ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/275974124

An algebraic approach to temporal network analysis based on temporal quantities

Article · May 2015

DOI: 10.1007/s13278-016-0330-4 · Source: arXiv

CITATIONS
11

READS
150

2 authors, including:

Vladimir Batagelj
IMFM, Ljubljana and UP IAM, Koper

206 PUBLICATIONS **7,573** CITATIONS

SEE PROFILE

Some of the authors of this publication are also working on these related projects:

Project Network visualization View project

Project Symbolic data analysis View project

All content following this page was uploaded by Vladimir Batagelj on 14 May 2015.

The user has requested enhancement of the downloaded file.

arXiv:1505.01569v1 [cs.SI] 7 May 2015

An algebraic approach to temporal network analysis based on temporal quantities

Vladimir Batagelj Selena Praprotnik

Abstract

In a temporal network, the presence and activity of nodes and links can change through time. To describe temporal networks we introduce the notion of temporal quantities. We define the addition and multiplication of temporal quantities in a way that can be used for the definition of addition and multiplication of temporal networks. The corresponding algebraic structures are semirings. The usual approach to (data) analysis of temporal networks is to transform it into a sequence of time slices – static networks corresponding to selected time intervals and analyze each of them using standard methods to produce a sequence of results. The approach proposed in this paper enables us to compute these results directly. We developed fast algorithms for the proposed operations. They are available as an open source Python library TQ (Temporal Quantities) and a program Ianus. The proposed approach enables us to treat as temporal quantities also other network characteristics such as degrees, connectivity components, centrality measures, Pathfinder skeleton, etc. To illustrate the developed tools we present some results from the analysis of Franzosi's violence network and Corman's Reuters terror news network.

Keywords: temporal network; time slice; temporal quantity; semiring; algorithm; network measures; Python library; violence; terror.
MSC(2010): 91D30; 16Y60; 90B10; 68R10; 93C55

1 Introduction

In a *temporal network*, the presence and activity of nodes and links can change through time. In the last two decades the interest for the analysis of temporal networks increased partially motivated by travel-support services and the analysis of sequences of interaction events (e-mails, news, phone calls, collaboration, etc.). The approaches and results were recently surveyed by Holme and Saramäki in their paper (Holme and Saramäki, 2012) and the book (Holme and Saramäki, 2013).

Most of temporal social networks data contain the information about activity time intervals of their links, sometimes augmented by the activity intensity. The usual approach to the (data) analysis of temporal networks is to transform it into a sequence of time slices – static networks corresponding to selected time intervals – see for example Moody et al. (2005); Kim, Yoon, and Crowfort (2012); Gulys et al. (2013). Afterward each time slice is analyzed using the standard methods for analysis of static networks. Finally the results are collected into a temporal sequence of results. In this paper we propose an alternative approach, based on the notion of temporal quantity, that bypasses explicit construction of time slices. The developed algorithms are transforming temporal networks directly into results in the form of temporal quantities, vectors, temporal vectors or partitions, and temporal networks.

In the paper, we first present the basic notions about temporal networks. In Section 3 we introduce the temporal quantities and propose an algebraic approach, based on semirings, to the analysis of temporal networks. In the following sections we show that most of the traditional network analysis concepts and algorithms such as degrees, clustering coefficient, closeness, betweenness, weak and strong connectivity, PathFinder skeleton, etc. can be straightforwardly extended to their temporal versions.

2 Description of temporal networks

For the description of temporal networks we propose an elaborated version of the approach used in Pajek (de Nooy, Mrvar, and Batagelj, 2012). In our approach we also consider values of links (in most cases measuring the intensity/frequency of the activity). Pajek supports two types of descriptions of temporal networks based on *presence* and on *events* (Pajek 0.47, July 1999). Here, we will describe only the approach to capturing the presence of nodes and links.

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, where $\mathcal{T}$ is a set of *time points*, $t \in \mathcal{T}$. $\mathcal{V}$ is the set of nodes, $\mathcal{L}$ is the set of links, $\mathcal{P}$ is the set of node properties, and $\mathcal{W}$ is the set of link properties or weights (Batagelj, 2009). The time $\mathcal{T}$ is usually either a subset of integers, $\mathcal{T} \subseteq \mathbb{Z}$, or a subset of reals, $\mathcal{T} \subseteq \mathbb{R}$. In Pajek $\mathcal{T} \subseteq \mathbb{N}$. In a general setting it could be any linearly ordered set.

In a temporal network, nodes $v \in \mathcal{V}$ and links $l \in \mathcal{L}$ are not necessarily present or active at all time points. Let $T(v)$, $T \in \mathcal{P}$, be the activity set of time points for the node v; and $T(l)$, $T \in \mathcal{W}$, the activity set of time points for the link l. The following *consistency* condition is imposed: If a link $l(u, v)$ is active at the time point t then its end-nodes u and v should be active at the time t. Formally we express this by

$$T(l(u,v)) \subseteq T(u) \cap T(v).$$

The activity set $T(e)$ of a node/link e is usually described as a sequence of activity time intervals $([s_i, f_i))_{i=1}^{k}$, where s_i is the *start*ing time and f_i is the *finish*ing time.

We denote a network consisting of links and nodes active in the time $t \in \mathcal{T}$ by $\mathcal{N}(t)$ and call it the (network) *time slice* or *footprint* of t. Let $\mathcal{T}' \subset \mathcal{T}$ (for example, a time interval). The notion of a time slice is extended to $\mathcal{T}'$ by

$$\mathcal{N}(\mathcal{T}') = \bigcup_{t \in \mathcal{T}'} \mathcal{N}(t).$$

2.1 Examples

Let us look at some examples of temporal networks.

Citation networks can be obtained from bibliographic data bases such as Web of Science (Knowledge) and Scopus. In a citation network $\mathcal{N} = (\mathcal{V}, \mathcal{L}, \mathcal{T}, \mathcal{P}, \mathcal{W})$, its set of nodes $\mathcal{V}$ consists of selected works (papers, books, reports, patents, etc.). There exists an arc $a(u, v) \in \mathcal{L}$ iff the work u cites the work v. The time set $\mathcal{T}$ is usually an interval of years $[\mathit{year}_{\mathit{first}}, \mathit{year}_{\mathit{last}}]$ in which the works were published. The activity set of the work v, $T(v)$, is the interval $[\mathit{year}_{\mathit{pub}}(v), \mathit{year}_{\mathit{last}}]$; and the activity set of the arc $a(u, v)$, $T(a)$, can be set to the interval $[\mathit{year}_{\mathit{pub}}(u), \mathit{year}_{\mathit{pub}}(u)]$ (instances approach) or to the interval $[\mathit{year}_{\mathit{pub}}(u), \mathit{year}_{\mathit{last}}]$ (cumulative approach). An example of a property $p \in \mathcal{P}$ is the number of pages or the number of authors. Other properties, such as work's authors and keywords, are usually represented as two-mode networks.

Project collaboration networks are usually based on some project data base such as Cordis. The set of nodes $\mathcal{V}$ consists of participating institutions. There is an edge $e(u{:}\, v) \in \mathcal{L}$ iff institutions u and v work on a joint project. The time set $\mathcal{T}$ is an interval of dates/days $[\mathit{day}_{\mathit{first}}, \mathit{day}_{\mathit{last}}]$ in which the collaboration data were collected. $T(v) = \mathcal{T}$ and $T(e) = \{[s, f]$: there exists a project P such that u and v are partners on P; s is the start and f is the finish date of $P\}$.

KEDS/WEIS networks are networks registering political events in critical regions in the world (Middle East, Balkans, and West Africa) on the basis of daily news. Originally they were collected by KEDS (Kansas Event Data System). Currently they are hosted by Parus Analytical Systems. The set of nodes $\mathcal{V}$ contains the involved actors (states, political groups, international organizations, etc.). The links are directed and are describing the events:

$$(date, actor_1, actor_2, action)$$

on a given $date$ the $actor_1$ made the $action$ on the $actor_2$. Different actions are determining different relations – we get a multirelational network with a set of links partitioned by actions $\mathcal{L} = \{\mathcal{L}_a : a \in \mathit{Actions}\}$. The time set is determined by the observed period $\mathcal{T} = [day_{\mathit{first}}, day_{\mathit{last}}]$. Since most of the actors are existing during all the observed period their node activity time sets are $T(v) = \mathcal{T}$. Another option is to consider as their node activity time sets the period of their engagement in the region. The activity time set $T(l)$ of an arc $l(u, v) \in \mathcal{L}_a$ contains all dates – intervals $[day, day + 1)$ – in which the actor u made an action a on the actor v. Another possibility is to base the description on a single relation network and store the information about the action a as a structured value in a triple $(day, day + 1, value)$

$$value = [(action_1, count_1), (action_2, count_2), \ldots, (action_k, count_k)]$$

and introduce an appropriate semiring over such values (see Section 3).

There are many other examples of temporal networks such as: genealogies, contact networks, networks of phone calls, etc.

3 Temporal quantities

Besides the presence/absence of nodes and links also their properties can change through time. To describe the changes we introduce the notion of a *temporal quantity* a with the activity set $T_a \subseteq \mathcal{T}$

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

where $a(t)$ is the value of a at an instant t, and ⌘ denotes the value *undefined*.

We assume that the values of temporal quantities belong to a set A which is a semiring $(A, \oplus, \odot, 0, 1)$ for binary operations $\oplus : A \times A \to A$ and $\odot : A \times A \to A$ (Gondran and

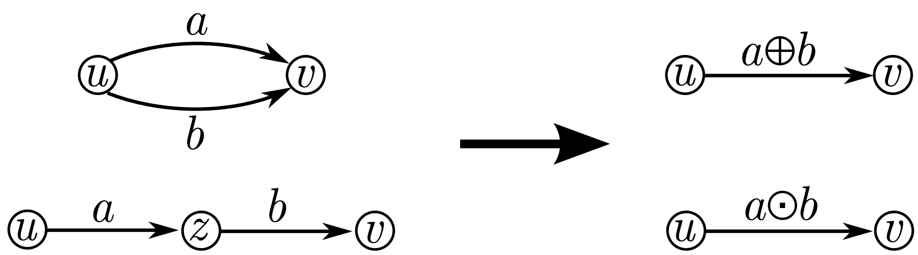

Figure 1: Semiring addition and multiplication in networks.

Minoux, 2008; Batagelj, 1994). This means that $(A, \oplus, 0)$ is an Abelian monoid – the addition $\oplus$ is associative and commutative, and has 0 as its neutral element; and $(A, \odot, 1)$ is a monoid – the multiplication $\odot$ is associative and has 1 as its neutral element. Also, multiplication distributes from both sides over the addition. Note that 0 and 1 denote the two elements of A that satisfy the required properties. In expressions the precedence of the multiplication $\odot$ over the addition $\oplus$ is assumed. We can extend both operations to the set $A_⌘ = A \cup \{⌘\}$ by requiring that for all $a \in A_⌘$ it holds

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

The structure $(A_⌘, \oplus, \odot, ⌘, 1)$ is also a semiring.

The "default" semiring is the *combinatorial* semiring $(\mathbb{R}_0^+, +, \cdot, 0, 1)$ where $+$ and $\cdot$ are the usual addition and multiplication of real numbers. In some applications other semirings are useful.

In applications of semirings in the analysis of graphs and networks the addition $\oplus$ describes the composition of values on parallel walks and the multiplication $\odot$ describes the composition of values on sequential walks – see Figure 1. For the combinatorial semiring these two schemes correspond to basic principles of combinatorics: the *Rule of Sum* and the *Rule of Product* (Riordan, 1958).

The semiring $(\overline{\mathbb{R}_0^+}, \min, +, \infty, 0)$, $\overline{\mathbb{R}_0^+} = \mathbb{R}_0^+ \cup \{\infty\}$, is suitable to deal with the shortest paths problem in networks; and the semiring $(\{0,1\}, \vee, \wedge, 0, 1)$ for reachability problems. The standard references on semirings are Carr (1979) and Gondran and Minoux (2008).

3.1 Semiring of temporal quantities

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

$$a \oplus b = s$$

as

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

and $T_s = T_a \cup T_b$; and the *product* (sequential links)

$$a \odot b = p$$

as

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

and $T_p = T_a \cap T_b$.

In these definitions and also in the following text, to avoid the 'pollution' with many different symbols, we use the symbols $\oplus$ and $\odot$ to denote the semiring operations. The appropriate semiring can be determined from the context. For example, in the definition of addition of temporal quantities the symbol $\oplus$ on the left hand side of the equation operates on temporal quantities and the symbol $\oplus$ on the right hand side denotes the addition in the basic semiring $A_\maltese$.

Let us define the temporal quantities $\mathbf{0}$ and $\mathbf{1}$ with requirements $\mathbf{0}(t) = \maltese$ and $\mathbf{1}(t) = 1$ for all $t \in \mathcal{T}$. It is a routine task to verify that the structure $(A_\maltese(\mathcal{T}), \oplus, \odot, \mathbf{0}, \mathbf{1})$ 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 $\mathbf{A} \odot \mathbf{B} = \mathbf{P}$

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

Again, the symbols $\oplus$ and $\odot$ on the left hand side operate on matrices and on the right hand side in the semiring of temporal quantities.

The matrix multiplication is closely related to traveling on networks. Consider an entry p_{ij} in an instant t

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

For a value $p_{ij}(t)$ to be defined (different from $\maltese$) there should exist in the instant t at least one node k such that both the link (i, k) and the link (k, j) exist – the transition from the node i to the node j through a node k is possible. Its contribution is $a_{ik}(t) \odot b_{kj}(t)$. This means that the matrix multiplication is taking into account only the links inside the time slice $\mathcal{N}(t)$.

3.2 Operationalization

In the following we shall limit our discussion to temporal quantities that can be described in the form of time-interval/value sequences

$$a = ((I_i, v_i))_{i=1}^{k}$$

where I_i is a time-interval and v_i is a value of a on this interval. In general, the intervals can be of different types: 1 – $[s_i, f_i]$; 2 – $[s_i, f_i)$; 3 – $(s_i, f_i]$; 4 – (s_i, f_i). Also the value v_i can be structured. For example $v_i = (w_i, c_i, \tau_i)$ – weight, capacity and transition time, or $v_i = (d_i, n_i)$

Algorithm 1 Addition of temporal quantities.

function *sum*(a, b)
  if *length*$(a) = 0$ **then return** b
  if *length*$(b) = 0$ **then return** a
  $c \leftarrow [\,]$; $(s_a, f_a, v_a) \leftarrow$ *get*(a); $(s_b, f_b, v_b) \leftarrow$ *get*(b)
  while $(s_a < \infty) \vee (s_b < \infty)$ **do**
    if $s_a < s_b$ **then**
      $s_c \leftarrow s_a$; $v_c \leftarrow v_a$
      if $s_b < f_a$ **then** $f_c \leftarrow s_b$; $s_a \leftarrow s_b$
      else $f_c \leftarrow f_a$; $(s_a, f_a, v_a) \leftarrow$ *get*(a)
    else if $s_a = s_b$ **then**
      $s_c \leftarrow s_a$; $f_c \leftarrow \min(f_a, f_b)$; $v_c \leftarrow$ *sAdd*(v_a, v_b)
      $s_a \leftarrow s_b \leftarrow f_c$; $f_d \leftarrow f_a$
      if $f_d \leq f_b$ **then** $(s_a, f_a, v_a) \leftarrow$ *get*(a)
      if $f_b \leq f_d$ **then** $(s_b, f_b, v_b) \leftarrow$ *get*(b)
    else
      $s_c \leftarrow s_b$; $v_c \leftarrow v_b$
      if $s_a < f_b$ **then** $f_c \leftarrow s_a$; $s_b \leftarrow s_a$
      else $f_c \leftarrow f_b$; $(s_b, f_b, v_b) \leftarrow$ *get*(b)
    c.*append*$((s_c, f_c, v_c))$
  return *standard*(c)

Algorithm 2 Multiplication of temporal quantities.

function *prod*(a, b)
  if *length*$(a) \cdot$ *length*$(b) = 0$ **then return** $[\,]$
  $c \leftarrow [\,]$; $(s_a, f_a, v_a) \leftarrow$ *get*(a); $(s_b, f_b, v_b) \leftarrow$ *get*(b)
  while $(s_a < \infty) \vee (s_b < \infty)$ **do**
    if $f_a \leq s_b$ **then** $(s_a, f_a, v_a) \leftarrow$ *get*(a)
    else if $f_b \leq s_a$ **then** $(s_b, f_b, v_b) \leftarrow$ *get*(b)
    else
      $s_c \leftarrow \max(s_a, s_b)$; $f_c \leftarrow \min(f_a, f_b)$; $v_c \leftarrow$ *sMul*(v_a, v_b)
      c.*append*$((s_c, f_c, v_c))$
      if $f_c = f_a$ **then** $(s_a, f_a, v_a) \leftarrow$ *get*(a)
      if $f_c = f_b$ **then** $(s_b, f_b, v_b) \leftarrow$ *get*(b)
  return *standard*(c)

– the length of geodesics and the number of geodesics, etc. We require $s_i \leq f_i$, for $i = 1, \ldots, k$ and $s_{i-1} < s_i$, for $i = 2, \ldots, k$.

To simplify the exposition we will assume in the following that all the intervals in our descriptions of temporal quantities are of type 2 – $[s_i, f_i)$ and $f_{i-1} \leq s_i$, for $i = 2, \ldots, k$.

Therefore we can describe the temporal quantities with sequences of triples

$$a = ((s_i, f_i, v_i))_{i=1}^k.$$

In the examples we will also assume that $\mathcal{T} = [t_{min}, t_{max}] \subset \mathbb{N}$.

To provide a computational support for the proposed approach we are developing in Python a library TQ (Temporal Quantities). In the examples we will use the Python notation for temporal quantities.

The following are two temporal quantities a and b represented in Python as a list of triples

```
a = [(1, 5, 2), (6, 8, 1), (11, 12, 3), (14, 16, 2),
     (17, 18, 5), (19, 20, 1)]
b = [(2, 3, 4), (4, 7, 3), (9, 10, 2), (13, 15, 5), (16, 21, 1)]
```

The temporal quantity a has on the interval $[1, 5)$ value 2, on the interval $[6, 8)$ value 1, on the interval $[11, 12)$ value 3, etc. Outside the specified intervals its value is undefined, ⌘.

The temporal quantities can also be visualized as it is shown for a and b at the top half of Figure 2.

For the simplified version of temporal quantities we wrote procedures *sum* (Algorithm 1) for the addition and *prod* (Algorithm 2) for the multiplication of temporal quantities over the selected semiring. Because, by assumption, the triples in a description of a temporal quantity are ordered by their starting times, we can base both procedures on the ordered lists merging scheme. The basic semiring operations of addition and multiplication are provided by functions *sAdd* and *sMul*.

The function *length*(a) returns the length (number of items) of the list a. The function *get*(a) returns the current item of the list a and moves to the next item; if the list is exausted it returns a 'sentinel' triple $(\infty, \infty, 0)$. The statement $(s, f, v) \leftarrow e$ describes the unpacking of the item e into its parts. The statement c.*append*(e) appends the item e to the tail of the list c. The function *standard*(a) joins, in the list a, adjacent time intervals with the same value into a single interval.

The following are the sum s and the product p of temporal quantities a and b. They are visually displayed at the bottom half of Figure 2.

```
s = [(1, 2, 2), (2, 3, 6), (3, 4, 2), (4, 5, 5), (5, 6, 3),
     (6, 7, 4), (7, 8, 1), (9, 10, 2), (11, 12, 3), (13, 14, 5),
     (14, 15, 7), (15, 16, 2), (16, 17, 1), (17, 18, 6),
     (18, 19, 1), (19, 20, 2), (20, 21, 1)]
p = [(2, 3, 8), (4, 5, 6), (6, 7, 3), (14, 15, 10), (17, 18, 5),
     (19, 20, 1)]
```

Let $l_a =$ *length*(a) and $l_b =$ *length*(b). Then, assuming that the semiring operations take constant time each, the time complexity of both algorithms is $O(l_a + l_b)$. The example in Figure 3 shows that in extreme cases the sum can be almost 4 times longer than each of its arguments, and the product almost twice as long as the arguments. If $\mathcal{T} = [t_{min}, t_{max}] \subset \mathbb{N}$ the length of a list describing a temporal quantity can not exceed $L = t_{max} - t_{min}$.

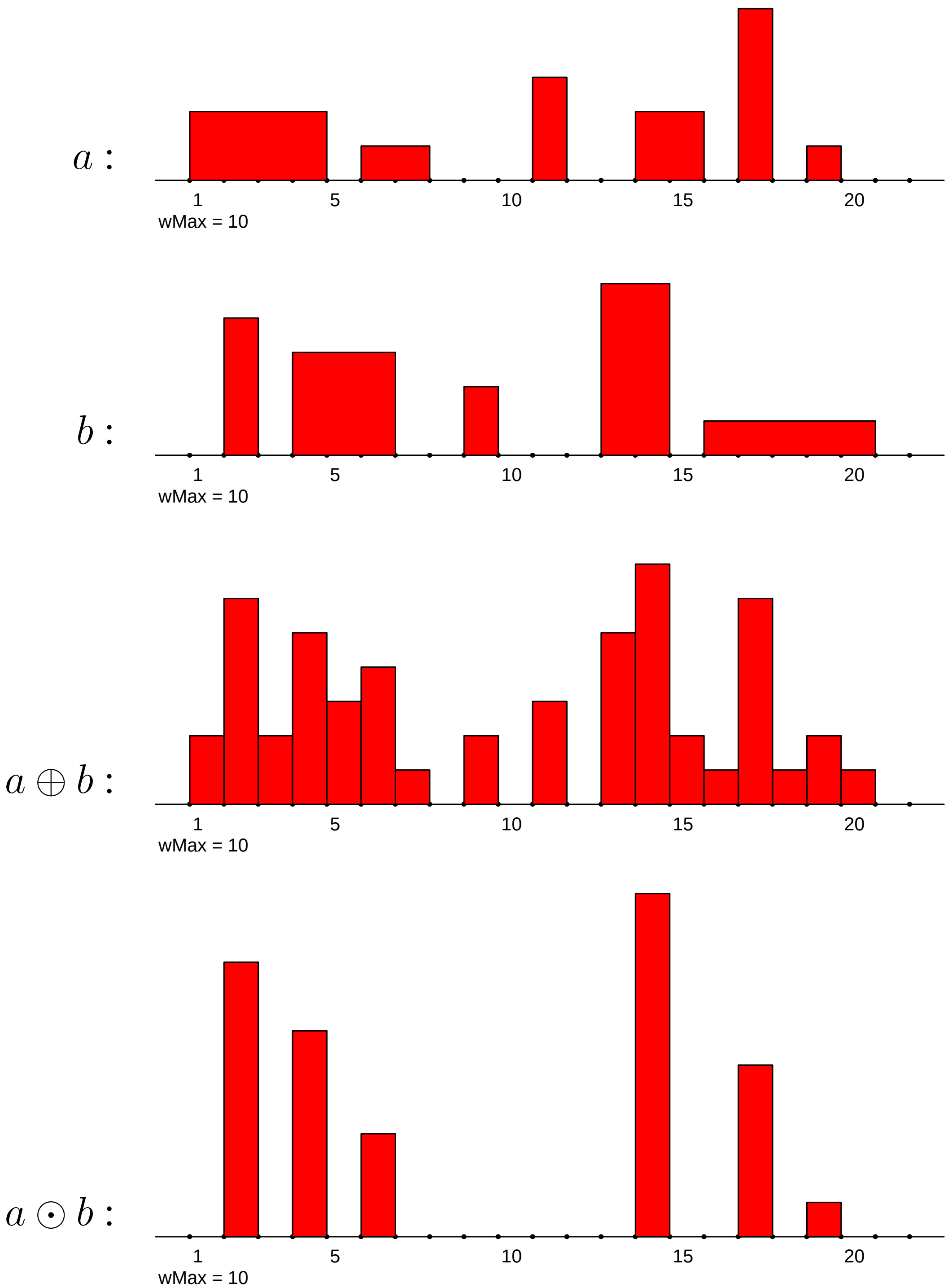

Figure 2: Addition and multiplication of temporal quantities.

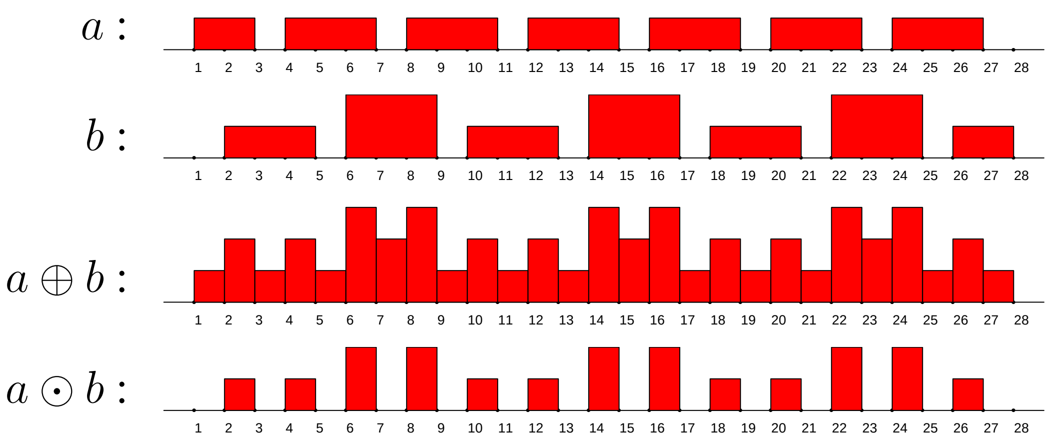

Figure 3: Addition and multiplication of temporal quantities – growth of size.

3.3 The aggregated value

In some applications over the combinatorial semiring we shall use the *aggregated value* of a temporal quantity $a = ((s_i, f_i, v_i))_{i=1}^k$. It is defined as

$$\Sigma a = \sum_{i=1}^{k} (f_i - s_i) \cdot v_i$$

and is computed using the procedure *total*. For example $\Sigma a = 23$ and $\Sigma b = 30$. Note that $\Sigma a + \Sigma b = \Sigma(a + b)$.

3.4 Temporal partitions

The description of temporal partitions has the same form as the description of temporal quantities $a = ((s_i, f_i, v_i))_{i=1}^k$. They differ only in the interpretation of values $v_i \in \mathbb{N}$. In case of partitions $v_i = j$ means that the unit described with a belongs to a class j in the time interval $[s_i, f_i)$. We shall use temporal partitions to describe connectivity components in Section 10.

We obtain a more adequate description of temporal networks by using vectors of temporal quantities (temporal vectors and temporal partitions) for describing properties of nodes and making also link weights into temporal quantities. In the current version of the library TQ we use a representation of a network $\mathcal{N}$ with its matrix $\mathbf{A} = [a_{uv}]$

$$a_{uv} = \begin{cases} w(u,v) & (u,v) \in \mathcal{L} \\ \text{⌘} & \text{otherwise} \end{cases}$$

where $w(u,v)$ is a temporal weight attached to a link (u,v).

3.5 Products of a temporal matrix and a temporal vector

In some applications the product of a temporal matrix with a temporal vector is useful. There are two products – left and right.

Let $\mathbf{A}$ be a temporal matrix of size $n \times m$, $\mathbf{v}$ a vector of size n, and $\mathbf{u}$ a vector of size m. The *product from left* of $\mathbf{A}$ with $\mathbf{v}$, denoted by $\mathbf{u} = \mathbf{v} \bullet \mathbf{A}$, is defined by

$$u_j = \bigoplus_{i=1}^{n} v_i \odot a_{ij}, \qquad j = 1, \ldots, m$$

and the *product from right* of $\mathbf{A}$ with $\mathbf{u}$, denoted by $\mathbf{v} = \mathbf{A} \bullet \mathbf{u}$, is defined by

$$v_i = \bigoplus_{j=1}^{m} a_{ij} \odot u_j, \qquad i = 1, \ldots, n.$$

In the TQ library both products are implemented as functions *MatVecMulL*(A, v) and *MatVecMulR*(A, v).

If a vector $\mathbf{v}$ of size n is considered as a column vector – an $n \times 1$ matrix – it holds $\mathbf{v} \bullet \mathbf{A} = (\mathbf{v}^T \odot \mathbf{A})^T$ and $\mathbf{A} \bullet \mathbf{u} = \mathbf{A} \odot \mathbf{u}$. T denotes the matrix transposition operation.

4 Node activities

In this section we show how we can use the proposed operations with temporal quantities (the addition) for a simple analysis of temporal networks.

Assume that the values in temporal quantities a_{uv} from a temporal network matrix $\mathbf{A}$ are positive real numbers measuring the intensity of the activity of the node u on the node v. We define the *activity* of a group of nodes $\mathcal{V}_1$ on a group $\mathcal{V}_2$ (using the combinatorial semiring) as

$$\text{act}(\mathcal{V}_1, \mathcal{V}_2) = \sum_{u \in \mathcal{V}_1} \sum_{v \in \mathcal{V}_2} a_{uv}.$$

To illustrate the notion of activity we applied it on Franzosi's violence temporal network (Franzosi, 1997). Roberto Franzosi collected from the journal news in the period (January 1919 – December 1922) information about the different types of interactions between political parties and other groups of people in Italy. The violence network contains only the data about violent actions and counts the number of interactions per month.

We determined the temporal quantities $pol = \text{act}(\{\text{police}\}, \mathcal{V}) + \text{act}(\mathcal{V}, \{\text{police}\})$, *fas* $= \text{act}(\{\text{fascists}\}, \mathcal{V}) + \text{act}(\mathcal{V}, \{\text{fascists}\})$ and $all = \text{act}(\mathcal{V}, \mathcal{V})$. They are presented in Figure 4. Comparing the intensity charts of police and fascists activity with overall activity we see that most of the violent activities in the first two years 1919 and 1920 were related to the police. In the next two years (1921 and 1922) they were taken over by the fascists.

5 Temporal degrees

For an ordinary graph with a (binary) adjacency matrix $\mathbf{A}$ we can compute the corresponding indegree, $\mathbf{i}$, and outdegree, $\mathbf{o}$, vectors using (over the combinatorial semiring) the relations

$$\mathbf{i} = \mathbf{e} \bullet \mathbf{A} \qquad \text{and} \qquad \mathbf{o} = \mathbf{A} \bullet \mathbf{e}$$

where $\mathbf{e}$ is a column vector of size $n = |\mathcal{V}|$ with all its entries equal to 1. The same holds for temporal networks. In this case the vector $\mathbf{e}$ contains as values the temporal unit $\mathbf{1} = [(0, \infty, 1)]$.

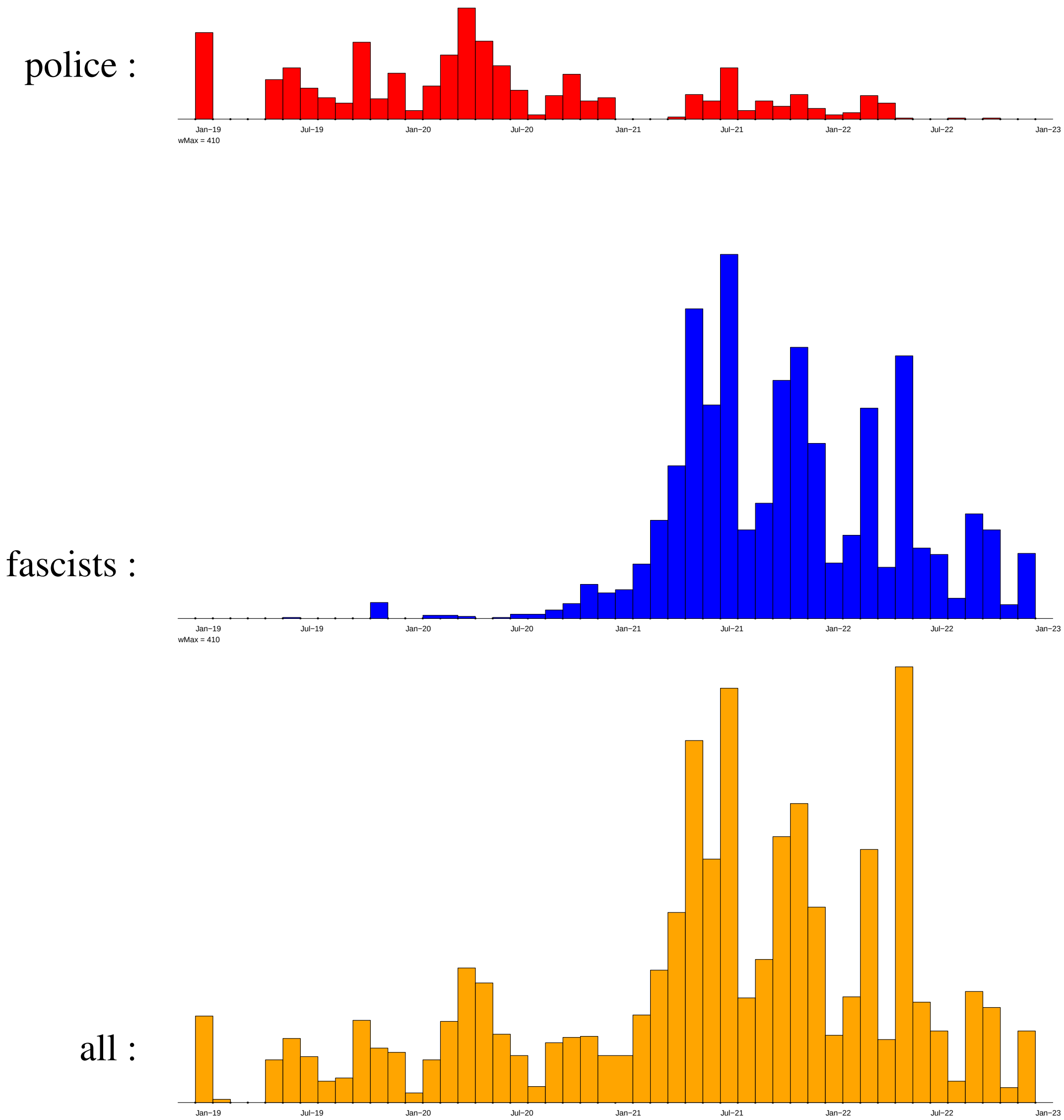

Figure 4: Intensity of violent activities of police, fascists and all.

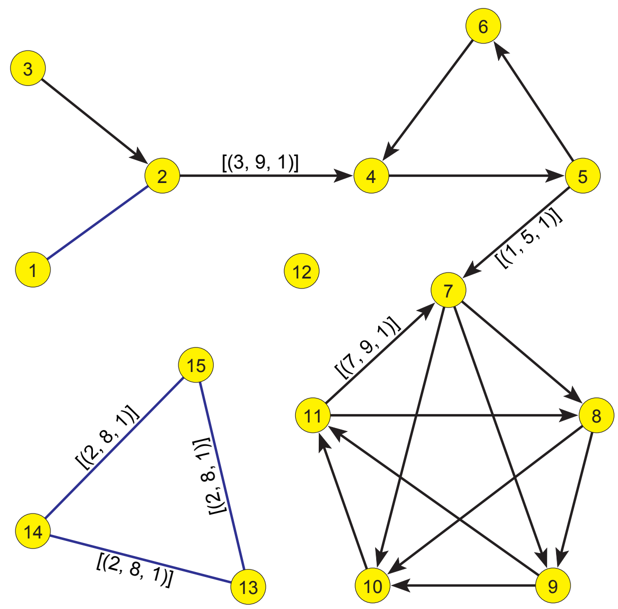

Figure 5: First example network. All unlabeled links have a value of $[(1, 9, 1)]$.

Table 1: Temporal indegrees and outdegrees for the first example network.

```
Indegrees                             Outdegrees
 1 : [(1, 9, 1)]                        1 : [(1, 9, 1)]
 2 : [(1, 9, 2)]                        2 : [(1, 3, 1), (3, 9, 2)]
 3 : []                                 3 : [(1, 9, 1)]
 4 : [(1, 3, 1), (3, 9, 2)]             4 : [(1, 9, 1)]
 5 : [(1, 9, 1)]                        5 : [(1, 5, 2), (5, 9, 1)]
 6 : [(1, 9, 1)]                        6 : [(1, 9, 1)]
 7 : [(1, 5, 1), (7, 9, 1)]             7 : [(1, 9, 3)]
 8 : [(1, 9, 2)]                        8 : [(1, 9, 2)]
 9 : [(1, 9, 2)]                        9 : [(1, 9, 2)]
10 : [(1, 9, 3)]                       10 : [(1, 9, 1)]
11 : [(1, 9, 2)]                       11 : [(1, 7, 1), (7, 9, 2)]
12 : []                                12 : []
13 : [(2, 8, 2)]                       13 : [(2, 8, 2)]
14 : [(2, 8, 2)]                       14 : [(2, 8, 2)]
15 : [(2, 8, 2)]                       15 : [(2, 8, 2)]
```

For a temporal network presented in Figure 5 the corresponding temporal indegrees and outdegrees are given in Table 1. For example, the node 5 has in the time interval $[1, 5)$ outdegree 2. Because the arc $(5, 7)$ disappears at the time point 5 the outdegree of the node 5 diminishes to 1 in the interval $[5, 9)$.

We will use the simple temporal network from Figure 5 also for the illustration of some other algorithms because it allows the users to manually check the presented results.

6 Temporal co-occurrence networks

Let the binary matrix $\mathbf{A} = [a_{ep}]$ describe a two-mode network on the set of events E and the set of of participants P:

$$a_{ep} = \begin{cases} 1 & p \text{ participated in the event } e \\ 0 & \text{otherwise} \end{cases}$$

The function $d : E \to \mathcal{T}$ assigns to each event e the date $d(e)$ when it happened. $\mathcal{T} = [\mathit{first}, \mathit{last}]$. Using these data we can construct two temporal affiliation matrices:

- **instantaneous** $\mathbf{Ai} = [ai_{ep}]$, where

$$ai_{ep} = \begin{cases} [(d(e), d(e) + 1, 1)] & a_{ep} = 1 \\ [\,] & \text{otherwise} \end{cases}$$

- **cumulative** $\mathbf{Ac} = [ac_{ep}]$, where

$$ac_{ep} = \begin{cases} [(d(e), last + 1, 1)] & a_{ep} = 1 \\ [\,] & \text{otherwise} \end{cases}$$

Using the multiplication of temporal matrices over the combinatorial semiring we get the corresponding instantaneous and cumulative co-occurrence matrices

$$\mathbf{Ci} = \mathbf{Ai}^T \cdot \mathbf{Ai} \qquad \text{and} \qquad \mathbf{Cc} = \mathbf{Ac}^T \cdot \mathbf{Ac}$$

A typical example of such a matrix is the papers authorship matrix where E is the set of papers, P is the set of authors and d is the publication year (Batagelj and Cerinek, 2013).

The triple (s, f, v) in a temporal quantity ci_{pq} tells that in the time interval $[s, f)$ there were v events in which both p and q took part.

The triple (s, f, v) in a temporal quantity cc_{pq} tells that in the time interval $[s, f)$ there were in total v accumulated events in which both p and q took part.

The diagonal matrix entries ci_{pp} and cc_{pp} contain the temporal quantities counting the number of events in the time intervals in which the participant p took part.

For example, in a data set on the stem cell research during 1997–2012 in Spain collected by Gisela Cantos-Mateos (Cantos-Mateos et al., 2014) we get from the basic two-mode network, where E is the set of papers and P is the set of institutions, for selected two institutions (HCL/B $=$ University Hospital Clnic de Barcelona, Barcelona and IDI/B $=$ Institut d'Investigacions Biomdiques August Pi i Sunyer, Barcelona) the collaboration temporal quantities presented in Table 2.

The first column in the table contains the yearly collaboration (co-authorship) data and the second column contains the cumulative collaboration data. Let's read the table:
$ci[\texttt{IDI/B}, \texttt{HCL/B}](2005, 2006) = 3$ — in the year 2005 researchers from both institutions published 3 joint papers;
$ci[\texttt{IDI/B}, \texttt{HCL/B}](2011, 2013) = 18$ — in the years 2011 and 2012 researchers from both institutions published 18 joint papers each year;
$ci[\texttt{HCL/B}, \texttt{HCL/B}](2010, 2011) = 78$ — in the year 2010 researchers from the institution `HCL/B` published 78 papers;

Table 2: Temporal collaboration.

```
ci[’IDI/B’,’HCL/B’]
 1 : (2003, 2004, 1)
 2 : (2004, 2005, 2)
 3 : (2005, 2006, 3)
 4 : (2006, 2007, 2)
 5 : (2007, 2008, 1)
 6 : (2008, 2009, 7)
 7 : (2009, 2010, 6)
 8 : (2010, 2011, 7)
 9 : (2011, 2013, 18)
```

```
cc[’IDI/B’,’HCL/B’]
 1 : (2003, 2004, 1)
 2 : (2004, 2005, 3)
 3 : (2005, 2006, 6)
 4 : (2006, 2007, 8)
 5 : (2007, 2008, 9)
 6 : (2008, 2009, 16)
 7 : (2009, 2010, 22)
 8 : (2010, 2011, 29)
 9 : (2011, 2012, 47)
10 : (2012, 2013, 65)
```

```
ci[’HCL/B’,’HCL/B’]
 1 : (1997, 1998, 2)
 2 : (1998, 1999, 5)
 3 : (1999, 2000, 8)
 4 : (2000, 2001, 7)
 5 : (2001, 2002, 5)
 6 : (2002, 2003, 6)
 7 : (2003, 2004, 14)
 8 : (2004, 2005, 20)
 9 : (2005, 2006, 10)
10 : (2006, 2007, 14)
11 : (2007, 2008, 20)
12 : (2008, 2009, 28)
13 : (2009, 2010, 56)
14 : (2010, 2011, 78)
15 : (2011, 2012, 84)
16 : (2012, 2013, 112)
```

```
cc[’HCL/B’,’HCL/B’]
 1 : (1997, 1998, 2)
 2 : (1998, 1999, 7)
 3 : (1999, 2000, 15)
 4 : (2000, 2001, 22)
 5 : (2001, 2002, 27)
 6 : (2002, 2003, 33)
 7 : (2003, 2004, 47)
 8 : (2004, 2005, 67)
 9 : (2005, 2006, 77)
10 : (2006, 2007, 91)
11 : (2007, 2008, 111)
12 : (2008, 2009, 139)
13 : (2009, 2010, 195)
14 : (2010, 2011, 273)
15 : (2011, 2012, 357)
16 : (2012, 2013, 469)
```

$cc[\texttt{IDI/B}, \texttt{HCL/B}](2008, 2009) = 16$ — till the year 2008 (included) researchers from both institutions published 16 joint papers.

Note that the violence network from Section 4 is essentially a co-occurrence network that could be obtained from the more primitive instantaneous two-mode network about violent actions reported in journal articles and the involved political actors.

7 Clustering coefficients

Let us assume that the network $\mathcal{N}$ is based on a simple directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ without loops. From a simple undirected graph we obtain the corresponding simple directed graph by replacing each edge with a pair of opposite arcs. In such a graph the *clustering coefficient*, $C(v)$, of the node v is defined as the proportion between the number of realized arcs among the node's neighbors and the number of all possible arcs among the node's neighbors $N(v)$, that is

$$C(v) = \frac{|\mathcal{A}(N(v))|}{k(k-1)}$$

where k is the number of neighbors of the node v. For a node v without neighbors or with a single neighbor we set $C(v) = 0$.

The clustering coefficient measures a local density of the node's neighborhood. A problem with its applications in network analysis is that the identified densest neighborhoods are mostly very small. For this reason we provided in Pajek the *corrected clustering coefficient*, $C'(v)$,

$$C'(v) = \frac{|\mathcal{A}(N(v))|}{\Delta(k-1)}$$

where Δ is the maximum number of neighbors in the network.

To count the number of realized arcs among the node's neighbors we use the observation that each arc forms a triangle with links from its end-nodes to the node v; and that the number of triangles in a simple undirected graph can be obtained as the diagonal value in the third power of the graph matrix (over the combinatorial semiring).

For simple directed graphs the counting of triangles is slightly more complicated. Let us denote $\mathbf{T} = \mathbf{A}^T$ and $\mathbf{S} = \mathbf{A} + \mathbf{T}$. From Figure 6 we see that each triangle (determined with a link opposite to the dark node) appears exactly once in

$$\mathbf{AAA} + \mathbf{AAT} + \mathbf{TAT} + \mathbf{TAA} = \mathbf{AAS} + \mathbf{TAS} = \mathbf{SAS}.$$

This gives us a simple way to count the triangles which is used in Algorithm 3. The function *nRows*$(\mathbf{A})$ returns the size (number of rows) of matrix $\mathbf{A}$. The function *VecConst*(n, v) constructs a vector of size n filled with the value v. The function *MatBin*$(\mathbf{A})$ transforms all values in the triples in the matrix $\mathbf{A}$ to 1. The function *MatSetDiag*$(\mathbf{A}, c)$ sets all the diagonal entries of the matrix $\mathbf{A}$ to the value c. The function *MatSym*$(\mathbf{A})$ makes the transformation $\mathbf{S} = \mathbf{A} \oplus \mathbf{T}$. Functions *VecSum* and *VecProd* implement a component wise composition of temporal vectors: *VecSum*$(a, b) = [a_i \oplus b_i,\ i = 1, \ldots, n]$ and *VecProd*$(a, b) = [a_i \odot b_i,\ i = 1, \ldots, n]$. Similarly *VecInv*$(a) = [$*invert*$(a_i),\ i = 1, \ldots, n]$ in the combinatorial semiring; where *invert*$(a) = [(s, f, 1/v)$ **for** $(s, f, v) \in a]$. The function *MatProd*$(\mathbf{A}, \mathbf{B})$ determines the product $\mathbf{A} \odot \mathbf{B}$. Since we need only the diagonal values of the matrix $\mathbf{SAS}$ we applied a special

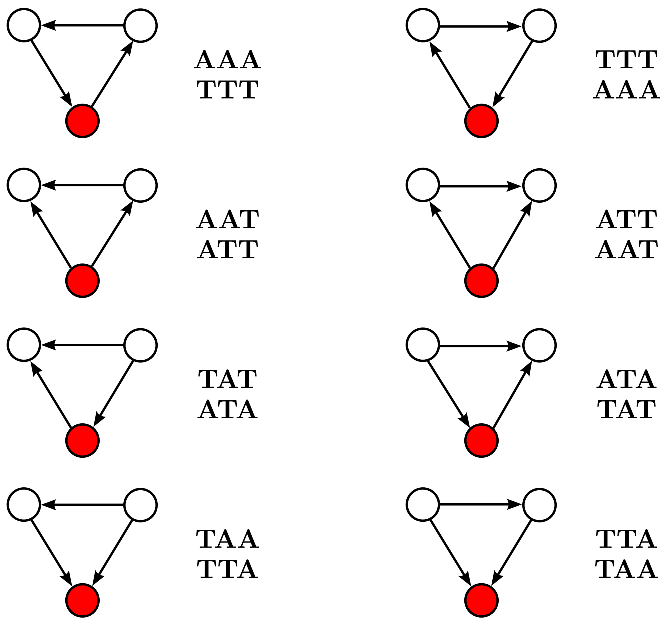

Figure 6: Counting triangles.

function *MatProdDiag*$(\mathbf{A}, \mathbf{B})$ that determines only the diagonal vector of the product $\mathbf{A} \odot \mathbf{B}$. Afterward, to get the clustering coefficient, we have to normalize the obtained counts. The number of neighbors of the node v is determined as its degree in the corresponding undirected temporal skeleton graph (in which an edge $e = (v : u)$ exists iff there is at least one arc between the nodes v and u). The maximum number of neighbors Δ can be considered either for a selected time point (*type* $= 2$) or for the complete time window (*type* $= 3$). Note that to determine the temporal Δ we used summing of temporal degrees over the *maxmin* semiring $(\mathbb{R}, \max, \min, -\infty, \infty)$.

The time complexity of Algorithm 3 is $O(n^3 \cdot L)$.

In Table 3 and Table 4 the ordinary and the corrected clustering coefficients are presented for the example network from Figure 5 and its undirected skeleton.

8 Closures in temporal networks

When the basic semiring $(A, \oplus, \odot, 0, 1)$ is *closed* – an unary *closure* operation $\star$ with the property

$$a^\star = 1 \oplus a \odot a^\star = 1 \oplus a^\star \odot a, \qquad \text{for all } a \in A$$

is defined in it – this property can be extended also to the corresponding matrix semiring. When it exists, a standard closure is obtained as

$$a^\star = \bigoplus_{i=0}^{\infty} a^i.$$

In some semirings different closures can exist. For computing the matrix closure we can apply the Fletcher's algorithm (Fletcher, 1980). The entry c_{uv} in the matrix $\mathbf{C} = \mathbf{A}^\star$ is equal to the

Algorithm 3 Clustering coefficients.

function *clusCoef*$(A, type = 1)$
\# *type* $= 1$ - standard clustering coefficient
\# *type* $= 2$ - corrected clustering coefficient / temporal degMax
\# *type* $= 3$ - corrected clustering coefficient / overall degMax
  SetSemiring(*combinatorial*)
  $n \leftarrow$ *nRows*(A); $ve \leftarrow$ *VecConst*$(n, [(0, \infty, -1)])$
  $B \leftarrow$ *MatSetDiag*(*MatBin*(A), $\mathbf{0}$)
  $S \leftarrow$ *MatBin*(*MatSym*(B))
  deg $\leftarrow$ *MatVecMulR*$(S$, *VecConst*$(n, \mathbf{1}))$
  if *type* $= 1$ **then**
    fac $\leftarrow$ *VecProd*(*deg*, *VecSum*(*deg*, ve))
  else
    SetSemiring(*maxmin*); $\delta \leftarrow \mathbf{0}$
    for $d \in$ *deg* **do** $\delta \leftarrow$ *sum*(δ, d)
    if *type* $= 3$ **then**
      $\Delta \leftarrow \max([v$ **for** $(s, f, v) \in \delta])$
      $\delta \leftarrow [(0, \infty, \Delta)]$
    SetSemiring(*combinatorial*)
    degm $\leftarrow$ *VecSum*(*deg*, ve); *fac* $\leftarrow \mathbf{0}$
    for $d \in$ *degm* **do** *fac*.*append*(*prod*(δ, d))
  tri $\leftarrow$ *MatProdDiag*(*MatProd*(S, B), S)
  return *VecProd*(*VecInv*(*fac*), *tri*)

Table 3: Clustering coefficients for the first example network.

```
Clustering coefficient
 1 : []
 2 : []
 3 : []
 4 : [(1, 3, 0.5), (3, 9, 0.1667)]
 5 : [(1, 5, 0.1667), (5, 9, 0.5)]
 6 : [(1, 9, 0.5)]
 7 : [(1, 5, 0.25), (5, 9, 0.5)]

 8 : [(1, 7, 0.4167), (7, 9, 0.5)]
 9 : [(1, 7, 0.4167), (7, 9, 0.5)]
10 : [(1, 7, 0.4167), (7, 9, 0.5)]
11 : [(1, 9, 0.5)]
12 : []
13 : [(2, 8, 1.0)]
14 : [(2, 8, 1.0)]
15 : [(2, 8, 1.0)]
```

```
Corrected clustering coefficient
 1 : []
 2 : []
 3 : []
 4 : [(1, 3, 0.25), (3, 9, 0.125)]
 5 : [(1, 5, 0.125), (5, 9, 0.25)]
 6 : [(1, 9, 0.25)]
 7 : [(1, 5, 0.25), (5, 7, 0.375),
      (7, 9, 0.5)]
 8 : [(1, 7, 0.4167), (7, 9, 0.5)]
 9 : [(1, 7, 0.4167), (7, 9, 0.5)]
10 : [(1, 7, 0.4167), (7, 9, 0.5)]
11 : [(1, 7, 0.375), (7, 9, 0.5)]
12 : []
13 : [(2, 8, 0.5)]
14 : [(2, 8, 0.5)]
15 : [(2, 8, 0.5)]
```

Table 4: Clustering coefficients for the skeleton of the first example network.

```
Clustering coefficient
 1 : []
 2 : []
 3 : []
 4 : [(1, 3, 1.0), (3, 9, 0.3333)]
 5 : [(1, 5, 0.3333), (5, 9, 1.0)]
 6 : [(1, 9, 1.0)]
 7 : [(1, 5, 0.5), (5, 9, 1.0)]

 8 : [(1, 7, 0.8333), (7, 9, 1.0)]
 9 : [(1, 7, 0.8333), (7, 9, 1.0)]
10 : [(1, 7, 0.8333), (7, 9, 1.0)]
11 : [(1, 9, 1.0)]
12 : []
13 : [(2, 8, 1.0)]
14 : [(2, 8, 1.0)]
15 : [(2, 8, 1.0)]
```

```
Corrected clustering coefficient
 1 : []
 2 : []
 3 : []
 4 : [(1, 3, 0.5), (3, 9, 0.25)]
 5 : [(1, 5, 0.25), (5, 9, 0.5)]
 6 : [(1, 9, 0.5)]
 7 : [(1, 5, 0.5), (5, 7, 0.75),
      (7, 9, 1.0)]
 8 : [(1, 7, 0.8333), (7, 9, 1.0)]
 9 : [(1, 7, 0.8333), (7, 9, 1.0)]
10 : [(1, 7, 0.8333), (7, 9, 1.0)]
11 : [(1, 7, 0.75), (7, 9, 1.0)]
12 : []
13 : [(2, 8, 0.5)]
14 : [(2, 8, 0.5)]
15 : [(2, 8, 0.5)]
```

sum of values of all walks from the node u to the node v. In most of the semirings, except the combinatorial, for which we are interested in determining the closures, also the *absorption law* holds

$$1 \oplus a = 1, \qquad \text{for all } a \in A.$$

In these semirings $a^\star = 1$, for all $a \in A$, and therefore the Fletcher's algorithm can be simplified and performed in place as implemented in Algorithm 4.

For a temporal quantity a over a closed semiring it holds $T_{a^\star} = \mathcal{T}$.

The time complexity of Algorithm 4 is $O(n^3 \cdot L)$.

Algorithm 4 Closure of a temporal matrix over an absorptive semiring.

function *MatClosure*(R, *strict* $=$ *False*)
    $n \leftarrow$ *nRows*(R)
    $C \leftarrow R$
    for $k \in 1 : n$ **do**
        for $u \in 1 : n$ **do**
            for $v \in 1 : n$ **do**
                $C[u,v] \leftarrow$ *sum*$(C[u,v],$ *prod*$(C[u,k], C[k,v]))$
        if $\neg$*strict* **then** $C[k,k] \leftarrow$ *sum*$(\mathbf{1}, C[k,k])$
    return C

9 Temporal node partitions

In the previous sections, the nodes of temporal networks were considered as being present all the time. We can describe the presence of nodes through time using a temporal binary (single valued) node partition $T : \mathcal{V} \to A_{\text{⌘}}(\mathcal{T})$,

$$T(u) = ((s_i, f_i, 1))_{i=1}^k, \quad \text{for } u \in \mathcal{V}$$

specifying that a node u is present in time intervals $[s_i, f_i)$, $i = 1, \ldots, k$.

The node partition T_{Min} determined from the temporal network links by

$$T_{Min}(u) = \bigcup_{l \in \mathcal{L}: u \in \text{ext}(l)} \textit{binary}(a_l),$$

for $u \in \mathcal{V}$, is the smallest temporal partition of nodes that satisfies the consistency condition from Section 2. The term $\text{ext}(l)$ denotes the set of endnodes of the link l, a_l is the temporal quantity assigned to the link l, and the function *binary* sets all values in a given temporal quantity to 1. In the library TQ the partition T_{Min} can be computed using the function *minTime*.

A temporal node partition q can also be used to extract a corresponding subnetwork from the given temporal network described with a matrix $\mathbf{A}$. The subnetwork contains only the nodes active in the partition q and the active links satisfying the consistency condition with respect to q.

To formalize the described procedure we first define the procedure *extract*$(p, a) = b$, where p is a binary temporal quantity and a is a temporal quantity, as

$$b(t) = \begin{cases} a(t) & t \in T_p \cap T_a \\ \maltese & \text{otherwise} \end{cases} .$$

Let $\mathbf{B}$ be a temporal matrix describing the links of the subnetwork determined by the partition q. Its entries for $l(u, v) \in \mathcal{L}$ are determined by

$$b_l = \textit{extract}(q(u) \cap q(v), a_l).$$

In TQ this operation is implemented as a procedure *MatExtract*$(\mathbf{q}, \mathbf{A})$.

10 Temporal reachability and weak and strong connectivity

For a temporal network represented with the corresponding binary matrix $\mathbf{A}$ its transitive closure $\mathbf{A}^\star$ (over the reachability semirings based on the semiring $(\{0,1\}, \vee, \wedge, 0, 1)$) determines its *reachability* relation matrix. We obtain its *weak connectivity* temporal matrix $\mathbf{W}$ as

$$\mathbf{W} = (\mathbf{A} \cup \mathbf{A}^T)^\star$$

and its *strong connectivity* temporal matrix $\mathbf{S}$ as

$$\mathbf{S} = \mathbf{A}^\star \cap (\mathbf{A}^\star)^T.$$

The use of the strict transitive closure instead of a transitive closure in these relations preserves the inactivity value $\mathbf{0}$ on the diagonal for all isolated nodes.

10.1 Reachability degrees

Let $\mathbf{R} = \overline{\mathbf{A}} = \mathbf{A} \odot \mathbf{A}^\star$ be the strict reachability relation of a given network. Then the temporal vectors *inReach* $=$ *inDeg*$(\mathbf{R})$ and *outReach* $=$ *outDeg*$(\mathbf{R})$ contain temporal quantities counting the number of nodes: from which a given node v is reachable (*inReach*$[v]$) / which are reachable from the node v (*outReach*$[v]$). The results for our example network are presented in Table 5. For example, 8 nodes $\{4, 5, 6, 7, 8, 9, 10, 11\}$ are reachable from node 6 in the time interval $[1, 5)$, and 3 nodes $\{4, 5, 6\}$ are reachable in the time interval $[5, 9)$.

Table 5: Temporal input and output reachability degrees for the first example network.

```
Input reachability
 1 : [(1, 9, 3)]
 2 : [(1, 9, 3)]
 3 : []
 4 : [(1, 3, 3), (3, 9, 6)]
 5 : [(1, 3, 3), (3, 9, 6)]
 6 : [(1, 3, 3), (3, 9, 6)]
 7 : [(1, 3, 3), (3, 5, 6), (7, 9, 5)]
 8 : [(1, 3, 8), (3, 5, 11), (5, 9, 5)]
 9 : [(1, 3, 8), (3, 5, 11), (5, 9, 5)]
10 : [(1, 3, 8), (3, 5, 11), (5, 9, 5)]
11 : [(1, 3, 8), (3, 5, 11), (5, 9, 5)]
12 : []
13 : [(2, 8, 3)]
14 : [(2, 8, 3)]
15 : [(2, 8, 3)]
```

```
Output reachability
 1 : [(1, 3, 2), (3, 5, 10), (5, 9, 5)]
 2 : [(1, 3, 2), (3, 5, 10), (5, 9, 5)]
 3 : [(1, 3, 2), (3, 5, 10), (5, 9, 5)]
 4 : [(1, 5, 8), (5, 9, 3)]
 5 : [(1, 5, 8), (5, 9, 3)]
 6 : [(1, 5, 8), (5, 9, 3)]
 7 : [(1, 7, 4), (7, 9, 5)]
 8 : [(1, 7, 4), (7, 9, 5)]
 9 : [(1, 7, 4), (7, 9, 5)]
10 : [(1, 7, 4), (7, 9, 5)]
11 : [(1, 7, 4), (7, 9, 5)]
12 : []
13 : [(2, 8, 3)]
14 : [(2, 8, 3)]
15 : [(2, 8, 3)]
```

10.2 Temporal weak connectivity

The function *weakConnMat*$(\mathbf{A})$ for a given temporal network matrix $\mathbf{A}$ determines the corresponding temporal weak connectivity matrix $\mathbf{W}$. Every time slice $\mathcal{N}(t)$, $t \in \mathcal{T}$, of the matrix $\mathbf{W}$ is an equivalence relation that can be compactly described with the corresponding partition.

To transform the temporal equivalence matrix $\mathbf{E}$ into the corresponding temporal partition $\mathbf{p}$ we use the fact that on a given time interval equivalent (in our case weakly connected) nodes get the same value on this interval in the product of the matrix $\mathbf{E}$ with a vector computed over the combinatorial semiring $(\mathbb{N}, +, \cdot, 0, 1)$. We take for the vector values randomly shuffled integers from the interval $1 : n$. With a very high probability the values belonging to different equivalence classes are different. This is implemented as a procedure *eqMat2Part*$(\mathbf{E})$ (see Algorithm 5). Maybe in the future implementations we shall add a loop with the check of the injectivity of this mapping. The classes of the obtained temporal partition are finally renumbered with consecutive numbers using the function $renumPart(p)$ (see Algorithm 6). The variable C in the description of the function $renumPart$ is a dictionary (data structure).

For our first example network we obtain the temporal weak partition presented on the left hand side of Table 6.

Algorithm 5 Transform temporal equivalence relation into partition.

function *eqMat2Part*(E)
  SetSemiring$($*combinatorial*$)$
  $v \leftarrow$ *shuffle*$([[(0, \infty, i+1)]$ **for** $i \in 1 : $ *nRows*$(E)])$
  $p \leftarrow$ *MatVecMulR*(E, v)
  return *renumPart*(p)

10.3 Temporal strong connectivity

The procedure *strongConnMat*$(\mathbf{A})$ for a given temporal network matrix $\mathbf{A}$ determines the corresponding temporal strong connectivity matrix $\mathbf{S}$. To determine the intersection of temporal

Algorithm 6 Renumber the classes of a partition.

function *renumPart*(p)
  $C \leftarrow \{\ \}; q = [\ ]$
  for $a \in p$ **do**
    $r \leftarrow [\ ]$
    for $(s_a, f_a, c_a) \in a$ **do**
      if $c_a \notin C$ **then** $C[c_a] \leftarrow 1 + \textit{length}(C)$
      $r.\textit{append}((s_a, f_a, C[c_a]))$
    $q.\textit{append}(r)$
  return q

Table 6: Temporal weak and strong connectivity partitions for the first example network.

```
Weak partition
 1 : [(1, 3, 1), (3, 5, 2), (5, 9, 3)]
 2 : [(1, 3, 1), (3, 5, 2), (5, 9, 3)]
 3 : [(1, 3, 1), (3, 5, 2), (5, 9, 3)]
 4 : [(1, 3, 4), (3, 5, 2), (5, 9, 3)]
 5 : [(1, 3, 4), (3, 5, 2), (5, 9, 3)]
 6 : [(1, 3, 4), (3, 5, 2), (5, 9, 3)]
 7 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
 8 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
 9 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
10 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
11 : [(1, 3, 4), (3, 5, 2), (5, 9, 5)]
12 : []
13 : [(2, 8, 6)]
14 : [(2, 8, 6)]
15 : [(2, 8, 6)]
```

```
Strong partition
 1 : [(1, 9, 1)]
 2 : [(1, 9, 1)]
 3 : []
 4 : [(1, 9, 2)]
 5 : [(1, 9, 2)]
 6 : [(1, 9, 2)]
 7 : [(7, 9, 3)]
 8 : [(1, 7, 4), (7, 9, 3)]
 9 : [(1, 7, 4), (7, 9, 3)]
10 : [(1, 7, 4), (7, 9, 3)]
11 : [(1, 7, 4), (7, 9, 3)]
12 : []
13 : [(2, 8, 5)]
14 : [(2, 8, 5)]
15 : [(2, 8, 5)]
```

network binary matrices $\mathbf{A}$ and $\mathbf{B}$ we use the function *MatInter*$(\mathbf{A}, \mathbf{B})$. Again, to get the strong connectivity partition we have to apply the function *eqMat2Part* to the strong connectivity matrix.

The time complexity of algorithms for temporal weak and strong connectivity partitions is $O(n^3 \cdot L)$.

For our first example network we obtain the temporal strong partition presented on the right hand side of Table 6. In the library TQ both matrices and partitions are based on the strict transitive closure.

11 Temporal closeness and betweenness

Closeness and betweenness are among the traditional social network analysis indices measuring the importance of nodes (Freeman, 1978). They are somehow problematic when applied to non (strongly) connected graphs. In this section we will not consider these questions. We will only show how to compute them for non-problematic temporal graphs.

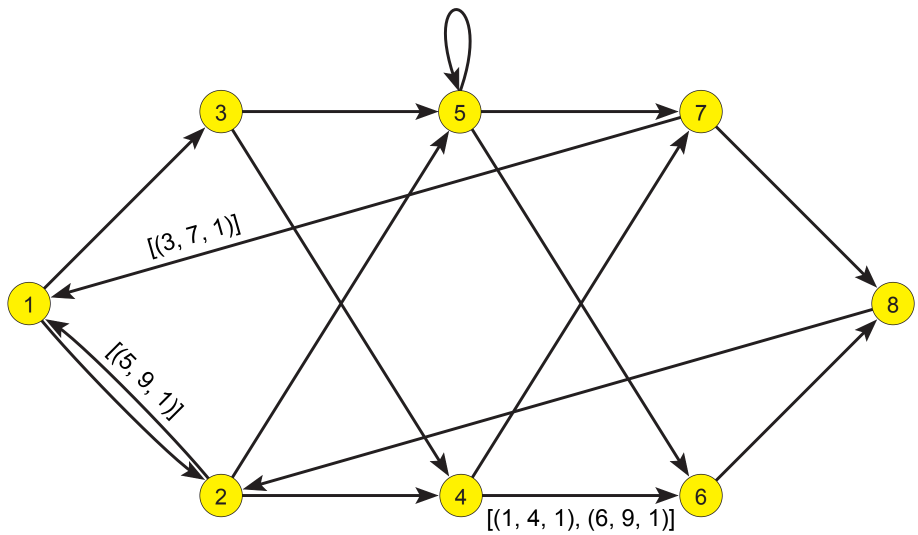

Figure 7: Second example network. All unlabeled arcs have the value $[(1, 9, 1)]$.

11.1 Temporal closeness

The *output closeness* of the node v is defined as

$$ocl(v) = \frac{n-1}{\sum\limits_{u \in \mathcal{V} \setminus \{v\}} d_{vu}}.$$

To determine the closeness we first need to compute the matrix $\mathbf{D} = [d_{uv}]$ of geodetic distances d_{uv} between the nodes u and v. It can be obtained as a closure of the network matrix $\mathbf{A}$ over the *shortest paths* semiring $(\overline{\mathbb{R}_0^+}, \min, +, \infty, 0)$. Note that the values in the matrix $\mathbf{A}$ can be any nonnegative real numbers.

In Figure 7 we present our second example temporal network which is an extended version of the example given in Figure 3 from Batagelj (1994).

Because a complete strict closure matrix $\mathbf{D}$ is too large to be listed we present only some of its selected entries:

```
D[3,1] = [(3, 7, 3), (7, 9, 5)]
D[4,6] = [(1, 4, 1), (4, 6, 5), (6, 9, 1)]
D[6,3] = [(3, 5, 6), (5, 9, 4)]
D[7,6] = [(1, 9, 4)]
```

To compute the vector of closeness coefficients of nodes we have to sum the temporal distances to other nodes over the combinatorial semiring. See Algorithm 7. While summing we replace gaps (inactivity intervals inside $\mathcal{T}$) with time intervals with the value infinity, using the procedure *fillGaps*. The time complexity of Algorithm 7 is $O(n^3 \cdot L)$.

The temporal closeness coefficients for our second example network are given in Table 7.

Algorithm 7 Temporal closeness.

function *closeness*$(A, \mathit{type} = 2)$
\# *type*: 1 - output, 2 - all, 3 - input
    $s \leftarrow \mathit{startTime}(A)$; $f \leftarrow \mathit{finishTime}(A)$; $n \leftarrow \mathit{nRows}(A)$
    $\mathit{SetSemiring}(path)$
    $D \leftarrow \mathit{MatClosure}(A, \mathit{strict} = True)$
    $\mathit{SetSemiring}(\mathit{combinatorial})$
    $k \leftarrow (2 - |\mathit{type} - 2|) \cdot (n-1); \mathit{fac} \leftarrow [(0, \infty, k)]$
    for $v \in 1 : n$ **do**
        $d \leftarrow \mathbf{0}$
        for $u \in 1 : n$ **do**
            if $u \neq v$ **then**
                if $\mathit{type} < 3$ **then** $d \leftarrow \mathit{sum}(d, \mathit{fillGaps}(D[v, u], s, f))$
                if $\mathit{type} > 1$ **then** $d \leftarrow \mathit{sum}(d, \mathit{fillGaps}(D[u, v], s, f))$
        $cl[v] \leftarrow \mathit{prod}(\mathit{fac}, \mathit{invert}(d))$
    return cl

Table 7: Output closeness for the second example network.

```
1 : [(1, 9, 0.4375)]
2 : [(1, 3, 0.0000), (3, 5, 0.4375), (5, 9, 0.5833)]
3 : [(1, 3, 0.0000), (3, 7, 0.4375), (7, 9, 0.3889)]
4 : [(1, 3, 0.0000), (3, 4, 0.4375), (4, 6, 0.3500),
     (6, 7, 0.4375), (7, 9, 0.3500)]
5 : [(1, 3, 0.0000), (3, 7, 0.4375), (7, 9, 0.3500)]
6 : [(1, 3, 0.0000), (3, 5, 0.2917), (5, 9, 0.3500)]
7 : [(1, 3, 0.0000), (3, 7, 0.4375), (7, 9, 0.3500)]
8 : [(1, 3, 0.0000), (3, 5, 0.3500), (5, 9, 0.4375)]
```

11.2 Temporal betweenness

The *betweenness* of a node v is defined as

$$b(v) = \frac{1}{(n-1)(n-2)} \sum_{\substack{u,w \in \mathcal{V} \\ |\{v,u,w\}|=3}} \frac{n_{u,w}(v)}{n_{u,w}}$$

where $n_{u,w}$ is the number of u-w geodesics (shortest paths) and $n_{u,w}(v)$ is the number of u-w geodesics passing through the node v.

Suppose that we know the matrix

$$\mathbf{C} = [(d_{u,v}, n_{u,v})]$$

where $d_{u,v}$ is the length of u-v geodesics. Then it is also easy to determine the quantity $n_{u,w}(v)$:

$$n_{u,w}(v) = \begin{cases} n_{u,v} \cdot n_{v,w} & d_{u,v} + d_{v,w} = d_{u,w} \\ 0 & \text{otherwise} \end{cases} .$$

This gives the following scheme of procedure for computing the nontemporal betweenness coefficients $\mathbf{b}$

compute $\mathbf{C}$
for $v \in \mathcal{V}$ **do**
  $r \leftarrow 0$
  for $u \in \mathcal{V}, w \in \mathcal{V}$ **do**
    if $n[u,w] \neq 0 \wedge |\{v,u,w\}| = 3 \wedge d[u,w] = d[u,v] + d[v,w]$ **then**
      $r \leftarrow r + n[u,v] \cdot n[v,w] / n[u,w]$
  $b[v] \leftarrow r/((n-1) \cdot (n-2))$

In Batagelj (1994) it is shown that the matrix $\mathbf{C}$ can be obtained by computing the closure of the network matrix over the *geodetic semiring* $(\overline{\mathbb{N}}^2, \oplus, \odot, (\infty, 0), (0, 1))$, where $\overline{\mathbb{N}} = \mathbb{N} \cup \{\infty\}$ and we define *addition* $\oplus$ with

$$(a,i) \oplus (b,j) = (\min(a,b), \begin{cases} i & a < b \\ i+j & a = b \\ j & a > b \end{cases})$$

and *multiplication* $\odot$ with:

$$(a,i) \odot (b,j) = (a+b, i \cdot j).$$

To compute the geodetic closure we first transform the network temporal adjacency matrix $\mathbf{A}$ to a matrix $\mathbf{G} = [(d,n)_{u,v}]$ which has for entries pairs defined by

$$(d,n)_{u,v}(t) = \begin{cases} (1,1) & \exists l \in \mathcal{L} : (l(u,v) \wedge t \in T(l)) \\ ⌘ & \text{otherwise} \end{cases}$$

where d is the length of a geodesic and n is the number of geodesics from u to v. In temporal networks the distance d and the counter n are temporal quantities.

Algorithm 8 Temporal betweenness.

function *betweenness*(A)
  $n \leftarrow \textit{nRows}(A)$; $G \leftarrow \textit{MatSetVal}(A, (1, 1))$
  $\textit{SetSemiring}(\textit{geodetic})$
  $C \leftarrow \textit{MatClosure}(G, \textit{strict} = True)$
  $\textit{SetSemiring}(\textit{combinatorial})$
  $\textit{fac} \leftarrow [(0, \infty, 1/(n-1)/(n-2))]$
  for $v \in 1 : n$ **do**
    $r \leftarrow \mathbf{0}$
    for $u \in 1 : n, w \in 1 : n$ **do**
      if $(C[u, w] \neq [\,]) \wedge (u \neq w) \wedge (u \neq v) \wedge (v \neq w)$ **then**
        $r \leftarrow \textit{sum}(r, \textit{between}(C[u, v], C[v, w], C[u, w]))$
    $b[v] \leftarrow \textit{prod}(r, \textit{fac})$
  return b

Following the presented scheme of computing the betweenness vector and adapting it to temporal quantities (see Algorithm 8) in the function *betweenness* we first transform the network matrix $\mathbf{A}$ into a matrix $\mathbf{G}$ with values $(1, 1)$ on arcs and compute its strict geodetic closure $\mathbf{C}$ over the geodetic semiring.

We present only some selected entries of the strict geodetic closure matrix $\mathbf{C}$ for our second example network:

```
C[1,7] = [(1, 9, (3, 4))]
C[2,2] = [(1, 3, (4, 4)), (3, 4, (4, 6)), (4, 5, (4, 5)), (5, 9, (2, 1))]
C[4,6] = [(1, 4, (1, 1)), (4, 6, (5, 3)), (6, 9, (1, 1))]
C[5,5] = [(1, 9, (1, 1))]
C[6,3] = [(3, 5, (6, 2)), (5, 9, (4, 1))]
C[7,6] = [(1, 3, (4, 2)), (3, 4, (4, 6)), (4, 6, (4, 3)), (6, 7, (4, 6)),
          (7, 9, (4, 2))]
```

For example, the value $\mathbf{C}[4, 6]$ reflects the facts that an arc exists from node 4 to node 6 in time intervals $[1, 4)$ and $[6, 9)$; and in the time interval $[4, 6)$ they are connected with 3 geodesics of length 5: $(4, 7, 8, 2, 5, 6)$, $(4, 7, 1, 3, 5, 6)$, $(4, 7, 1, 2, 5, 6)$.

We continue and using the combinatorial semiring we compute the temporal betweenness vector $\mathbf{b}$. The specificity of temporal quantities $d[u, v]$ and $n[u, v]$ is considered in the function *between* (see Algorithm 9) that implements the temporal version of the statement

if $d[u, w] = d[u, v] + d[v, w]$ **then** $r \leftarrow r + n[u, v] \cdot n[v, w]/n[u, w]$

from the basic betweenness algorithm. Again we have to apply the merging scheme. The time complexity of Algorithm 8 is $O(n^3 \cdot L)$.

The temporal betweenness coefficients for our second example network are presented in Table 8.

12 Temporal PathFinder

The Pathfinder algorithm was proposed in the eighties (Schvaneveldt, Dearholt, and Durso, 1988; Schvaneveldt, 1990) for the simplification of weighted networks – it removes from the

Algorithm 9 Temporal betweenness merge operation.

function *between*(a, b, c)
  if *length*$(a) = 0$ **then return** $[\,]$
  if *length*$(b) = 0$ **then return** $[\,]$
  if *length*$(c) = 0$ **then return** $[\,]$
  $r = [\,]$
  $(s_a, f_a, v_a) \leftarrow$ *get*(a); $(s_b, f_b, v_b) \leftarrow$ *get*(b); $(s_c, f_c, v_c) \leftarrow$ *get*(c)
  if *isTuple*(v_a) **then** $(d_a, c_a) \leftarrow v_a$
  if *isTuple*(v_b) **then** $(d_b, c_b) \leftarrow v_b$
  if *isTuple*(v_c) **then** $(d_c, c_c) \leftarrow v_c$
  while $(s_a < \infty) \vee (s_b < \infty) \vee (s_c < \infty)$ **do**
    $s_r \leftarrow \max(s_a, s_b, s_c)$; $f_r \leftarrow \min(f_a, f_b, f_c)$
    if $f_a \leq s_r$ **then**
      $(s_a, f_a, v_a) \leftarrow$ *get*(a)
      if *isTuple*(v_a) **then** $(d_a, c_a) \leftarrow v_a$
    else if $f_b \leq s_r$ **then**
      $(s_b, f_b, v_b) \leftarrow$ *get*(b)
      if *isTuple*(v_b) **then** $(d_b, c_b) \leftarrow v_b$
    else if $f_c \leq s_r$ **then**
      $(s_c, f_c, v_c) \leftarrow$ *get*(c)
      if *isTuple*(v_c) **then** $(d_c, c_c) \leftarrow v_c$
    else
      if $d_a + d_b = d_c$ **then** r.*append*$((s_r, f_r, c_a \cdot c_b / c_c))$
      if $f_r = f_a$ **then**
        $(s_a, f_a, v_a) \leftarrow$ *get*(a)
        if *isTuple*(v_a) **then** $(d_a, c_a) \leftarrow v_a$
      if $f_r = f_b$ **then**
        $(s_b, f_b, v_b) \leftarrow$ *get*(b)
        if *isTuple*(v_b) **then** $(d_b, c_b) \leftarrow v_b$
      if $f_r = f_c$ **then**
        $(s_c, f_c, v_c) \leftarrow$ *get*(c)
        if *isTuple*(v_c) **then** $(d_c, c_c) \leftarrow v_c$
  return *standard*(r)

Table 8: Betweenness for the second example network.

```
1 : [(3, 4, 0.2500), (4, 6, 0.2754), (6, 7, 0.2500), (7, 9, 0.1429)]
2 : [(1, 3, 0.3452), (3, 4, 0.4048), (4, 6, 0.4187), (6, 7, 0.4048), (7, 9, 0.6071)]
3 : [(1, 3, 0.0595), (3, 4, 0.0952), (4, 6, 0.1052), (6, 7, 0.0952), (7, 9, 0.0595)]
4 : [(1, 3, 0.1667), (3, 4, 0.2500), (4, 5, 0.1762), (5, 6, 0.1048), (6, 9, 0.1786)]
5 : [(1, 3, 0.1667), (3, 4, 0.2500), (4, 5, 0.3476), (5, 6, 0.2762), (6, 9, 0.1786)]
6 : [(1, 3, 0.1190), (3, 4, 0.0952), (4, 6, 0.0544), (6, 7, 0.0952), (7, 9, 0.1786)]
7 : [(1, 3, 0.1190), (3, 4, 0.4048), (4, 5, 0.4694), (5, 6, 0.3266), (6, 7, 0.2619),
     (7, 9, 0.1786)]
8 : [(1, 3, 0.3095), (3, 4, 0.2500), (4, 6, 0.2484), (6, 7, 0.2500), (7, 9, 0.5238)]
```

network all links that do not satisfy the triangle inequality – if for a weighted link there exists a shorter path connecting its endnodes then the link is removed. The basic idea of the Pathfinder algorithm is simple. It produces a network PFnet$(\mathbf{W}, r, q) = (\mathcal{V}, \mathcal{L}_{PF})$ determined by the following scheme of procedure

compute $\mathbf{W}^{(q)}$;
$\mathcal{L}_{PF} \leftarrow \emptyset$;
for $e(u,v) \in \mathcal{L}$ **do**
    if $\mathbf{W}^{(q)}[u,v] = \mathbf{W}[u,v]$ **then** $\mathcal{L}_{PF} \leftarrow \mathcal{L}_{PF} \cup \{e\}$

where $\mathbf{W}$ is a network ***dissimilarity*** matrix and $\mathbf{W}^{(q)} = \bigoplus_{i=1}^{q} \mathbf{W}^i = (\mathbf{1} \oplus \mathbf{W})^q$ is the matrix of the values of all walks of length at most q computed over the *Pathfinder* semiring $(\overline{\mathbb{R}_0^+}, \oplus, \boxdot, \infty, 0)$ with $a \boxdot b = \sqrt[r]{a^r + b^r}$ and $a \oplus b = \min(a,b)$. The value of $w_{uv}(q)$ in the matrix $\mathbf{W}^{(q)}$ is equal to the value of all walks of length at most q from the node u to the node v.

The scheme of Pathfinder is implemented as the function *pathFinder*. The temporal version of the statement

if $\mathbf{W}^{(q)}[u,v] = \mathbf{W}[u,v]$ **then** $\mathcal{L}_{PF} := \mathcal{L}_{PF} \cup \{e\}$

is implemented in the function *PFcheck* using the merging scheme.

The function *MatPower*(A,k) computes the k-th power of the matrix $\mathbf{A}$.

The time complexity of Algorithm 10+11 is $O(L \cdot n^3 \cdot \log q)$ (Guerrero-Bote et al, 2006).

Algorithm 10 Temporal PathFinder.

function *pathFinder*$(W, r = 1, q = \infty)$
    $n \leftarrow$ *nRows*(W); *SetSemiring*(*pathfinder*$, r, q$)
    if $q > n$ **then** $Z \leftarrow$ *MatClosure*(W)
    else $Z \leftarrow$ *MatPower*(*MatSetDiag*$(W, \mathbf{1}), q$)
    for $u \in 1 : n, v \in 1 : n$ **do**
        $PF[u,v] \leftarrow$ *PFcheck*$(W[u,v], Z[u,v])$
    return PF

The bottom network in Figure 8 presents the Pathfinder skeleton PFnet$(\mathcal{N}, 1, \infty)$ of a network $\mathcal{N}$ presented in the top part of the same figure. Because $r = 1$ a link e is removed if there exists a path, connecting its initial node to its terminal node, with the value (sum of link values) smaller than the value of the link e. The arc $(1,2)$ is removed because $3 = v(1,2) >$

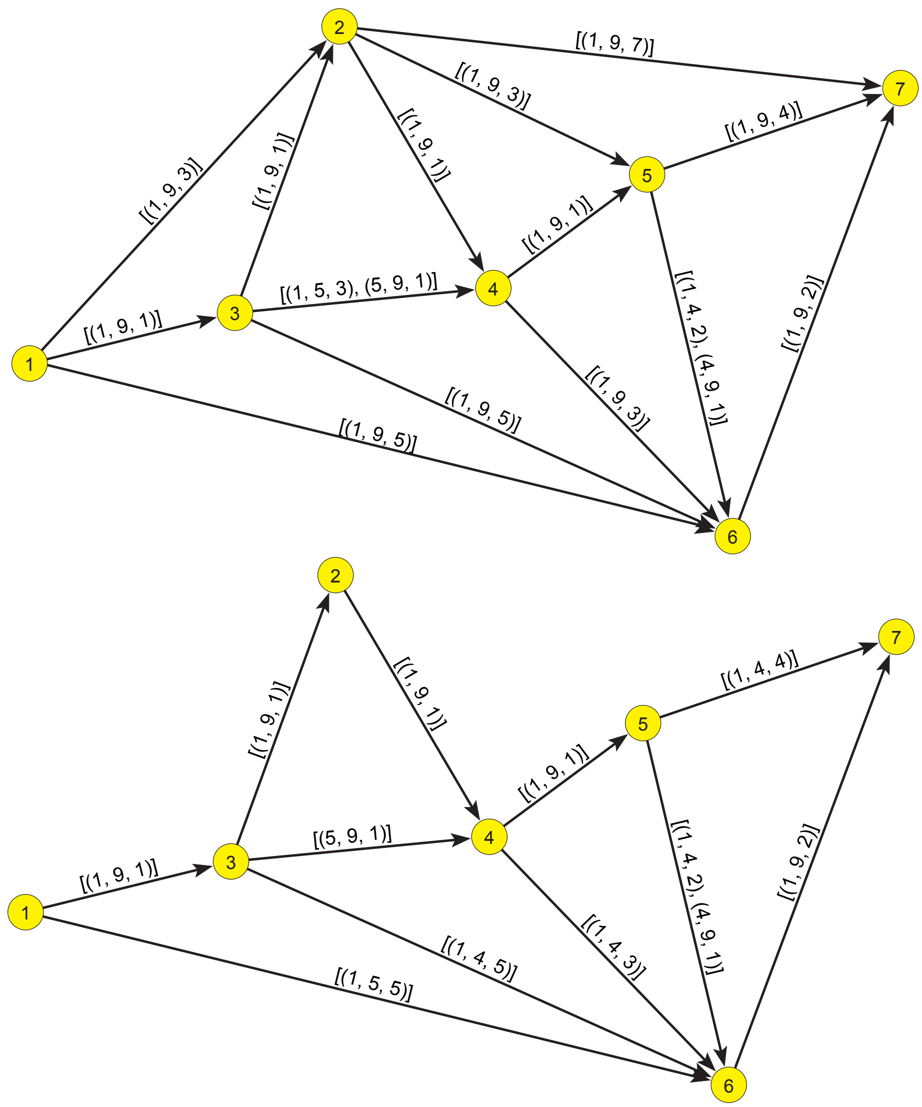

Figure 8: Pathfinder example.

Algorithm 11 Temporal PathFinder merge operation.

function *PFcheck*(a, b)
  if *length*$(a) = 0$ **then return** a
  if *length*$(b) = 0$ **then return** a
  $c \leftarrow [\,]$
  $(s_a, f_a, v_a) \leftarrow$ *get*(a); $(s_b, f_b, v_b) \leftarrow$ *get*(b)
  while $(s_a < \infty) \vee (s_b < \infty)$ **do**
    if $f_a \le s_b$ **then** $(s_a, f_a, v_a) \leftarrow$ *get*(a)
    else if $f_b \le s_a$ **then** $(s_b, f_b, v_b) \leftarrow$ *get*(b)
    else
      $s_c \leftarrow \max(s_a, s_b)$; $f_c \leftarrow \min(f_a, f_b)$
      if $v_b = v_a$ **then** c.*append*$((s_c, f_c, v_a))$
      if $f_c = f_a$ **then** $(s_a, f_a, v_a) \leftarrow$ *get*(a)
      if $f_c = f_b$ **then** $(s_b, f_b, v_b) \leftarrow$ *get*(b)
  return *standard*(c)

$v(1,3)+v(3,2) = 2$. The arc $(1,6)$ is removed in the time interval $[5,9)$ because in this interval $5 = v(1,6) > v(1,3) + v(3,4) + v(4,5) + v(5,6) = 4$.

13 September 11th Reuters terror news

The Reuters terror news network was obtained from the CRA (Centering Resonance Analysis) networks produced by Steve Corman and Kevin Dooley at Arizona State University. The network is based on all the stories released during 66 consecutive days by the news agency Reuters concerning the September 11 attack on the U.S., beginning at 9:00 AM EST 9/11/01. The nodes of this network are important words (terms). There is an edge between two words iff they appear in the same utterance (for details see the paper Corman et al. (2002)). The weight of an edge is its frequency. The network has $n = 13332$ nodes (different words in the news) and $m = 243447$ edges, 50859 with value larger than 1. There are no loops in the network.

The Reuters terror news network 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.

We transformed the Pajek version of the network into the Ianus format used in TQ. To identify important terms we computed their aggregated frequencies and extracted the subnetwork of the 50 most frequently used (during 66 days) nodes. They are listed in Table 9.

Trying to draw this subnetwork it turns out to be almost a complete graph. To obtain something readable we removed all temporal edges with a value smaller than 10. The corresponding underlying graph is presented in Figure 9. The isolated nodes were removed.

For each of the 50 nodes we determined its temporal activity and drew it. By visual inspection we identified 6 typical activity patterns – types of terms (see Figure 10). For all charts in the figure the displayed values are in the interval $[0, 200]$ – the largest activity value for the term Wednesday is larger than 200.

Table 9: 50 most frequent terms in the Terror news network.

n	term	Σfreq	n	term	Σfreq
1	united_states	15000	26	terrorism	2212
2	attack	10348	27	day	2128
3	taliban	6266	28	week	2017
4	people	5286	29	worker	1983
5	afghanistan	5176	30	office	1967
6	bin_laden	4885	31	group	1966
7	new_york	4832	32	air	1962
8	pres_bush	4506	33	minister	1919
9	washington	4047	34	time	1898
10	official	3902	35	hijack	1884
11	anthrax	3563	36	strike	1818
12	military	3394	37	afghan	1775
13	plane	3078	38	flight	1775
14	world_trade_ctr	3006	39	tell	1746
15	security	2906	40	terrorist	1745
16	american	2825	41	airport	1741
17	country	2794	42	pakistan	1714
18	city	2689	43	tower	1685
19	war	2679	44	bomb	1674
20	tuesday	2635	45	new	1650
21	pentagon	2620	46	buildng	1634
22	force	2516	47	wednesday	1593
23	government	2380	48	nation	1589
24	leader	2375	49	police	1587
25	world	2213	50	foreign	1558

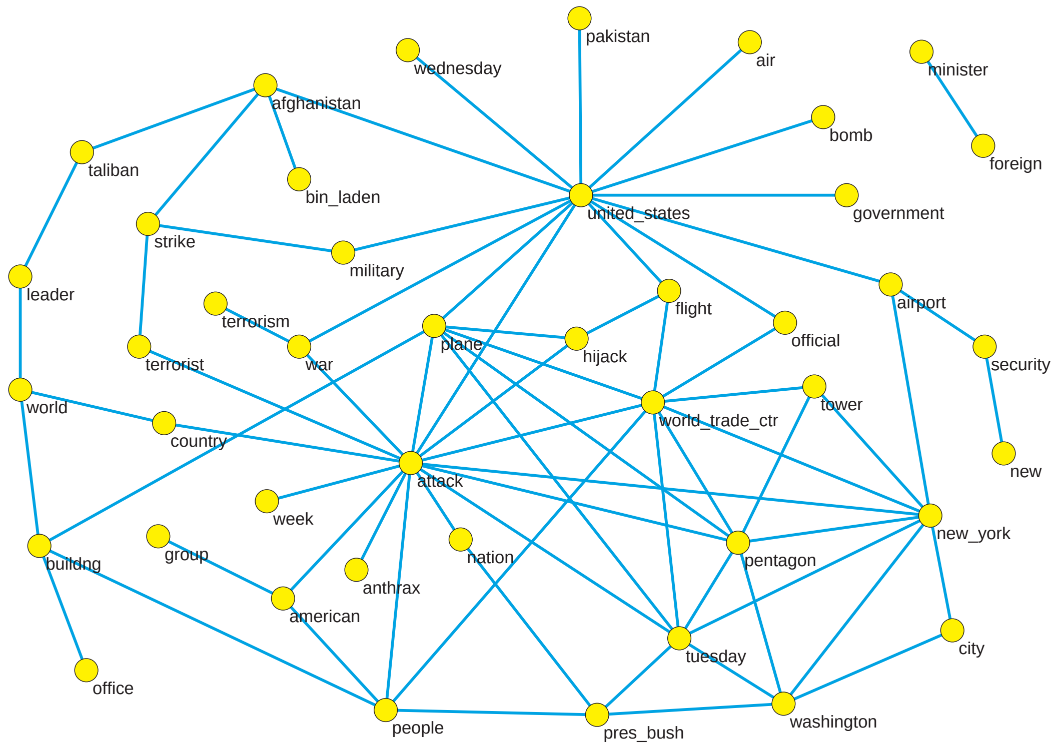

Figure 9: September 11th.

The *primary* terms are the terms with a very high frequency of appearance in the first week after September 11th and smaller, slowly declining values in the following period. The representative of this group in Figure 10 is **hijack** and other members are: airport, american, attack, city, day, flight, nation, New York, official, Pentagon, people, plane, police, president Bush, security, tower, United States, Washington, world, World Trade center. These are the terms describing the event.

The *secondary* terms are a reaction to the event. There are no big changes in their values. We identified three subgroups: a) *slowly declining* represented with **bin Laden** (country, foreign, government, military, minister, new, Pakistan, tell, terrorism, terrorist, time, war, week); b) *stationary* represented with **taliban** (afghan, Afghanistan, force, group, leader); and c) *occasional* with several peaks, represented with **bomb** (air, building, office, strike, worker).

There are three special patterns – two *periodic* **Wednesday** and Tuesday; and one *episodic* **anthrax**.

To consider in a measure of importance of the node $u \in \mathcal{V}$ also the node's position in the network we constructed the attraction coefficient $\text{att}(u)$.

Let $\mathbf{A} = [a_{uv}]$ be a network matrix of temporal quantities with positive real values. We define the *node activity* $\text{act}(u)$ as (see Section 4)

$$\text{act}(u) = \text{act}(\{u\}, \mathcal{V} \setminus \{u\}) = \sum_{v \in \mathcal{V} \setminus \{u\}} a_{uv}.$$

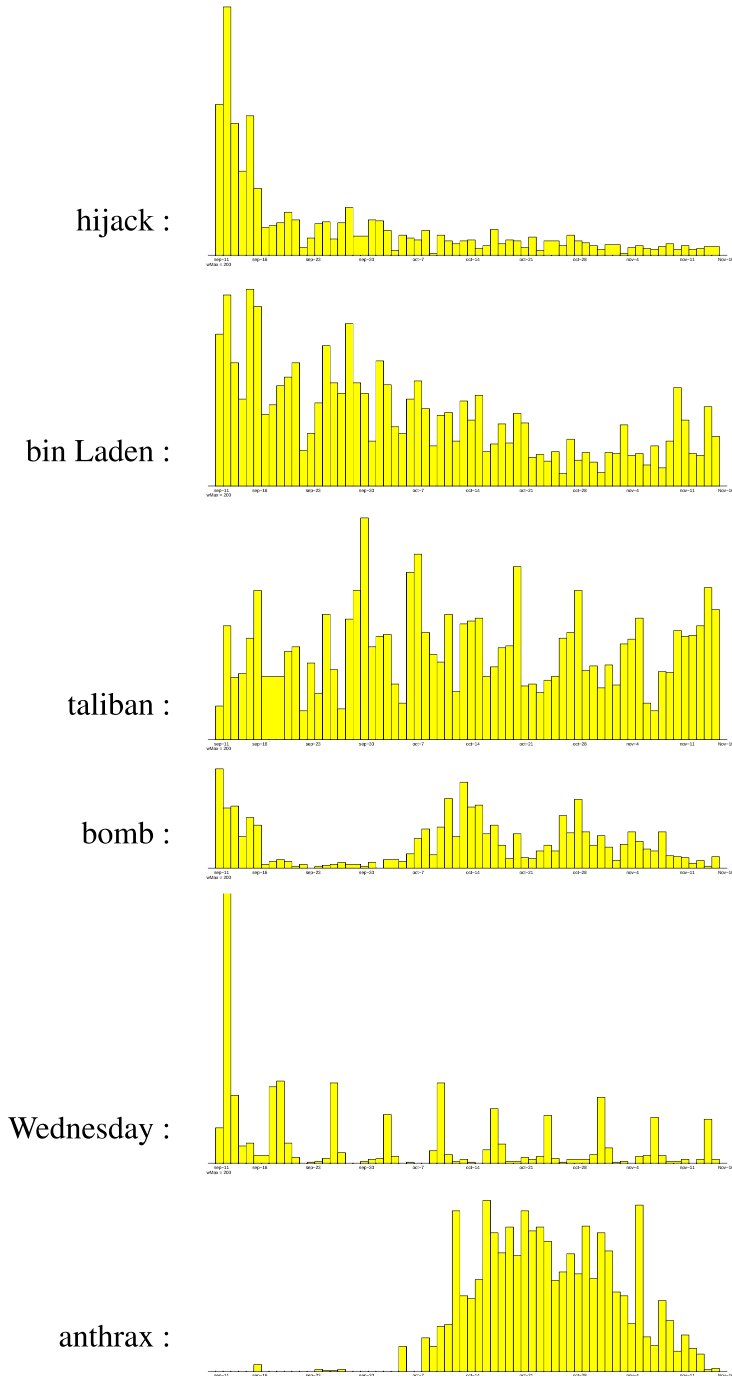

Figure 10: Types of activity.

Then the *attraction* of the node u is defined as

$$\text{att}(u) = \frac{1}{\Delta} \sum_{v \in \mathcal{V} \setminus \{u\}} \frac{a_{vu}}{\text{act}(v)}.$$

Note that the fraction $\frac{a_{vu}}{\text{act}(v)}$ is measuring the proportion of the activity of the node v that is shared with the node u.

From $0 \leq \frac{a_{vu}}{\text{act}(v)} \leq 1$ and $\deg(v) = 0 \Rightarrow a_{vu} = 0$ it follows that

$$\sum_{v \in \mathcal{V} \setminus \{u\}} \frac{a_{vu}}{\text{act}(v)} \leq \deg(u) \leq \Delta$$

where Δ denotes the maximum degree. Therefore we have $0 \leq \text{att}(u) \leq 1$, for all $u \in \mathcal{V}$.

The maximum possible attraction value 1 is attained exactly for nodes: a) in an undirected network: that are the root of a star; b) in a directed network: that are the only out-neighbors of their in-neighbors – the root of a directed in-star.

We computed the temporal attraction and the corresponding aggregated attraction values for all the nodes in our network. We selected 30 nodes with the largest aggregated attraction values. They are listed in Table 10. Again we visually explored them. In Figure 11 we present temporal attraction coefficients for the 6 selected terms. For all charts in the figure the displayed attraction values are in the interval $[0, 0.2]$.

Comparing on the common terms (taliban, bomb, anthrax) the activity charts in Figure 10 with the corresponding attraction charts in Figure 11 we see that they are "correlated" (obviously $\text{act}(a; t) = 0$ implies $\text{att}(a; t) = 0$), but different in details.

For example, the terms taliban and bomb have small attraction values at the beginning of the time window – the terms were disguised by the primary terms. On the other hand, the terms taliban and Kabul get increased attraction towards the end of the time window.

14 Conclusions

In the paper we proposed an algebraic approach to the "deterministic" analysis of temporal networks based on temporal quantities and presented algorithms for the temporal variants of basic network analysis measures and concepts. We expect that the support for many temporal variants of other network analysis notions can be developed in similar ways. Our results on temporal variants of eigen values/vectors based indices (Katz, Bonacich, hubs and authorities, page rank) will be presented in a separate paper (Praprotnik and Batagelj, 2015).

The proposed approach is an alternative to the traditional cross sectional approach based on time slices. Its main advantages are:

- the data and the results are expressed using temporal quantities that are natural descriptions of properties changing through time;
- the user does not need to be careful about the intervals on which the time slices are determined – exactly the right intervals are selected by the merging (sub)operations. This also improves, on average, the efficiency of the proposed algorithms.

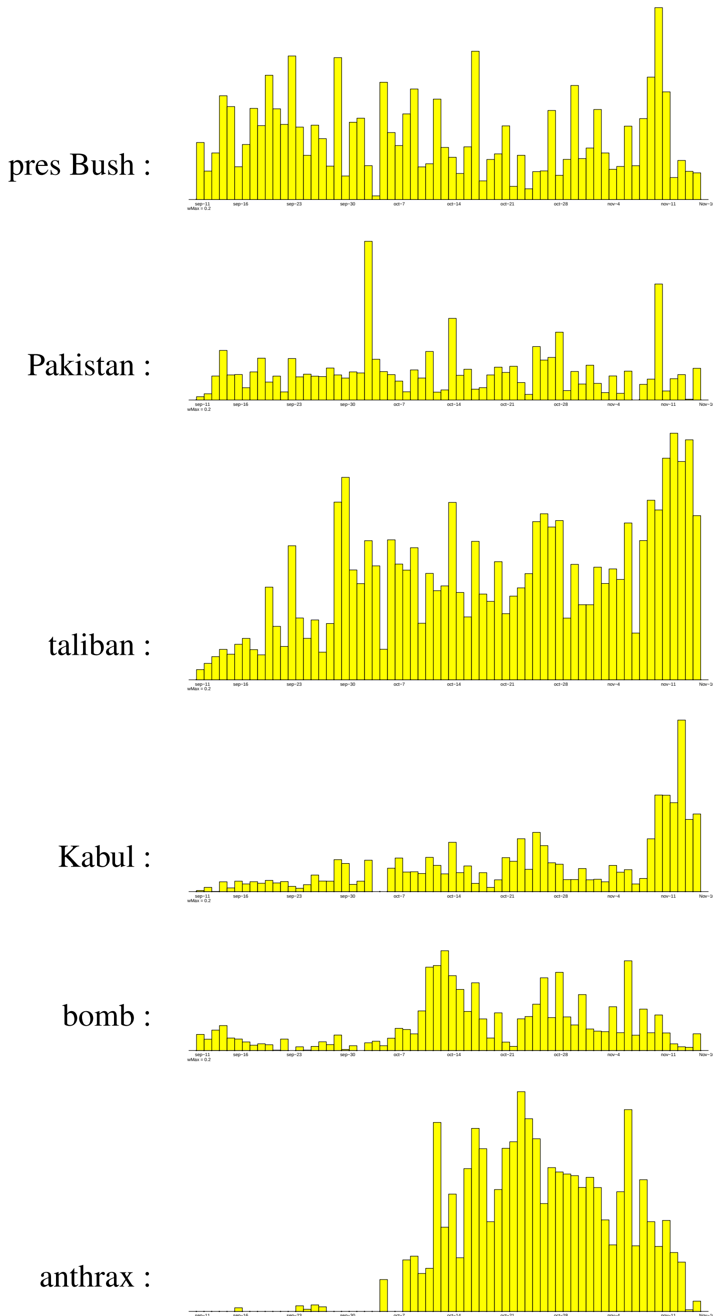

Figure 11: Attraction patterns.

Table 10: 30 most attractive terms in the Terror news network.

n	term	Σatt	n	term	Σatt
1	united_states	12.216	16	war	2.758
2	taliban	7.096	17	force	2.596
3	attack	7.070	18	new_york	2.590
4	afghanistan	5.142	19	government	2.496
5	people	5.023	20	day	2.338
6	bin_laden	4.660	21	leader	2.305
7	anthrax	4.601	22	terrorism	2.202
8	pres_bush	4.374	23	time	2.182
9	country	3.317	24	group	2.072
10	washington	3.067	25	afghan	2.040
11	security	2.939	26	world	1.995
12	american	2.922	27	week	1.961
13	official	2.831	28	pakistan	1.943
14	city	2.798	29	letter	1.866
15	military	2.793	30	new	1.851

All the described algorithms (and some others) are implemented in a Python library TQ (Temporal Quantities) available at http://vlado.fmf.uni-lj.si/pub/networks/progs/TQ/ . We started to develop a program Ianus that will provide a user-friendly (Pajek like) access to the capabilities of the TQ library.

The main goal of the paper was to show: it can be done. Therefore we based the current version of the library TQ on a matrix representation of temporal networks as it is presented in the paper. For this representation most of the network algorithms have the time complexity of $O(n^3 \cdot L)$ and the space complexity of $O(n^2 \cdot L)$. This implies that their application is limited to networks of moderate size (up to some thousands of nodes). Large networks are usually sparse. On this assumption more efficient algorithms can be developed based on a graph (sparse matrix) representation – one of the directions of our future research.

In a description of a temporal network $\mathcal{N}$ we can consider also a transition time or latency $\tau \in \mathcal{W}$: $\tau(l, t)$ is equal to the time needed to traverse the link l starting at the instant t. Problems considering latency are typical for operations research but could be important, when such data are available, also in social network analysis (Moody, 2002; Xuan et al., 2003; George, Kim, and Shekhar, 2007; Casteigts et al., 2012; Kontoleon, Falzon, and Pattison, 2013). The analysis of temporal networks considering also the latency seems a much harder task – for example, in such temporal networks the strongly connected components problem is NP-complete (Bhadra and Ferreira, 2003).

The results obtained from temporal procedures are relatively large. To identify interesting elements we used in the paper the aggregated values and the visualization of selected elements. Additional tools for browsing and presenting the results should be developed.

Acknowledgements

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

The paper is based on our talks presented at the 1st European Conference on Social Networks, Barcelona (UAB), July 1-4, 2014.

References

Batagelj, V., 1994. Semirings for social networks analysis. Journal of Mathematical Sociology, 19(1), 53-68.

Batagelj, V., 2009. Social Network Analysis, Large-Scale. R.A. Meyers, ed., Encyclopedia of Complexity and Systems Science, Springer, 8245-8265

Batagelj, V, Cerinek, M, 2013. On bibliographic networks. Scientometrics 96 (3), 845-864.

Bhadra, S., Ferreira, A., 2003. Complexity of Connected Components in Evolving Graphs and the Computation of Multicast Trees in Dynamic Networks. In ADHOC-NOW, LNCS 2865, Springer, 259-270.

Cantos-Mateos, G., Zulueta, M.., Vargas-Quesada, B., Chinchilla-Rodrguez, Z., 2014. Estudio evolutivo de la investigacin espaola con clulas madre. Visualizacin e identificacin de las principales lneas de investigacin. El Profesional de la Informacin, 23(3), 259-271.

Carr, B., 1979. Graphs and Networks. Clarendon, Oxford.

Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N., 2012. Time-varying graphs and dynamic networks. International Journal of Parallel, Emergent and Distributed Systems, 27(5), 387-408.

Corman, S.R., Kuhn, T., McPhee, R.D., Dooley, K.J., 2002. Studying complex discursive systems: Centering resonance analysis of communication. Human Communication Research, 28(2), 157-206.

Fletcher, J.G., 1980. A more general algorithm for computing closed semiring costs between vertices of a directed graph. CACM 23, 350-351.

Franzosi, R., 1997. 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(2-3), 275-304.

Freeman, L.C., 1978. Centrality in Social Networks; Conceptual Clarification. Social Networks 1, 215-239.

George, B., Kim, S., Shekhar, S., 2007. Spatio-temporal Network Databases and Routing Algorithms: A Summary of Results. in D. Papadias, D. Zhang, and G. Kollios (Eds.): SSTD 2007, LNCS 4605, Springer-Verlag, Berlin, Heidelberg, 460-477.

Gondran, M., Minoux, M., 2008. Graphs, Dioids and Semirings – New Models and Algorithms. Springer.

Guerrero-Bote, V.P., Zapico-Alonso, F., Espinosa-Calvo, M.E., Crisstomo, R.G., de Moya-Anegn, F., 2006. Binary Pathfinder: An improvement to the Pathfinder algorithm. Information Processing and Management, 42 (6), 1484-1490.

Gulys, L., Kampis, G., Legendi, R.O., 2013. Elementary models of dynamic networks. The European Physical Journal, Special Topics 222, 13111333.

Holme, P., Saramäki, J., 2012. Temporal networks. Physics Reports 519 (3), 97125.

Holme, P., Saramäki, J. (Eds.), 2013. Temporal Networks. Understanding Complex Systems. Springer.

Kim, H., Yoon, J.W., Crowcroft, J., 2012. Network analysis of temporal trends in scholarly research productivity. Journal of Informetrics **6**, 97 110

Kolaczyk, E.D., 2009. Statistical Analysis of Network Data: Methods and Models. New York: Springer.

Kontoleon, N., Falzon, L., Pattison, P., 2013. Algebraic structures for dynamic networks. Journal of Mathematical Psychology, **57**(6), 310319.

Moody, J., 2002. The Importance of Relationship Timing for Diffusion. Social Forces, **81**(1), 25-56.

Moody, J., McFarland, D., Bender-deMoll, S., 2005. Dynamic Network Visualization. American Journal of Sociology, **110**(4), 1206-1241.

de Nooy, W., Mrvar, A., Batagelj, V., 2012. *Exploratory Social Network Analysis with Pajek* (Structural Analysis in the Social Sciences), revised and expanded second edition. Cambridge University Press, Cambridge.

Praprotnik, S., Batagelj, V., 2015. Spectral centrality measures in temporal networks. Accepted in Ars Mathematica Contemporanea.

Riordan, J., 1958. Introduction to combinatorial analysis. New York: Wiley.

Schvaneveldt, R. W., Dearholt, D. W., Durso, F. T., 1988. Graph theoretic foundations of Pathfinder networks. Comput. Math. Applic. **15**(4), 337-345.

Schvaneveldt, R.W. (Ed.), 1990. Pathfinder Associative Networks: Studies in Knowledge Organization. Norwood, NJ: Ablex.

Xuan, B.B., Ferreira, A., Jarry, A., 2003. Computing shortest, fastest, and foremost journeys in dynamic networks. International Journal of Foundations of Computer Science, **14**(2), 267285.

View publication stats