Scalable Algorithms to Measure User Influence in Social Networks

Nouamane Arhachoui, Esteban Bautista, Maximilien Danisch, Anastasios Giovanidis, Lionel Tabourier

Scalable Algorithms to Measure User Influence in Social Networks

Nouamane Arhachoui, Esteban Bautista, Maximilien Danisch,
Anastasios Giovanidis, and Lionel Tabourier

Sorbonne Université, CNRS – LIP6, F-75005 Paris, France
{nouamane.arhachoui, esteban.bautista-ruiz, anastasios.giovanidis,
lionel.tabourier}@lip6.fr

Abstract. Measuring user influence in social networks is crucial for a variety of applications. While traditional centrality metrics evaluate structural graph importance, a more recent metric known as the ψ-score takes into account users' posting and re-posting activities to provide richer information. The ψ-score is a powerful tool that generalizes PageRank for non-homogeneous node activity. However, for large datasets with N users, it becomes computationally expensive, requiring solving N linear systems of N equations. To tackle this issue, we propose three new scalable algorithms that can quickly approximate the ψ-score. The Power-ψ and Push-ψ algorithms are based on a novel equation that shows it is sufficient to solve one system of equations of size N to calculate the ψ-score. These algorithms take advantage of the fact that the solution of such a system can be recursively and distributedly approximated. Consequently, the ψ-score, which summarizes the nodes' structural and behavioral information, can be computed as quickly as PageRank. The third proposed algorithm is Push-NF. Despite aiming to solve all N systems to extract additional information on the information dynamics, it still manages to converge to the accurate user ranking faster than the current state-of-the-art alternative. To validate the effectiveness of our proposed algorithms, we release them as an open-source Python library and test them on various real-world datasets.

Keywords: Social Networks · PsiScore · Ranking · Algorithms · Influence.

1 Introduction

1.1 Context

Measuring the importance of an individual in a network has been a focus of interest in Social Network Analysis for long. Indeed, a wealth of metrics have been developed to quantify this idea, including the well-known closeness centrality in the 1950s [1] or the betweenness centrality in the 1970s [2].
Nowadays, the ubiquitous nature of Online Social Platforms (OSPs) makes it all the more important to have relevant measurements available. For instance, it is essential for social scientists to identify the main leaders of opinion in a network in order to gain a better understanding of the dynamics that determine the roots of political polarization [3]. Also, it is crucial for companies to identify the so-called *influencers* that are believed to play a key role in the emergence of new trends [4] and thus are targeted to develop effective marketing strategies [5]. From a technical perspective, machine learning algorithms also rely on the identification of a reduced set of users with features present throughout the network, which is critical in reducing the number of parameters and avoiding the curse of dimensionality [6, 7].
In this context, many OSPs may be described using a comparable structure: users have access to a *wall* and a *news feed*, they can publish messages (*posts*) on their wall, while their news feed shows the messages posted by other specific users, known as their leaders (or followees). This archetype is well illustrated by Twitter, Facebook and Weibo among others. Users can also re-post messages from their news feed to their wall, which is the main mechanism allowing posts to spread around the network [8]. A crucial issue is to quantify the extent to which their posts and re-posts circulate throughout the network and are accessible to others, as this process can be understood as the measure of their influence.
To address the aforementioned challenge, a variety of centrality metrics have been proposed to rank network nodes according to their structural relevance [9]. But given the potentially massive size of these networks, it is crucial that these measures remain scalable. For instance, betweenness centrality measures to what extent a node falls on the shortest paths between others and thus its potential to control communication, however, the most efficient algorithms have a temporal complexity in $O(NM)$ [10] (where N is the number of nodes and M the number of edges) which makes it inappropriate for large OSPs. In addition to that, most centralities are unsuitable for assessing influence in OSPs, as they rely on a static description of the social network, while these are known to be dynamic platforms, where the activity of users is typically bursty. Nonetheless, these scores do not include user posting and sharing activity. The importance of this information is emphasized in [11], which demonstrates that people with the most followers (in-degree) may not necessarily create the most retweets or views. Therefore, influence is not solely based on the number of followers, which is mainly related to a user's popularity. In the same line of inquiry, the authors in [12] ask th question: *"Are social links valid indicators of real user interaction?"* and underline the difference between these two ideas by examining a significant user trace from Facebook. It is thus essential to get past simple structural measures.
A new metric, coined as ψ-score, has been recently introduced in [13] to overcome this limitation. The diffusion of information model presented in [13] takes into account users' behaviors by including activity features for each user. The ψ-score

results from this model and provides a more accurate ranking metric of user influence in OSPs by integrating both structural information related to the graph and user activity. For instance, if users post and re-post content at different frequencies, the model incorporates this variation in the users' activity, thus yielding a more meaningful ranking metric. Furthermore, it was shown in [13, Theorem 5] that the ψ-score is equivalent to PageRank when user activity is homogeneous, *i.e.*, when all users create the same number of posts within a given time interval and share at the same rate within that interval. As a consequence, the ψ-score can be seen as a generalization of the PageRank, which is designed for situations where we have access to the nodes' activity in the network.

1.2 Goal and contributions

A limitation of the ψ-score calculation as described in [13] is that it does not scale to large graphs. Indeed, it requires the resolution of N linear systems of N equations each, where N denotes the number of users in the network. In contrast, PageRank only requires finding the solution of a single system with N variables. This paper addresses this shortcoming by proposing new ψ-score calculation algorithms that solve this issue.

For that purpose, we reorganize these N systems into a single one whose solution can be used to easily derive the ψ-score for all users. Then, we propose two different algorithms to solve this new system:

- `Power-`ψ which is based on power-iteration rules in order to approximate the solution (see also the conference version of this work [14]). This process allows approximating the sum of a geometric series for vectors without computing any matrix inversion.
- `Push-`ψ, an algorithm inspired by the push-flow algorithm for PageRank [15, 16]. It uses an approximation vector and a residual representing the difference between the exact solution and the approximation. During each push iteration, the approximated score of a node is updated and its residual is pushed to the residual of its neighbors in order to be minimized.

In addition to this contribution, we also propose to use the push method to make a tool to analyze the influence mechanisms at a finer grain. For this purpose, we describe another push-based algorithm, named `Push-NF`, that aims to solve the original N linear systems presented in [13] instead of using the new system. While it does not allow to avoid the scalability issue aforementioned, by solving all N systems, `Push-NF` can not only extract the influence of each user on the entire network but also their influence on a specified individual. This feature provides valuable information for various applications, such as identifying opinion leaders or understanding information flow dynamics between specific users.

Finally, we provide an open-source software library[1] to facilitate the implementation of the ψ-score. This comprehensive library includes its initial baseline implementation as well as implementations of each of the improved algorithms proposed in this paper. By making this library available to the community, we aim to encourage the widespread use of the ψ-score in various fields and its adaptation to each developer's specific needs. In particular, the `psi-score` library can be used to compare the metric with PageRank, since they are equivalent when the network activity is homogeneous.

[1] https://github.com/NouamaneA/psi-score

1.3 Related Works

Centrality measures have been largely used in various fields of network science, including of course social network analysis [17–19], but also transportation networks for example to improve traffic monitoring [20], communication networks to design routing protocols [21] or to improve network control [22], or even in biology, where they allow for instance to point out critical proteins in metabolic networks [23]. Another essential application of centrality measures is information retrieval: with the expansion of the web, there is an ever-increasing need for algorithms that allow one to look for precise information in a very large database. Among those, the most famous example is certainly the PageRank [24, 25]. Although these metrics are commonly used to rank nodes of a network [9], they only consider the network structure. For instance, PageRank can be described in non-technical terms as a random walk on a graph with a probability of teleporting on a random node at each step, so it does not take into account aspects such as node activity.

A standard way to compute the PageRank in a graph is to rely on a power iteration method [25, Chapter 4]. Various existing algorithms, such as Push [15] or methods using Chebyshev polynomials [26], exploit the random walk interpretation of PageRank to speed up its calculation. The Push method [15] iterates through nodes by allowing each of them to update the approximation of its own PageRank value and its neighbors' values. The method proposed in [26] uses Chebyshev polynomials to approximate the PageRank vector more efficiently than the known power-method introduced with the metric [24].

In a short version of this work [14], we have investigated the algebraic limitations of the ψ-score and proposed a novel approach to have it as scalable as the PageRank is. However, we acknowledge that other strategies to accelerate graph eigenvalue-based measures have the potential to further improve the performance of our method. Moreover, they can also give us insight into the mechanisms of social influence from one individual to another. Therefore, in this study, we aim to build upon this work by considering the Push method and integrating it with our approach. Our goal is to create a novel algorithm that not only addresses the limitations of the ψ-score but also surpasses the existing methods in terms of speed.

1.4 Outline of the paper

In Section 2, we describe the principles underlying the definition of the ψ-score and to what extent it measures the spread of influence on a social platform, as well as the baseline algorithm to compute it. Section 3 introduces the `Power-`ψ algorithm, which improves upon the baseline by reducing the number of equations to solve. In section 4, we present the `Push-*` approach, which provides an alternative way to compute the ψ-score and also allows to compute the influence between nodes at the individual scale. Finally, in Section 5, we evaluate numerically the different algorithms under study in terms both of accuracy to approximate the result and in terms of time efficiency of the process.

2 Measuring the Influence of OSP Users

The authors of [13] proposed a model to describe the diffusion of posts on a generic social platform. Within this model, each user is not only a node of the social graph, but is also characterized by their own Wall and Newsfeed, as well as their own posting and re-posting activities. Using this spreading model, they compute in closed form the probabilities that a particular user's posts will appear on the Wall and Newsfeed of any other user. This allows them to derive a measure of influence in the network, that they name the ψ-score of a node.
In this section, we first give some basic elements of description of the OSP and the notations, summarized in Table 1, that will be used afterward. Then, we detail the spreading model proposed in [13] and describe how the ψ-score is defined in this context. Note however that this score is not restricted to the specifics of this model and can be defined on any network where nodes can be characterized with their activity rate.

Table 1. Notations used in the paper

Notation	Description
ψ_i	ψ-score of user i
$\boldsymbol{\psi}$	ψ-score column-vector
$p_i^{(n)}$	impressions of i on the Newsfeed of n
$q_i^{(n)}$	(influence) impressions of i on the Wall of n
$\mathbf{p}_i$	vector with every $p_i^{(n)}$
$\mathbf{q}_i$	vector with every $q_i^{(n)}$
$\mathbf{P}$	matrix with $\mathbf{p}_i$ as column i
$\mathbf{Q}$	matrix with $\mathbf{q}_i$ as column i
$\mathbf{b}_i$	input vector of normalized posting activity
$\mathbf{d}_i$	vector with the posting ratio of a user i
$\mathbf{B}$	matrix with $\mathbf{b}_i$ as column i
$\mathbf{D}$	diagonal matrix with $\mathbf{d}_i$ as column i
$\mathbf{1}$	column-vector of $\mathbb{R}^N$ full of ones, i.e. $(1\ 1\ \cdots\ 1)^T$

2.1 Network user model and notations

Network structure. The platform is represented as a directed and unweighted graph denoted $\mathcal{G} = (\mathcal{N}, \mathcal{E})$. Its vertices represent the users in the OSP and its edges represent the follower-leader relations: $(i, j) \in \mathcal{E}$ means that user i follows user j. The cardinalities of $\mathcal{N}$ and $\mathcal{E}$ are denoted $|\mathcal{N}| = N$ and $|\mathcal{E}| = M$. Note that a social platform where the relationships between users are reciprocal (as for instance, friendships on Facebook) can also be described by this representation, by connecting users with edges in both directions.

Posting activity. As mentioned above, users are also characterized by their posting and re-posting activities. Let λ represent the vector of posting activity over the vertex set. Thus, $\lambda^{(n)}$ represents the number of wall posts per unit of time (posting frequency) created by user n. Similarly, let μ represent the vector of re-posting activity of the vertices. Then, $\mu^{(n)}$ refers to the frequency with which user n checks their Newsfeed and selects one of the current posts uniformly at random for re-posting on their Wall.

2.2 The ψ-score: metric of influence

Here we present the ψ-score following the way it was introduced in [13]. It measures the nodes' influence in the OSP, not only based on the graph structure but also on the nodes' activity, as a user who shares information more frequently is bound to have more influence than users who don't. Similarly, a user whose posts are shared often by their followers is also bound to have a higher influence. The score can be computed from the graph structure and the activity vectors λ and μ only, however, it is helpful to describe the information spreading process that inspired it in order to have a better understanding of its meaning.

Information spreading model. In [13], the authors present the ψ-score as a natural way to evaluate influence within the context of a specific information spreading model on the OSP. In short, the model can be described as follows: a user in the network posts at a constant rate on their wall while messages from their leaders appear on their news feed, replacing a random older post. Regularly, this user selects randomly a post from their news feed to re-post it on the wall. This process is summarized in Figure 1. Again, we underline that this is not necessarily a lifelike description of information spreading on an OSP, nonetheless, it is realistic enough to give a good intuition of the interpretation of the ψ-score and other intermediary quantities that are defined further.
Precisely, this model is a continuous-time Markov chain relying on the following set of assumptions:

- *Poisson arrivals*: any user n posts on their wall according to a Poisson process with rate $\lambda^{(n)}$, and re-posts from the news feed to the wall according to a Poisson process with rate $\mu^{(n)}$.
- *Random selection policy*: we suppose that when a user checks their news feed, they randomly select one of the listed posts to re-post on their wall.
- *Random eviction policy*: a new entry on the wall or on the news feed list replaces an older entry chosen at random.

Influence quantification. Now that the hypotheses of the model are set, we focus on how to quantify the influence of a user. The authors of [13] introduce two vectors related to the influence of user i in $\mathcal{N}$, which can be computed from the activity attributes of the nodes:

- $\mathbf{p}_i = (p_i^{(1)}\ p_i^{(2)}\ \cdots\ p_i^{(N)})^T$ where, for $n \in \mathcal{N}$, $p_i^{(n)}$ represents the expected proportion of posts from user i on the news feed of user n.
- $\mathbf{q}_i = (q_i^{(1)}\ q_i^{(2)}\ \cdots\ q_i^{(N)})^T$ where, for all $n \in \mathcal{N}$, $q_i^{(n)}$ represents the expected proportion of posts from user i on the wall of user n.

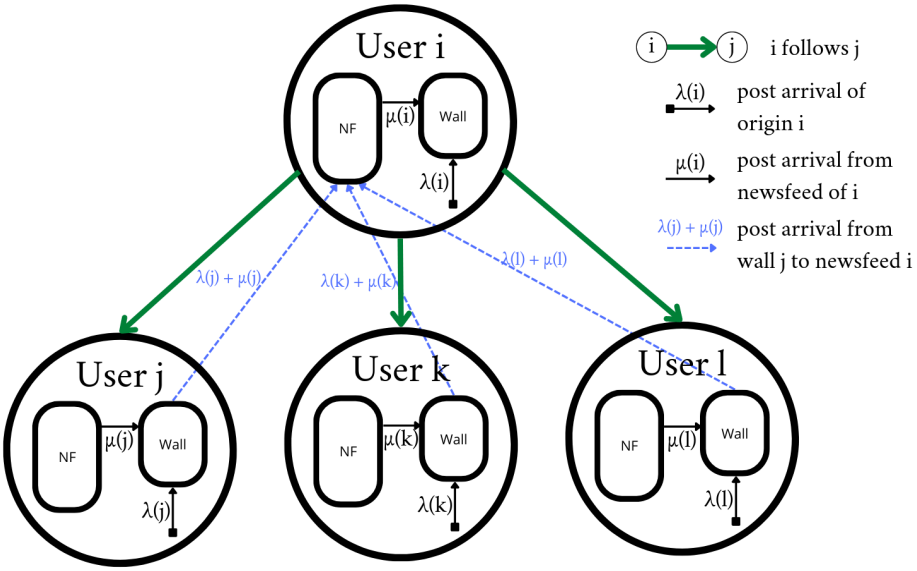

Fig. 1. A small example of the Online Social Platform from the point of view of user i.

So, $q_i^{(n)}$ can be understood as the influence of user i on user n, which means that the more posts from i appear on the wall of n, the greater is i's influence on n. This is used to define the influence of user i throughout the whole network as the average influence of i has on all other users. This concept is formalized by the ψ-score of user i, which is defined as

$$\psi_i = \frac{1}{N} \sum_{n \in \mathcal{N}} q_i^{(n)}. \tag{1}$$

ψ-score computation. In order to calculate the influence score ψ_i of user i, we have to compute all entries of vector $\mathbf{q}_i$ as they appear in its definition (1). However, $\mathbf{q}_i$ is not directly available: a system of equations must be solved to determine the complete vector $\mathbf{q}_i$. Precisely, each $q_i^{(n)}$ can be obtained from $p_i^{(n)}$, so we have to compute the vector $\mathbf{p}_i$. $\mathbf{p}_i$ is the solution to a fixed-point problem, which is determined in [13, Section III]. This fixed-point problem and the derivation of the $\mathbf{q}_i$ vector are summarized in the following linear system, which we present in a matrix form:

$$\mathbf{p}_i = \mathbf{A}.\mathbf{p}_i + \mathbf{b}_i \tag{2}$$

$$\mathbf{q}_i = \mathbf{C}.\mathbf{p}_i + \mathbf{d}_i \tag{3}$$

$$\text{where } \begin{cases} \mathbf{A} \in \mathbb{R}_+^{N\times N}, & a_{ji} = \frac{\mu^{(i)}}{\sum_{\ell\in\mathcal{L}^{(j)}}(\lambda^{(\ell)}+\mu^{(\ell)})}\mathbb{1}_{\{i\in\mathcal{L}^{(j)}\}}, \\ \mathbf{b}_i \in \mathbb{R}_+^N, & b_{ji} = \frac{\lambda^{(i)}}{\sum_{\ell\in\mathcal{L}^{(j)}}(\lambda^{(\ell)}+\mu^{(\ell)})}\mathbb{1}_{\{i\in\mathcal{L}^{(j)}\}}, \\ \mathbf{C} \in \mathbb{R}_+^{N\times N}, & c_{ji} = \frac{\mu^{(j)}}{\lambda^{(j)}+\mu^{(j)}}\mathbb{1}_{\{j=i\}}, \\ \mathbf{d}_i \in \mathbb{R}_+^N, & d_{ji} = \frac{\lambda^{(i)}}{\lambda^{(i)}+\mu^{(i)}}\mathbb{1}_{\{j=i\}}. \end{cases}$$

As shown in [13, Theorem 4], it is possible to solve (2) using a power iteration method as what is typically done to compute the PageRank. It consists in applying iteratively the following equation:

$$\mathbf{p}_i(t) = \mathbf{A}\mathbf{p}_i(t-1) + \mathbf{b}_i \tag{4}$$

t being the current iteration. As $t \to \infty$, the algorithm converges towards the solution $\mathbf{p}_i(t)$ independently of the initialization $\mathbf{p}_i(0)$ because $\mathbf{A}$ is a sub-stochastic matrix[2]. The algorithm 1 given in [13] uses this equation to compute the vector $\mathbf{p}_i$.

Algorithm 1 `Power-NF`: Power method to compute the news feed probabilities $\mathbf{p}_i$ related to user i.

Input: origin user i, $N \times N$ matrix $\mathbf{A}$, vector $\mathbf{b}_i$, $\mathbf{p}$-tolerance ε
Output: vector $\mathbf{p}_i$
$\mathbf{p}_i \leftarrow \mathbf{b}_i$
$t \leftarrow 0$
$gap \leftarrow 1$
while $(gap > \varepsilon)$ **do**
  $\mathbf{p}_i^{old} \leftarrow \mathbf{p}_i$
  $\mathbf{p}_i \leftarrow \mathbf{A}\mathbf{p}_i^{old} + \mathbf{b}_i$
  $gap \leftarrow \left\|\mathbf{p}_i - \mathbf{p}_i^{old}\right\|$
  $t \leftarrow t + 1$
end while
return p

Now, ranking the N users requires to compute ψ_i for each $i \in \mathcal{N}$, *i.e.*, the complete vector $\psi = (\psi_1, \psi_2, \cdots, \psi_N)^T \in \mathbb{R}_+^N$. To solve the linear system (2), the iterative approach in (4) must be used N times for each origin $i \in \mathcal{N}$. Then, mapping each vector $\mathbf{p}_i$ to $\mathbf{q}_i$ thanks to (3) and finally using (1) gives the vector ψ with all the ψ-score values.
As a side note, which will be useful for comparison purposes, the paper [13] also establishes a relationship between the ψ-score and the PageRank. Indeed,the vector ψ equals the PageRank vector π in the scenario of homogeneous activity, meaning that $\forall n \in \mathcal{N}$, $\lambda^{(n)} = \lambda$ and $\mu^{(n)} = \mu$. In that specific case, the PageRank damping coefficient α is given by $\alpha = \frac{\mu}{\lambda+\mu}$ [13, Theorem 5].

[2] A sub-stochastic matrix is a real square matrix having each row summing to a value strictly lower than 1.

Problem Statement. The algorithm 1 for the ψ-score vector computation – which we are going to refer to as `Power-NF` in this paper – is highly time-consuming. By contrast with PageRank, which demands solving 1 system of N equations, we need to solve N systems of N equations separately. This is particularly challenging when attempting to calculate the ψ-score in large real-world networks in a reasonable time.

Consequently, the problem discussed in this work is the following: given a directed social graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and the vectors λ and μ which contain the information about the nodes posting and sharing activities, can we design an algorithm that computes the ψ-score of all nodes in the graph with a comparable speed to PageRank?

3 The Power-ψ Algorithm

In this section, we provide a first algorithm to compute the ψ-score which is faster than the baseline algorithm 1. The principle here is to reformulate the equations as presented in Section 2.2 to reduce the computational cost from solving N linear systems of size N to solving a single linear system of size N that includes all users. It will allow to use the power iteration method to approximate the solution of the new linear system, which can also benefit from distributed computing. We call this process Power-ψ: an algorithm for the derivation of the ψ-score that is comparable to the PageRank in terms of computational complexity, while benefiting from the score expressiveness because of the addition of the node activity information.

3.1 Expressing the ψ-score with a single linear system

We describe the ψ-score of the network in terms of the matrices $\mathbf{P}, \mathbf{Q}, \mathbf{B}, \mathbf{D}$, which are defined in Table 1. This matrix notation enables a vectorized expression of the ψ-score vector:

$$\psi^T = \frac{1}{N}\mathbf{1}^T\mathbf{Q} \tag{5}$$

We remind that the matrix $\mathbf{Q}$ contains the $\mathbf{q}_i = (q_i^{(1)}\ q_i^{(2)}\ \cdots\ q_i^{(N)})^T$ vectors, which components $q_i^{(n)}$ represent the influence of user i on user n. So, the task of obtaining the ψ-score is equivalent to computing a matrix $\mathbf{Q}$, the columns of which require having the vectors $\mathbf{p}_i$, according to Eq. 3.
Now, finding each $\mathbf{p}_i$ is equivalent to solving a linear system, as can be seen in Eq.(2). The equations (2) and (3) call to the matrices $\mathbf{A}$ and $\mathbf{C}$, which are themselves derived from the adjacency matrix of the graph and the posting and re-posting activities of the users (λ and μ). Using [13, Lemma 2] each system solution can be written as

$$\mathbf{p}_i = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{b}_i = \sum_{t=0}^{\infty} \mathbf{A}^t\mathbf{b}_i \tag{6}$$

Eq. (6) is a linear equation in $\mathbf{b}_i$, which implies that the solution in matrix form for all users is

$$\mathbf{P} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} = \sum_{t=0}^{\infty} \mathbf{A}^t\mathbf{B} \tag{7}$$

Thus, by using Eq. (3) in its matrix form and the series in eq. (7), we can develop Eq. (5):

$$\begin{aligned}
\psi^T &= \frac{1}{N}\mathbf{1}^T\mathbf{Q} = \frac{1}{N}\mathbf{1}^T(\mathbf{C}\mathbf{P} + \mathbf{D}) \\
&= \frac{1}{N}\mathbf{1}^T\left[\mathbf{C}(\mathbf{I} - \mathbf{A})^{-1}\mathbf{B} + \mathbf{D}\right] \\
&= \frac{1}{N}\mathbf{1}^T\left[\mathbf{C}\left(\sum_{t=0}^{\infty}\mathbf{A}^t\right)\mathbf{B} + \mathbf{D}\right] \\
&= \frac{1}{N}\left[\left(\sum_{t=0}^{\infty}\mathbf{1}^T\mathbf{C}\mathbf{A}^t\right)\mathbf{B} + \mathbf{1}^T\mathbf{D}\right]
\end{aligned}$$

Let us define $\mathbf{c}^T := \mathbf{1}^T\mathbf{C}$ and $\mathbf{d}^T := \mathbf{1}^T\mathbf{D}$. $\mathbf{c}$ and $\mathbf{d}$ are vectors in $\mathbb{R}_+^N$ (the diagonals of the matrices $\mathbf{C}$ and $\mathbf{D}$). This allows us to formulate the following equation to compute the ψ-score:

$$\psi^T = \frac{1}{N}\left[\left(\sum_{t=0}^{\infty}\mathbf{c}^T\mathbf{A}^t\right)\mathbf{B} + \mathbf{d}^T\right] \tag{8}$$

Eq. (8) demonstrates that it is not essential to solve N linear systems of size N to calculate the ψ-score. Instead, it is sufficient to solve a single N-dimensional linear system, which is written as the infinite sum of matrix-vector multiplications.
There is a drawback to this new approach, which is the fact that we do not compute explicitly the values of $\mathbf{p}_i$ and $\mathbf{q}_i$, which quantify the influence at the individual level. As (8) demonstrates, these are not required any longer to calculate the ψ-score, however, they include valuable information for practical applications, where evaluating the influence of a specific user on others is important.

3.2 Convergence and truncation of the sum

We state here that the infinite sum in Eq. (8) may be evaluated using a power iteration method. To show this, let us define:

$$\mathbf{s}_t^T = \sum_{\tau=0}^{t}\mathbf{c}^T\mathbf{A}^\tau, \quad \mathbf{s} = \lim_{t\to\infty}\mathbf{s}_t, \tag{9}$$

where T denotes the matrix transpose and t is the iteration index. We emphasize that Eq. (9) converges if $\mathbf{A}$'s spectral radius is strictly lower than 1 (which is a standard property of geometric series). Lemmas 1 and 2 in [13] guarantee this convergence, using the fact that $\mathbf{A}$ is a sub-stochastic matrix. Consequently, the sum in 8 may be truncated to a finite number of terms to approximate the ψ-score. Now, let us evaluate the approximation error made when truncating the sum. The recursive expression associated with the truncated sum is

$$\mathbf{s}_t^T = \mathbf{s}_{t-1}^T\mathbf{A} + \mathbf{c}^T \tag{10}$$

where $\mathbf{s}_0 = \mathbf{c}$. This allows us to define the following gap parameter

$$\varepsilon_t = \left\|\mathbf{s}_t^T - \mathbf{s}_{t-1}^T\right\| \tag{11}$$

which is used to stop the algorithm. Any norm can be used, we opt for using the L^1-norm because it is standard when approximating series with power iterations [24]. So practically, we terminate the iterations when the following condition is met:

$$\varepsilon_t \leq \varepsilon \tag{12}$$

A key question here is to evaluate how the tolerance ε impacts the approximation of the ψ-score. To answer this question, we define ψ_t to approximate ψ by substituting (10) with (8) in t terms. Then, we define the difference δ_t between two ψ-score approximations truncated at consecutive steps:

$$\delta_t = \left\|\psi_t^T - \psi_{t-1}^T\right\| \tag{13}$$

Now, we describe δ_t as a function of ε_t and identify an upper-bound relation between the two to deduce a convergence condition on the vector ψ_t from the one on $\mathbf{s}_t$. Indeed, we know from (8) that

$$\begin{aligned}\psi_t^T - \psi_{t-1}^T &= \frac{1}{N}(\mathbf{s}_t^T\mathbf{B} + \mathbf{d}^T) - \frac{1}{N}(\mathbf{s}_{t-1}^T\mathbf{B} + \mathbf{d}^T) \\ &= \frac{1}{N}(\mathbf{s}_t^T - \mathbf{s}_{t-1}^T)\mathbf{B} \end{aligned} \tag{14}$$

Eq. (14) then allows to deduce that:

$$\begin{aligned}\delta_t = \frac{1}{N}\left\|(\mathbf{s}_t^T - \mathbf{s}_{t-1}^T)\mathbf{B}\right\| &\leq \frac{1}{N}\left\|\mathbf{s}_t^T - \mathbf{s}_{t-1}^T\right\| \|\mathbf{B}\| \\ \Rightarrow \delta_t &\leq \frac{\varepsilon_t \|\mathbf{B}\|}{N} \end{aligned} \tag{15}$$

Using this upper bound, we can choose a tolerance ε that ensures the ψ-score evolution does not deviate by more than δ. Precisely, if the algorithm is terminated by the condition $\varepsilon_t \|\mathbf{B}\| \leq \varepsilon$, then δ_t is smaller than $\frac{\varepsilon}{N}$. The corresponding power iteration algorithm is designated as `Power-`ψ and described in Algorithm 2.

Algorithm 2 `Power-`ψ**:** Power method to compute the ψ-score vector.

Input: $N \times N$ matrices $\mathbf{A}$ and $\mathbf{B}$, size N vectors $\mathbf{c}$ and $\mathbf{d}$, $\mathbf{s}$-tolerance ε
Output: vector ψ with the ψ-score of all users
$\mathbf{s} \leftarrow \mathbf{c}$
$B\_norm \leftarrow \|\mathbf{B}\|$
$t \leftarrow 0$
$gap \leftarrow 1$
while $(gap > \varepsilon)$ **do**
  $\mathbf{s}_{old} \leftarrow \mathbf{s}$
  $\mathbf{s}^T \leftarrow \mathbf{s}_{old}^T\mathbf{A} + \mathbf{c}$
  $gap \leftarrow B\_norm \|\mathbf{s}_{old} - \mathbf{s}\|$
  $t \leftarrow t + 1$
end while
$\psi^T \leftarrow \frac{1}{N}\left(\mathbf{s}^T\mathbf{B} + \mathbf{d}^T\right)$
return ψ

3.3 Relationship between ψ-score and PageRank

In this section, we discuss the relationship between the ψ-score in the homogeneous activity scenario and PageRank, which highlights the contrast between our proposed method and the usual power-iteration algorithm for PageRank.
Let π denotes the PageRank vector and $\mathbf{W} = \mathbf{D}_{out}^{-1}\mathbf{L}$ is the random walk transition matrix, where $\mathbf{L}$ is the adjacency matrix of the graph that points to the leaders and $\mathbf{D}_{out}$ is the diagonal matrix whose entries (i, i) are the out-degree of node i. Theorem 5 in [13] ensures that, in the homogeneous activity scenario, where $\forall i$, $\lambda^{(i)} = \lambda$ and $\mu^{(i)} = \mu$, $\psi = \pi$ with $\alpha = \frac{\mu}{\lambda+\mu}$ as the PageRank damping factor.
In the homogeneous activity case, the vectors and matrices in the ψ-score equation of Eq. (8) are as follows:

$$\begin{cases} \mathbf{A} = \alpha\mathbf{W} \\ \mathbf{B} = (1-\alpha)\mathbf{W} \\ \mathbf{c} = \alpha\mathbf{1} \\ \mathbf{d} = (1-\alpha)\mathbf{1} \end{cases}$$

So, after replacing $\mathbf{A}$ and $\mathbf{c}$ by their expression in Eq. (10), we obtain

$$\mathbf{s}_t^T = \alpha\mathbf{s}_{t-1}^T\mathbf{W} + \alpha\mathbf{1}^T \tag{16}$$

By substituting the above expression in Eq. (8) and using the homogeneous values for $\mathbf{B}$ and $\mathbf{d}$ we get

$$\begin{aligned} \psi_t^T &= \frac{1}{N}\left(\mathbf{s}_t^T(1-\alpha)\mathbf{W} + (1-\alpha)\mathbf{1}^T\right) \\ &= \frac{1}{N}\left((\alpha\mathbf{s}_{t-1}^T\mathbf{W} + \alpha\mathbf{1}^T)(1-\alpha)\mathbf{W} + (1-\alpha)\mathbf{1}^T\right) \\ &= \alpha\left(\frac{1}{N}(\mathbf{s}_{t-1}^T(1-\alpha)\mathbf{W} + (1-\alpha)\mathbf{1}^T)\right)\mathbf{W} + \frac{(1-\alpha)}{N}\mathbf{1}^T \\ &= \alpha\psi_{t-1}^T\mathbf{W} + \frac{(1-\alpha)}{N}\mathbf{1}^T \end{aligned} \tag{17}$$

The power method for PageRank computation translates as

$$\pi_t^T = \alpha\pi_{t-1}^T\mathbf{W} + \frac{1-\alpha}{N}\mathbf{1}^T. \tag{18}$$

so, we can see that we find similar expressions, consistently with the result from [13, Theorem 5].

Now, the proposed `Power-`ψ algorithm in the general case uses a power iteration to first estimate the series $\mathbf{s}$ up to a truncation based on the desired tolerance. Following convergence, the product by $\mathbf{B}$ and addition of $\mathbf{d}$ at the end of Algo. 2 prevent superfluous matrix-vector multiplications. This is the main implementation difference compared to PageRank power iteration. As shown above, both approaches converge to the same result in the homogeneous situation; however, the tolerance for ψ-score is defined on the $\mathbf{s}$-vector, whereas for PageRank, it is defined on π itself.

3.4 Computational complexity

We analyze the computational complexity of the `Power-`ψ algorithm. The main computational cost of the algorithm is the power iteration, which is performed until the convergence condition is met. Let us denote $k(\varepsilon)$ the number of iterations required to reach convergence. Each iteration requires a matrix-vector multiplication, which is of complexity $O(M)$ where M is the number of non-zero entries in the matrix, i.e. the number of edges in $\mathcal{G}$. There is also a vector addition, which is of complexity $O(N)$, and a scalar-vector multiplication, which is in $O(N)$. So the total complexity of each iteration is in $O(M+N)$. The total complexity of the algorithm is then $O(k(\varepsilon)(M+N))$. This result is comparable to the complexity of the PageRank power iteration, which is also in $O(k(\varepsilon)(M+N))$.

4 Push approach for influence quantification

In this section, we design a push-based algorithm approach to our problem of influence quantification on a social network. This approach can be applied to two different sets of equations with two distinct purposes. On the one hand, we can use it on Equation (10) in a similar manner as what has been done for PageRank [15, 16]. In this case, the goal is to find more efficiently the vector $\mathbf{s}$, which in turn allows for the direct calculation of the ψ-score of each node. On the other hand, the push method can also be employed to solve Equation (2), and the purpose here is to compute faster the ψ-score of a single user; by providing $\mathbf{p}_i$ and $\mathbf{q}_i$, it gives much more details about the mechanism of influence on the network at an individual level. Note that the application of the push approach to our problem is the main theoretical addition to the shorter version of this work [14].

4.1 Intuition behind the push approach

The push method can be thought of as a message-passing algorithm. The intuitive general idea is to iteratively propagate information across the graph, in order to calculate the importance of each node based on its connections to other nodes.
Practically, it works by initializing a residual value for each node and its approximate score, where the residual represents the amount of information that still needs to be propagated from that node to its neighbors. The algorithm then iteratively updates the approximate score for each node based on the residual that it gets from its neighbors. At each iteration, a node "pushes" its residual to its neighbors, proportionally to the strength of the connection between the two nodes (encoded by the matrix $\mathbf{A}$ in the case of ψ-score and the propagation matrix for PageRank). The node then updates its own approximate score based on the residuals that it received from its neighbors. This process continues until the residuals converge towards 0, indicating that the approximate score vector has converged to the actual score.
Note that the push method can be implemented in a distributed way since each node updates its own score, making it very efficient at solving linear systems of a large number of equations.

4.2 General formulation of the approach Push-*

Let us first rewrite equation (10) as follows:

$$\mathbf{s} = \mathbf{A}^T.\mathbf{s} + \mathbf{c} \tag{19}$$

so that it has exactly the same form as Eq. (2). Now, while they contain and return different quantities, Eq. (2) on $\mathbf{p}_i$ and (19) on $\mathbf{s}$, they allow us to describe the application of the push method on both of them with some generic notations:

$$\mathbf{x} = \mathbf{M}.\mathbf{x} + \mathbf{r} \tag{20}$$

where $\mathbf{x} \in \mathbb{R}^N$ is the unknown vector, $\mathbf{r} \in \mathbb{R}^N$ and $\mathbf{M} \in \mathbb{R}^{N \times N}$. In the case of Eq. (2), $\mathbf{x} = \mathbf{p}_i$, $\mathbf{M} = \mathbf{A}$ and $\mathbf{r} = \mathbf{b}_i$ and in the case of Eq. (19), $\mathbf{x} = \mathbf{s}$, $\mathbf{M} = \mathbf{A}^T$ and $\mathbf{r} = \mathbf{c}$. Taking advantage of this formal similarity, from now on we refer to the method and algorithm that we develop as `Push-*`, then we will describe them as `Push-NF` for Eq. (2) and `Push-`ψ for Eq. (19) when necessary.

It is important here to notice that Equation (20) does not have the same form as the one that computes the PageRank since $\mathbf{r}$ has different components while the second term of the PageRank equation is homogeneous. Consequently, we cannot use the push-based algorithm described in [15] or [16]. Thus we adapt the method to the specific case under consideration.
Let us first observe that the solution $\mathbf{x}$ is a function of the vector $\mathbf{r}$, so that we can re-write Eq. (20) as

$$\mathbf{x}(\mathbf{r}) = \mathbf{M}.\mathbf{x}(\mathbf{r}) + \mathbf{r} \tag{21}$$

Rearranging this equation, it becomes

$$\mathbf{x}(\mathbf{r}) = (\mathbf{I} - \mathbf{M})^{-1}\mathbf{r} = \sum_{n=0}^{\infty} \mathbf{M}^n \mathbf{r} \tag{22}$$

The next step is key and specific to the push method. Let χ_u be the unit vector with only the u-th entry equal to 1 and all the other components equal to 0. Also, we denote $\mathbf{r}(u)$ the u-th entry of $\mathbf{r}$. Using only Eq. (21) and Eq. (22) (which is linear), we have the following expressions:

$$\begin{aligned}
\mathbf{x}(\mathbf{r}) &= \mathbf{x}\left(\mathbf{r} - \mathbf{r}(u)\chi_u + \mathbf{r}(u)\chi_u\right) \\
&= \mathbf{x}\left(\mathbf{r}(u)\chi_u\right) + \mathbf{x}\left(\mathbf{r} - \mathbf{r}(u)\chi_u\right) \\
&= \mathbf{M}.\mathbf{x}(\mathbf{r}(u)\chi_u) + \mathbf{r}(u)\chi_u + \mathbf{x}\left(\mathbf{r} - \mathbf{r}(u)\chi_u\right) \\
&= \mathbf{M}\sum_{n=0}^{\infty} \mathbf{M}^n \mathbf{r}(u)\chi_u + \mathbf{r}(u)\chi_u + \mathbf{x}\left(\mathbf{r} - \mathbf{r}(u)\chi_u\right) \\
&= \sum_{n=0}^{\infty} \mathbf{M}^n \mathbf{M}\mathbf{r}(u)\chi_u + \mathbf{r}(u)\chi_u + \mathbf{x}\left(\mathbf{r} - \mathbf{r}(u)\chi_u\right) \\
&= \mathbf{x}\left(\mathbf{M}\mathbf{r}(u)\chi_u\right) + \mathbf{x}\left(\mathbf{r} - \mathbf{r}(u)\chi_u\right) + \mathbf{r}(u)\chi_u \\
&= \mathbf{x}\left(\mathbf{r} - \mathbf{r}(u)\chi_u + \mathbf{M}\mathbf{r}(u)\chi_u\right) + \mathbf{r}(u)\chi_u \\
&= \mathbf{r}(u)\chi_u + \mathbf{x}(\mathbf{r}'),
\end{aligned} \tag{23}$$

where we define the vector $\mathbf{r}'$ as

$$\mathbf{r}' := \mathbf{r} - \mathbf{r}(u)\chi_u + \mathbf{M}\mathbf{r}(u)\chi_u. \tag{24}$$

So, we have derived a new expression of the solution as the sum of two components:

$$\mathbf{x}(\mathbf{r}) = \mathbf{r}(u)\chi_u + \mathbf{x}(\mathbf{r}') \tag{25}$$

where the first part is an approximation of the solution $\mathbf{x}(\mathbf{r})$ and the second, $\mathbf{x}(\mathbf{r}')$ can be interpreted as a residual and denoted $\mathbf{x}'$.

4.3 Convergence of the residual

Following the above approach, we can put (24) and (25) in an iterative algorithm to derive the solution of (20) with a convergence criterion related to the residual. The problem is that the exact expression of the residual $\mathbf{x}(\mathbf{r}')$ is not available. However, we know its argument $\mathbf{r}'$ from (24). In addition, $\mathbf{r} \mapsto \mathbf{x}(\mathbf{r})$ is a linear map, as shown by Eq. (22), so that $\mathbf{x}(\mathbf{0}) = \mathbf{0}$, where $\mathbf{0}$ is a vector of 0. Thus, we

set a convergence criterion related to the infinite norm of the residual argument of the form $\|\mathbf{r}'\|_\infty < \varepsilon$.
Now, we investigate if we can bound the volume of the residuals. As long as $\|\mathbf{M}\| < 1$, we have that

$$\|\mathbf{x}(\mathbf{r}')\| = \left\|\sum_{n=0}^{\infty} \mathbf{M}^n \mathbf{x}'\right\| \leq \left\|\sum_{n=0}^{\infty} \mathbf{M}^n\right\| \cdot \|\mathbf{x}'\|$$
$$\leq \varepsilon \cdot \sum_{n=0}^{\infty} \|\mathbf{M}\|^n = \frac{\varepsilon}{1 - \|\mathbf{M}\|}, \tag{26}$$

Note that this inequality is true for the L^1 norm of a right sub-stochastic matrix, such as $\mathbf{A}$, and for the L^∞ norm of a left sub-stochastic matrix. In the case of Eq. (2) (`Push-NF`), $\mathbf{M} = \mathbf{A}$ and in the case of Eq. (19) $\mathbf{M} = \mathbf{A}^T$ (`Push-`ψ). So, we have the following cases of induced norms

- in the case of `Push-NF`:

$$\|\mathbf{x}(\mathbf{r}')\|_1 \leq \frac{\varepsilon}{1 - \|\mathbf{A}\|_1} = \frac{\varepsilon}{1 - \max_{1 \leq u \leq N} \sum_{v \in \mathcal{F}(u)} \mathbf{A}(v,u)}, \tag{27}$$

- in the case of `Push-`ψ:

$$\|\mathbf{x}(\mathbf{r}')\|_\infty \leq \frac{\varepsilon}{1 - \left\|\mathbf{A}^T\right\|_\infty} = \frac{\varepsilon}{1 - \max_{1 \leq u \leq N} \sum_{v \in \mathcal{L}(v)} \mathbf{A}^T(u,v)} \tag{28}$$

Where $\mathcal{F}(u)$ designates the set of followers of user u while $\mathcal{L}(v)$ is the set of leaders of user v.
As a consequence, we can choose the tolerance ε to be an upper bound of $\frac{\varepsilon}{1-\|\mathbf{M}\|}$ as a convergence criterion by taking the appropriate norm for $\mathbf{M}$.

4.4 The Push-* algorithm

We propose our `Push-*` algorithm following this expression to derive the solution $\mathbf{x}(\mathbf{r})$ of the generic equation. Each iteration of `Push-*` works as follows: the u-th entry of the approximation vector is incremented by $\mathbf{r}(u) > \varepsilon$, which will be then set to 0 as indicated by the first part of (24). Next, for every v follower (respectively leader) of u for (2) (respectively for (19)), the value $\mathbf{M}(v,u).\mathbf{r}(u) > 0$ is added to $\mathbf{r}(v)$ (second part of (24)). The algorithm stops when the value of the residual is small enough according to what has been discussed in Section 4.3.

Algorithm 3 `Push-*`: the Push algorithm for $\mathbf{x}$

Input: $N \times 1$ vector $\mathbf{r}$, $N \times N$ matrix $\mathbf{M}$, ε
Output: Calculate the $N \times 1$ vector $\mathbf{x}(\mathbf{r})$
Initialization: $x(v) = 0$, for $v = 1, \ldots, N$
FIFO $\leftarrow$ CreateFIFO()
for user u with $\mathbf{x}(u) > \varepsilon$ **do**
    FIFO.add(u)
    Mark(u)
end for
while FIFO not empty **do**
    $u \leftarrow FIFO.pop()$
    $\mathbf{x}(u) \leftarrow \mathbf{x}(u) + \mathbf{r}(u)$
    for user v in $\mathbf{M}(:, u) > 0$ **do**
        $\mathbf{r}(v) \leftarrow \mathbf{r}(v) + \mathbf{M}(v, u) \cdot \mathbf{r}(u)$
        if $\mathbf{r}(v) > \varepsilon$ and (v not marked) **then**
            FIFO.add(v)
            Mark(v)
        end if
    end for
    $\mathbf{r}(u) \leftarrow 0$
    Unmark(u)
end while

5 Numerical evaluation

In this section, we evaluate the performances of the three algorithms presented in this work: `Power-`ψ, `Push-`ψ and `Push-NF`. We compare their performances to the baseline algorithm `Power-NF` described in [13]. In the case where the posting and re-posting activities of nodes are identical for all nodes, the ψ-score is equivalent to the PageRank, it allows us to compare the results to the ones obtained with the PageRank in this homogeneous scenario.
First, we briefly present the package which has been developed and released for this project. Second, we describe the different datasets that are investigated in the experiments. Then we precise some important elements of the experimental protocol, before reporting the results of the performance evaluation.

5.1 The `psi-score` Python package

The *psi-score* project is a software library that we have developed in Python. It is open source and licensed under the terms of the MIT license, allowing free usability. It is currently available at the address https://github.com/NouamaneA/psi-score.
This package is a practical tool for the community to easily use the ψ-score metric in projects without the need to develop each algorithm individually. All methods mentioned in this work are accessible: the baseline `Power-NF` as well as the proposed algorithms `Power-`ψ, `Push-`ψ and `Push-NF`. To simplify its usage, the project has a similar Application Programming Interface (API) as the one used in *scikit-network* [27] and *scikit-learn* [28].

5.2 Datasets

For the numerical evaluation of our methods, we choose four real-world datasets which are listed in Table 2. Three of them are publicly available in *the KONECT project*[3] [29] which is a network dataset repository. These three datasets do not include any information about the nodes' activity in the network so we decided to generate λ and μ uniformly at random. The last dataset has been downloaded from *Kaggle* dataset repository[4] [30]. It is a Twitter trace that contains a million tweets and retweets from 181,621 users during the 2018 Russian elections, we can therefore derive the actual posting and re-posting activities (respectively λ and μ) of all users.

Table 2. Datasets used for evaluation

Dataset name	Type	#Nodes	#Edges	Sources
`DBLP`	Citation Network	12 590	49 759	[31]
`Twitter-ICWSM`	Social Network	465 017	834 797	[32]
`ego-Twitter`	Social Network	23 370	33 101	[33]
`Russian`	Social Network	181 621	515 419	[30]

[3] http://konect.cc/
[4] https://www.kaggle.com

5.3 Experiments

Unit of comparison. The complexity of the methods is compared using the number of messages required to reach a targeted tolerance ε, which is the number of vector entries updated at each iteration during the process. We use the term "messages" because we can interpret each update as a signal that propagates across the network from a node to one of its neighbors. By contrast with the computation time, this measure has the advantage of allowing the comparison between sequential and distributed algorithms, regardless of the technology used to implement them.
Note here that each one of the algorithms (`Power-`ψ, `Power-NF`, `Push-*` and `PageRank`) have a different notion of tolerance, depending on the criterion of convergence. Specifically, for `Push-*` we name it **r**-tolerance because the convergence is based on the norm of the residual vector, as expressed by the equation $\|\mathbf{x}'\|_\infty < \varepsilon$. For `Power-NF` we name it **p**-tolerance because the convergence is based on the norm of the news-feed **p**-values. For `Power-`ψ we refer to **s**-tolerance because the convergence is based on the norm of the series **s**-values. For `PageRank` we consider π-tolerance because the convergence is based on the norm of the PageRank vector. Also, we underline that further processing is needed after convergence to eventually derive the resulting ψ-scores. So, when we set some value ε for the x-tolerance according to a specific method, we obtain some approximation of the ψ-score ψ_ε, however, we are interested in evaluating the relative error of this method compared to the exact value ψ_{true}, defined as follows:

$$error = \frac{\|\psi_{true} - \psi_\varepsilon\|_2}{\|\psi_{true}\|_2} \tag{29}$$

Note that to compute the ψ_{true} vector, we use the solver from the `sparse` module of the SciPy Python library [34] which employs matrix factorization to solve the system (10).

Design of the experiments. We conducted three series of experiments to evaluate the performance of our proposed algorithms. The first series of experiments aims to validate their ability to approximate the exact ψ-score by comparing the approximation error of the algorithms to the exact ψ-score value. In the second series, we aim at validating the algorithms ability to return the actual user ranking based on the ψ-score. In the third series, we compare the algorithms' performance with that of `Power-NF`, in terms of messages, which is the unit of comparison previously defined. To do this, we used the Kendall τ correlation coefficient to compare the rankings returned by each algorithm to the rankings returned by the true ψ-score values.

Scenarios. For each series of experiments, we consider two scenarios:

(i) In the *heterogeneous scenario*, we set the activity parameters λ and μ to be heterogeneous. It means that in the cases of `ego-Twitter`, `DBLP` and `Twitter-ICWSM`, where no temporal activity data is available, we draw λ and μ uniformly at random in the interval $[0, 1]$. In the case of `Russian`, we use the actual activities measured on the dataset.

(*ii*) In the *homogeneous scenario*, we set the activity parameters λ and μ to be homogeneous, *i.e.*, we set $\lambda = 0.15$ and $\mu = 0.85$ for all nodes. According to Section 3.3, this corresponds to the case where the ψ-score is equivalent to the PageRank with a damping factor $\alpha = 0.85$ (which is a typical value for the damping factor). Besides, we run in this scenario the experiments on the giant connected component of the graph, as the equivalence between the ψ-score and the PageRank is ensured only within these settings, as stated in [13].

Series 1: quality of the ψ-score approximation. In the first series of experiments, we evaluate the accuracy of our proposed algorithms in approximating the ψ-score vector by measuring the error and comparing them to `Power-NF` in both scenarios and to `PageRank` in the homogeneous scenario only.
The corresponding results for the heterogeneous and homogeneous activity scenarios are respectively presented in Figure 2 and Figure 3. The x-axis of the figure represents the fixed **p**-, **s**-, **r**- and π-tolerance for each algorithm, while the y-axis represents the computed error from equation (29). Notably, the `Power-`ψ method exhibits significantly lower approximation error than the other algorithms, which can be attributed to the fact that the measured gap at each iteration is multiplied by $\|\mathbf{B}\|$ to ensure that the gap in the ψ-score approximation is below the chosen tolerance when the algorithm converges. We also notice that the power method is systematically more accurate than the corresponding push method: the error committed with `Power-`ψ is lower than the one of `Push-`ψ and similarly, the error with `Power-NF` is lower than with `Push-NF`. It is expected, as the tolerance is more directly related to the quantity targeted with a power method than it is with a push method since the latter bases its convergence on the norm of the residual. In the homogeneous scenario, Figure 3 shows that `Power-`ψ is more accurate than `PageRank` in the sense that a similar tolerance implies a lower error.

Series 2: quality of the ranking approximation. When using centrality measures, we are in general less concerned by the absolute value of the score than we are by the ordering of the nodes. Thus, to evaluate how the error presented in the first series of experiments translates in terms of ranking, we investigate in this series the quality of the approximations in view of the ranking that they produce. To assess this, we compute the correlation between the rankings produced by each algorithm to the one generated by the exact ψ-score using Kendall's τ correlation coefficient. Introduced in [35], this metric is used to measure the similarity between two rankings. It involves counting the number of concordant and discordant pairs (resp. n_C and n_D) between the two rankings x and y, as well as the number of ties t_x and t_y within each ranking. Then, the coefficient is computed as follows:

$$\tau(x, y) = \frac{n_C - n_D}{\sqrt{(n_C + n_D + t_x)(n_C + n_D + t_y)}}$$

The results of this series of experiments in the heterogeneous and homogeneous scenarios are respectively displayed in Figure 4 and Figure 5. We represent Kendall's τ scores as a function of the tolerance on all four datasets and for each algorithm. As can be seen, all three proposed algorithms (`Push-`ψ, `Push-NF`, and `Power-`ψ) perform very well in terms of user ranking, in the sense that the Kendall

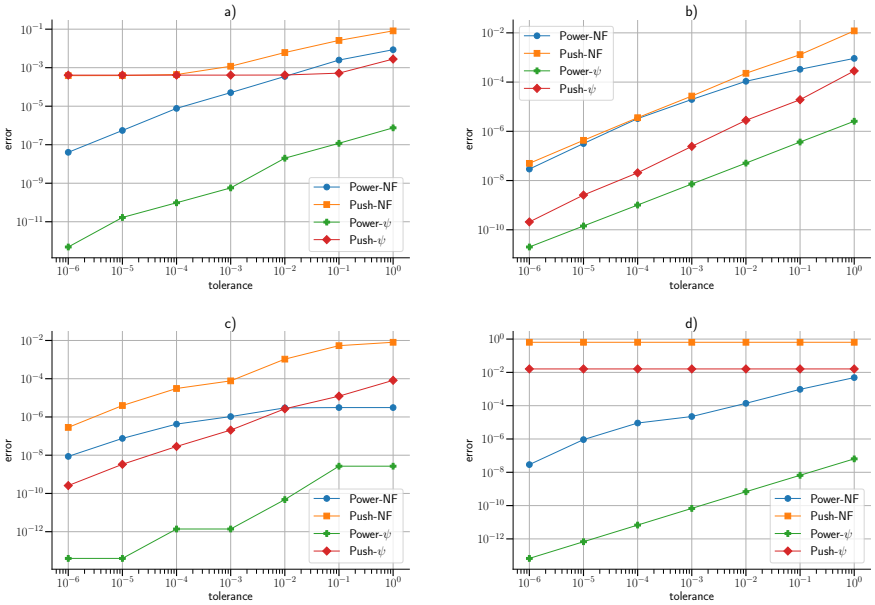

Fig. 2. (*i*) Heterogeneous scenario. Experiment 1 with a) `DBLP`, b) `ego-Twitter`, c) `Twitter-ICWSM` and d) `Russian`: Precision assessment of the proposed algorithms compared to the baseline.

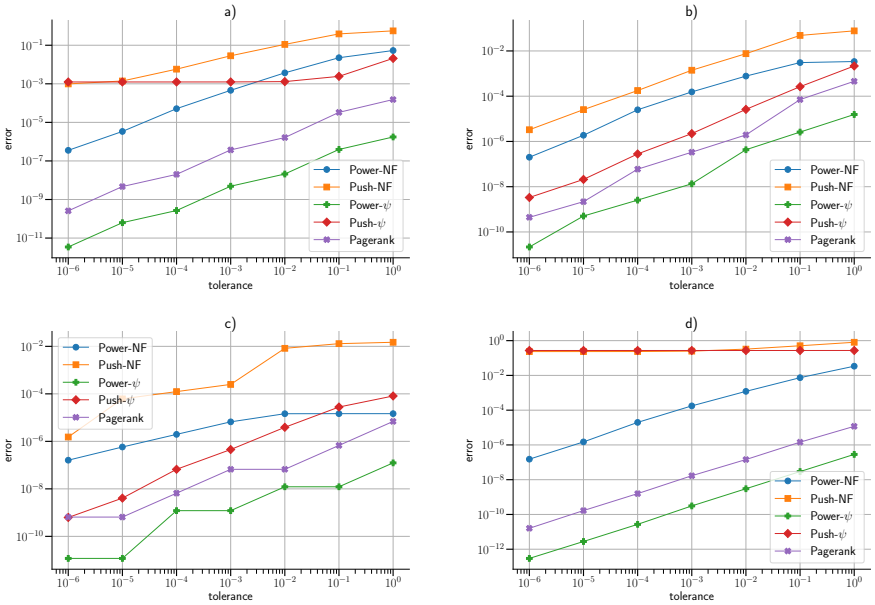

Fig. 3. (*ii*) Homogeneous scenario. Experiment 1 with a) `DBLP`, b) `ego-Twitter`, c) `Twitter-ICWSM` and d) `Russian`: Precision assessment of the proposed algorithms compared to the baseline.

τ scores are very close to 1, even with relatively high tolerance values. This means that the algorithms are able to effectively rank users in the same way as the actual ψ-score does. In the homogeneous activity scenario, we notice a decrease in the ranking quality of `Push-NF` when increasing the tolerance: a significant drop occurs around a tolerance of 10^{-2} in the cases of `DBLP`, `ego-Twitter` and `Russian` and around a tolerance of 10^{-3} concerning `Twitter-ICWSM`.

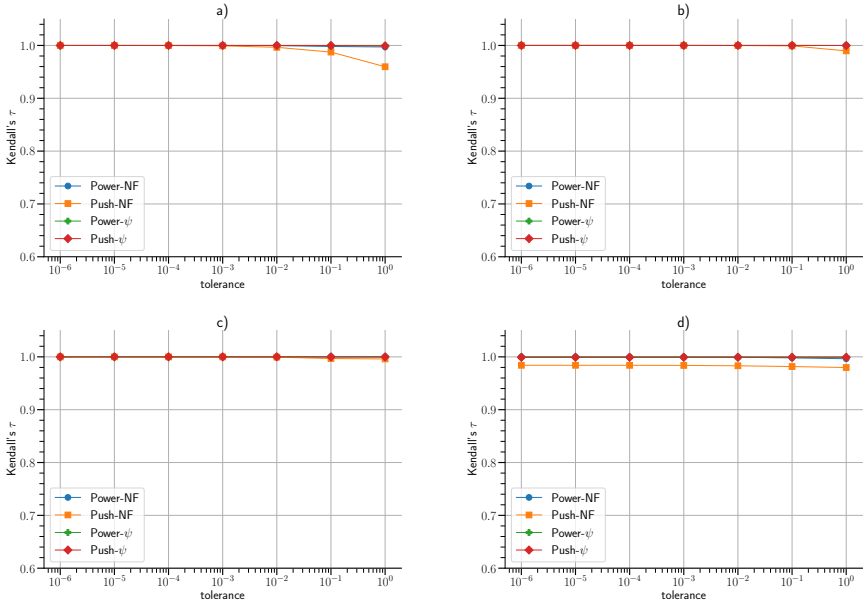

Fig. 4. (*i*) Heterogeneous scenario. Experiment 2 with a) `DBLP`, b) `ego-Twitter`, c) `Twitter-ICWSM` and d) `Russian`: Evaluation of the algorithms ability to return an accurate ranking compared to the one returned by the exact ψ-score. 1 means a perfect correlation, 0 means no correlation at all and -1 means a full anti-correlation.

Series 3: message complexity. In the third series of experiments, we focus on the scalability of these algorithms by evaluating their performance in terms of the number of messages required to achieve a given level of precision in approximating the result. It is worth noticing that achieving the same precision for all algorithms is not always possible. This is because, as discussed previously, the tolerance is not equivalent for all algorithms, so the resulting approximation error on the exact score may vary. To compare the performances of the algorithms to the baseline, we run them with various tolerance parameters (the same ones used in the first series of experiments) and measure the number of messages required to reach convergence. We then calculate the relative error using (29) to compare the algorithms performance under a specific level of precision. Note that the message-based complexity can be simply related to the number of iterations

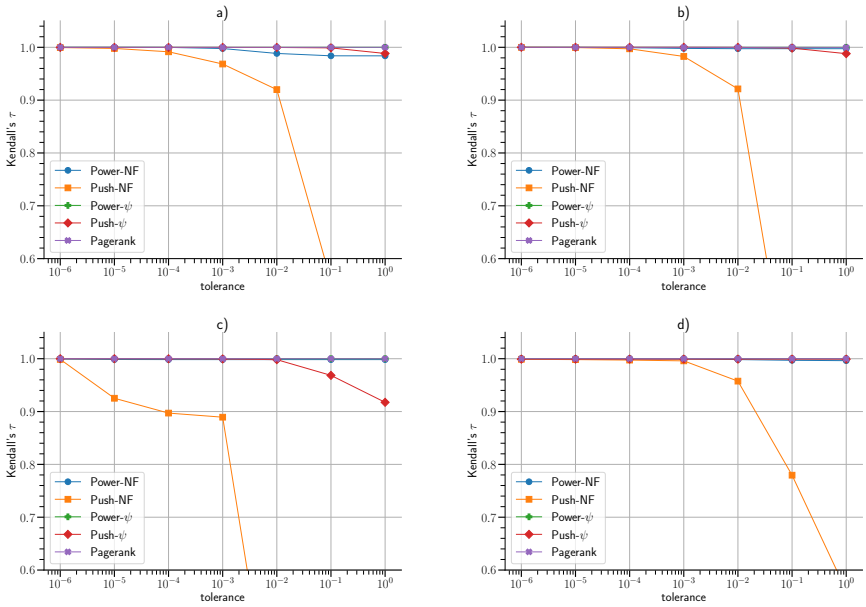

Fig. 5. (*ii*) Homogeneous scenario. Experiment 2 with a) `DBLP`, b) `ego-Twitter`, c) `Twitter-ICWSM` and d) `Russian`: Evaluation of the algorithms ability to return an accurate ranking compared to the one returned by the exact ψ-score/PageRank. 1 means a perfect correlation, 0 means no correlation at all and -1 means a full anti-correlation.

of a power method. Indeed, a complete iteration consists of sending M messages through the network, where M is the number of edges in the network. So the value $k(\varepsilon)$ mentioned in 3.4 is related to the total number of messages n_{mes} sent by the relation $k(\varepsilon) = n_{mes}/M$.

The results of this series of experiments are respectively reported in Figure 6 and 7 for the heterogeneous and homogeneous scenarios. Consistently with the findings of the short version of this work [14], we observe that `Power-`ψ significantly outperforms `Power-NF` due to the reduced number of equations to solve. The `Push-*` algorithms also bring some very interesting results, as `Push-NF` requires fewer messages than `Power-NF`, and `Push-`ψ requires fewer messages that `Power-`ψ. The authors of [26] reported a specificity of push methods, which is that they are not competitive to obtain very precise approximations as they require a large number of messages to decrease the error. Such a trend is confirmed by the `DBLP` and `Russian` datasets. We even notice a peculiar trend in Figure 6-d) with `Russian`, for which we do not succeed in improving the approximation error even when largely increasing the number of messages of the push methods. This might be due to the fact that, unlike the other datasets, the activity parameters λ and μ are measured from the dataset (see Section 5.2) itself and we noticed that almost half of the nodes in this social network are inactive. However, we do not notice such a dramatic trend with datasets `ego-Twitter` and `Twitter-ICWSM`.

An important observation in Figure 7 is that `Power-`ψ and `Push-`ψ converge faster than `PageRank` in the homogeneous scenario. This makes the ψ-score a very interesting alternative to PageRank for ranking users in social networks as its complexity has been reduced to the same level as PageRank.

Finally, the overall picture that we draw from these experimental investigations is that push methods are steadily faster than power methods but allow less control over the error committed on the ψ-score. Consequently, we suggest that both methods will not be used for the same applications, depending on the precision or speed required for the computation. Also, in the case of the datasets investigated, the fact that the push methods are limited to larger errors does not significantly impact the algorithms performance at ranking users, as we have seen in the previous series of experiments.

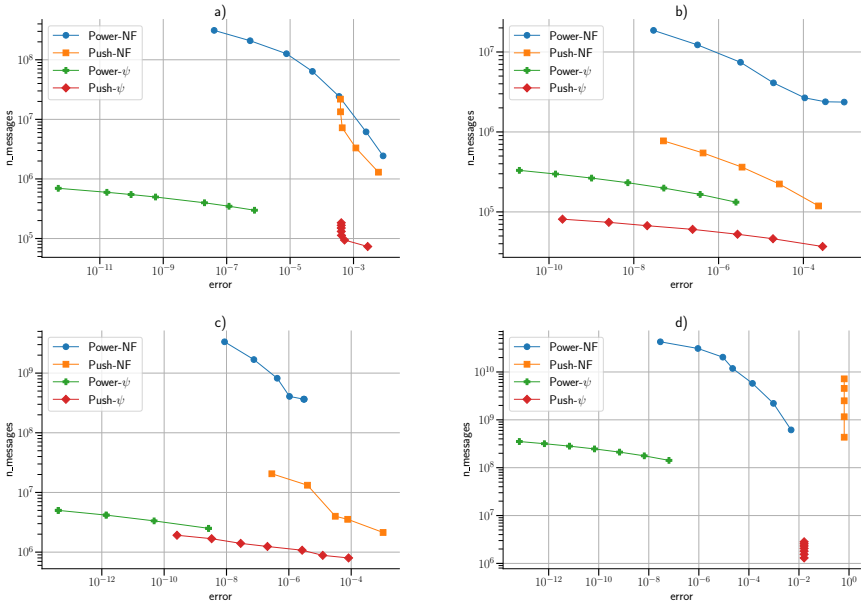

Fig. 6. (*i*) Heterogeneous scenario. Experiment 3 with a) `DBLP`, b) `ego-Twitter`, c) `Twitter-ICWSM` and d) `Russian`: Comparison of the proposed algorithms with the baseline (Power-NF) in terms of the number of messages to reach convergence.

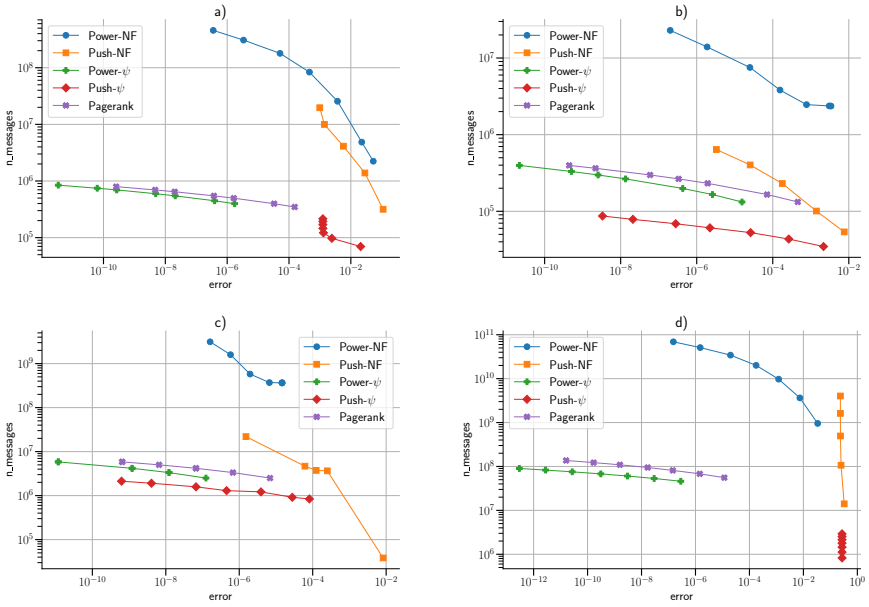

Fig. 7. (*ii*) Homogeneous scenario. Experiment 3 with a) `DBLP`, b) `ego-Twitter`, c) `Twitter-ICWSM` and d) `Russian`: Comparison of the proposed algorithms with the baselines `Power-NF` and `PageRank` in terms of the number of messages to reach convergence.

6 Conclusion and future work

In conclusion, the ψ-score is a powerful metric that generalizes PageRank for non-homogeneous node activities. It combines the structural information of the graph with the temporal information of the users' activities to rank their influence in the social network. In this work, we have proposed three scalable algorithms for approximating the ψ-score. The `Power-`ψ and `Push-`ψ algorithms take advantage of a novel equation that reduces the system we need to solve for the ψ-score calculation, while the `Push-NF` algorithm provides additional insights on the information dynamics at the individual level. Through our experiments, we have shown that these algorithms perform well on various real-world datasets both in terms of precision of the approximation and in terms of speed; however, someone interested in controlling the error committed will favor power methods while someone interested in time efficiency will favor push-methods. We have released them as an open-source Python package for the research community to use and improve upon.

This score suits many real-world applications. It has been specially designed to evaluate the notion of influence on online social platforms, typically micro-blogging social networks. Indeed, while the PageRank only takes into account the structural information, the ψ-score also considers nodes posting and re-posting activities, which are known to be important features on these platforms. It could thus help identify influential users on social networks. However, it could also be an interesting score to evaluate influence in other contexts, such as scientific citation networks, where an author posting corresponds to a new publication and re-posting would be a citation. Here as well, we expect the users' activity to have a significant effect on their visibility and influence in the network.

This study opens several interesting avenues for future work. A possibility is to investigate in what way the heterogeneous patterns of activities in real networks affect the spread of influence on the platform: for instance, does it allow some specific posts to reach some distant users faster? Another one could be to measure to what extent the idealized process underlying the ψ-score definition differs from the actual diffusion process and possibly identify anomalous behaviors from these differences. For instance, we can think that if a post of a user is largely re-posted while they have a relatively low ψ-score, it might convey a specific message that is worthy of analysis.

Acknowledgement. This work is funded by the ANR (French National Agency of Research) through the FairEngine project (grant ANR-19-CE25-0011) and the FiT LabCom project (grant FiT-ANR-19-LCV1-0005).

References

1. A. Bavelas, The journal of the acoustical society of America **22**(6), 725 (1950)
2. L.C. Freeman, Sociometry pp. 35–41 (1977)
3. M.S. Alvim, B. Amorim, S. Knight, S. Quintero, F. Valencia, in *International Conference on Formal Techniques for Distributed Objects, Components, and Systems* (Springer, 2021), pp. 22–41
4. D. Kempe, J. Kleinberg, E. Tardos, in *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (Association for Computing Machinery, New York, NY, USA, 2003), KDD '03, p. 137–146. https://doi.org/10.1145/956750.956769. URL https://doi.org/10.1145/956750.956769
5. K. Bok, Y. Noh, J. Lim, J. Yoo, Electronic Commerce Research **21**(3), 671 (2021)
6. C. Musto, P. Lops, M. de Gemmis, G. Semeraro, Knowl. Based Syst. **216**, 106806 (2021). https://doi.org/10.1016/j.knosys.2021.106806. URL https://doi.org/10.1016/j.knosys.2021.106806
7. V. Cabannes, L. Pillaud-Vivien, F. Bach, A. Rudi, Advances in Neural Information Processing Systems **34** (2021)
8. S. Goel, D.J. Watts, D.G. Goldstein, in *Proceedings of the 13th ACM Conference on Electronic Commerce* (Association for Computing Machinery, New York, NY, USA, 2012), EC '12, p. 623–638. https://doi.org/10.1145/2229012.2229058. URL https://doi.org/10.1145/2229012.2229058
9. Z. Wan, Y. Mahajan, B.W. Kang, T.J. Moore, J.H. Cho, IEEE Access **9**, 104773 (2021). https://doi.org/10.1109/ACCESS.2021.3094196
10. U. Brandes, Journal of mathematical sociology **25**(2), 163 (2001)
11. M. Cha, H. Haddadi, F. Benevenuto, K. Gummadi, Proceedings of the International AAAI Conference on Web and Social Media **4**(1), 10 (2010). URL https://ojs.aaai.org/index.php/ICWSM/article/view/14033
12. C. Wilson, A. Sala, K.P.N. Puttaswamy, B.Y. Zhao, ACM Trans. Web **6**(4) (2012). https://doi.org/10.1145/2382616.2382620. URL https://doi.org/10.1145/2382616.2382620
13. A. Giovanidis, B. Baynat, C. Magnien, A. Vendeville, IEEE/ACM Trans. Netw. **29**(5), 2198 (2021). https://doi.org/10.1109/TNET.2021.3085201. URL https://doi.org/10.1109/TNET.2021.3085201
14. N. Arhachoui, E. Bautista, M. Danisch, A. Giovanidis, in *2022 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)* (2022), pp. 526–533. https://doi.org/10.1109/ASONAM55673.2022.10068673
15. J.J. Whang, A. Lenharth, I.S. Dhillon, K. Pingali, in *Euro-Par 2015: Parallel Processing*, ed. by J.L. Träff, S. Hunold, F. Versaci (Springer Berlin Heidelberg, Berlin, Heidelberg, 2015), pp. 438–450

16. R. Andersen, F. Chung, K. Lang, in *2006 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS'06)* (2006), pp. 475–486. https://doi.org/10.1109/FOCS.2006.44
17. A. Landherr, B. Friedl, J. Heidemann, Wirtschaftsinformatik **52**, 367 (2010)
18. C. Correa, T. Crnovrsanin, K.L. Ma, IEEE Transactions on Visualization and Computer Graphics **18**(1), 106 (2012). https://doi.org/10.1109/TVCG.2010.260
19. P.R. Miller, P.S. Bobkowski, D. Maliniak, R.B. Rapoport, Political Research Quarterly **68**(2), 377 (2015). https://doi.org/10.1177/1065912915580135. URL https://doi.org/10.1177/1065912915580135
20. R. Puzis, Y. Altshuler, Y. Elovici, S. Bekhor, Y. Shiftan, A. Pentland, J. Intell. Transp. Syst. **17**(1), 91 (2013). https://doi.org/10.1080/15472450.2012.716663. URL https://doi.org/10.1080/15472450.2012.716663
21. P. Hui, J. Crowcroft, E. Yoneki, in *Proceedings of the 9th ACM international symposium on Mobile ad hoc networking and computing* (2008), pp. 241–250
22. A. Tizghadam, A. Leon-Garcia, IEEE Network **24**(6), 10 (2010). https://doi.org/10.1109/MNET.2010.5634437
23. M. Ashtiani, A. Salehzadeh-Yazdi, Z. Razaghi-Moghadam, H. Hennig, O. Wolkenhauer, M. Mirzaie, M. Jafari, bioRxiv (2017). https://doi.org/10.1101/149492. URL https://www.biorxiv.org/content/early/2017/10/09/149492
24. L. Page, S. Brin, R. Motwani, T. Winograd. The pagerank citation ranking: Bringing order to the web (1998)
25. A.N. Langville, C.D. Meyer, *Google's PageRank and beyond: The science of search engine rankings* (Princeton university press, 2006)
26. E. Bautista, M. Latapy, Soc. Netw. Anal. Min. **12**(1), 31 (2022). https://doi.org/10.1007/s13278-022-00860-5. URL https://doi.org/10.1007/s13278-022-00860-5
27. T. Bonald, N. de Lara, Q. Lutz, B. Charpentier, Journal of Machine Learning Research **21**(185), 1 (2020). URL http://jmlr.org/papers/v21/20-412.html
28. L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, R. Layton, J. VanderPlas, A. Joly, B. Holt, G. Varoquaux, in *ECML PKDD Workshop: Languages for Data Mining and Machine Learning* (2013), pp. 108–122
29. J. Kunegis, in *Proc. Int. Conf. on World Wide Web Companion* (2013), pp. 1343–1350. URL http://dl.acm.org/citation.cfm?id=2488173
30. B. Chumichev. Russian election 2018 — twitter user activity. https://www.kaggle.com/datasets/borisch/russian-election-2018-twitter (2018)
31. M. Ley, in *String Processing and Information Retrieval, 9th International Symposium, SPIRE 2002, Lisbon, Portugal, September 11-13, 2002, Proceedings*, *Lecture Notes in Computer Science*, vol. 2476, ed. by A.H.F. Laender, A.L. Oliveira (Springer, 2002), *Lecture Notes in Computer Science*, vol. 2476, pp. 1–10. https://doi.org/10.1007/3-540-45735-6"¨1. URL https://doi.org/10.1007/3-540-45735-6_1
32. M. De Choudhury, Y.R. Lin, H. Sundaram, K.S. Candan, L. Xie, A. Kelliher, in *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media* (2010). URL http://www.public.asu.edu/˜mdechoud/pubs/icwsm_10.pdf

33. J. McAuley, J. Leskovec, in *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1* (Curran Associates Inc., Red Hook, NY, USA, 2012), NIPS'12, p. 539–547
34. P. Virtanen, R. Gommers, T.E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S.J. van der Walt, M. Brett, J. Wilson, K.J. Millman, N. Mayorov, A.R.J. Nelson, E. Jones, R. Kern, E. Larson, C.J. Carey, İ. Polat, Y. Feng, E.W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E.A. Quintero, C.R. Harris, A.M. Archibald, A.H. Ribeiro, F. Pedregosa, P. van Mulbregt, SciPy 1.0 Contributors, Nature Methods **17**, 261 (2020). https://doi.org/10.1038/s41592-019-0686-2
35. M.G. Kendall, Biometrika **30**(1-2), 81 (1938). https://doi.org/10.1093/biomet/30.1-2.81. URL https://doi.org/10.1093/biomet/30.1-2.81