ARS MATHEMATICA CONTEMPORANEA

Also available at http://amc-journal.eu
ISSN 1855-3966 (printed edn.), ISSN 1855-3974 (electronic edn.)
ARS MATHEMATICA CONTEMPORANEA 11 (2016) 11–33

Spectral centrality measures in temporal networks

Selena Praprotnik
FMF, University of Ljubljana, Jadranska 19, 1000 Ljubljana, Slovenia

Vladimir Batagelj *
FMF, University of Ljubljana, Jadranska 19, 1000 Ljubljana, Slovenia

Received 11 February 2015, accepted 14 April 2015, published online 6 July 2015

Abstract

In our previous article we defined temporal quantities used for the description of temporal networks with zero latency and we showed that some centrality measures (e.g. degree, betweenness, closeness) can be extended to the case of temporal networks. In this article we broaden the scope of centrality measures in temporal networks to centrality measures derived from the eigenvectors of network matrices, namely the eigenvector in-centrality, the eigenvector out-centrality, the Katz centrality, the Bonacich α and $(\alpha, \beta)-$centrality, the HITS algorithm (also known as Hubs and Authorities) introduced by Kleinberg, and the PageRank algorithm defined by Page and Brin.

We extended our Python library TQ (Temporal Quantities) to include the algorithms from our research. The library is available online. The procedures will also be added to the user friendly program called Ianus. We tested the proposed algorithms on Franzosi's violence network and on Corman's Reuter terror news network and show the results.

Keywords: Temporal network, semiring, algorithm, network measures, Python library, violence.

Math. Subj. Class.: 91D30,16Y60,90B10,68R10.

1 Introduction

Many real-life problems can be represented as networks in which the actors are represented with nodes (or vertices) and interactions between the actors are represented with links – arcs or edges, according to the nature of interactions (whether the interactions are directed or

*The work was supported in part by the ARRS, Slovenia, grant J5-5537, as well as by a grant within the EURO-CORES Programme EUROGIGA (project GReGAS) of the European Science Foundation.

E-mail addresses: selena.praprotnik@fmf.uni-lj.si (Selena Praprotnik), vladimir.batagelj@fmf.uni-lj.si (Vladimir Batagelj)

not). Networks have been widely studied in mathematics, computer science, biology, social sciences and other disciplines. There are many examples of data that have an underlying network structure, such as the Internet, the phone-calls data, the co-authorship graphs, the email graphs, the biological and the chemical networks, the transaction networks, the trade networks, etc. These networks are tipically generated by human activity and often exhibit similar structure. The network analysis has seen an ever increasing research activity in the past years due to the amount of data available and to the global interest in data analysis. See for example [7] and [22]. In the last two decades, there has been an increased interest in temporal network analysis where a time dimension is also considered.

The node centrality has been a fundamental tool in the study of social networks since the late 1940s, beginning with the Group Networks Laboratory at MIT directed by Alex Bavelas (see [3], [4], [21]). The node degree is probably the oldest measure of a node's importance in a network. In a network, every node has some measure of influence or importance within the rest of the network and the importance of a node is determined by the structure of the network it belongs to. Centrality measures are designed to rank nodes based on their structural position inside the network and different centrality measures aim to quantitatively measure the importance of a node in the network. Various measures of centrality were employed in different contexts. There is no consensus on what centrality is and there is little agreement on the best way to measure it. It still falls to the network analysts to decide which centrality measure is the most appropriate for the given network and context and to define exactly what the purpose of the computation is. The usual questions that are approximately answered using network centrality measures are - Who are the influential people in a social network? Which roads are most often used? Which web pages are important?

In this article, we make a step towards connecting two of the most frequent questions arising in contemporary network analysis: we consider the temporal changes of the centralities of nodes.

The paper is organized as follows: in Sections 2 and 3 we review some basic ideas and notations on centrality measures and temporal networks. In Section 4 we present the algorithms for computing the spectral centrality measures, and we give examples on real-life data in Section 5. We conclude with possible directions of future research in Section 6.

2 Centrality Measures and Graph Matrices

Let $G = (V, L)$ be a graph with a node set $V = \{v_1, v_2, \ldots, v_n\}$ and a link set $L \subseteq V \times V$. An adjacency matrix $\mathbf{A}(G) = [a_{uv}]$ of the graph G is a binary $n \times n$ matrix with elements

$$a_{uv} = \begin{cases} 1, & (u, v) \in L, \\ 0, & \text{otherwise.} \end{cases}$$

Therefore, the undirected graphs have a symmetric adjacency matrix and the graphs with no loops have adjacency matrices with zero diagonal elements. If the network has an arc without its opposite arc, the adjacency matrix is not symmetric. In directed networks, we have two types of links adjacent to a node – links pointing to the node (incoming) and links pointing away from the node (outgoing). The number of incoming links is the node indegree, the number of outgoing links is the node outdegree.

If G has weights on the arcs we let a_{uv} be the weight of the arc (u, v). Let $\mathbf{A} \in \mathbb{R}^{n \times n}$ be the corresponding matrix. An eigenvector of $\mathbf{A}$ is a non-zero vector x such

that $\mathbf{A}x = \lambda x$ for some complex λ, which is called an eigenvalue of $\mathbf{A}$ belonging to the vector x. The eigenvalues of a graph G are defined as the eigenvalues of its adjacency matrix $\mathbf{A}(G)$. The set of the eigenvalues of G is called the spectrum of G. There are many connections between the eigenvalues of a graph and its combinatorial properties. These include eigenvalues that are not dominant and are beyond the scope of this article.

It is well known that if $\mathbf{A}$ is a real symmetric matrix then its eigenvalues are real. Even more is true for nonnegative matrices ([5]).

Theorem 2.1. *(The Perron-Frobenius Theorem) If an $n \times n$ matrix has nonnegative entries then it has a nonnegative real eigenvalue λ which has a maximum absolute value among all eigenvalues. This eigenvalue λ has a nonnegative real eigenvector. If, in addition, the matrix has no block – triangular decomposition (i.e., it does not contain a $k \times (n-k)$ block of zeros disjoint from the diagonal), then λ has a multiplicity of 1 and the corresponding eigenvector is positive. If the matrix is positive, λ has the strictly largest absolute value.*

Theorem 2.1 implies that if a graph G is strongly connected (and the link weights are nonnegative in case of weighted graphs), then the strongly largest eigenvalue $\lambda_{\max}$ of $\mathbf{A}(G)$ has a multiplicity of 1 and the corresponding eigenvector is positive. If the graph is not strongly connected, the uniqueness of the largest eigenvalue is not guaranteed.

All centrality measures are real valued functions on the nodes of the network. Spectral centrality measures are based on the (left) dominant eigenvector of a network adjacency matrix or some other matrix derived from it. Existence and uniqueness of the spectral measures follow from the theory of nonnegative matrices.

The rationalization behind using an eigenvector as a centrality measure is that important nodes have many connections, but the nodes with the highest degree are not necessarily the most important. It is not just the number of neighbors that counts, but also the importance of these neighbors.

3 Temporal Quantities in Networks

In [2], we proposed a definition of temporal networks with zero latency that is based on temporal quantities. Here, we repeat some of the definitions and describe our approach to the temporal networks which we will use in the rest of the paper.

A temporal network $\mathcal{N}_T = (\mathcal{V}, \mathcal{L}, \mathcal{T}, \mathcal{P}, \mathcal{W})$ is obtained by attaching the time $\mathcal{T}$ to an ordinary network of nodes $\mathcal{V}$ and links $\mathcal{L}$. The sets $\mathcal{P}$ and $\mathcal{W}$ represent the node properties and the link properties or weights, respectively. The time $\mathcal{T}$ is a set of time points, $t \in \mathcal{T}$, and is usually a subset of positive integers, $\mathcal{T} \subseteq \mathbb{N}$.

In a temporal network, nodes $v \in \mathcal{V}$ and links $\ell \in \mathcal{L}$ are not necessarily present or active at all time points. We denote the activity sets of time points for the nodes v with $T(v), T \in \mathcal{P}$, and for the links ℓ with $T(\ell), T \in \mathcal{W}$. The activity set $T(e)$ of a node/link e is described as a sequence of time intervals $([s_i, f_i))_{i=1}^{k}$, where s_i is the starting time and f_i is the finishing time of the activity.

Besides the presence/absence of nodes and links also their properties can change through time. Let a describe the temporal property of a node/link. The activity set of the corresponding node/link is denoted with T_a. To describe the changes we introduce a notion of a temporal quantity

$$\tilde{a}(t) = \begin{cases} a(t) & t \in T_a \\ \text{⌘} & t \in \mathcal{T} \setminus T_a \end{cases}$$

where $a(t)$ is the value of a at an instant t, and $\mathbb{H}$ denotes the value undefined. In the following we are talking about temporal quantities and we write simply a instead of $\tilde{a}$. We assume that the values of temporal properties belong to a set A which is a semiring $(A, \oplus, \odot, 0, 1)$. We can extend both operations to the set $A_{\mathbb{H}} = A \cup \{\mathbb{H}\}$ by requiring that for all $a \in A_{\mathbb{H}}$ it holds

$$a \oplus \mathbb{H} = \mathbb{H} \oplus a = a \quad \text{and} \quad a \odot \mathbb{H} = \mathbb{H} \odot a = \mathbb{H}.$$

The structure $(A_{\mathbb{H}}, \oplus, \odot, \mathbb{H}, 1)$ is also a semiring. For more about semirings see [1] or [16].

Let $A_{\mathbb{H}}(\mathcal{T})$ denote the set of all temporal quantities over $A_{\mathbb{H}}$ in the time $\mathcal{T}$. To extend the operations to networks and their matrices we define the sum of temporal quantities (corresponds to parallel links)

$$a \oplus b = s$$

as

$$s(t) = \begin{cases} a(t) \oplus b(t) & t \in T_a \cap T_b \\ a(t) & t \in T_a \setminus T_b \\ b(t) & t \in T_b \setminus T_a \\ \mathbb{H} & \text{otherwise} \end{cases}$$

and $T_s = T_a \cup T_b$; and the product of temporal quantities (corresponds to sequential links)

$$a \odot b = p$$

as

$$p(t) = \begin{cases} a(t) \odot b(t) & t \in T_a \cap T_b \\ \mathbb{H} & \text{otherwise} \end{cases}$$

and $T_p = T_a \cap T_b$. This definition of product is restricted to temporal networks with zero latency – the time needed to traverse the link is equal to zero and there is no waiting in nodes for the next transition.

We define the temporal quantities $\mathbf{0}$ and $\mathbf{1}$ with $\mathbf{0}(t) = \mathbb{H}$ and $\mathbf{1}(t) = 1$ for all $t \in \mathcal{T}$. The structure $(A_{\mathbb{H}}(\mathcal{T}), \oplus, \odot, \mathbf{0}, \mathbf{1})$ is also a semiring, and therefore so is the set of square matrices of order n over it for the addition $\mathbf{A} \oplus \mathbf{B} = \mathbf{S}$

$$s_{ij} = a_{ij} \oplus b_{ij}$$

and multiplication $\mathbf{A} \odot \mathbf{B} = \mathbf{P}$

$$p_{ij} = \bigoplus_{k=1}^{n} a_{ik} \odot b_{kj}.$$

The operations $\oplus$ and $\odot$ on the left hand side operate on matrices and on the right hand side in the semiring of temporal quantities.

The static network consisting of links and nodes that are present in the temporal network $\mathcal{N}$ at the time $t \in \mathcal{T}$ is denoted by $\mathcal{N}(t)$ and is called a *time slice* of the network $\mathcal{N}$. The addition and the multiplication of temporal quantities operate inside the chosen time slice. They are defined as pointwise operations on functions. The operations in the matrix semiring also operate on the network time slices. Using these operations on a temporal network is equivalent to using the usual operations on a sequence of static networks that represent the time slices of the temporal network and then combining them into one result.

Using our algebraic approach avoids creating the network time slices and the problem of choosing the time intervals for which the time slices should be computed. The appropriate intervals are chosen by the operations on temporal quantities.

The procedures we have developed for the analysis of temporal networks using temporal quantities are available as a Python library TQ (Temporal Quantities) at `http://vladowiki.fmf.uni-lj.si/doku.php?id=tq`. In the TQ library the temporal quantities are represented with the sequences of ordered triples $[(s_i, f_i, v_i)]_{i\in I}, I \subseteq \mathbb{N}$, where $[s_i, f_i)$ is the time interval in which the temporal quantity has the value $v_i \in A$. Note that this means that the temporal quantities are constant functions on time intervals. The procedures that will be used for the computation of spectral centrality measures use and extend this library. A user friendly program Ianus is also being developed.

4 Algorithms

4.1 Eigenvector Centrality

The most intuitive notion of centrality is the degree centrality which says that the most important nodes in the network are the ones with the highest degree. In many applications, the degree centrality is flawed as it measures the exposure (the number of arcs) and not the actual influence of the node. Wasserman and Faust [23] discuss what they call prestige measures of centrality where the centralities of nodes in a network are recursively related to the centralities of the nodes to which they are linked, the idea being "It is better to have less friends who are powerful than to have a lot of non-powerful friends." This measure has the following form.

Let $\mathbf{A}$ be the (weighted) adjacency matrix of the network in which $a_{vu} \neq 0$ implies that there exists an arc $\ell = (v, u)$ and let x be the in-centrality vector. The form of the prestige measure is

$$x_v = a_{1v}x_1 + a_{2v}x_2 + \cdots + a_{nv}x_n = \sum_{u:u\to v} a_{uv}x_u,$$

where we denote "u links to v" with $u \to v$. The in-centrality of the node v is a combination of the in-centralities of the in-neighbors of v. This set of equations has a matrix representation

$$\mathbf{A}^T x = x. \tag{4.1}$$

In the equation (4.1), x is an eigenvector of $\mathbf{A}^T$ corresponding to the eigenvalue of 1. It has no non-zero solutions unless $\mathbf{A}^T$ has an eigenvalue of 1. One way to solve this problem is to normalize the rows so that each row sums to 1. Then the normalized matrix has an eigenvalue 1 and there is a solution to the equation (4.1).

Eigenvector centrality, first proposed in [8], generalizes the equation to the general eigenvector equation for $\mathbf{A} \in \mathbb{R}^{n\times n}$. The underlying assumption is that the node's in-centrality is proportional to the weighted sum of the in-centralities of its neighbors

$$\lambda x_v = a_{1v}x_1 + a_{2v}x_2 + \cdots + a_{nv}x_n = \sum_{u:u\to v} a_{uv}x_u$$

which (in the matrix notation) is equivalent to

$$\mathbf{A}^T x = \lambda x. \tag{4.2}$$

This equation always has a non-zero solution.

In some cases, it is more appropriate to define the out-centrality of the node v as a combination of the centralities of the out-neighbors of v. In this case, with the same reasoning, the eigenvector out-centrality is the solution to the equation

$$\mathbf{A}x = \lambda x. \tag{4.3}$$

When computing both (in- and out-) eigenvector centralities, we are looking for the dominant eigenvalue and the corresponding eigenvector of the network adjacency matrix $\mathbf{A}$. The simplest numerical method to compute them is the power iteration, see [14] for a more detailed description and for the convergence conditions.

1: **function** *power*(A, x_0) ▷ x_0 is the initial approximation for the eigenvector
2: $i = 0$
3: **while** no convergence **do**
4: $y_{i+1} = Ax_i$
5: $x_{i+1} = y_{i+1}/\|y_{i+1}\|_2$ ▷ Approximate eigenvector.
6: $i = i + 1$
7: $\overline{\lambda} = x_i^T Ax_i$ ▷ Approximate eigenvalue.

Our implementation of the power iteration algorithm for temporal networks with zero latency as a function *eigTemp* is described in Algorithm 1. The algorithm returns the approximate eigenvector x, the approximate eigenvalue ev and the parameter *convergence,* which tells us whether the algorithm ended when the required tolerance was achieved (its value is **True**) or not (its value is **False**). The function *MatVecRight*(A, x) computes the product $\mathbf{A}x$ for a temporal matrix $\mathbf{A}$ and a temporal vector x, the function *normalize*(x) implements the temporal version of $x/\|x\|_2$. The function *test_dif*(x, y) finds the maximal value (over time) of $\|x - y\|_2$, which we compare to the desired tolerance in line 7. If we achieved the desired tolerance *tol,* we exit the loop. We compute the approximate eigenvalue after the algorithm exits the loop to avoid numerous matrix multiplications. The function *scalProd*(x, y) computes the scalar product of two temporal vectors x and y. Line 11 is the temporal version of $\lambda = x^T\mathbf{A}x$.

Algorithm 1 Temporal power iteration.

1: **function** *eigTemp*$(A, x, tol = 10^{-6}, maxIter = 100)$
2: $i = 0$
3: *convergence* = **False**
4: **while** $i <$ *maxIter* **do**
5: $x\_old = x$
6: $x =$ *normalize*(*MatVecRight*(A, x))
7: **if** *test_dif*$(x, x\_old) < tol$ **then**
8: *convergence* = **True**
9: **break**
10: $i = i + 1$
11: $ev =$ *scalProd*$(x,$ *MatVecRight*$(A, x))$
12: **return** $(x, ev,$ *convergence*$)$ ▷ x is an approximate eigenvector and ev is an approximate eigenvalue

Theorem 4.1. *The temporal power iteration algorithm converges if and only if the non-temporal power iteration converges for every time slice matrix* $\mathbf{A}(t),\ t \in \mathcal{T}$.

Proof. The addition and the multiplication of temporal quantities correspond to pointwise operations with functions $a : A_{\mathfrak{K}} \to A_{\mathfrak{K}}$ as we noted on page 4 after the definition of the operations. For every $t \in \mathcal{T}$ the values of the temporal quantities describe a static network – the time slice of the temporal network at the time t. Therefore the conditions of convergence for static matrices translate pointwise to temporal matrices. In the algorithm, the function *test_dif* checks the maximum difference over all times. Since the lifetime is a finite set, the pointwise convergence implies that this maximum converges to 0. □

Corollary 4.2. *Let $\lambda_1(t)$ and $\lambda_2(t)$ be the eigenvalues with the greatest absolute values of a network time slice matrix $\mathbf{A}(t)$. The rate of convergence is $\mu = \max\{|\lambda_2(t)/\lambda_1(t)|,\ t \in \mathcal{T}\}$. The temporal power iteration algorithm converges for $\mu < 1$ and the convergence is slower when the value of μ is near to* 1.

Proof. The temporal power iteration algorithm converges at the time point t when the quotient $|\lambda_2(t)/\lambda_1(t)| < 1$. The proof of convergence for static matrices can be found in [14]. The rate is calculated pointwise and the maximum over a finite set of time points is computed. □

Note that by the proof of Theorem 4.1 the temporal power iteration algorithm can converge for some times $t \in \mathcal{T}$ and not converge for others. We give two stopping conditions for the loop: The first condition is the desired tolerance *tol* which has a default value of 10^{-6} and the second condition is the number of iterations *maxIter* with a default value of 100. In our implementation we set the convergence parameter to **True** when convergence of all the time slice matrices is achieved. It could easily be altered to require convergence of at least one of the time slices.

We use the temporal power iteration algorithm to compute the eigenvector in-centrality (function *inEig*) and the eigenvector out-centrality (function *outEig*). Both algorithms are written in the Algorithm 2. The function *MatTrans*(A) computes the transpose of a temporal matrix $\mathbf{A}$. The function *VecConst*(n) creates a temporal vector of the dimension n, which has components equal to $\mathbf{1}$. The function *numInv*(a) replaces the value of the temporal quantity with its inverse value, leaving the time component intact. The function *numVecProd*(a, x) computes the product of a temporal quantity a and a temporal vector x. In line 2 (or 6), we compute the approximate eigenvalue and eigenvector for $\mathbf{A}^T$ (or $\mathbf{A}$) with an initial vector of (temporal) ones. In line 3 (or 7), we scale the vector according to the eigenvalue.

Algorithm 2 Temporal eigenvalue centrality.

```
1: function inEig(A)
2:     (x, ev, conv) = eigTemp(MatTrans(A), VecConst(len(A)))
3:     x = numVecProd(numInv(ev), x)
4:     return (x, conv)
5: function outEig(A)
6:     (x, ev, conv) = eigTemp(A, VecConst(len(A)))
7:     x = numVecProd(numInv(ev), x)
8:     return (x, conv)
```

If the network is not strongly connected, the network matrix (with the right renumeration of nodes) has a block form

$$\mathbf{A} = \begin{bmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{0} & \mathbf{D} \end{bmatrix}.$$

In this case, the corresponding dominant eigenvector is not necessarily unique and there is some debate on how to interpret the result. A lot of the times, the right eigenvector has the form $[\tilde{x}, 0]^T$ which means that we get no information about a lot of the nodes. When the given network is not connected, the matrix has a block diagonal form. Let

$$\mathbf{A} = \begin{bmatrix} 0.8000 & 0.7500 & 0 & 0 \\ 0.2000 & 0.2500 & 0 & 0 \\ 0 & 0 & 0.4000 & 0.5455 \\ 0 & 0 & 0.6000 & 0.4545 \end{bmatrix}$$

be the matrix of a disconnected network. It has the eigenvalues $\lambda_{1,2} = 1, \lambda_3 = -0.1455$ and $\lambda_4 = 0.0500$. The dominant eigenvectors (corresponding to the eigenvalues of 1) are $v_1 = [0.9662, 0.2577, 0, 0]$ and $v_2 = [0, 0, -0.6727, -0.7399]$. Because they correspond to the same eigenvalue, also their sum $v_1 + v_2 = [0.9662, 0.2577, -0.6727, -0.7399]$ or any linear combination of v_1 and v_2 is also an eigenvector. How do we choose the right one? No definite answer to this question has been given. The first two eigenvectors correspond to the centralities of nodes in each component, which makes sense, but there is no good way to compare the two scores.

Another problem with disconnected networks is that the node scores in the largest component do not neccessarily get non-zero values, and the highest scores are often those, that correspond to dyads (strongly connected components with two nodes), which are usually not of high interest. The nodes in the largest strongly connected component (that are of greatest interest most of the time) are not likely to have scores higher than those of the dyadic component. This is usually solved by introducing some normalization factor, which we have not implemented in our algorithms.

The problem of finding the strongly / weakly connected components in temporal networks with zero latency and no waiting in nodes has been addressed in our article [2]. If the network is not strongly connected, the user can choose how to proceed – one can either extract the strongly connected components and compute the eigenvector centralities separately for each component, or use one of the other, more elaborate measures that are described in the later sections of this article and have no such limitation to their use.

4.2 Katz Centrality

In his article [18] Katz describes the centrality index which computes the centrality of a node v by taking into account the centralities of all the nodes from which the node v is reachable. In the proposed approach a weight α is used to dampen the effects of more distant nodes. The weight α could depend on the group and the context and could also vary through time. We only consider the case when it is constant through time. We assume that it is known or we compute it in a way that guarantees the convergence of the algorithm. The constant α can be viewed as the probability of success of the link: the value $\alpha = 0$ means that even the neighboring nodes have no impact on the node and the value $\alpha = 1$ means that the distant nodes are as important as the neighbors.

This idea is modelled with powers of the binary adjacency matrix $\mathbf{A}$ of the network, as the element a_{vu} from $\mathbf{A}^r$ equals to the number of walks of length r from the node v to the

node u through other nodes. The column sums of $\mathbf{A}$ give the indegrees of nodes (walks of length 1) and the column sums of $\mathbf{A}^r$ give the number of walks of length r from other nodes.

The idea is to find the column sums of the matrix

$$\mathbf{T} = \alpha\mathbf{A} + \alpha^2\mathbf{A}^2 + \cdots + \alpha^k\mathbf{A}^k + \cdots = (\mathbf{I} - \alpha\mathbf{A})^{-1} - \mathbf{I}.$$

It has been shown in [18] that this is equivalent to solving the system of linear equations

$$\left(\frac{1}{\alpha}\mathbf{I} - \mathbf{A}^T\right) t = d, \tag{4.4}$$

where d is a vector of indegrees. The vector t has elements t_v which are the column sums of the matrix $\mathbf{T}$, i.e. the answers to the original question. This means that for a given network with the binary adjacency matrix $\mathbf{A}$ and for a given α we only need to solve the system of linear equations (4.4). In his article, Katz states that reasonable values of $1/\alpha$ are those between the largest eigenvalue of $\mathbf{A}$ and about twice that value. For smaller values of $1/\alpha$ the effect of distant nodes is greater.

The usual centrality indices are normalized – in case of degree, for example, by $n-1$, the number of possible choices. Using the same notion, Katz [18] defined the divisor of t_v by

$$m = \alpha(n-1) + \alpha^2(n-1)^{(2)} + \alpha^3(n-1)^{(3)} + \ldots,$$

where $(n-1)^{(k)} = (n-1)(n-2)\cdots(n-k)$. A good approximation for m is

$$m \doteq (n-1)!\alpha^{n-1}e^{1/\alpha},$$

which improves with increasing n.

The Katz centrality vector is given by $\frac{1}{m}t$, where t is the solution to the equation (4.4).

We used the Jacobi's method (see [14]) to compute the solution to the linear system of equations. It is an iterative method for solving linear systems of the form $\mathbf{A}x = b$. The idea of the Jacobi's method is to rewrite the original system in the form $\mathbf{A} = \mathbf{L} + \mathbf{D} + \mathbf{U}$, where $\mathbf{D} = \mathrm{diag}(\mathbf{A})$ and $\mathbf{L}$ and $\mathbf{U}$ are the lower and upper triangles of $\mathbf{A}$, respectively. Then iterate

$$\mathbf{D}x_{m+1} = -(\mathbf{L} + \mathbf{U})x_m + b.$$

The impletation of the Jacobi's method for solving a system $\mathbf{A}x = b$, where $\mathbf{A}$ and b have elements that are temporal quantities, is written in the Algorithm 3 as a function *jacobi*. We give two conditions for exiting the loop: when we reach the desired precision *tol* of the solution or when we compute a predetermined number of steps *maxIter*. In line 2, we compute the inverse of the diagonal matrix $\mathbf{D}$, and in line 3, we compute the matrix $\mathbf{B} = -(\mathbf{L} + \mathbf{U})$, by setting the diagonal of $\mathbf{A}$ to undefined (semiring neutral element) and negating the values. Line 8 computes the next approximation to the solution as a temporal version of $x_n = invD(Bx + b)$. Lines 9-11 test whether the desired tolerance has been achieved and end the computation if that is the case.

Theorem 4.3. *The temporal Jacobi iteration algorithm converges if and only if the non-temporal Jacobi iteration converges for all the time slice network matrices* $\mathbf{A}(t),\ t \in \mathcal{T}$.

Proof. The reasoning is the same as in the proof of Theorem 4.1. The addition and the multiplication of temporal quantities correspond to pointwise operations on functions. The operations on temporal matrices can therefore be viewed as if we were operating on sequences of static matrices and the convergence conditions for static matrices translate to temporal matrices. □

Definition 4.4. The static matrix $\mathbf{A}$ is *strictly diagonally dominant* if it holds

$$|a_{jj}| > \sum_{\substack{i=1 \\ i \neq j}}^{n} |a_{ij}|, \quad j = 1, 2, \ldots, n.$$

Corollary 4.5. *If all the time slice matrices are strictly diagonally dominant the temporal Jacobi iteration converges with any temporal vector as the initial aproximation to the solution of the linear system* $\mathbf{A}x = b$

Proof. The proof of convergence for static matrices can be found in [14]. □

Similarly to the power iteration, the Jacobi iteration algorithm can converge for some times $t \in \mathcal{T}$ and not converge for others. We set the convergence parameter to **True** if it converges in all time points for which the values of temporal quantities are defined.

Algorithm 3 Temporal Jacobi iteration.

1: **function** *jacobi*$(A, b, x, tol = 10^{-6}, maxIter = 100)$ ▷ x is the initial approximation for the solution
2: $invD = MatSetDiagVec(vecInv(diag(A)))$
3: $B = MatMinus(MatSetDiagZero(A))$
4: $i = 0$
5: $convergence =$ **False**
6: **while** $i < maxIter$ **do**
7: $i = i + 1$
8: $xn = MatVecRight(invD, VecSum(MatVecRight(B, x), b))$
9: **if** $test\_dif(x, xn) < tol$ **then**
10: $convergence =$ **True**
11: **break**
12: $x = xn$
13: **return** $(xn, convergence)$

The algorithm for computing the Katz centrality for temporal networks is written as Algorithm 4. In the algorithm for computing the Katz centrality, the input parameter a, corresponding to α, can be left out.

Corollary 4.6. *The Algorithm 4 computes the parameter* a *in a way that insures that the Jacobi's algorithm converges when* a *is not given as an input parameter.*

Proof. In lines 4-9 of the algorithm we compute a from the maximum of all the column sums (indegrees), so that a is a little bigger than this maximum and every time slice matrix in the equation (4.4) is strictly diagonally dominant. Therefore the algorithm converges by the Corollary 4.5. □

Lines 10-13 compute $\mathbf{B} = \frac{1}{a}\mathbf{I} - \mathbf{A}^T$, and line 14 computes the solution to the linear equation $\mathbf{B}t = d$ with the initial approximation equal to the temporal vector with all elements equal to $\mathbf{1}$. Lines 15-17 normalize the solution with an appropriate m.

Note that the algorithm also works for weighted adjacency matrices. In this case, the powers of the adjacency matrix are the weighted sums of the walks and the above explanation is not that straightforward.

Algorithm 4 Temporal Katz centrality.

```
1:  function katz(A, a = Null)
2:      n = len(A)
3:      d = MatVecLeft(A, VecConst(n))          ▷ Column sums – temporal indegrees.
4:      if a = Null then                        ▷ Compute a if it is not given.
5:          max = 0
6:          for i = 1 : len(d) do
7:              if VecMax(d[i]) > max then
8:                  max = VecMax(d[i])
9:          a = 0.999/max
10:     B = n × n temporal matrix
11:     for i = 1 : n do
12:         B[i][i] = [(1, ∞, 1/a)]
13:     B = MatDiff(B, MatTrans(A))
14:     (t, conv) = jacobi(B, d, VecConst(n))
15:     m = math.factorial(n − 1) ∗ (a ∗ ∗(n − 1)) ∗ math.exp(1/a)
16:     m = [(1, ∞, 1/m)]
17:     return (numVecProd(m, t), conv)
```

4.3 Bonacich α and (α, β) Centrality

The dominant eigenvector from Section 4.1 is one of the standard measures of network centrality but it also has its flaws. The nodes with zero indegree also have a zero centrality. Nodes pointed at by nodes with zero centrality also have a zero centrality and the effect propagates to other nodes. In many cases the eigenvector centrality gives no information about a lot of nodes. Some solutions to this problem were given, see for example [10], [9] and [22].

We can assign each node v some status s_v that is independent of the connections. It is possible for the vector s to reflect the effects of external status but it is often assumed to be a vector of ones. The new equation is

$$x = \alpha(\mathbf{A}^T x) + s.$$

The parameter α weighs the relative importance of the network sources versus the outside sources. This measure is called $\alpha-$centrality. It has a matrix solution

$$x = (\mathbf{I} - \alpha\mathbf{A}^T)^{-1}s$$

and is almost identical to the measure proposed by Katz in [18] which we study in Section 4.2. The temporal version of $\alpha-$centrality is written in Algorithm 5. The parameter a in

the algorithm corresponds to the parameter α from the definition. If the status vector s is not given, we set it to be a temporal vector of ones in line 3. The solution to the linear system is computed with the temporal version of Jacobi's iteration (Algorithm 3).

Algorithm 5 Temporal Bonacich $\alpha-$centrality.

```
1: function alpha(A, a, s = None)
2:     if s = None then
3:         s = VecConst(len(A))
4:     return jacobi(MatSum(MatEye(len(A)),
   numMatProd([(1, ∞, −a)], MatTrans(A))), s, VecConst(len(A)))
```

Another proposed solution from [9], written in Algorithm 6, is also very similar to Katz's centrality measure. It depends on two parameters α and β. The parameter β affects how much of the node's influence is due to the node's neighborhood. If β is positive the status of the node is increasing with its connections. This would be the case in a communication network, for example, where the amount of information available to the individual is increasing with the amount of information available to its contacts. A positive β is chosen in situations in which the node's status (power, influence) increases with connections to influential nodes.

In some situations it is advantageous to have connections to people who have few other options (e.g. in bargaining). In this case power comes with connections to powerless nodes and the node's power reduces with connections to powerful nodes. In such cases a negative β is chosen. The main difference between this measure and Katz's is that we allow $\beta < 0$.

The magnitude of β affects the influence of more distant nodes. When $\beta = 0$, the $(\alpha, \beta)-$centrality measure is proportional to the degree. With increasing $|\beta|$ the distant (reachable) nodes influence the node's centrality in a greater proportion.

The $(\alpha, \beta)-$centrality of a node v is defined as

$$c_v(\alpha, \beta) = \sum_u (\alpha + \beta c_u) a_{uv},$$

which we write in matrix notation as

$$c(\alpha, \beta) = \alpha(\mathbf{I} - \beta\mathbf{A})^{-1}\mathbf{A}\mathbf{e}, \tag{4.5}$$

where $\mathbf{e}$ is a column vector of ones.

From (4.5) we see that α only affects the length of the solution vector. If α is not given, we normalize the solution in such a way that $\|c(\alpha, \beta)\|_2^2 = n$. Using this normalization, $c_v(\alpha, \beta) = 1$ means that the node v has no special standing in the network.

Our temporal version of Bonacich $(\alpha, \beta)-$centrality is given as a function *bonacich* and is described in Algorithm 6. The parameters a and b in the algorithm correspond to the parameters α and β from the definition, respectively. We introduce an auxiliary variable $normB$ that tells whether the solution is normalized in a way we described above (we do that in line 10) or not. We compute the temporal version of the statements $b_1 = a\mathbf{A}\mathbf{e}$ in line 6 and $\mathbf{B} = \mathbf{I} - b\mathbf{A}$ in line 7. We use Jacobi's iteration with the initial approximation of all (temporal) ones to compute the solution to the equation.

Algorithm 6 Temporal Bonacich $(\alpha, \beta)-$centrality.

function *bonacich*$(A, b, a = None)$
  $normB =$ **False**
  if $a =$ **None then**
    $a = 1$
    $normB =$ **True**
  $b1 =$ *numVecProd*$([(1, \infty, a)],$ *MatVecRight*$(A,$ *VecConst*$(len(A))))$
  $B =$ *MatSum*$($*MatEye*$(len(A)),$ *numMatProd*$([(1, \infty, -b)], A))$
  $(x, conv) =$ *jacobi*$(B, b1,$ *VecConst*$(len(A)))$
  if $normB$ **then**
    $x =$ *numVecProd*$([(1, \infty, \sqrt{len(A)})],$ *normalize*$(x))$
  return $(x, conv)$

4.4 Hubs and Authorities

This centrality measure is motivated by the problem of searching the Web but its use is not limitted to text search networks. It is useful in arbitrary networks, especially those that present data with some duality of actor roles (for example, agressors and victims, bidders and recipients, providers and consumers, etc.). At the time when it first appeared, search engines relied on indexing the Web and creating a structured collection of the indexed pages. The problem was the fast growth of the Internet. Because of the enormous size of the network, text-based searching became slow and inefficient. The idea was to use the structure of the hyperlink network to infer the importance of the page from its connections to other pages on the Internet – more relevant pages will be pointed at by many other pages. But the simple indegree measure does not discriminate between the relevant pages for the query and the universally popular pages. Human judgement of relevance is in some way underlying the network structure. The creator of the page v inferred some authority on the page u when he included the link to u on his page. Kleinberg [19] defined two roles of Web pages – hubs and authorities. The idea behind the HITS algorithm for computing hubs and authorities is that inlinks endorse the importance of a page – the page referred to by many other pages is preferred by many (such pages are authorities for a given query). But also, there exist pages that compile lists of relevant resources (these are hubs for a given query). If a page lists a high number of relevant sources it should score high. Good hubs point to good authorities and good authorities are pointed at by good hubs. A page gets authority ranking from the hub rankings of the pages pointing to it, and gets a hub ranking from the authority rankings of the pages it points to. Kleinberg defined the authority update rule and the hub update rule. Both scores are applied iteratively. For an overview, see also [22].

The algorithm operates on focused subnetworks of the Web that are constructed from the output of a text-based search engine. We will not deal with the construction of such a subnetwork and will assume that it is given. We denote its adjacency matrix by $\mathbf{A}$. To each node v of a network (the node represents a Web page) two scores are assigned: the hub score x_v and the authority score y_v. The scores are stored in two distinct vectors. We get

coupled relations

$$\lambda y_v = \sum_{u:u\to v} x_u = \sum_u a_{uv}x_u = (\mathbf{A}^T x)_v,$$
$$\mu x_v = \sum_{u:v\to u} y_u = \sum_u a_{vu}y_u = (\mathbf{A}y)_v,$$

which can be rewritten in matrix notation as

$$\lambda\mu x = \mathbf{A}\mathbf{A}^T x, \quad \lambda\mu y = \mathbf{A}^T\mathbf{A}y.$$

This means that the hub and authority scores are just the elements of the dominant eigenvectors of the matrices $\mathbf{A}\mathbf{A}^T$ and $\mathbf{A}^T\mathbf{A}$, respectively. Our version of the HITS algorithm for temporal networks is given in Algorithm 8.

For the computation of the eigensystem of $\mathbf{A}^T\mathbf{A}$ we implemented a more efficient algorithm that computes the eigensystem directly, without computing the product $\mathbf{A}^T\mathbf{A}$. It is implemented as a function *singTemp* and is written in Algorithm 7. The algorithm is similar to Algorithm 1, the difference is in lines 6 and 11, where we multiply with $\mathbf{A}^T$, using the function *MatTransVecRight*.

Algorithm 7 Power iteration for computing the eigenvalues of $\mathbf{A}^T\mathbf{A}$.

```
1:  function singTemp(A, x, tol = 10^-6, maxIter = 100)
2:      i = 0
3:      convergence = False
4:      while i < maxIter do
5:          x_old = x
6:          x = normalize(MatTransVecRight(A, MatVecRight(A, x)))
7:          if test_dif(x, x_old) < tol then
8:              convergence = True
9:              break
10:         i = i + 1
11:     ev = scalProd(x, MatTransVecRight(A, MatVecRight(A, x)))
12:     return (x, ev, convergence)
```

We compute the hubs and authorities scores in Algorithm 8 by first computing the eigensystem of the matrix $\mathbf{A}^T\mathbf{A}$ in line 2, using the initial approximation of ones, from which we get the authority scores vector y. In line 3, we compute the hub scores vector x from y. In lines 4-6 we scale them according to the eigenvalue.

4.5 PageRank

PageRank is the centrality measure used by Google to rank Web pages. Because of the success of Google there is a lot of literature on PageRank, see for example [11], [12], [6], [17], [20] and [22]. Brin and Page first described the calculation of PageRank in their original paper [11].

The PageRank algorithm can be viewed in two different ways – as a random walk on a graph and as an eigenvector of a network matrix. We briefly explain both and compute the

Algorithm 8 Hubs and authorities (HITS algorithm).

1: **function** *hits*(A)
2: $(y, evy, conv) =$ *singTemp*(A, *VecConst*($len(A)$))
3: $x =$ *normalize*(*MatVecRight*(A, y))
4: $evInv =$ *numInv*(evy)
5: $y =$ *numVecProd*($evInv, y$)
6: $x =$ *numVecProd*($evInv, x$)
7: **return** $(x, y, conv)$ ▷ x is the hub scores vector and y is the authority scores vector

PageRank using the eigenvector. Due to the size of the Internet, the random walk version is used in practice.

A random walk is a stationary process on any undirected graph. The centrality of a node derived from a random walk is defined as the number of times that the walker stops at the node in the random process. In directed graphs, the process may not be stationary as the nodes with zero outdegree (dangling ends) act as sinks for the process. Once we get to a node with a zero outdegree we cannot leave it. To make the process stationary, the random walker is given the opportunity to leave a dangling end.

The random walker of PageRank simulates the behaviour of a user browsing the Internet. Most of the time, the user is clicking links on the pages (is surfing), but sometimes he types an URL (jumps). These jumps are added to the random walk in the model. They occur with a probability q and take the simulated user to a random page. The process is described by a simple set of relations

$$p_v = \frac{q}{n} + (1-q) \sum_{u:u\to v} \frac{p_u}{\text{outdeg}(u)},\ v = 1, 2, \ldots, n, \tag{4.6}$$

where n is the number of nodes, p_v is the PageRank value of the node v, and outdeg(u) is the outdegree of the node u. The sum runs over all the nodes incoming to v.

Typically, the probability of jumps is chosen as $q = 0.15$. Small values of q preserve the information about the network connections better. When $q = 0$ the process may not be stationary and PageRank is ill-defined. When $q = 1$ the jumps dominate and all the nodes have the same PageRank value equal to $\frac{1}{n}$.

For the (equivalent) matrix version of PageRank: Let $\mathbf{A}$ be the adjacency matrix of the network and let $\mathbf{D}$ be the diagonal matrix of outdegrees so that the scaled matrix $\mathbf{S} = \mathbf{D}^{-1}\mathbf{A}$ has row sums equal to 1. When a page v has no outgoing links the row sum corresponding to v in $\mathbf{A}$ is equal to zero and we cannot compute the corresponding row of the matrix $\mathbf{S}$. In this case, we take $\mathbf{S}_{vu} = \frac{1}{n}$ for all u. We construct the matrix $\mathbf{M}$ as

$$\mathbf{M} = \frac{q}{n}\ \mathbf{1} + (1-q)\ \mathbf{S}, \tag{4.7}$$

where $\mathbf{1}$ is a temporal matrix of all (temporal) ones. This matrix is positive and has a unique normed positive left eigenvector x, so that $x\mathbf{M} = x$. The PageRank of a node v is the value of x_v.

The version of PageRank for temporal networks is given as Algorithm 9. In line 4 we compute the vector of outdegrees and in lines 5-7 we compute the matrix $\mathbf{S}$. In line 5, we use the function *vecInvPR* that returns a vector of the inverses of the degrees or, when the

degree is undefined (zero), the vector with the value $[(1, \infty, \frac{1}{n})]$ and the matrix, which has elements $[(1, \infty, 1)]$ in the rows that correspond to the nodes with zero outdegree. Line 6 basically changes the original matrix so that the rows corresponding to the nodes with zero outdegree contain all ones. Line 7 scales the matrix according to outdegrees. We do this using a special function *DiagMatProd*(x, A) instead of full matrix multiplication to make the algorithm more efficient. This function computes the product of a matrix that has the vector x on the diagonal and the matrix A. In line 8 we compute $\mathbf{M}$, creating a matrix with all values equal to $[(1, \infty, \frac{1}{n})]$ using the function *constantMat*. We compute the left eigenvector of $\mathbf{M}$ as a right eigenvector of $\mathbf{M}^T$. Finally, we normalize the result. The function *norm1* normalizes the vector using the first norm, meaning that the sum of the vector components is equal to 1 at all times.

Algorithm 9 The temporal PageRank algorithm.

1: **function** *pageRank*$(A, q = 0.15)$
2: $n = len(A)$
3: $S = n \times n$ temporal matrix
4: $s =$ *MatVecRight*$(A,$ *VecConst*$(n))$ ▷ vector of outdegrees
5: $(S, s) =$ *vecInvPR*(S, s)
6: $S =$ *MatSum*(A, S)
7: $S =$ *DiagMatProd*(s, S)
8: $M =$ *MatSum*(*numMatProd*$([(1, \infty, 1 - q)], S)$, *numMatProd*$([(1, \infty, q)],$ *constantMat*$(n, [(1, \infty, 1/n)]))$)
9: $(x, ev, conv) =$ *eigTemp*(*MatTrans*(M), *VecConst*(n))
10: $x =$ *norm1*(x)
11: **return** $(x, conv)$

Corollary 4.7. *The temporal pageRank algorithm always converges.*

Proof. The matrix $\mathbf{M}$ from the equation (4.7) is positive and has a unique eigenvalue that has the strictly largest absolute value by the Theorem 2.1. Therefore the temporal power iteration converges by the Theorem 4.1. □

4.6 A Note on the Time Complexity of the Algorithms

We use n for the number of nodes of the given network, m for the number of arcs, and k for the number of iterations of the iterative algorithms (Algorithms 1, 3and 7). Because of the assumption that $\mathcal{T} \subseteq \mathbb{N}$, the length of the temporal quantities describing the network vectors and matrices is bounded with the lifetime of the network. We denote the lifetime with L. The underlying semiring is plain floating point numbers field so the time complexity of the operations is $\mathcal{O}(1)$.

We showed in [2], that the addition and the multiplication of temporal quantities have the time complexity of $\mathcal{O}(L)$. Therefore the complexity of the multiplication of two temporal vectors is $\mathcal{O}(nL)$, the complexity of the multiplication of a temporal matrix with a temporal vector is $\mathcal{O}(n^2L)$ and the complexity of the multiplication of two temporal matrices is $\mathcal{O}(n^3L)$.

From this, it follows that all the algorithms we proposed have a time complexity of $\mathcal{O}(kn^2L)$. The time complexity of Algorithm 1 follows from the complexities of the operations in the temporal quantities semiring. The functions of Algorithm 2 have the same

complexity, as *eigTemp* is the major part of them. This is also true for Algorithm 9. (Note that the matrix product in line 7 would have the time complexity of $\mathcal{O}(n^3L)$ if we computed the full matrices.) Algorithm 3 also has a time complexity of $\mathcal{O}(kn^2L)$, the major part is line 8. We use the function *jacobi* in line 14 of Algorithm 4, in line 4 of Algorithm 5 and in line 8 of Algorithm 6. It is the major part of the computation in all cases, so these algorithms have the same complexity. The computation of the singular values in Algorithm 7 also has this complexity with our implementation (note that if we were to compute the matrix product and compute its the eigenvalues, the complexity would be $\mathcal{O}(kn^3L)$). We use the results as a major part of Algorithm 8, again of the same complexity.

5 Examples of Spectral Centralities in Temporal Networks

5.1 Spectral centrality measures – test case

We will test our algorithms on the temporal network from Figure 1. The network changes are outlined with the weights on the arcs and with dotted arcs as follows:

The full arcs are present through all of the network lifetime, that is in the time interval $[1, 9)$. In the time intervals $[1, 3) \cup [7, 9)$ the weight of these arcs is equal to one, on the interval $[3, 7)$, the weight is equal to the number written on the arc (note that some of the weights remain 1). The dashed arcs are present only in the time interval $[5, 9)$. In the interval $[5, 7)$ the weight on the arc is equal to the number on the arc in the figure, in the interval $[7, 9)$ all the weights are equal to 1.

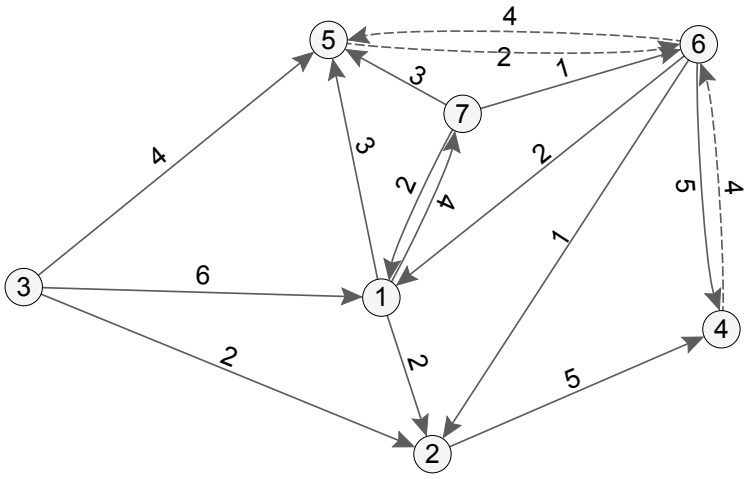

Figure 1: Test temporal graph.

The temporal vectors describing the centrality measures from Section 4 for the test graph are too long to be written in full. The changes in the standings of the nodes that are usually what interests us are written in Table 1. From the Table, we can see that some centrality measures remain undefined for certain nodes in some time intervals. For example, the nodes 2,4,5 in the time interval $[1, 3)$ are missing in the row, corresponding to out-eig. That can also be seen from the Figure, as the nodes 4 and 5 have outdegree equal to zero in

	time 1–3	time 3–5	5–7	7–9
in-eig	4,5,2,1,7,6	5,4,7,1,2,6	4,6,5,1,2,7	6,5,4,2,1,7
out-eig	7,1,3,6	3,1,7,6	6,3,1,4,7,2,5	7,6,1,3,4,5,2
Katz $a = 0.15$	2,5,1,4,7,6	5,4,7,1,2,6	5,6,4,1,7,2	5,6,2,1,4,7
Bonacich $\alpha = 0.85$	1,2,5,4,6,7,3	1,4,5,2,7,6,3	5,1,4,6,2,7,3	5,1,2,6,4,7,3
Bonacich $\beta = 0.15$	7,1,3,6,2	3,1,6,7,2	3,6,1,4,7,2,5	6,7,1,3,4,5,2
hub	3,6,1,7,2	3,7,6,1,2	3,6,7,1,2,4,5	6,3,7,1,2,4,5
authority	1,2,5,4,6,7	1,5,2,4,7,6	5,1,4,2,7,6	5,1,2,4,6,7
pageRank $q = 0.15$	4,2,5,1,7,6,3	4,5,7,1,2,6,3	6,4,5,1,7,2,3	6,4,5,2,1,7,3

Table 1: The order of the nodes of the test graph by their centralities through time.

this interval and the node 2 only points to 4, which has zero centrality.

The second interesting thing is that all the centrality measures return similar results, if we put them into two groups: One group chooses nodes that are central as the ones that have "more inlinks" (in-eig, Katz, α centrality, authority score), the other group chooses the nodes that have "more outlinks" (out-eig, $(\alpha, \beta)-$ centrality, hub score, pageRank).

5.2 Franzosi's violence network

We applied our algorithms to compute the centrality scores of the nodes in Franzosi's violence temporal network [15]. From the newspapers in the period from January 1919 to December 1922, Roberto Franzosi collected data about the reported violent actions – interactions between different political groups and other groups of people in Italy. The network nodes represent the involved groups of people (for example, "people", "police", "fascists", "communists", "socialists", "workers") and the arc weights correspond to the number of interactions between two groups (the arc (u, v) with a weight 3 would mean that the group u committed 3 violent actions on the group v). The temporal network contains data about violent activities for each month in the given time period – the temporal quantities corresponding to an arc tell the information about the violent activities for the whole 4 years.

We get the clearest results with the hub and authority scores for the nodes, which is expected because of the nature of the network – the underlying duality of the actors. The actors can be seen as the aggressors and as the groups at which the aggression was directed. For the sake of clarity, we created the timeline of changes in the highest scores. The hub scores can be seen in Figure 2. The time points are months and the heights of the symbols correspond to the value of the normalized authority score. From Figure 2 it is clearly seen that at one time the violent actions of police were replaced with that of the fascists. That happens at the time point 23 which corresponds to November 1920. From the Figure on the right, we see that for some time, the police retained some control and was second by the violent activities, but later it dissapeared altogether.

The authority scores are outlined in Figure 3. There is no clear trend and it seems that the violent activities were not limited to one particular group through time which is in

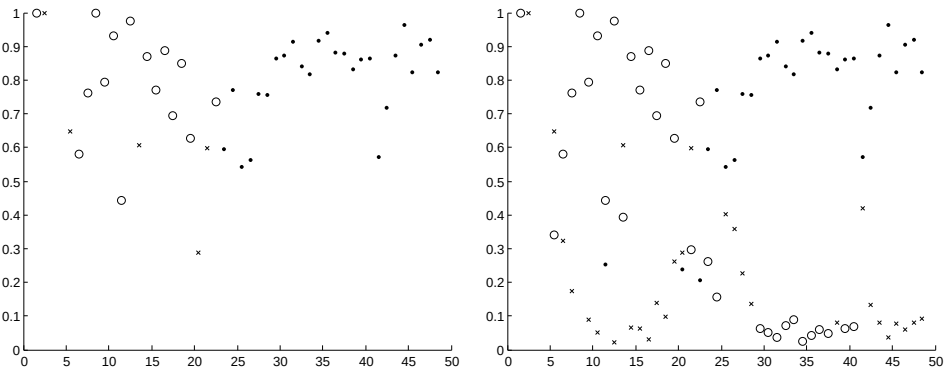

Figure 2: The highest hub score of the Franzosi's network through time (left) and the two highest scores (right). The "fascists" are the black dots, the "police" are circles, others are crosses.

accordance with our intuition.

With the other centrality scores the results are similar, but the boundary is not that obvious. For eigenvector in-centrality we get 24 counts of "workers," "workers (agricultural)" or "socialists," and 14 counts of"undefined," "people" or "protesters." There are three others (once "police" and two times "fascists").

The eigenvector out-centrality returns a mix of "police" (4), "protesters" (3), "?" (3), "undefined" (2), "workers" (2), "workers (agricultural)" (1) and "republicans" (1). The first appearance of "fascists" is at the time point 23 (November 1920). The fascists have the highest centrality score until the end od the timeline, except for 4 instances ("the right", "?", "workers", "police").

The pageRank centrality for $q = 0.15$ gives us 18 counts of "fascists", starting from the time point 22 (October 1920), which is then interrupted with "workers" (3), "people" (3), "undefined" (2) and "police." Until that time, we have a mix of "police" (6), "undefined" (5), "people" (4), "socialists" (3), "war affected" and "the right."

As it seems that the aggressor is more distinct than the groups that were targeted, we computed the Katz and the $\alpha-$centrality measure on the transpose of the original matrix.

The Bonacich $\alpha-$centrality for $\alpha = 0.9$ returns 17 counts of "police" until the time point 23 (others with the maximal centrality score until this time are "thugs," "undefined," and twice "workers"). After the time point 23, we have 23 counts of "fascists," others are "thugs," "police," and twice "workers."

The Katz centrality measure has 16 counts of "police," and one appearance of "thugs," "undefined," "protesters" and "?". After the time point 23 the "fascists" are the only ones with the highest centrality score. The Bonacich $(\alpha, \beta)-$centrality returns the same score.

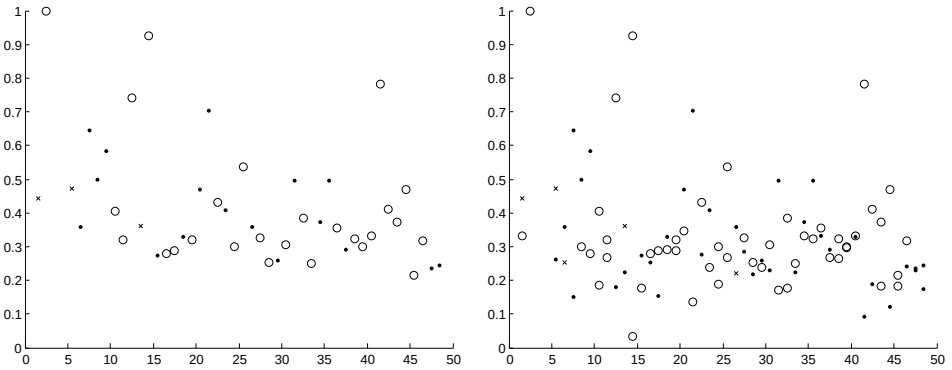

Figure 3: The highest authority score of the Franzosi's network through time (left) and the two highest scores (right). The "workers," "workers (agricultural)" and "socialists" are the black dots, the "undefined," "people" and "protesters" are circles, and other groups are crosses.

These results are summarized in Table 2 in which we have written the count of "police," "fascists," and others with the maximum value of centrality for different centrality measures, divided into two columns – the first for the count before November 1920, the second after that. We only do this for the centrality measures that correspond to the aggressor. From the table, we can see that the fascist aggression was central in the studied news after November 1920 in all cases. Because of the undefined values on some intervals the number of data in the columns varies.

group of people	hub score		pageRank		out-eig		α		α, β		Katz	
police	15	0	6	1	4	1	17	1	16	0	16	0
fascists	0	26	1	18	0	21	0	23	0	25	0	25
other	5	0	14	8	12	3	4	3	4	0	4	0

Table 2: The summary of the maximum centrality scores before and after November 1920 for the Franzosi's violence network.

5.3 9/11/2001 Reuters terror news network

The Reuters terror news network about the 9/11 attack on the United States was obtained from the CRA (Centering Resonance Analysis) networks created by Steve Corman and Kevin Dooley at Arizona State University [13] and was used as a case network for the

Viszards visualization session on the Sunbelt XXII International Sunbelt Social Network Conference, New Orleans, USA, 13-17. February 2002.

The network is based on the September 11 attack news that were released by the news agency Reuters during the 66 consecutive days after the attack. The nodes of the network are words and the edges tell whether the two words appear in the same news sentence. The weight of the edge is the frequency of these common appearances. The network has $n = 13332$ nodes (different words) and $m = 243447$ edges, of which 50859 have weights larger than 1. There are no loops in the network. We extracted a subnetwork of the 50 most active nodes as in [2]. We tested our algorithms on this smaller network.

The methods *inEig* and *outEig* do not converge with the initial approximation vector of all temporal ones. Also, the PageRank ranking of nodes tells almost nothing about the importance of nodes as it jumps around – in value as well as in the node with the highest value of centrality.

The other methods are twofold: The first group corresponds to the question "Which words are the news pointing at the most? What's the end-game?" All methods return the terms "attack," "afghanistan," and "anthrax" as the most frequent terms with the highest value of centrality. The methods belonging to this group are the Katz centrality index computed on the transposed adjacency matrix, Kleinberg's hub score, Bonacich α centrality on the transposed matrix, and Bonacich $(\alpha, \beta)-$centrality. The value of the maximal centrality is getting smaller as the time increases.

The second group answers to the question "From which words do the news spread? What started it all?" and all the centrality measures have the most frequent term "united_states," except for the first week after the attack during which the term with the highest centrality is "world_trade_ctr." The methods belonging to this group are the Katz centrality index, Kleinberg's authority score, Bonacich α centrality, and Bonacich $(\alpha, \beta)-$centrality computed on the transposed adjacency matrix.

We list the count of the terms with the highest α centrality (for the transposed matrix) through time as an example of the first group: 50 times "attack," 10 times "afghanistan," 4 times "anthrax" and once "leader."

As an example of the second group, we list the count of the terms that have the highest Katz centrality measure through time: 49 times "united_states," 7 times "world_trade_ctr," 4 times "washington," 2 times "taliban" and "war," once "world" and "wednesday."

6 Conclusions and Future Work

In the article, we show that spectral centrality measures can be extended to the analysis of temporal networks with zero latency described with temporal quantities. In the application we are using only the combinatorial semiring, but the underlying linear algebra could be extended to other semirings in the future, providing some reasonable motivation is found. Also, the meaning of non-dominant eigenvalues and/or eigenvectors could be explored. With the theory of perturbations of eigenvectors, we feel that it would be possible to continue this research to predict the changes in the standing of the nodes in the network for the near future.

Algorithms for the efficient computation of eigenvalues and for solving linear systems in other semirings could be developed. The problem is that, in semirings, the inverse is not necessarily available. There has been some research on this topic which we have not approached yet.

Methods for the visualisation of temporal networks and for the visualisation of the changes in the node importance through time should be developed.

Our current representation is based on the network matrix, which means that it is not very efficient for large sparse networks. In the future, data structures for the representation of sparse temporal networks could be studied and implemented.

References

[1] V. Batagelj, Semirings for social network analysis, *Journal of Mathematical Sociology* **19(1)** (1994), 53–68.

[2] V. Batagelj and S. Praprotnik, An algebraic approach to temporal network analysis, *Submitted, preprint available at* `http://arxiv.org/abs/1505.01569` (2014).

[3] A. Bavelas, A mathematical model of group structure, *Human Organizations* **7** (1948), 16–30.

[4] A. Bavelas and D. Barrett, *An Experimental Approach to Organizational Communication*, Publications: Industrial Relations, American Management Association, 1951.

[5] A. Berman and R. J. Plemmons, *Nonnegative Matrices in the Mathematical Sciences (Classics in Applied Mathematics)*, Society for Industrial Mathematics, January 1987.

[6] M. Bianchini, M. Gori and F. Scarselli, Inside pagerank, *ACM Trans. Internet Technol.* **5** (2005), 92–128, doi:10.1145/1052934.1052938.

[7] P. Boldi and S. Vigna, Axioms for centrality, *Internet Math.* **10(3-4)** (2014), 222–262.

[8] P. Bonacich, Factoring and weighting approaches to status scores and clique identification, *Journal of Mathematical Sociology* **2** (1972), 113–120.

[9] P. Bonacich, Power and centrality: A family of measures, *American Journal of Sociology* **92** (1987), 1170–1182.

[10] P. Bonacich and P. Lloyd, Eigenvector-like measures of centrality for asymmetric relations, *Social Networks* **23** (2001), 191–201, doi:10.1016/S0378-8733(01)00038-7.

[11] S. Brin and L. Page, The anatomy of a large-scale hypertextual web search engine, in: *Computer Networks and ISDN Systems*, Elsevier Science Publishers B. V., 1998 pp. 107–117.

[12] K. Bryan and T. Leise, The $25,000,000,000 eigenvector: the linear algebra behind google, *SIAM Review* **48** (2006), 569–581.

[13] S. R. Corman, T. Kuhn, R. D. McPhee and K. J. Dooley, Studying complex discursive systems, *Human communication research* **28** (2002), 157–206.

[14] J. W. Demmel, *Applied Numerical Linear Algebra*, Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1997.

[15] R. Franzosi, Mobilization and counter-mobilization processes: From the "red years"(1919–20) to the "black years" (1921–22) in Italy. A new methodological approach to the study of narrative data., *Theory and Society* **26** (1997), 275–304.

[16] M. Gondran and M. Minoux, *Graphs, Dioids and Semirings: New Models and Algorithms (Operations Research/Computer Science Interfaces Series)*, Springer Publishing Company, Incorporated, 1st edition, 2008.

[17] T. Haveliwala and S. Kamvar, *The Second Eigenvalue of the Google Matrix*, Technical Report 2003-20, Stanford InfoLab, 2003.

[18] L. Katz, A new status index derived from sociometric analysis, *Psychometrika* **18** (1953), 39–43.

[19] J. M. Kleinberg, Authoritative sources in a hyperlinked environment, *J. ACM* **46** (1999), 604–632, doi:10.1145/324133.324140.

[20] A. N. Langville and C. D. Meyer, A survey of eigenvector methods for web information retrieval, *SIAM Rev.* **47** (2005), 135–161, doi:10.1137/S0036144503424786.

[21] H. J. Leavitt, Some effects of certain communication patterns on group performance, *Journal of Abnormal and Social Psychology* **46** (1951), 38–50.

[22] N. Perra and S. Fortunato, Spectral centrality measures in complex networks, *Physical Review E (Statistical, Nonlinear, and Soft Matter Physics)* **78** (2008), 036107+, doi:10.1103/physreve.78.036107.

[23] S. Wasserman and K. Faust, *Social network analysis: Methods and applications*, Cambridge University Press, 1994.