinction of the exploration process jointly depends on the function $f(k)$, as well as on the graph topology. At generation k, a node j has an expected number of children equal to $d_j \cdot f(k)$. Thus, nodes with large degree can compensate small levels of fertility and, analogously, a high level of fertility can overcome issues with making derivations from nodes with small degrees, thus avoiding an early extinction of the process. As a further difference, observe that classical branching processes assume that the probability an individual has a particular number of children is constant across generations; in our model, in contrast, the fertility ratio $f(k)$ decreases with the generation counter k.

We re-interpret well-known centrality metrics The mainstream Degree centrality, eigenvector centrality, PageRank, betweenness and closeness centrality [43] can be regarded as single-parameter functions ϕ that take as input the adjacency matrix $\mathbf{A}$ of a network and generates a vector of centrality scores.[1] In some cases (e.g., Katz's Centrality [28] and Subgraph centrality [16]), the adjacency matrix $\mathbf{A}$ is coupled with some additional parameter. Our approach, instead, jointly requires the specification of the adjacency matrix $\mathbf{A}$ and a discount function $f(\cdot)$ which quantifies the speed at which our process becomes extinct. As a consequence, the most innovative feature of our study is the combination of topological information with a dynamic process, that is the reproduction of individuals constrained by graph topology in order to calculate the importance of nodes.

We provide an efficient Monte Carlo algorithm (called Monte Carlo Potential Gain - MCPG*) to efficiently calculate the SPG* Unlike other centrality metrics, the computation of the SPG does not require to take the whole graph G as input but only to look up the portion of nodes (and edges) which the MCPG algorithm traverses before our graph-driven branching process becomes extinct. The number of nodes being traversed depends on the form of the function $f(\cdot)$ and, thus, with a proper choice of $f(\cdot)$ we can significantly reduce this number, resulting in computational efficiency.

We have experimentally tested[2] the performance of the MCPG algorithm on two large datasets, namely **FB-Friends**, a dataset about friendship data of Facebook users [48] and **GitHub**, a dataset describing a social network of GitHub developers [45]. In both datasets we found that it was necessary to traverse only one thousandth of the nodes to complete computation.

This article is organised as follows: in Section 2 we describe related approaches, while in Section 3 we introduce basic terminology used throughout the article. We introduce branching processes on networks in Section 4 and the MCPG algorithm in the following section. Section 6 describes our experimental results while in Section 8 we report our conclusions and illustrate potential future work. The proofs of our formal results are given in Appendix A.

2. Related work

In this section we briefly review applications of branching processes to the analysis of large real-world networks. Many online social networks expose large volumes of data, often endowed with temporal labels [36,32]; available datasets make possible the

[1] Recall that the adjacency matrix $\mathbf{A}$ is an $|N| \times |N|$ matrix such that $\mathbf{A}_{ij} = 1$ if and only if nodes i and j are connected by an edge, 0 otherwise.

[2] Please find the datasets and the Python code for the experiments at https://github.com/ale66/Montecarlo-Potential-Gain.

investigation of how information flows in human societies. Moreover, branching processes have emerged as an appealing option to describe information cascades [3,17,39].

In our literature review we found that most existing studies focus on Twitter and two types of information cascade are usually considered, namely *retweet diffusion* (user forwards/re-shares, e.g., a tweet to her/his followers) and *reply diffusion* (user replies to a tweet). Other researchers focused on online discussion platforms such as Digg or Reddit; in these platforms, users submit their content (e.g., posts, videos and links) and interact with each other mostly by commenting posts. User interactions lead to the creation of reply cascades known as *discussion trees* [31].

One of the early studies of the reply diffusion is due to Nishi et al. [44], who classified *reply trees* in Twitter into three categories: *a)* long path-like, *b)* large star-like and *c)* large irregular reply trees. The Galton-Watson process [22] was applied to predict the frequency of trees in each of the categories above. Nishi et al. [44] also investigated the distributions of the size and the depth of reply trees. The matching between empirical data and predictions made with the Galton-Watson process turned out to be unsatisfactory; hence several extensions of the Galton-Watson process were considered. One of the main findings of Nishi et al. [44] is that the in-degree of the "root tweet" (that is the tweet from which the cascade started) has a key role in determining the shape of the reply tree.

Zhao et al. [50] applied *Hawkes processes* [23] to predict the total number R_t of re-tweets received by a tweet at the time step t. Hawkes processes extend the Galton Watson branching process by incorporating a further parameter, called a *memory kernel*, which measures the time between a cause (in our case, a tweet) and its effect (a retweet). The estimation of R_t depends on the memory kernel and the *infectiousness,* that is the probability that a given user will retweet a given tweet.

Kobayashi and Lambiotte [29] extended the approach of Zhao et al. [50] by considering time-dependent Hawkes processes (that is, they assume that the parameters defining the process can vary daily). In this way, Kobayashi-Lambiotte can predict not only the cascade size but also how the popularity of a tweet (defined as the number of retweets per day) evolve over time.

The approaches cited so far mostly aim at predicting the cascade size. An important extension is due to Gleeson et al. [18], who investigated how the design features of a given web platform affect the offspring distribution; they found that a model which considers users' limited attention (see [34]) was more appropriate to describe the process of cascade propagation than those like, e.g., the well-known Independent Cascade Model, which did not.

More recently, [14] applied deep learning methods to integrate tweet features (derived from the tweet's content), network features (that is, the followers of the user who posted it first) and external signals (e.g., the popularity and novelty of the news corresponding to the tweet), in order to predict the cascade size.

Medvedev et al. [40] studied the shape and the temporal evolution of discussion trees in a Reddit dataset. The analysis of reply/retweet trees totally differs from the corresponding discussion trees; in case of reply/retweet trees (e.g., again, Twitter) the online social network mapping the follower-followed relationships has a strong influence on the size and depth of the reply/retweet tree, as shown by Nishi et al. [44]. In contrast, platforms such as Reddit make content available to all visitors (and, often, posts are only discriminated on the basis of their recency). Another contribution of Medvedev et al. [40] is to apply Hawkes processes to model the growth of discussion trees with the goal of predicting which posts are more likely to be commented in the near future. Predictions were validated on a public Reddit dataset with posts collected between 2008 and 2015. Medvedev et al. proved that discussion trees in Reddit are properly described by a variant of branching trees in which the root offspring follows a specific distribution.

3. Background

In this section we introduce some basic definitions about graphs.

Definition 1 *(Network)*. A *network* G (also called graph) is a pair $G = \langle N, E\rangle$ where N is the *node* set and E is the *edge* set. A graph is *undirected* if $\langle i, j\rangle \in E$ implies $\langle j, i\rangle \in E$, that is, for any edge from a node i to a node j there is also an edge from j to i. Otherwise, the graph is *directed.* We set $n = |N|$ and $m = |E|$ as the number of nodes and edges of G, respectively.

Here we will focus on undirected networks and leave the extension to directed ones as future work. Given a node $i \in N$, the *neighbourhood* $N(i)$ of i is the set of nodes connected with i, that is $N(i) = \{j \in N : \langle i, j\rangle \in E\}$. The degree d_i of i is the size of the neighbourhood of i: $d_i = |N(i)| = |\{j \in N : \langle i, j\rangle \in E\}|$. A network G can be represented through its *adjacency matrix* $\mathbf{A}$, that is a $n \times n$ matrix such that the entry $\mathbf{a}_{ij}$ is equal to one if and only if there is an edge from i to j, zero otherwise. In the following, we will also find useful the definition of *walks* and *paths* in a graph.

Definition 2 *(Paths and walks in a Network)*. Let $G = \langle N, E\rangle$ be a network. A sequence of nodes $w = \{i_0, \dots, i_l\}$ is called a *walk* of length ℓ if pairs of consecutive nodes in w are connected by an edge, that is $\langle i_t, i_{t+1}\rangle \in E$ for any t such that $0 \le t \le \ell - 1$. A *path* $\{i_0, \dots, i_\ell\}$ *of length* ℓ is a walk which does not contain repeated nodes. A network G is *connected* if, for any pair of nodes i and j, there is at least one path connecting them.

Paths in networks/graphs can be easily discovered by a *graph traversal algorithms* such as the classical Depth First Search (DFS) and Breadth First Search (BFS) [7]. Practically, if we square the adjacency matrix $\mathbf{A}$ of a graph G we obtain a matrix $\mathbf{A}^2$ with entries $a_{ij}^{(2)} = \sum_{r=1}^{n} a_{ir}a_{rj}$. It is easy to see how each value $a_{ij}^{(2)}$ will count the number of walks of length two going from i to j [8] in G. This notation easily extends to arbitrary integers ℓ: consider the matrix $\mathbf{A}^\ell$ obtained by multiplying $\mathbf{A}$ by itself ℓ times; we have that the generic entry $a_{ij}^{(\ell)}$ of $\mathbf{A}^\ell$ is equal to the number of walks of length ℓ between nodes i and j [8].

For the undirected networks we consider here the adjacency matrix **A** will be symmetric so that all its eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n$ will be real. The largest eigenvalue λ_1 of **A** is also called its *principal eigenvalue* or *spectral radius* of G.

3.1. The potential gain

The *potential gain* (PG) was originally introduced by Levene and Wheeldon [35] with the purpose of discovering "good" starting pages (nodes) for navigating a network of Web pages and their hyperlinks. Potential gain was originally defined for directed networks, however for the purpose of this article, there is no loss of generality in assuming that the graph is undirected, that is, hyperlinks are bidirectional.

Web pages displaying "high" PG scores should satisfy the following criteria: *i)* they should be *relevant*, that is they should match user's goals; *ii)* they should be *central*, that is the distance of a Web page to all other pages should be as small as possible: the distance between two pages p_1 and p_2 is the length of the shortest chain of hyperlinks from p_1 to p_2; *iii)* they should be *well connected,* that is, they are able to reach the largest number of Web pages. Algorithm 1 reports the pseudo-code for computing the PG of the network represented by matrix **A**. It begins by initialising the PG of all nodes to one.

Algorithm 1 Computation of the Potential Gain (see [35]).

```
cur ← 1_n
PG ← 1_n
for d ∈ {1, ..., MAX} do
    cur ← A × cur
    PG ← PG + δ(d) · cur
end for
return PG
```

Next, PG values are iteratively updated; at most MAX iterations will run, MAX one of the inputs of the algorithm. In this way, for each node i we explore G in a breadth-first fashion and count the number of nodes at distance d from i, where d ranges from 1 to MAX. To such value we apply a penalty coefficient $\delta(d)$—the discount function described earlier–then add it to the current PG value for i. Here, $\delta(\cdot)$ is a monotonically-decreasing function; in other words we *normalise* the PG value so that its growth is bounded.

Levene and Wheeldon [35] suggested two possible variants for $\delta(d)$ (the analog to our $f(d)$) namely the *reciprocal* function $\delta(d) = \frac{1}{d}$ and the *discount* function $\delta(d) = \gamma^{d-1}$ where $0 < \gamma < 1$. Both variants encode the assumption that the utility of a page decreases with its distance from the 'start' page; this hypothesis was supported by experiments carried out over Web data sets [25].

Recently, De Meo et al. [13,12] generalised the concept of the Potential Gain to account for node centrality in generic networks. They proved that the PG is able to capture the notion of *graph navigability*, understood as the ease at which it is possible to reach a target node i in G regardless of the node from where the navigation is started.

Intuitively, the higher the PG of a node i, the easier it will be to reach all other nodes from there. More specifically, [12] proved that the PG of node i depends on the number of walks of any length between it and all other nodes. As discussed above, the contribution of each walk to the computation of the PG should decrease with its length. Hence a weighting function $\phi(\ell)$ was introduced to penalise longer walks. The following two possible definitions for $\phi(\cdot)$ were introduced.

1. The *geometric potential gain, GPG:* $\phi(l) = \delta^l$ where δ is a parameter ranging between 0 and λ_1^{-1}, i.e., the inverse of the spectral radius λ_1 of the network. In this case, the PG of a node i is equal to the product its degree, d_i, times its *Katz centrality score* [28,33].
2. The *exponential potential gain, EPG* where $\phi(k) = \frac{1}{(k-1)!}$. Here the PG of a node i is equal to the product of its degree d_i times its *Communicability score*, which is also known as *Subgraph centrality* [15,4].

3.2. Branching processes

The simplest formalism of a branching process is the *Galton-Watson process,* a simple but powerful time-discrete model to describe the growth of a population [22,21] in mathematical terms. Consider a population of reproducing individuals; each one reproduce independently and with identically distributed probability. The number of offspring i will be represented by a random variable Y_i. In fact, every individual i in the population lives one unit of time, generate Y_i offspring then dies at the end of the time unit. We denote the family size of each individual as $Y_1, Y_2, \ldots, Y_m, \ldots$ and $Y_i \sim Y$ where Y is a random variable called the *number of young*. The probability for an individual to generate r offspring is equal to p_r, that is $P(Y = r) = p_r$.

Now suppose that at time instant $x = 0$ there is only one individual. The lone individual lives one unit of time, generates one or more Offspring then dies. As we repeat the process for the following time units, we introduce Z_t as the size of the population at time t. By definition Z_0 is always equal to 1 and the branching process is thus defined by the sequence $\{Z_0, Z_1, \ldots, Z_t, \ldots\}$.

A key question regarding the computation of the *probability of ultimate extinction*, that is the probability θ that no individual exists after a finite (and fixed) number of generations. We have that $\theta = \lim_{x \to +\infty} P\left(Z_x = 0\right)$. If we denote μ the expected number of children of an individual, we have that the expected number of individuals at the x-th generation is μ^x; if $\mu < 1$, then $Z_x \to 0$ and, thus, θ converges to one. On the other hand, if $\mu > 1$ (that is, an individual has, on average, more than one child) then θ is strictly less than one. An interesting situation occurs when $\mu = 1$: here, θ is equal to one, unless each individual has exactly one child.

3.3. The binomial distribution

A *binomial random variable* Z with parameters n and p is a discrete random variable, whose probability distribution $\mathcal{B}(n,p)$ represents the *number of successes* in a sequence of n independent experiments (also called *trials*), assuming that each experiment *succeeds* with probability p (and thus *fails* with probability $q = 1 - p$).

The outcome of each trial can be equivalently described by random variable $X \sim \mathrm{Ber}(p)$ where $\mathrm{Ber}(p)$ is Bernoulli random variable of p; this implies that any binomial random variable with parameters n and p can be thought as the sum of n independent random variables $X_i \sim \mathrm{Ber}(p)$. Since the random variables X_i are independent and identically distributed, we have that the expectation $\mathbb{E}[Z]$ of Z is equal to np while the variance $\mathrm{Var}[Z]$ is equal to $np(1-p)$. We will use the following classic result, which is elegantly proved, e.g., in [30].

Theorem 1. *Let $X \sim \mathcal{B}(n,p)$ and $Y \sim \mathcal{B}(m,p)$ be two independent binomial variables with the same probability distribution p. The random variable $Z = X + Y$ is a binomial variable with distribution $Z = X + Y \sim B(n+m,p)$.*

The result above can be easily extended to the sum of three or more binomial random variables.

4. Graph-driven branching processes and the potential gain

In this section we introduce the notion of a network-driven branching process (see Section 4.1). We then use it in Section 4.2 to derive the definition of the Stochastic Potential Gain (SPG).

4.1. Graph-driven branching processes

Consider a network $G = \langle N, E \rangle$ and let $i \in N$ be the *starting node*, i.e. the first individual of a population, at time step $x = 0$. As described above, i can reproduce and its offspring will coincide with its neighbouring nodes (or a subset of them). The process above repeats generation after generation and it may eventually terminate after a certain number of generations.

We can apply the process above by choosing any node in N as the starting node and, for each node, we can compute the expected size of the population that is being generated. Hence nodes can be classified (or ranked) on the basis of the expected size of the population they generate. We assume that the most prolific nodes shall also be the most central ones.

To make our discussion more precise we need to specify the mechanism that regulates node reproduction. To this purpose, the simplest option it to apply the Galton-Watson (hereafter GW) process described in Section 3.2. Unfortunately, the GW process is not effective in ranking nodes: it assumes that the number of offspring is modelled as a random variable Y which is the same for all nodes and for all generations. Thus, the expected number of offspring is the same for all nodes and we cannot search for "prolific" nodes. To discriminate nodes on the bases of their descendants we need to alter the traditional GW branching process; this is done in two ways:

1. A node can have as children a subset of its neighbours and, possibly, it can have all its neighbours as children.
2. The "fecundity" of a node decreases from one generation to the next one. We suppose that there exists a non-increasing function $f(k)$ that specifies the probability that a node will have an offspring at the k-th generation.

We show that modifications (1) and (2) will enable the modified branching process to discriminate between nodes, and thus rank them. Also, a proper choice of $f(k)$ will yield the stochastic formulation of the potential gain described in Section 3.1. Start with modification (1) and assume, as a first approximation, that the function $f(k)$ is constant across generations and fixed to 1. Now our process starts from node i at time $k = 1$, where i has d_i children (since *f(k) = 1)*. Next, each neighbour j of i will, in turn, reproduce and generate d_j children. The process described here, which could continue indefinitely, is called *the unfolding of G at node i;* it produces a branching tree rooted at i and called $\Psi(G,i)$. The unfolding $\Psi(G,i)$ will contain the same node multiple times. To visualise this it suffices to observe that if i generates j then j will also generate back i as on offspring at the next level of the unfolding. This observation is generalised to the fact that nodes positioned at depth ℓ in the unfolding are certainly connected to i via a walk of length ℓ. In real networks the node degree is often distributed unevenly: consequently, unlike the GW process our modified process enables some nodes to have more children than others.

We can now relax the assumption that $f(k) = 1$ for any k, we will study the case when it decreases over time, i.e., when k grows. Recall that $f(k) = \alpha^k$ with $0 < \alpha < 1$ and consider the first k levels of the unfolding. In this case the number of leaves in $\Psi(G,i)$ at level ℓ is equal to the number of walks of length ℓ coming out of i. Each of those leaves will be assigned weight equal to $f(\ell)$. Finally, we can define the *stochastic potential gain* (SPG) as the random variable given by the sum of the weights of all the walks, of any length, which start from the origin node. In the next subsection we will show that the SPG random variable is in fact closely related to the Potential Gain of [35].

4.2. The stochastic potential gain

In this section we provide a formal definition of the stochastic potential gain. Again, assume a connected and undirected network $G = \langle N, E \rangle$ as input. The next three definitions are needed to properly define the SPG in Definition 6. We begin by defining the *unfolding* of a network.

Definition 3 *(Unfolding)*. Let $G = \langle N, E \rangle$ be a graph and let $i \in N$ be a node of G. The *unfolding* $\Psi(G, i)$ is a, possibly infinite, tree rooted in $i \in N$ which results from traversing G starting from i. Each time a node is *revisited* during the traversal a duplicate of it is added to the node set of $\Psi(G, i)$. For any non-negative integer $\ell \in \mathbb{N}^+$ we denote by $\mathcal{L}(G, i, \ell)$ the set of nodes in the ℓ-th level of $\Psi(G, i)$.

From Definition 3 we see that nodes in $\mathcal{L}(G, i, \ell)$ are connected to the root i by means of walks of length ℓ. Since multiple walks (of different length) could join any arbitrary pair of nodes, we have that a node can appear more than once in $\Psi(G, i)$ and on different levels. We are now ready to introduce *discount functions* to our model.

Definition 4 *(Discount Function)*. Let $f : \mathbb{N} \to (0, 1]$ be a strictly monotonically decreasing function. We say that $f(\cdot)$ is a *discount function* if it satisfies the following two properties:

1. $f(0) = 1$, and
2. $0 < f(k) < 1$ for any $k > 0$.

For instance, function $f(k) = \left(\frac{1}{2}\right)^k$ is a discount function for $k = 0, 1, 2, \ldots$. A *graph-driven branching process* on G and starting at $i \in N$ is defined next.

Definition 5 *(Graph-Driven Branching Process)*. Let $G = \langle N, E \rangle$ be an undirected and connected network and let $i \in N$ be one of its nodes. Let also $\Psi(G, i)$ be the unfolding tree associated with G and rooted at i.

We define the *graph-driven branching process* associated with G and i as the branching process corresponding to the unfolding $\Psi(G, i)$ as follows: if a node j appears in the (ℓ)-th level of $\Psi(G, i)$, then j is allowed to visit only a subset of its neighbours. Specifically, each neighbour of j is visited with probability equal to $f(\ell)$ (or, equivalently, it is discarded with probability equal to $1 - f(\ell)$). As a consequence, the number of offsprings of j is a binomial random variable $\mathcal{B}(d_j, f(\ell))$ with d_j trials and probability of success equal to $f(\ell)$.

We are now ready to define the *Stochastic Potential Gain* (SPG), our new centrality index, which is derived from Definition 5 and generalises the PG [35].

Definition 6 *(Stochastic Potential Gain)*. Let $G = \langle N, E \rangle$ be an undirected connected graph and let $f(\cdot)$ be a discount function. For an arbitrary node $i \in N$ for its corresponding unfolding $\Psi(G, i)$, the SPG of i is defined as

$$SPG(i) = 1 + \sum_{k=1}^{+\infty} \sum_{j \in \mathcal{L}(G,i,k)} A_{jk} \tag{1}$$

Here $A_{jk} \sim \mathrm{Ber}(f(k))$ with $1 \le j \le n$ and $k \in \mathbb{N}^+$ are Bernoulli random variables defined as follows: $A_{jk} = 1$ if and only if the node $j \in N$ is successfully selected in the k-th level of the unfolding, and 0 otherwise.

Thus we can set the probabilities $P(A_{jk} = 1) = f(k)$ and $P(A_{jk} = 0) = 1 - f(k)$.

We are now ready to discuss the computation of SPG and its computational cost. Computing the SPG as in Equation (1) is clearly unfeasible, we have developed a Monte Carlo algorithm to efficiently and accurately approximate the SPG of nodes.

5. A Monte Carlo algorithm to compute the stochastic potential gain

5.1. Algorithm description

In this subsection we describe our a Monte Carlo algorithm to compute the SPG. We do so by defining a function, called MCPG, which operates on an undirected and connected graph G and makes use of a discount function $f(k)$ (see Algorithm 2). Function MCPG takes as input a node (say *cur_node*) and an integer (say *cur_level*); to compute the SPG node i we call MCPG with *cur_node* $= i$ and *cur_level* $= 0$.

Algorithm 2 The MCPG algorithm.

```
MCPG(cur_node, cur_level)
cur_pg ← 0
for all children ch_node of cur_node do                    ▷ B(d_cur_node, f(cur_level + 1))
    toss a coin with success probability f(cur_level + 1)  ▷ B(1, f(cur_level + 1))
    if success then
        cur_pg ← cur_pg + MCPG(ch_node, cur_level + 1)
    end if
end for
return 1 + cur_pg
```

MCPG initialises a variable *cur_pg* to zero then it updates *cur_pg* and returns 1+*cur_pg* as an estimation of the SPG. We say that the node *ch_node* is a *child* of *cur_node* if *ch_node* belongs to the neighbourhood of *cur_node*.

Next, MCPG tosses a coin with success probability $f(cur_level+1)$ for each child of *cur_node*: success leads to a recursive call with; with failure MCPG stops exploring the sub-tree of the unfolding rooted at *ch_node*. This coin toss procedure is equivalent to drawing a sample of the neighbours of *cur_node* whose size is distributed as $\mathcal{B}(d_{cur_node}, f(cur_level+1))$.

The main component of function MCPG is the **for** loop in lines 3-8; it never executes if: i) *cur_node* has no children or ii) the coin toss always comes up with a failure. Condition i) would be satisfied only by isolated nodes but here G is always assumed to be connected so it will never hold. For condition ii) we prove (see Theorem 2) that a suitable choice of the discount function $f(k)$ leads to the number of children of *cur_node* tending to zero as *cur_level* increases. In other words, with probability approaching one there exists some level in the unfolding of G beyond which the recursive calls will terminate.

We refer to condition ii) above as the *extinction condition*; we also say that the MCPG *becomes extinct* at node j and level k if j belongs to the k-th level of the unfolding and the coin tosses associated with j all end up with a failure.

Consider now the case of calling *MCPG(i, 0)* for, say, T times and let $P_\ell(i,0)$ be the output *MCPG(i, 0)* at the ℓ-th iteration, with $1 \le \ell \le T$. We define the *Empirical stochastic potential gain* (ESPG) $\hat{P}(i)$ as follows:

$$\hat{P}(i) = \frac{1}{T}\sum_{t=1}^{T} P_t(i,0). \tag{2}$$

To better understand how function MCPG computes $P_t(i,0)$, consider the execution of a generic call $P_t(i,k)$. If the extinction condition holds true, then MCPG terminates and returns $1+P_t(i,k)$. Vice versa, if the extinction condition is not satisfied, then MCPG will update $P_t(i,k)$ as follows:

$$P_t(i,k) \;=\; P_t(i,k+1) + \sum_{j\in\mathcal{L}(G,i,k)} P_t(j,k+1)\cdot a_{jk}^{(t)} \;=\; 1 + \sum_{j\in\mathcal{L}(G,i,k)} P_t(j,k+1)\cdot a_{jk}^{(t)}. \tag{3}$$

Here $a_{jk}^{(t)}$ is a binary variable which equals 1 if MCPG does not become extinct at node j and level k, and 0 otherwise. Coefficients $P_t(i,k)$ can be interpreted as *samples* drawn from a random variable $P(i,k)$ which represents the output of the *MCPG(i, k)* call. With the notation above, we have that the SPG(i) is equal to $P(i,0)$. Analogously, the $a_{jk}^{(t)}$ coefficient is the realisation of the random variable A_{jk} introduced in Definition 6. In the next subsections we investigate the convergence and correctness of Algorithm 2.

5.2. Convergence analysis

In this sub section we provide our results on the convergence of the Algorithm 2. Consider the function call *MCPG(i, 0)* and let Z_k be random variable representing the number of nodes visited during the 'exploration' of level k of $\Psi(G,i)$. We prove that if the discount function $f(k)$ satisfies some criteria (see Theorem 2) then the sequence of random variables Z_k will converge *in probability* to 0. Recall, e.g., from [21], that a sequence of random variables $Z_1 \dots Z_k$ converges in probability to $\mu \in \mathbb{R}$ if, for any arbitrarily small $\varepsilon > 0$, we have that $\lim_{k\to+\infty} Pr(|X_k - \mu| \ge \varepsilon) = 0$.

We have the following convergence result.

Theorem 2. *Consider the execution of MCPG on a network $G = \langle N, E\rangle$ with $d_M = \max_{h\in N} d_h$ as the largest degree. Also, let $f(k)$ be the discount function such that $f(k)\cdot d_M^k \to 0$ as $k \to +\infty$.*

For any node $i \in N$, let Z_k be the number of nodes explored by the MCPG function at level k of the unfolding $\Psi(G,i)$.

For any (arbitrarily small) $\varepsilon > 0$ the random variable Z_k will converge to zero in probability:

$$\lim_{k\to+\infty} Pr(|Z_k| \ge \varepsilon) = 0.$$

Proof. See Appendix A.1 □

5.3. Correctness analysis

In this section we prove that our Algorithm 2 is *correct*, that is, if we run it a sufficiently large number of times (parameter T in Equation (2)), then $\hat{P}(i)$ will converge to the expected value of *SPG(i)*. This intuition is formalised by the following theorem.

Theorem 3. *Consider (as we did above) the execution of MCPG on a network $G = \langle N, E\rangle$ with $d_M = \max_{h\in N} d_h$ as the largest degree. Also, let $f(k)$ be the discount function s. t. $f(k)\cdot d_M^k \to 0$ if $k \to +\infty$. For any node i, suppose we call function MCPG(i, 0) for T times and let $\hat{P}(i)$ be the empirical stochastic potential gain (see Equation (2)) obtained. Under these assumptions, as $T \to +\infty$ we that $\hat{P}(i)$ converges in probability to the SPG of i.*

Proof. See Appendix A.2. □

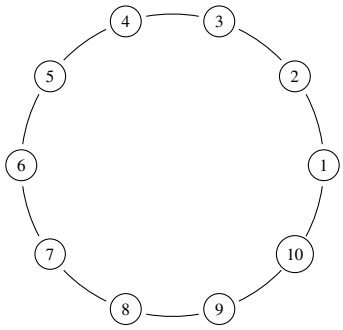


Fig. 1. A ring with $n = 10$ nodes.

5.4. Network topology impacts the convergence rate of MCPG

In the previous subsections we have provided some conditions regarding the discount function in the SPG which guarantee the convergence and correctness of Algorithm 2. Intuitively, the topology of the input graph will affect the rate of convergence of Algorithm 2 and, in many cases, we can choose a discount function that converges to zero slower than what is required by Theorem 2 to obtain convergence. To illustrate the point, consider a simple ring, sometimes denoted C_{10}, as in Fig. 1: n nodes, all of degree two. In this case, Theorem 2 suggests a function $f(k) = \alpha^k$ with $\alpha < \frac{1}{d_M}$. We found that Algorithm 2 converges quite quickly even for α values above 0.33.

This observation can be explained as follows: suppose Algorithm 2 is at the k-th generation and it is currently visiting the node i, with $1 \le i \le 10$. It can generate at most two children, which we know[3] will be $i-1$ and $i+1$. At each step, we will 'see' Algorithm 2 tossing a coin twice; the process will end when it gets tails (failure) both times. By the independence assumption, the probability of obtaining two consecutive failures is equal to $(1-f(k))\cdot(1-f(k)) = (1-f(k))^2$. Vice versa, the probability that the process continues is $1-(1-f(k))^2 = 2f(k) - f(k)^2$. If we set $f(k) = \alpha^k$, then we can rewrite the probability of survival as

$$2f(k) - f(k)^2 \;=\; 2\alpha^k - \alpha^{2k} \;=\; \alpha^k \cdot \left(2 - \alpha^k\right) \ge \alpha^k$$

since α^k is at most 1 when $k = 0$. By this mechanism, the branching process will reach generation ν only if it survives for the first $\nu - 1$ generations and then fails/becomes extinct at the ν-th generation. Therefore, the probability P_ν of becoming extinct at the ν-th generation is:

$$\begin{aligned} P_\nu &= \left(2f(0) - f^2(0)\right)\cdot\left(2f(1) - f^2(1)\right)\cdot\ldots\cdot\left(2f(\nu-1) - f^2(\nu-1)\right)\cdot(1-f(\nu))^2 = \\ &= (2-\alpha)\cdot\left(2f(1) - f^2(2)\right)\cdot\ldots\cdot\left(2f(\nu-2) - f^2(\nu-1)\right)\cdot(1-f(\nu))^2 \ge \\ &\ge \alpha^{\nu-2}\cdot(1-\alpha^\nu)^2 \simeq \alpha^{\frac{\nu(\nu-1)}{2}} \simeq \alpha^{\nu-2}. \end{aligned} \tag{4}$$

In the last step we assume that $(1-\alpha^\nu)^2 \simeq 1$ if ν is large enough. We have that P_ν converges to zero at least as fast as $a^{\nu-2}$ and, thus, the constraint that each node has degree equal to two implies the quick extinction of our process even if we choose values of α larger than those prescribed by Theorem 2. At the opposite end of the spectrum, for very dense networks, i.e. those that resemble a complete graph, the number of potential children at generation k grows as fast as n^k. Thus, for function $f(k) = a^k$ we need to ensure $\alpha < \frac{1}{d_M}$.

Real-world networks often display a highly skewed distribution with few nodes displaying a very large degree and the remaining ones having very few connections. It is also reasonable to expect that in these type of graphs we can relax the requirement $\alpha < \frac{1}{d_M}$ to obtain convergence.

Finally, observe that by setting $\alpha < \frac{1}{d_M}$ the SPG will converge to the PG; it is also true that it converges to the GPG we introduced in Section 3.1. We further note the well-known fact (see, e.g., [42]) that the maximum degree d_M of a network is also an upper bound to its spectral radius λ_1: $\lambda_1 \le d_M$, which concludes our analysis.

6. Experimental results

In this section we describe several experiments we carried out to validate our MCPG algorithm. We will begin by fixing some general research questions and then design experiments that will answer them.

6.1. Research questions and evaluation metrics

The first research question is about the amount of computation needed for MCPG to complete.

[3] Of course if $i = 10$ (resp. $i = 1$) we consider nodes 1 and 9 (resp. 10 and 2) as potential children.

Table 1
The main measures and features of the graphs used in our experimental evaluation.

Dataset	No. of nodes	No. of edges	Clustering Coeff.	Diameter
FB-Friends	63,731	817,035	0.148	15
GitHub	37,700	289,003	0.013	7

RQ$_1$ *How many nodes does the MCPG algorithm traverse before it terminates?*

This question is important vis-a-vis centrality indices such as, e.g., the Katz coefficient [43] or the subgraph centrality [16]) that need to process the entire input network to obtain the centrality of a specific node. In contrast, MCPG is invoked on a specific node and will traverse only a fraction of the nodes to complete its execution. For the sake of ensuring scalability, it is of practical interest to quantify the (average) number of nodes that MCPG will need to visit before termination.

Suppose we invoke it node i and monitor its execution. Let $v(i)$ be the number of nodes it actually explores before terminating. Since a node can be visited several times, each one could contribute multiple times to the calculation of $v(i)$. So $v(i)$ is not bounded by $n = |N|$ in general. To better visualise the efficiency of MCPG and make it comparable to alternative centrality algorithms we normalise $v(i)$ by n, i.e., $s(i) = \frac{v(i)}{n}$, so that the computational cost will be independent of graph size. In the light of the observations above, we have that, for our choice of discount functions, $s(i)$ ranges between 0 and 1; the smaller $s(i)$, the better because MCPG will have to visit a fraction of nodes before terminating. To obtain a robust estimation of $s(i)$ for each input we ran MCPG T times then took the average over T runs. Equation (5) defines the *Average Saving (or sparsity)* (AS) measure, denoted $\overline{s}(i)$:

$$\overline{s}(i) = \frac{1}{T}\sum_{\ell=1}^{T}\frac{v(i)}{n} \tag{5}$$

Clearly $\overline{s}(i)$ ranges between 0 and 1 and the smaller it is the larger the space/time saving wrt. Katz and other centralities. We can now formulate our next research question.

RQ$_2$ *How good is the approximation of potential gain obtained by MCPG?*

Theorem 3 guarantees that, as $k \to +\infty$, the product $d_M^k \cdot f(k)$ converges to 0, hence the Empirical Stochastic Potential Gain (ESPG) converges to the SPG. We would like to check how network topology affects the convergence of MCPG (see Section 5.4). To this end, we wish to determine if the SPG scores returned by MCPG are "close enough" to the exact PG even if we relax the hypotheses of Theorem 3 and, specifically, if we consider a discount function of the type $f(k) = \alpha^k$ with $\alpha > \frac{1}{d_M}$.

We have already observed (see Section 3.1) how the spectral radius λ_1 (or more precisely $\frac{1}{\lambda_1}$) of the relative adjacency matrix has a fundamental role in the calculation of the PG and the GPG. In fact, if we choose α less than λ_1^{-1} then the PG and GPG of each node will be equivalent (remember also that the Potential gain of a node was proved to be equivalent to the node degree multiplied by its Katz centrality).

We are interested in studying what happens when α takes on values smaller/larger than λ_1^{-1} and, for specific values of α, we wish to determine how much the ESPG deviates from the potential gain. So, let **PG** be the output of Algorithm 1 and let $\hat{\mathbf{p}}_T$ be the ESPG scores vector, obtained from Equation (2) after T repetitions. We will use the standard *cosine similarity* measure to determine how close the two output vectors are:

$$\cos(\mathbf{PG},\ \hat{\mathbf{p}}_T) = \frac{\mathbf{PG}\cdot\hat{\mathbf{p}}_T}{||\mathbf{PG}||\cdot||\hat{\mathbf{p}}_T||}. \tag{6}$$

Of course, the higher the cosine similarity the more similar the two measures are.

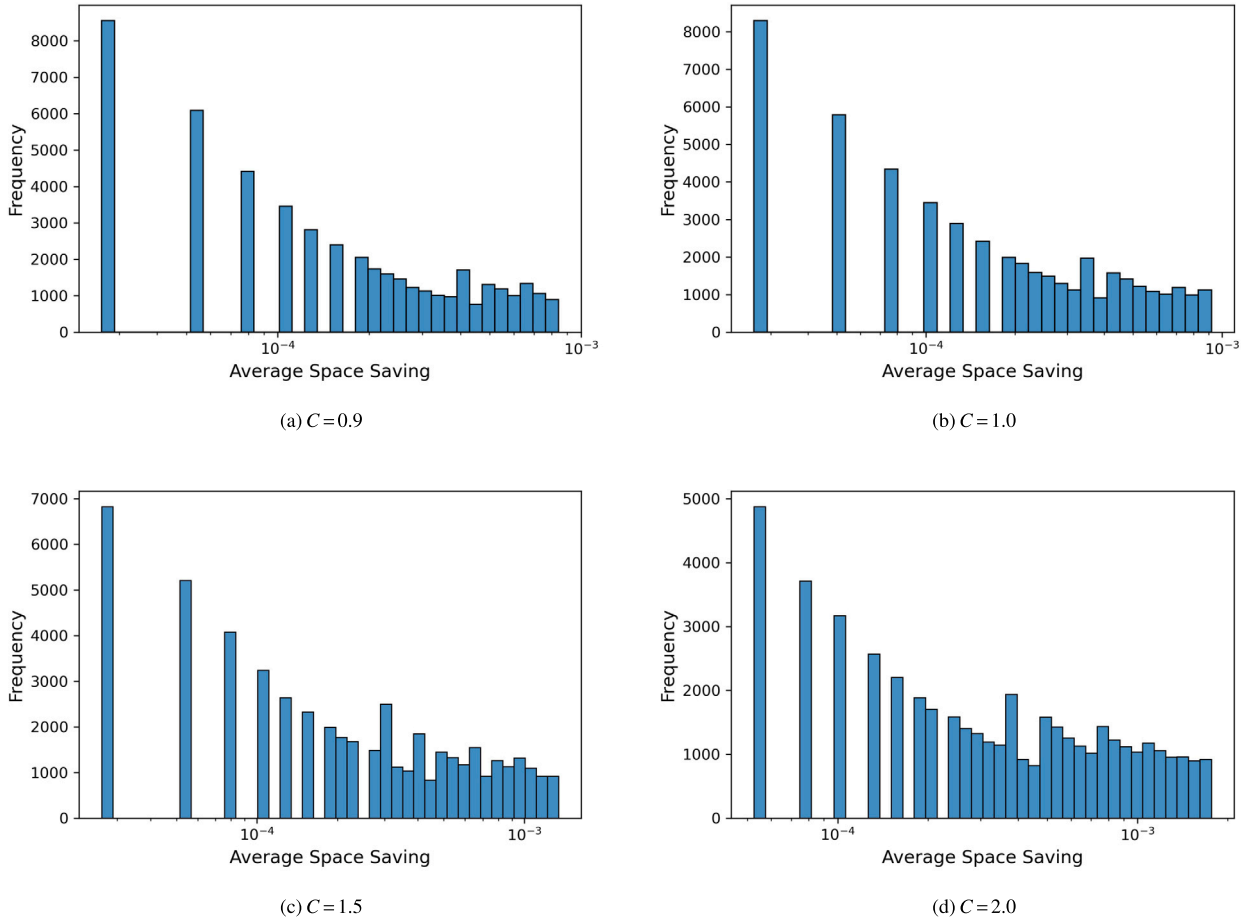
6.2. Dataset description

Our experimental phase focused on two large and publicly-available networks: Facebook Friendship and GitHub. Their main features are summarised in Table 1.

FB-Friends. This classic dataset contains data on Facebook friendship collected by Viswanah et al. [48]. A node represents a user and an edge represents a friendship between two users.

GitHub. This dataset was collected from the public GitHub API in June 2019 by [45] and it describes a social network of GitHub developers. Nodes represent GitHub users who have starred at least 10 repositories; edges identify mutual 'follow' relationships between users.

We observe that node degree distribution is right-skewed for all datasets considered in our experimental trials. In fact, differences in observed distributions are likely to derive from the mechanisms regulating the formation and growth of each social network. For instance, **GitHub** collects mutual likes between members who are quite actively contributing. So the average node degree is higher than in other social networks. We inspected the dataset and found that only about one thousand nodes have a degree ranging between 50 and 110. On the other hand, node degree distribution in **FB-Friends** is even more skewed with a large number of nodes displaying a degree less than ten while only a tiny fraction of nodes have degree of 120 or more.



(a) $C = 0.9$ (b) $C = 1.0$ (c) $C = 1.5$ (d) $C = 2.0$

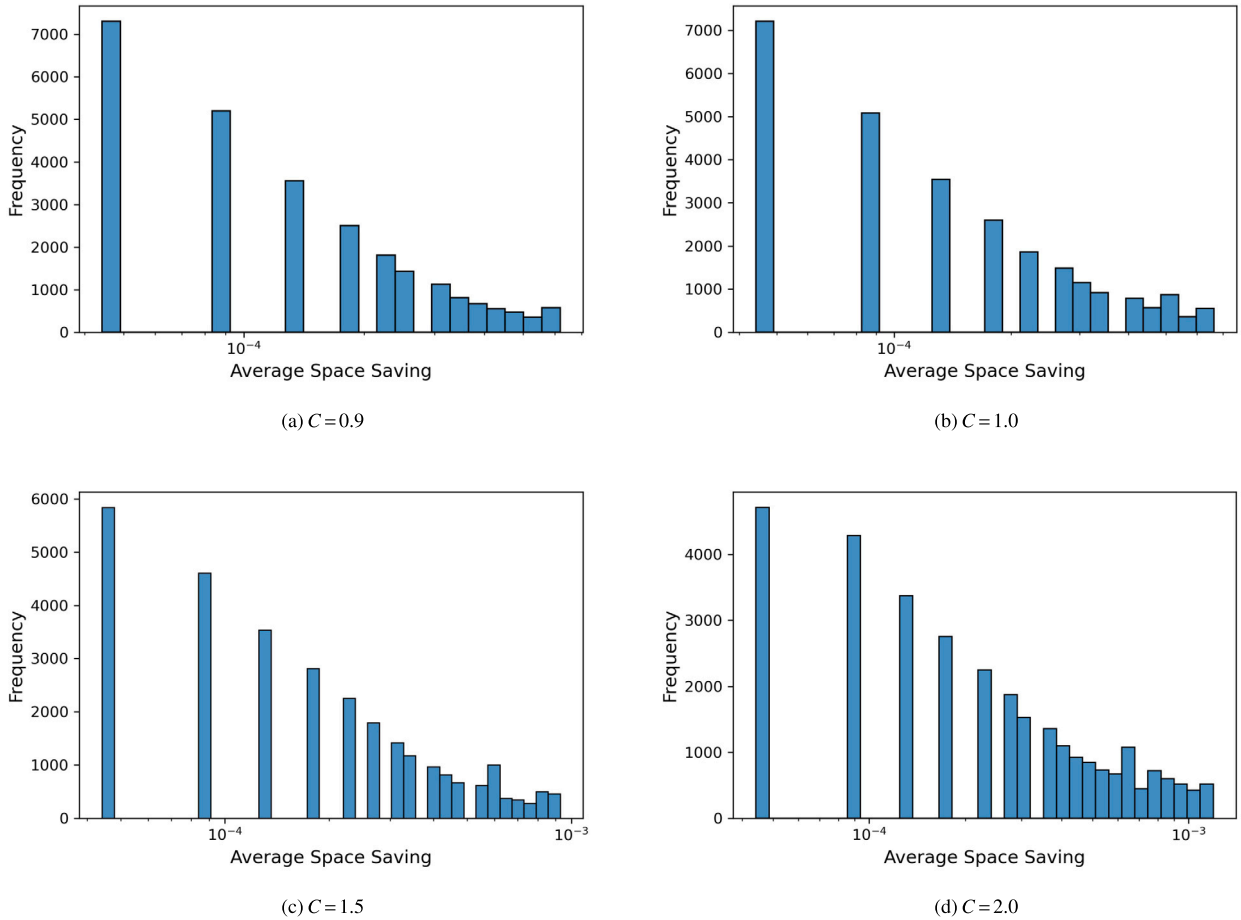
Fig. 2. Distribution of the Average Space Saving for the **FB- Friends** dataset with $C = 0.9$ (top left), $C = 1.0$ (top right), $C = 1.5$ (bottom left) and $C = 2.0$ (bottom right).

6.3. How many nodes does MCPG traverse before it terminates?

In our experimental trials we focused on a discount function of the type $f(k) = \left(\frac{C}{\lambda_1}\right)^k$, where λ_1^{-1} is again the spectral radius and C is a constant. We considered values of C less than one (namely $C = 0.9$) as well as bigger than or equal to one (namely $C = 1.0, 1.5, 2.0$). The distribution of the Average Saving value $\overline{s}(i)$ is reported in Figs. 2 and 3 with log-scale histograms.

From these figures the following observations can be made.

1. The constant parameter C affects the average savings, as we detail below. For instance, in the **FB-Friends** dataset with $C = 0.9$, the percentage of nodes with an $AS \leq 10^{-4}$ is 50.75% while the percentage of nodes with an $AS \leq 10^{-5}$ decreases to 17.51%. For $C = 2.0$, then 26.85% of nodes has an AS less than or equal to 10^{-4} is while the percentage of nodes with an AS less than 10^{-5} decreased to 11.13%.
   In the **GitHub** dataset with $C = 0.9$ we have that 62% of all nodes has an AS less than 10^{-4} while 25.28% of all nodes displays an AS less than 10^{-5}; if we set $C = 2.0$, then around 29.59% of nodes has an AS less than 10^{-4} but there are no nodes having an AS less than 10^{-5}. These results are explained as follows: the smaller C is, the smaller $\alpha = \frac{C}{\lambda_1}$ and, thus, the faster the discount function converges to zero as k increases. Vce versa for a smaller α the exploration of the unfolding of G terminates earlier and then a smaller number of node visits are needed.
2. Independently of the input dataset as well as the value of C, we report very small values of the AS. For instance if we set $C = 2.0$, then 11.13% (resp., 2.05%) of nodes in the **FB-Friends** (resp., **GitHub**) dataset display an AS larger than 10^{-3}. We can thus provide a *positive answer to* $\mathbf{RQ_1}$: the MCPG algorithms explore only a tiny portion of the input network, much smaller than other centrality indices (e.g. Katz') which need to visit multiple times all nodes of the input graph.
3. The distribution of AS scores in the **FB-Friends** dataset is more skewed than in the **GitHub** one. It follows that, in the **FB-Friends** dataset we generally obtain better AS scores than that observed for **GitHub**; however, the percentage of nodes in the **FB-Friends** dataset with AS bigger than 10^{-3} is roughly five times more than in the **GitHub** dataset. Such a behaviour is likely



(a) $C = 0.9$ (b) $C = 1.0$

(c) $C = 1.5$ (d) $C = 2.0$

Fig. 3. Distribution of the Average Space Saving for the **GitHub** dataset with $C = 0.9$ (top left), $C = 1.0$ (top right), $C = 1.5$ (bottom left) and $C = 2.0$ (bottom right).

Table 2
Cosine Similarity in the **FB-Friends** dataset as function of the number of trials.

T	$C = 0.9$	$C = 1.0$	$C = 1.5$	$C = 2.0$
2	0.9349	0.9275	0.9009	0.8735
6	0.9817	0.9780	0.9572	0.9343
11	0.9869	0.9835	0.9636	0.9414
16	0.9884	0.9849	0.9655	0.9435

to depend on the node degree distribution in each dataset. Specifically, in the **FB-Friends** dataset, most of the nodes has degree less than ten; a low-degree node can generate only few children, thus causing a quick extinction of the exploration process. In contrast, high degree nodes can generate a large number of children even if the discount function goes to zero rapidly. Thus, the exploration process in high degree nodes become extinct more slowly.

6.4. How good is the MCPG approximation of the potential gain?

We compute the cosine similarity between the ESPG and the PG using Equation (6). We ran a preliminary test in which we choose the discount function $f(k) = f^{\star}(k) = \left(\frac{1}{d_M+1}\right)^k$ with d_M the largest degree in the input graph. We obtained a cosine similarity close to 0.99 after five trials, in line with our theoretical results.

Next, let us considered the—perhaps more interesting—case where the discount function decreases slower than $f^{\star}(k)$ while, as in the previous test, we assumed $f(k) = \left(\frac{C}{\lambda_1}\right)^k$ with $C \in \{0.9, 1.0, 1.5, 2.0\}$. Recall that Theorem 3 cannot be applied and thus we lack any theoretical guarantee on the convergence of Algorithm 2. Tables 2 and 3 report our experimental results.

For this set of tests, our main findings are as follows.

Table 3
Cosine Similarity in the GitHub dataset as function of the number of trials.

T	$C = 0.9$	$C = 1.0$	$C = 1.5$	$C = 2.0$
2	0.8955	0.8730	0.7876	0.7237
6	0.9210	0.9024	0.8170	0.7550
11	0.9182	0.9045	0.8223	0.7575
16	0.9193	0.9049	0.8223	0.7562

1. If T is increased, we observe an increase in the cosine similarity. Our experiment leads us to hypothesise that the MCPG converges to the potential gain even if we opt for a discount function which converges to zero slower than function $\frac{1}{d_M^k}$ does; this opens the door to a possible generalisation of Theorem 3. Our results above derive from the fact that our branching process is guided by the topology of the input graph. This in fact is the reason why we called it *graph-driven branching process*: if $f(k) \simeq 0$ but a node has many children then our process might still survive extinction. Indeed, one of those children could have a large number of neighbours, and this would give rise a strengthening of the process rather than a weakening of it.
2. We obtain the highest similarity results at $C = 0.9$ and, in general, an increase in C causes a decrease in cosine similarity. Moreover, cosine similarity scores over **FB-Friends** dataset are higher than those recorded in the **GitHub** dataset. Such results depend on the speed with which the exploration process becomes extinguished.
   In fact, both the exact and approximate algorithms initialise all nodes to PG (resp. SPG) of one. If the branching process terminates almost immediately, then the PG and ESPG scores returned by Algorithms 1 and 2 will be very close to one for all nodes and, thus, the cosine similarity will also be close to 1. Such outcomes are somewhat unwelcome as having lots of nodes with score (exact or approximate) around 1 invalidate our power to effectively discriminate nodes.

7. Comparison with other centrality metrics

This section completes the presentation of the SPG centrality by an experimental comparative analysis of how the SPG apportion centrality (and the subsequent node ranking) vis-a-vis the centralities assigned by Degree and Eigenvector centrality, two of the main measures in literature. Betweenness centrality was not included in the current comparison due to its high computational complexity and is left from future work.

In general, there is no standard way to compare two centrality measures nor to assess their similarity/difference. After careful evaluation we adopted the method which was developed by Grando et al. [20] to evaluate centralities and the output of search engines; its description now follows. For each given network of size n we computed the Degree (d), Eigenvector (e) and SPG (s) centrality of its nodes. Thus we obtain three vectors, called **d**, **e** and **s** of size n with the respective centrality values. Next, we sort each array by value to create a node ranking for each method. Finally, the difference between centralities will be assessed in terms of the correlation between the obtained rankings. In principle, centrality calculation provides a complete ranking for all the nodes. In practice, however, the main attention is on the top-ranked nodes, i.e., on whether centrality measures characterise the same few nodes as the most important in the network. The final step of Grando et al.'s method is to compare the top-k nodes according to the centrality produced by each of the three methods, which we carried out for values of $k = 10$, 20, 50 and 100.

Unfortunately, the standard measures of ranking correlation, especially *Kendall's tau (τ) coefficient* [1] are not adequate for this type of comparison. An intuitive definition of Kendall's τ is as follows: given two rankings and an arbitrary pair of objects which co-occur in them, the τ will measure the probability that they appear in the same order in both rankings. Moreover, the aggregated τ of two rankings will always range between 0 (misalignment) and 1 (perfect alignment). The use of τ to compare centrality rankings is not fully convincing since we would like a larger penalisation (i.e., less correlation) for ranking disagreement on the top item than for disagreement on, say, the 10-th or 20-th element of the ranking. To capture this important aspect we adopted a recent measure of ranking correlation called Rank-Biased Overlap (RBO) [49].

The RBO coefficient is widely used to compare rankings generated by search engines. It is based on a simple user-centred model where an individual compares two rankings starting from the top elements to the lower-ranked ones. As with Web search engines, the ranking vectors might be very long but sooner or later the user will become distracted and, with high probability, truncate the comparison at some k relative to the top ranked element. Thus, the RBO is defined as the expected average overlap that the user has observed in the two lists. As with tau, RBO ranges between 0 and 1 (perfect alignment of the top elements).

We have pairwise compared the rankings induced by the centrality measures by computing their RBO for the top-k ordered vectors, for $k = 10$, 20, 50 and 100. The results obtained are shown in Tables 4 and 5. The main findings of our experimental analysis can be summarised and itemised as follows.

1. The correlation between the three centrality measures under consideration is generally high and often perfect (RBO = 1). The topology of the input graph affects the obtained results, in particular for the **FB-Friends** dataset we get significantly lower RBO scores than for the **GitHub** dataset. That is likely to emanate from the underlying mechanisms that governs the formation and growth of these graphs. In fact, the **GitHub** dataset maps connections between developers working on related software projects. Hence, the most influential nodes in **GitHub** will coincide, by construction, with users with the highest reputation (or popularity) in the community. For the **GitHub** dataset we observe almost-perfect RBO scores, in particular, centrality measures agree on which the most important nodes are. On the other hand, in the **FB-Friends** dataset edges simply identify friendship; so a very

Table 4
Pairwise RBO correlation between centrality metrics on the **FB-Friends** dataset.

Comparing	k = 10	k = 20	k = 50	k = 100
Deg-Eig	0.493	0.546	0.637	0.688
Deg-SPG	0.382	0.467	0.545	0.588
Eig-SPG	0.808	0.829	0.843	0.829

Table 5
Pairwise RBO correlation between centrality metrics on the **GitHub** dataset.

Comparing	k = 10	k = 20	k = 50	k = 100
Deg-Eig	1.000	1.000	1.000	1.000
Deg-SPG	0.946	0.926	0.928	0.919
Eig-SPG	0.946	0.926	0.928	0.919

active user who accumulates a large number of connections does not necessarily achieve a leadership role. Perhaps as a result of that, on the **FB-Friends** dataset we found lower levels of RBO correlation.

2. As k increases, the RBO scores increase too; this observation has a straightforward explanation. For small values of k two ranks might in fact contain different elements, i.e., some nodes will be in one ranking but not in the other. Such nodes do not contribute to the RBO calculation, which remains low. On the other hand, for higher values of k we expect the make-up of the rankings to be largely the same, that is, nodes might 'fall down,' i.e., have a smaller ranking but not entirely off it. So almost all nodes contribute to the RBO which explains why for higher values of k we find higher RBO scores.

7.1. Centrality as an approximation for the k-nodes deletion problem

To complete our comparison of the SPG to other commonly used centrality metrics, we have designed and executed a new experiment that goes beyond the correlation analysis described in the previous section. We consider the following computational problem, which is known to be NP-hard [46].

Problem: k-node deletion
Instance: an undirected graph $G = \langle N, E \rangle$ and an integer k.
Solution: a set of k nodes to be removed from G.
Measure: decrease of the spectral radius λ_1.

The state-of-the-art solution for k-node Deletion is the *NetShield* algorithm [5]. It provides and approximate, yet accurate, solution with a worst-case time complexity in $O(|N|k^2 + |E|)$. Intuitively, NetShield will remove nodes which are well-connected.

We have explored the question of whether centrality measures are good predictors of which nodes will be removed by *NetShield*. This can be formulated as asking whether the k removed nodes coincide, or at least overlap, with the top-k nodes ranked by centrality? The practical implication is that centrality measures could be used as a pre-processing step for NetShield, i.e., they would select the nodes whose removal is most likely to cause a decrease of the spectral radius of the resulting graph after the removal of the selected nodes.

We ran an experimental comparison of the three centralities on the basis of their ability to predict which nodes NetShield would remove. For the same two datasets we have used above, and for all values of k between 1 and 20, we computed the top-k ranks for SPG, Degree and Eigenvalue centrality. Next, we compared each ranking with the output of NetShield for the same input graph, called $NS(k)$. Finally, for each top-k centrality, here denoted $\mathcal{T}(k)$, we computed their Jaccard similarity index:

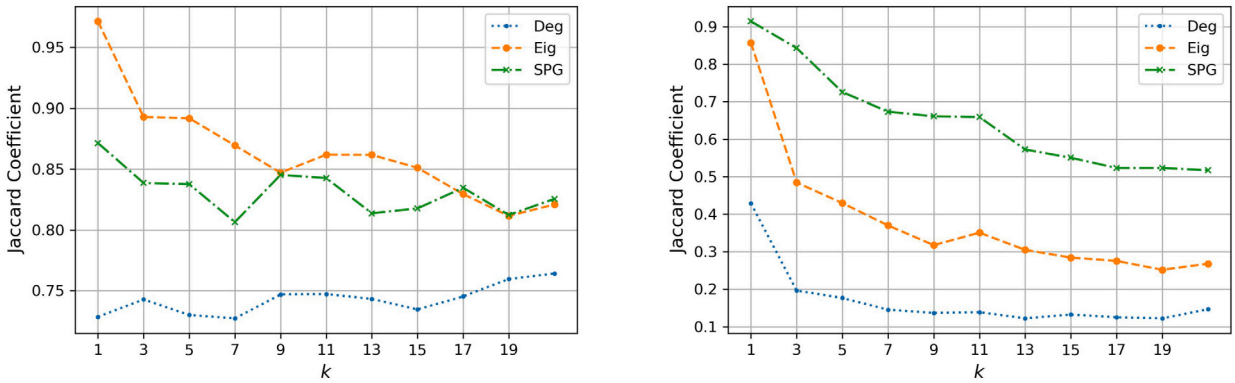
$$J(k) = \frac{|NS(k) \cap \mathcal{T}(k)|}{|NS(k) \cup \mathcal{T}(k)|} \tag{7}$$

The $J(k)$ similarity index will range between 0 (no similarity) and 1 (identity). The results of the comparison are plotted in Fig. 4.

We make several observations. First and foremost, and perhaps counter-intuitively, Degree centrality always gives the worst results. Next, we see that the parameter k greatly affects the values of $J(k)$. Indeed, as k increases, the complexity of the underlying problem increases and a heuristics prediction based on centrality becomes imprecise. Second, on the **GitHub** dataset the SPG achieves significantly better performances than Degree and Eigenvector centralities. On the other hand, on **FB-Friends** and for lower values of k Eigenvector centrality performs far better than SPG; for higher values ($k > 7$) the difference vanishes and both become dissimilar from NetShield.

7.2. A qualitative assessment of SPG

In this section we discuss and compare the SPG to what we see today as the state of the art in centrality metrics. We identify three major advantages and two potential disadvantages that are associated with the SPG.



(a) **FB-Friends** (b) **GitHub**

Fig. 4. Jaccard Coefficient between the K nodes returned by NetShield and the output of Degree, Eigenvector and SPG Centrality on the **FB-Friends** dataset (left) and the **GitHub** dataset (right).

+ From a conceptual standpoint, the SPG is an entirely novel centrality measure as it combines *structural information* (i.e., network topology) with *dynamic processes* (encoded through the discount function $f(k)$). Unlike traditional centrality metrics such as Degree or Eigenvector, the SPG simulates a random exploration process; yet unlike dynamic processes (such as standard branching processes), the SPG imposes additional constraints arising from the topology of the network. As observed earlier, if a node has a branching process stemming from it is likely to extinguish itself. On the other hand, a process stemming from a large-degree node has good chances to survive even for relatively high values of k.
+ SPG centralities are computed through randomised simulations; consequently, they are best interpreted as probability distributions (via the frequencies for each observed SPG value) rather than a scalar, which is the case with standard centrality measures. This might be seen as indirectness but in fact is *nuance* with good potential for application.
+ As seen in the previous section, often questions about centrality are restricted to a subset of "candidate" nodes and not to whole node set N. SPG is perfectly suited to accommodate these queries: we only need to call the MCPG algorithm on the required nodes. Unless specific algorithms are developed to, e.g., leverage the sparsity of the adjacency matrix, computing traditional centralities (except the hyper-local Degree centrality) relies on matrix operations which need to compute all centralities at once, even when only a small minority of nodes are of interest.

Let us now discuss the potential limitations of SPG.

- The branching process model we used here lacks a memory mechanism so successive runs (to compute the centrality for two or more nodes) will require re-exploring regions of the network. This inefficiency can be addressed with further algorithmic solutions, e.g. harnessing parallelism or dynamic programming solutions.
- The discriminating power of SPG centrality (and its efficient computation) relies on the choice of the discount function $f(k)$. In fact, by determining the number of nodes that can be explored at level k, $f(k)$ has a strong impact on the computational costs. Moreover, the choice of $f(k)$ is context-dependent: only for some network topologies a rapidly-declining $f(k)$ is adequate; sometimes a slower-decreasing function will be a better choice.

7.3. Replacing random walks with random paths?

Let us now discuss an algorithmic method that could provide an alternative to the random walks seen so far. We have seen how the MCPG begins the computation of the SPG of node i by launching a random walk from i; next, the discount function $f(\cdot)$ penalises long walks and we the process will stop after some a few more steps away from i. This simple schema makes SPG computation scalable. However, direct neighbours of i are likely to be repeatedly visited thus increasing, somehow artificially, their SPG centrality. This side-effect to the repeated exploration of a node is known as the *localisation effect* and, e.g., Martin et al. [38] examined this issue in depth. Consider for instance Eigenvector centrality: its computation, essentially, involves finding the dominant eigenvector of the given adjacency matrix. Martin et al. observed how, depending on the underlying topology, most of the mass of the leading eigenvector tends to concentrate around a handful of nodes. Then, as an undesirable side effect, most of the remaining nodes would have a centrality very close or equal to zero. When this happens, Eigenvector centrality looses its discriminating power and cannot help in selecting the most relevant nodes. The localisation effect is a direct consequence of the mathematical formulation of Eigenvector centrality since centrality of a node is defined as the sum of its neighbours, *hub* nodes 'give' centrality, and 'receive it back' from their neighbours.

We analysed the distribution of the SPG (and Eigenvector) centrality in our benchmark networks and found evidence of the localisation effect. A possible counter-measure to localisation could be to prevent random walks to "return" to previously-visited nodes. This strategy is realised by *self-avoiding random walks (SAWs)* by, e.g., [41,24]. A simple (but not very efficient) implementation

of SAWs consists in generating random walks and then stopping the process every time the walker "returns" to an already-visited node. Alternative definitions of SAWs have been considered in the literature. For instance, [41] defines SAWs as walkers who choose the next node to visit uniformly at random among the unvisited neighbours of the current node. If no unvisited neighbour remains, there is a restart, i.e., the next node is chosen uniformly at random among *all* neighbours. In the physics literature [2] *myopic SAWs* have been introduced, that is random processes which select among all neighbours but with probability proportional to a negative exponential of the number of times the neighbour has been visited already.

In our previous research work [11] we deployed SAWs to Community detection in large networks: we ran SAW walkers to re-weight and rank "local" edges, then we used those edge weights to detect communities. SAWs have also been extensively used to search for resources in peer-to-peer networks [6,9]. Hence a potential research avenue for improving the performance of MCPG is to base it on SAWs instead of purely-random walks or, equivalently, by replacing random walks with randomly-generated paths. Unfortunately, SAWs are not Markovian random processes, since their choice of next node intrinsically depends on past choices/visits. So even though SAWs are only slightly less randomised than fully-random walkers their conceptualisation and the analysis of their properties is much harder, see, e.g., [41] for a discussion.

8. Conclusions and future work

With the Stochastic Potential Gain we have introduced a new type of network centrality that is essentially different from traditional methods based on Linear Algebra. Our approach differs from the current literature on centrality indices in that we combine structural information describing the topology of the network with a novel branching process (called *graph-driven branching process*) exploration of the network.

Further, we defined a Monte Carlo approximation algorithm for SPG, called MCPG, for which we give formal proof of its properties: convergence, space-efficiency and a bound on the error. The expected value of the SPG of a node is equal to its Potential Gain centrality value, an index which we had previously introduced to generalise centrality indices such as Katz's and Subgraph.

Experiments on two real-world and large graphs indicate that the MCPG algorithm only requires the transversal of a small fraction of the nodes in a graph to accurately approximate the PG of a node. These results led us to claim that SPG is a centrality measure apt to scale up to Web and Online social network analysis.

One natural way to extend this research is to consider distributions other than the binomial distribution for defining the graph-driven branching process. For example, we may consider the Poisson distribution [26] or the negative binomial distribution [27].

A further research avenue consists of investigating how topological parameters of the input network (e.g., its edge density) affect the expected length of the random walks exploring it. So, an ambitious research goal would be to consider discount functions of the type $f(k) = \alpha^k$ with $\alpha \in (0, 1)$ and determine, for a specific value of edge density, the largest value of α which guarantees that the process will become extinct.

CRediT authorship contribution statement

Authors declare equal contribution.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Appendix A. Proofs

A.1. Proof of Theorem 2

To prove Theorem 2, we will need a well-known result in probability theory known as *Chebycheff's Inequality* [21]. It holds for any random variable X (with finite mean μ and variance σ^2) and for any $\varepsilon > 0$ that the probability that X deviates from μ more than ε is bounded above by $\frac{\sigma^2}{\varepsilon^2}$, i.e., $Pr(|X - \mu| \geq \varepsilon) \leq \frac{\sigma^2}{\varepsilon^2}$.
Another background result which will be needed to prove Theorem 2 is the Lemma below.

Lemma 1. *Let X_k be a sequence of random variables with mean μ_k and variance σ_k^2 such that the limits $\lim_{k\to+\infty} \mathbb{E}[X_k] = \mu$ and $\lim_{k\to+\infty} \mathrm{Var}[X_k] = 0$ hold.*
Then the sequence X_k will converge in probability to μ: $X_k \xrightarrow{P} \mu$.

Proof. We start with the Chebyshev's inequality on X_k and we will show that, for an arbitrary error threshold $\varepsilon > 0$:

$$Pr(|X_k - \mu| \geq \varepsilon) \leq \frac{\mathbb{E}[(X_k - \mu)^2]}{\varepsilon^2}. \tag{A.1}$$

In general, $\mathbb{E}[X_k] = \mu_k \neq \mu$ and, thus, $\mathbb{E}[(X_k - \mu)^2]$ is not necessarily equal to the variance of X_k.
Now we exploit the linearity of expectation (i.e., $\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$) to expand the term $\mathbb{E}[(X_k - \mu)^2]$ as follows:

$$\begin{aligned}
\mathbb{E}[(X_k - \mu)^2] &= \mathbb{E}[(X_k - \mu_k + \mu_k - \mu)^2] = \\
&= \mathbb{E}[((X_k - \mu_k) + (\mu_k - \mu))^2] = \\
&= \mathbb{E}[(X_k - \mu_k)^2 + 2(X_k - \mu_k)(\mu_k - \mu) + (\mu_k - \mu)^2] = \\
&= \mathbb{E}[(X_k - \mu_k)^2] + 2\mathbb{E}[(X_k - \mu_k)(\mu_k - \mu)] + \mathbb{E}[(\mu_k - \mu)^2] = \\
&= \mathbb{E}[(X_k - \mu_k)^2] + 2(\mu_k - \mu)\mathbb{E}[X_k - \mu_k] + (\mu_k - \mu)^2.
\end{aligned} \tag{A.2}$$

Observe that, by definition $\mathrm{Var}[X_k] = \sigma_k^2 = \mathbb{E}[(X_k - \mu_k)^2]$.
Also, we apply linearity again to obtain $\mathbb{E}[X_k - \mu_k] = \mathbb{E}[X_k] - \mathbb{E}[\mu_k] = \mathbb{E}[X_k] - \mu_k = \mu_k - \mu_k = 0$.
So the middle term in Equation (A.2) vanishes and we can rewrite Equation (A.1) as follows:

$$P(|X_k - \mu| \geq \varepsilon) \leq \frac{\mathbb{E}[(X_k - \mu)^2]}{\varepsilon^2} = \frac{1}{\varepsilon^2}\left[\mathbb{E}[(X_k - \mu_k)^2] + (\mu_k - \mu)^2\right] = \frac{\sigma_k^2}{\varepsilon^2} + \frac{1}{\varepsilon^2}(\mu_k - \mu)^2. \tag{A.3}$$

We take the limit, as $k \to +\infty$ of the left and right hand side of Equation (A.3).
Recall that we assumed that $\lim_{k\to+\infty} \sigma_k^2 = \lim_{k\to+\infty} \mathrm{Var}(X_k) = 0$ so the first term also goes to 0.
We also assumed that $\mu_k \to \mu$ as k goes to infinity and, thus, by the continuity property of the expectation, we get $\lim_{k\to+\infty} \mathbb{E}[(\mu_k - \mu)^2] = 0$.
Which proves that $\lim_{k\to+\infty} P(|X_k - \mu| \geq \varepsilon) = 0$, i.e., X_k converges in probability to μ. □

We are now ready to prove Theorem 2 (the termination of MCPG); for convenience we restate it below.

Theorem. *Consider a graph $G = \langle N, E\rangle$ with $d_M = \max_{h\in N} d_h$ as the largest degree and let $f(\cdot)$ be the discount function such that $f(k)\cdot d_M^k \to 0$ for $k \to +\infty$.*
For an arbitrary node $i \in N$, we define two sequences of random variables.
First, let $X_0 = \{i\}$ and $X_k \subseteq \mathcal{L}(G, i, k)$ be the multiset *of nodes visited by the* MCPG *function (see Algorithm 2) at the k-th level of the unfolding $\Psi(G, i)$.*
Second, let $Z_0 = |X_0| = 1$ and $Z_k = |X_k|$ at all levels k of $\Psi(G, i)$.
For any (arbitrarily small) $\varepsilon > 0$ we have that Z_k converges to zero in probability: $\lim_{k\to+\infty} Pr(|Z_k| \geq \varepsilon) = 0$.

Proof. Consider X_1, that is the multi-set of nodes reached by the MCPG function at the first level of the unfolding. From Algorithm 2, line 3, we have that $Z_1 \sim \mathcal{B}(d_i, f(1))$.
We introduce an auxiliary Binomial random variable, called $W_1 \sim \mathcal{B}(d_M, f(1))$, where d_M again is the largest degree in G.
Clearly, $\mathbb{E}[W_1] \geq \mathbb{E}[Z_1]$.
That is, the expected number of successes we can obtain with d_M trials (i.e., starting from a maximum-degree node) will be greater or equal to the number of successes we would obtain in d_i trials (i is chosen arbitrarily): the success probability is $f(1)$ in all cases.
We now construct a sequence of Binomial random variables W_k such that each $\mathbb{E}[W_k] \geq \mathbb{E}[Z_k]$ at any level k of the unfolding.
To see how this can be done, consider the multiset X_2 and let $Z_2 = |X_2|$.
Also, let j be an arbitrary node in X_1; its degree d_j and we observe that the number of nodes $H(j, 2)$ that j can explore is distributed as $H(j, 2) \sim \mathcal{B}(d_j, f(2))$.
Since $d_j \leq d_M$, we have $\mathbb{E}[H(j, 2)] \leq \mathbb{E}[\overline{W_2}]$, with $\overline{W_2} \sim \mathcal{B}(d_M, f(2))$.
As a result of this observation, we can get a bound on $\mathbb{E}[Z_2]$ as follows:

$$\mathbb{E}[Z_2] = \mathbb{E}[|X_2|] = \mathbb{E}\left[\sum_{j\in\mathcal{L}(G,i,1)} H(j, 2)\right] = \sum_{j\in\mathcal{L}(G,i,1)} \mathbb{E}[H(j, 2)] \leq \sum_{j\in\mathcal{L}(G,i,1)} \mathbb{E}[\overline{W_2}] = d_i \cdot \mathbb{E}[\overline{W_2}] \leq d_M \cdot \mathbb{E}[\overline{W_2}].$$

Let us define $W_2 = d_M \cdot \overline{W_2}$ and observe that W_2 is the sum of d_M Binomial random variables, each of them with parameters $n = d_M$ and $P = f(2)$.
From Theorem 1 we have that W_2 is a Binomial random variable with parameters $n = d_M \cdot d_M = d_M^2$ and $P = f(2)$.
We can generalise the procedure above for an arbitrary integer $k > 2$ for which we will study Z_k along with a binomial random variable $W_k \sim \mathcal{B}(d_M^k, f(k))$.
The expected value $\mathbb{E}[W_k]$ and the variance $\mathrm{Var}[W_k]$ of W_k are known: $\mathbb{E}[W_k] = n \cdot p = d_M^k \cdot f(k)$ and $\mathrm{Var}[W_k] = n \cdot p \cdot (1 - p) = d_M^k \cdot f(k) \cdot (1 - f(k))$.

If we assume that $d_M^k \cdot f(k) \to 0$ as $k \to +\infty$, then both $\mathbb{E}[W_k]$ and $\mathrm{Var}[W_k]$ will converge to zero.
But since for any (arbitrarily small) $\varepsilon > 0$ we have that $0 \le |Z_k| \le |W_k|$. Thus:

$$P(Z_k > \varepsilon) \;=\; P(|Z_k| > \varepsilon) \;\le\; P(|W_k| > \varepsilon).$$

At this point $\mathbb{E}[W_k]$ and $\mathrm{Var}[W_k]$ converge to zero as $k \to +\infty$ so we can apply Lemma 1 to obtain that Z_k converges in probability to 0. □

A.2. Proof of Theorem 3

We begin with the following preliminary result.

Lemma 2. *Let $G = \langle N,\ E\rangle$ be a graph and let $j \in N$ be any of its nodes.*
For any integer $k > 0$ let $P(j,\ k)$ be the random variable modelling the output of the MCPG *function called on parameters j and k.*
Let A_{jk} be the Bernoulli random variable of parameter $f(k)$ from Definition 6.
The expected value of $P(j,\ k)$ is given by:

$$\mathbb{E}[P(j,\ k)] = \mathbb{E}[P(j,\ k) \mid A_{jk} = 1] \cdot f(k).$$

Proof. From Definition 6 we have that A_{jk} equals 1 if and only if the extinction condition is false. We apply the well-known (see, e.g., [21]) Law of Total Expectations to obtain:

$$\mathbb{E}[P(j,\ k+1)] \;=\; \mathbb{E}[P(j,\ k+1) \mid A_{jk} = 1] \cdot P(A_{jk} = 1) \;+\; \mathbb{E}[P(j,\ k+1) \mid A_{jk} = 0] \cdot P(A_{jk} = 0).$$

Observe how whenever MCPG() extinguishes at node j and level k, then j won't contribute to the SPG, hence $\mathbb{E}[P(j,\ k+1) \mid A_{jk} = 0] \;=\; 0$.
Now, by the hypothesis, $P(A_{jk} = 1) = f(k)$, which enables us to conclude that

$$\mathbb{E}[P(j,\ k+1)] \;=\; \mathbb{E}[P(j,\ k+1) \mid A_{jk} = 1] \cdot f(k). \quad \square$$

Now we restate Theorem 3 then prove it.

Theorem. *Let $G = \langle N, E\rangle$ be an undirected and connected graph with $|N| = n$ nodes and $|E| = m$ edges.*
Let d_M be the largest degree of all nodes in G and let $f(k)$ be a discount function such that $d_M^k \cdot f(k) \to 0$ as $k \to +\infty$. For any node i, suppose to call the function MCPG$(i,\ 0)$ *for T times, being T a positive integer and let $\hat{P}(i)$ be the empirical stochastic potential gain (see Equation (2)).*
Then $\hat{P}(i)$ converges to the SPG of i when $T \to +\infty$.

Proof. We prove for any integer k we have that $\hat{P}(i,k) = \frac{1}{T}\sum_{\ell=1}^{T} P_\ell(i,k)$ converges to $\mathbb{E}[P(i,k)]$; as a consequence, because the SPG of a node i is equal to $\mathbb{E}[P(i,0)]$ the thesis follows if we take $k = 0$.
Our proof is by induction, we start from the observation that if k is large enough, then the extinction condition is satisfied because of Theorem 2.
Thus, for any ℓ, we have that $P_\ell(i,\ k)$ is always equal to one and $\hat{P}(i,\ k) = 1$. Our theorem is then trivially true because if none of the children of i is explored then the SPG of i is one, by Equation (1).
We need to prove the statement only for "small" values of k which guarantees that the extinction condition is not satisfied and the MCPG function continues building up and exploring $\Psi(G,i)$.
Suppose that the statement is true at the level $k+1$ that is $\hat{P}(i,\ k+1) \to \mathbb{E}[P(i,\ k+1)]$ if T is large enough.
We now prove that our assertion is true also at the level k.
Let $P_t(j,\ k)$ be the output of the function MCPG$(i,\ k)$ at iteration t. We have that

$$P_t(i,\ k) = 1 + \sum_{j \in \mathcal{L}(G,i,k+1)} P_t(j,\ k+1) \cdot a_{j,k+1}^{(t)}.$$

Consider $\hat{P}(i,\ k)$, i.e., the average of all $p_t(i,\ k)$ coefficients; we can rewrite it as follows:

$$\begin{aligned}\hat{P}(i,\ k) &= \frac{1}{T}\sum_{t=1}^{T} P_t(i,\ k) = \frac{1}{T}\sum_{t=1}^{T}\left(P_t(i,\ k+1) + \sum_{j \in \mathcal{L}(G,j,k+1)} P_t(j,\ k+1) \cdot a_{jk}^{(t)}\right) \\ &= \frac{1}{T}\sum_{t=1}^{T} P_t(i,\ k+1) + \underbrace{\frac{1}{T}\sum_{t=1}^{T}\sum_{j \in \mathcal{L}(G,j,k+1)} P_t(j,\ k+1) \cdot a_{jk}^{(t)}}_{\Delta}.\end{aligned} \tag{A.4}$$

By the inductive hypothesis we have

$$\frac{1}{T}\sum_{t=1}^{T} P_t(i,\ k+1) \xrightarrow{T\to+\infty} \mathbb{E}[P(i,\ k+1)].$$

Thus, we can swap the summations in Δ as follows:

$$\Delta = \frac{1}{T}\sum_{t=1}^{T}\sum_{j\in\mathcal{L}(G,j,k+1)} P_t(j,\ k+1)\cdot a_{jk}^{(t)} = \sum_{j\in\mathcal{L}(G,j,k+1)}\left(\frac{1}{T}\sum_{t=1}^{T} P_t(j,\ k+1)\cdot a_{jk}^{(t)}\right). \tag{A.5}$$

Let us define $\Gamma_{jk} = \sum_{t=1}^{T} a_{jk}^{(t)}$ and observe that Γ_{jk} is the number of times the MCPG function does not extinct at node j and level k. We can, in other words, consider the subset $T' \subseteq [1,T]$ of tests where node j provides a contribution equal to $P_t(j,\ k+1)$, while in the remaining tests ($t \notin T'$) j does not give any contribution. So, thanks to the notation introduced above, we rewrite Δ as follows:

$$\frac{1}{T}\sum_{t=1}^{T} P_t(j,\ k+1)\cdot a_{jk}^{(t)} = \frac{1}{T}\sum_{t\in T'} P_t(j,\ k+1) = \frac{\Gamma_{jk}}{T}\left(\frac{1}{\Gamma_{jk}}\sum_{t\in T'} P_t(j,\ k+1)\right).$$

Thus, if T is large enough, then $\frac{\Gamma_{jk}}{T}$ converges to the probability of success, which is in fact $f(k)$. By the Law of Large Numbers (see, e.g., [21]), $\frac{1}{\Gamma_{jk}}\sum_{t\in T'} P_t(j,k+1)$ will converge to its expected value $P(j,\ k+1)$ conditioned on $A_{jk}=1$, that is $\mathbb{E}[P(j,\ k+1) \mid A_{jk}=1]$. From Lemma 2 we have that $\mathbb{E}[P(j,\ k+1)] = \mathbb{E}[P(j,\ k+1) \mid A_{jk}=1]\cdot f(k)$, and, thus, if T is large enough, we have that Δ converges to $\sum_{j\in\mathcal{L}(G,j,k+1)} \mathbb{E}[P(j,\ k+1)]$.

At this point, by merging results above and by the linearity of the expectation we have that $\hat{P}(i,\ k)$ converges to

$$\hat{P}(i,\ k) \to \mathbb{E}[P(i,\ k+1)] + \sum_{j\in\mathcal{L}(G,i,k)} \mathbb{E}[P(j,\ k+1)] = \mathbb{E}\left[P(i,\ k+1) + \sum_{j\in\mathcal{L}(G,i,k)} P(j,\ k+1)\right] = \mathbb{E}[P(i,\ k)], \tag{A.6}$$

which ends the proof. □

References

[1] H. Abdi, The Kendall rank correlation coefficient, in: Encyclopedia of Measurement and Statistics, Sage, Thousand Oaks, CA, 2007, pp. 508–510.
[2] D. Amit, G. Parisi, L. Peliti, Asymptotic behavior of the "true" self-avoiding walk, Phys. Rev. B 27 (3) (1983) 1635.
[3] P. Aragón, V. Gómez, D. García, A. Kaltenbrunner, Generative models of online discussion threads: state of the art and research challenges, J. Internet Serv. Appl. 8 (1) (2017) 1–17.
[4] M. Benzi, C. Klymko, On the limiting behavior of parameter-dependent network centrality measures, SIAM J. Matrix Anal. Appl. 36 (2) (2015) 686–706.
[5] C. Chen, H. Tong, B. Prakash, C. Tsourakakis, T. Eliassi-Rad, C. Faloutsos, D. Chau, Node immunization on large graphs: theory and algorithms, IEEE Trans. Knowl. Data Eng. 28 (1) (2016) 113–126.
[6] V. Cholvi, Pascal P. Felber, E. Biersack, Efficient search in unstructured peer-to-peer networks, in: Proc. of the Sixteenth Annual ACM Symposium on Parallelism in Algorithms and Architectures, 2004, pp. 271–272.
[7] T. Cormen, C. Leiserson, R. Rivest, C. Stein, Introduction to Algorithms, MIT Press, 2022.
[8] D. Cvetkovic, P. Rowlinson, S. Simic, Eigenspaces of Graphs, Cambridge University Press, 1997.
[9] Luciano da Fontoura Costa, Gonzalo Travieso, Exploring complex networks through random walks, Phys. Rev. E 75 (1) (2007) 016102.
[10] K. Das, S. Samanta, M. Pal, Study on centrality measures in social networks: a survey, Soc. Netw. Anal. Min. 8 (2018) 11, https://doi.org/10.1007/s13278-018-0493-2.
[11] P. De Meo, E. Ferrara, G. Fiumara, A. Provetti, Enhancing community detection using a network weighting strategy, Inf. Sci. 222 (2013) 648–668, https://doi.org/10.1016/j.ins.2012.08.001.
[12] P. De Meo, M. Levene, F. Messina, A. Provetti, A general centrality framework-based on node navigability, IEEE Trans. Knowl. Data Eng. 32 (11) (2020) 2088–2100, https://doi.org/10.1109/TKDE.2019.2947035.
[13] P. De Meo, M. Levene, A. Provetti, Potential gain as a centrality measure, in: Proc. of the IEEE/WIC/ACM International Conference on Web Intelligence, WI 2019, Thessaloniki, Greece, ACM, 2019, pp. 418–422.
[14] S. Dutta, S. Mittal, D. Das, S. Chakrabarti, T. Chakraborty, Incomplete gamma integrals for deep cascade prediction using content, network, and exogenous signals, IEEE Trans. Knowl. Data Eng. 35 (6) (2022) 5991–6002, https://doi.org/10.1109/TKDE.2022.3174206.
[15] E. Estrada, N. Hatano, M. Benzi, The physics of communicability in complex networks, Phys. Rep. 514 (3) (2012) 89–119.
[16] E. Estrada, J.A. Rodriguez-Velazquez, Subgraph centrality in complex networks, Phys. Rev. E 71 (5) (2005) 056103.
[17] J. Gleeson, D. Cellai, J.P. Onnela, M. Porter, F. Reed-Tsochas, A simple generative model of collective online behavior, Proc. Natl. Acad. Sci. 111 (29) (2014) 10411–10415.
[18] J. Gleeson, T. Onaga, P. Fennell, J. Cotter, R. Burke, D. O'Sullivan, Branching process descriptions of information cascades on Twitter, J. Complex Netw. 8 (6) (2020) cnab002.
[19] D. Gleich, Pagerank beyond the web, SIAM Rev. 57 (3) (2015) 321–363.
[20] F. Grando, L. Granville, L. Lamb, Machine learning in network centrality measures: tutorial and outlook, ACM Comput. Surv. 51 (5) (2018) 1–32.
[21] G. Grimmett, D. Stirzaker, Probability and Random Processes, Oxford University Press, 2020.
[22] P. Haccou, P. Jagers, V.A. Vatutin, Branching Processes: Variation, Growth, and Extinction of Populations, Cambridge Studies in Adaptive Dynamics, vol. 5, Cambridge University Press, Cambridge, UK, 2005.
[23] A. Hawkes, Spectra of some self-exciting and mutually exciting point processes, Biometrika 58 (1) (1971) 83–90.
[24] C. Herrero, Self-avoiding walks on scale-free networks, Phys. Rev. E 71 (1) (2005) 016103.
[25] B. Huberman, P. Pirolli, J. Pitkow, R. Lukose, Strong regularities in World Wide Web surfing, Science 280 (5360) (1998) 95–97.
[26] N.L. Johnson, A.W. Kemp, S. Kotz, Chapter 4. Poisson distribution, in: Discrete Univariate Distributions, in: Wiley Series in Probability and Statistics, third edition, John Wiley & Sons, New York, NY, 2005, pp. 156–207.
[27] N.L. Johnson, A.W. Kemp, S. Kotz, Chapter 5. Negative binomial distribution, in: Discrete Univariate Distributions, in: Wiley Series in Probability and Statistics, third edition, John Wiley & Sons, New York, NY, 2005, pp. 208–250.
[28] L. Katz, A new status index derived from sociometric analysis, Psychometrika 18 (1) (1953) 39–43.

[29] R. Kobayashi, R. Lambiotte, Tideh: Time-dependent Hawkes process for predicting retweet dynamics, in: Proc. of the International Conference on Web and Social Media (ICWSM), Cologne, Germany, AAAI Press, 2016, pp. 191–200.
[30] C. Kraaikamp, H. Meester, A Modern Introduction to Probability and Statistics, 2005.
[31] R. Kumar, M. Mahdian, M. McGlohon, Dynamics of conversations, in: Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, ACM, 2010, pp. 553–562.
[32] H. Kwak, C. Lee, H. Park, S.B. Moon, What is Twitter, a social network or a news media?, in: Proc. of the International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, ACM, 2010, pp. 591–600.
[33] E. Leicht, P. Holme, M. Newman, Vertex similarity in networks, Phys. Rev. E 73 (2) (2006) 026120.
[34] K. Lerman, Information is not a virus, and other consequences of human cognitive limits, Future Internet 8 (2) (2016) 21.
[35] M. Levene, R. Wheeldon, Navigating the World-Wide-Web, in: M. Levene, A. Poulovassilis (Eds.), Web Dynamics, Springer-Verlag, Berlin, 2004, pp. 117–151.
[36] D. Liben-Nowell, J. Kleinberg, Tracing information flow on a global scale using internet chain-letter data, Proc. Natl. Acad. Sci. 105 (12) (2008) 4633–4638.
[37] L. Lü, D. Chen, X. Ren, Q. Zhang, Y. Zhang, T. Zhou, Vital nodes identification in complex networks, Phys. Rep. 650 (2016) 1–63.
[38] T. Martin, X. Zhang, M. Newman, Localization and centrality in networks, Phys. Rev. E 90 (5) (2014) 052808.
[39] J. McSweeney, Single-seed cascades on clustered networks, Netw. Sci. 9 (1) (2021) 59–72.
[40] A. Medvedev, J. Delvenne, R. Lambiotte, Modelling structure and predicting dynamics of discussion threads in online boards, J. Complex Netw. 7 (1) (2019) 67–82.
[41] V. Lopez Millan, V. Cholvi, L. López, A. Fernandez Anta, A model of self-avoiding random walks for searching complex networks, Networks 60 (2) (2012) 71–85.
[42] H. Minc, Nonnegative Matrices, John Wiley & Sons, 1988.
[43] M. Newman, Networks: an Introduction, Oxford University Press, 2010.
[44] R. Nishi, T. Takaguchi, K. Oka, T. Maehara, M. Toyoda, K. Kawarabayashi, N. Masuda, Reply trees in Twitter: data analysis and branching process models, Soc. Netw. Anal. Min. 6 (1) (2016) 1–13.
[45] B. Rozemberczki, C. Allen, R. Sarkar, Multi-scale attributed node embedding, arXiv preprint, arXiv:1909.13021, 2019.
[46] H. Tong, B. Prakash, T. Eliassi-Rad, M. Faloutsos, C. Faloutsos, Gelling, and melting, large graphs by edge manipulation, in: Proc. of the ACM International Conference on Information and Knowledge Management (CIKM 2012), Maui, ACM, 2012, pp. 245–254.
[47] S. Vigna, Spectral ranking, Netw. Sci. 4 (2016) 422–445.
[48] B. Viswanath, A. Mislove, M. Cha, K. Gummadi, On the evolution of user interaction in Facebook, in: Proc. of the 2nd ACM Workshop on Online Social Networks, Barcelona, Spain, 2009, pp. 37–42.
[49] W. Webber, A. Moffat, J. Zobel, A similarity measure for indefinite rankings, ACM Trans. Inf. Syst. 28 (4) (2010) 1–38.
[50] Q. Zhao, M. Erdogdu, H. He, A. Rajaraman, J. Leskovec, SEISMIC: a self-exciting point process model for predicting tweet popularity, in: Proc. of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Sydney, Australia, ACM, 2015, pp. 1513–1522.