A modeling environment for dynamic and adaptive network models implemented in matlab

Mohammadi Ziabari, S. Sahand; Treur, Jan

published in
Fourth International Congress on Information and Communication Technology
2020

DOI (link to publisher)
10.1007/978-981-15-0637-6_8

document version
Publisher's PDF, also known as Version of record

A Modeling Environment for Dynamic and Adaptive Network Models Implemented in MATLAB

S. Sahand Mohammadi Ziabari and Jan Treur

Abstract In this paper, a software environment to support Network-Oriented Modeling is presented. The environment has been implemented in MATLAB. This code covers the principles of temporal-causal network models. The software environment has built-in options for network adaptation principles such as the Hebbian learning principle from neuroscience and the adaptation principle for bonding based on homophily from social science. The implementation is illustrated for an adaptive temporal-causal network model under acute stress for decision-making.

Keywords Network-oriented modeling · Temporal-causal network · Adaptive · Software environment · Hebbian learning · Bonding by homophily · MATLAB

1 Introduction

In this paper, a dedicated software environment to support Network-Oriented Modeling is presented. The Network-Oriented Modeling approach addressed uses temporal-causal network models. This means that any scientific field in which causal relations are used to explain hypotheses, findings, and theories can be used in Network-Oriented Modeling [1]. Such domains vary from mental processes in individuals to social processes. For example, the interactions among individuals can be modeled as a network taking into account a network adaption principle like bonding based on homophily principle [2, 3]. Individual mental processes can be modeled as an interaction between mental states taking into account a network adaption principle based on Hebbian learning [4]. The latter represents the notion of plasticity described in Neuroscience which means that the communications within the brain are often adaptive and change over time.

S. S. Mohammadi Ziabari (✉) · J. Treur
Behavioural Informatics Group, Vrije Universiteit Amsterdam, Amsterdam, Netherlands
e-mail: sahandmohammadiziabari@gmail.com

J. Treur
e-mail: j.treur@vu.nl

X.-S. Yang et al. (eds.), *Fourth International Congress on Information and Communication Technology*, Advances in Intelligent Systems and Computing 1041, https://doi.org/10.1007/978-981-15-0637-6_8

There are two different representations of a temporal-causal network model named a conceptual representation (labeled graph or matrix representation) and a numerical representation (representation by difference or differential equations). Using the software environment presented here, a conceptual representation can be used as a basis. By the software, it is automatically translated into a numerical representation, which can be used for numerical simulation, mathematical analysis, validation by comparing to empirical data or properties, and tuning of parameters to characteristics of domain, person, or social context.

The example model presented in [5] has been used as an illustration. This model incorporates adaptation principles based on Hebbian learning and on suppression of connections due to acute stress. There are other implementations for other types of network-oriented modeling, some of which can be found in [6–11].

The sections of paper are as follows. In Sect. 2, the Network-Oriented Modeling approach based on temporal-causal networks is briefly described. In Sect. 3, modeling a temporal-causal network in **MATLAB** is introduced, and in Sect. 4 an Illustration for an example network model has been described. Finally, Sect. 5 is the discussion section.

2 The Network-Oriented Modeling Approach Addressed

This software environment covers the principles of Network-Oriented Modeling based on temporal-causal networks discussed in the book [12]. The Network-Oriented Modeling format used is based on a dynamic and adaptive variant of modeling, reasoning, and simulation in a causal way which is a topic with a long history in artificial intelligence [13]. In this respect, any scientific field of study in which causal relations are applied can be addressed on the basis of this Network-Oriented Modeling approach. Among the wide variety of application areas, there are two types of applications that in a sense are dominant: describing individual mental processes specifically and describing how individuals interact with each other [1]. Table 1 shows the overview of some combination functions. The following three notions are central

Table 1 Overview of some combination functions $c(V_1, \ldots, V_k)$

Name	Description	Formula $c(V_1, \ldots, V_k) =$
sum(…)	Sum	$V_1 + \cdots + V_k$
$\mathbf{ssum}_\lambda(\ldots)$	Scaled sum function	$\frac{V_1+\cdots+V_k}{\lambda}$ with $\lambda > 0$
min(…) **max(…)**	Minimal value Maximal value	$\mathrm{Min}(V_1, \ldots, V_k)$ $\mathrm{Max}(V_1, \ldots, V_k)$
$\mathbf{slogistic}_{\sigma\tau}(\ldots)$	Simple logistic sum function	$\frac{1}{1+e^{-\sigma(V_1+\cdots+V_k-\tau)}}$ with $\sigma, \tau \geq 0$
$\mathbf{alogistic}_{\sigma,\tau}(\ldots)$	Advanced logistic sum function	$\left[\frac{1}{1+e^{-\sigma(V_1+\cdots+V_k-\tau)}} - \frac{1}{1+e^{-\sigma\tau}}\right](1+e^{\sigma\tau})$ with $\sigma, \tau \geq 0$

in the Network-Oriented Modeling approach and define a temporal-causal network model, and therefore are part of a conceptual representation of a temporal-causal network model [14]:

- **Connections strength $\omega_{X,Y}$**

The connection strength between a state X to a state Y is called weight value $\omega_{X,Y}$ which is normally between 0 and 1.

- **Aggregation of impacts of states $\mathbf{c}_Y(..)$**

Each state needs a combination function $\mathbf{c}_Y(..)$ to aggregate the impacts of other states on state Y.

- **Speed of change of a state η_Y**

There is a speed factor η_Y shows how fast a state changes over a period of time based on the impact.

A conceptual representation of a temporal-causal network model can be transformed in a systematic or automated manner into a numerical representation of the model as follows [1, 12]:

- $Y(t)$ represents the value of Y at time point t in the model which is in the interval [0, 1].
- $\mathbf{impact}_{X,Y}(t) = \omega_{X,Y}X(t)$ shows the influence of a state X connected to a state Y at time point t where $\omega_{X,Y}$ represents the weight of the connection.
- The *aggregated impact* of some states X_i on Y at t is calculated using a *combination function* $\mathbf{c}_Y(..)$:

$$\begin{aligned}\mathbf{aggimpact}_Y(t) &= \mathbf{c}_Y(\mathbf{impact}_{X_1,Y}(t), \ldots, \mathbf{impact}_{X_k,Y}(t)) \\ &= \mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \ldots, \omega_{X_k,Y}X_k(t))\end{aligned}$$

- The impact of $\mathbf{aggimpact}_Y(t)$ on Y is applied over time gently, based on speed factor η_Y:

$$Y(t + \Delta t) = Y(t) + \eta_Y[\mathbf{aggimpact}_Y(t) - Y(t)]\Delta t$$

or

$$\mathbf{d}Y(t)/\mathbf{d}t = \eta_Y[\mathbf{aggimpact}_Y(t) - Y(t)]$$

- Therefore, the *difference* and *differential equations* for Y are achieved:

$$\begin{aligned}Y(t + \Delta t) &= Y(t) + \eta_Y[\mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \ldots, \omega_{X_k,Y}X_k(t)) - Y(t)]\Delta t \\ \mathbf{d}Y(t)/\mathbf{d}t &= \eta_Y[\mathbf{c}_Y(\omega_{X_1,Y}X_1(t), \ldots, \omega_{X_k,Y}X_k(t)) - Y(t)]\end{aligned}$$

Adaptation principles covered
The following adaptation principles are covered.

Hebbian learning
For *Hebbian learning* of a connection from state X_i to state X_j, the following model is used

$$\omega(t + \Delta t) = \omega(t) + \eta_\omega[\mathbf{c}_\omega(X_i(t), X_j(t), \omega(t)) - \omega(t)]\Delta t$$

with

$$\mathbf{c}_\omega(V_1, V_2, W) = \mathbf{hebb}_\mu(V_1, V_2, W) = V_1 V_2(1-W) + \mu W$$

where μ is the persistence factor with 1 as full persistence.

State-connection modulation
For the adaptation principle for *state-connection modulation* with control state cs, the following model is used:

$$\omega(t + \Delta t) = \omega(t) + \eta_\omega[\mathbf{c}_\omega(\mathrm{cs}_2(t), \omega(t)) - \omega(t)]\Delta t$$

with

$$\mathbf{c}_\omega(V, W) = \mathbf{scm}_\alpha(V, W) = W + \alpha V W(1-W)$$

where α is the adjustment parameter for ω from cs. In combination, these two adaptive combination functions can be used as a weighted average with $0 \leq \theta \leq 1$ as follows:

$$\mathbf{c}_\omega(V_1, V_2, V, W) = \theta\, \mathbf{hebb}_\mu(V_1, V_2, W) + (1 - \theta)\, \mathbf{scm}_\alpha(V, W)$$
$$\omega(t + \Delta t) = \omega(t) + \eta_\omega[\mathbf{c}_\omega(X_i(t), X_j(t), \mathrm{cs}(t), \omega(t)) - \omega(t)]\, \Delta t$$

All these difference equations can be used for simulation.

This state-connection adaptation principle can also be applied in a social context. The hypothesis is based on that whenever a more intensive interplay between two persons occurs, the connection will become solid, e.g., [15].

Bonding based on homophily
Bonding based on homophily shows that the more look like the states of two connected states, the stronger their connection will become: 'the more you are alike, the more you like (each other)' [12]; see, for example, [2, 16, 17]. When also the states are assumed dynamic, this principle can be combined with contagion of states into a circular causal relation [18]: State ↔ Link. See also, for example [19–22]. The homophily principle can be as represented numerically by a combination function $\mathrm{c}_{A,B}(V_1, V_2, W)$ as follows:

$$\omega_{A,B}(t+\Delta t) = \omega_{A,B}(t) + \eta_{A,B}[c_{A,B}(X_A(t), X_B(t), \omega_{A,B}(t)) - \omega_{A,B}(t)]\,\Delta t$$
$$\mathrm{d}Y(t)/\mathrm{d}t = \eta_{A,B}[c_{A,B}(X_A(t),\ X_B(t),\ \omega_{A,B}) - \omega_{A,B}]$$

Three variants of models for the homophily axiom are the linear, quadratic, and logistic variants:

Linear

$$c(V_1,\ V_2,\ W) = \mathbf{slhomo}(V_1,\ V_2,\ W) = W + W(1-W)(\tau - |V_1 - V_2|)$$

Quadratic

$$c(V_1,\ V_2,\ W) = \mathbf{sqhomo}(V_1,\ V_2,\ W) = W + W(1-W)(\tau^2 - (V_1 - V_2)^2)$$

Logistic

$$\begin{aligned} c(V_1,\ V_2,\ W) =& \mathbf{sloghomo}(V_1,\ V_2,\ W) \\ =& W + W(1-W)(0.5 - 1/(1+\mathrm{e}^{-\sigma(|V_1 - X_2| - \tau)})) \end{aligned}$$

Based on these options that can be chosen the following numerical differential and difference equations are generated

$$\mathrm{d}\omega_{C,D}/\mathrm{d}t = \eta_{C,D}\,\omega_{C,D}(1-\omega_{C,D})(\tau_{C,D} - |X_C - X_D|)$$
$$\omega_{C,D}(t+\Delta t) = \omega_{C,D} + \eta_{C,D}\,\omega_{C,D}(t)\,(\tau_{C,D} - |X_C - X_D|)\,\Delta t$$

$$\mathrm{d}\omega_{C,D}/\mathrm{d}t = \eta_{C,D}\,\omega_{C,D}(1-\omega_{C,D})(\tau_{C,D}^2 - |X_C - X_D|)^2)$$
$$\omega_{C,D}(t+\Delta t) = \omega_{C,D} + \eta_{C,D}\,\omega_{C,D}(t)(1-\omega_{C,D})(\tau_{C,D}^2 - (X_A(t) - X_B(t))^2)\,\Delta t$$

$$\mathrm{d}\omega_{C,D}/\mathrm{d}t = \eta_{C,D}\,\omega_{C,D}(1-\omega_{C,D})(0.5 - 1/(1+\mathrm{e}^{-\sigma(|X_C - X_D| - \tau_{C,D})}))$$
$$\omega_{C,D}(t+\Delta t) = \omega_{C,D} + \eta_{A,B}\,\omega_{C,D}(t)(1-\omega_{C,D})(0.5 - 1/(1+\mathrm{e}^{-\sigma(|X_C - X_D| - \tau_{C,D})}))\Delta t$$

Here, X_C, X_D are the states of person C and D; $\omega_{C,D}$ is the connection weight from person C to person D, $\eta_{C,D}$ the update speed factor for the connection from person C to person D, and $\tau_{C,D}$ the threshold or tipping point for connection adaption.

3 Modeling a Temporal-Causal Network in MATLAB

The advantage of using M**ATLAB** for simulation is that (as the abbreviation of that says) 'Matrix laboratory' can easily work with matrices. The format is defined in Table 2, if there is a connection between X_1 and X_2 as states, the connection weight assigned to the matrix representation (`w`) of states between aforementioned states.

Table 2 Notions and MATLAB representations

Notions	MATLAB representations
Number of nodes (states)	`N`
The notion of states ($X_1, X_2, \ldots$) and weights ω among states	`W`
The speed factors	`Sp_f`
Initial values	`STDX`
Combination function (identity)	First row of matrix '`O`' `id=O(1,:);`
Combination function (sum)	Second row of matrix '`O`' `sum function`

Combination function (scaled sum)	3rd and fourth rows of matrix '`O`'
	`Scaled sum,`
	`Scaling factor`

Combination function (normalized sum)	5th and 6th rows of matrix '`O`' `normalised sum, normalizing factor`
Combination function (adaptive normalized)	7th row of matrix '`O`' `adnorsum`
Combination function (simple logistic)	8th, 10th and 11th rows of matrix '`O`' `slogistic(…)`
Combination function (advance logistic)	9th and 10th and 11th rows of matrix '`O`' `Alogistic, steepness, threshold`
Combination function (adaptive advanced logistic)	12th and 13th and 14th rows of matrix '`O`' `adaptive advanced logistic, steepness, steepness`

(continued)

Table 2 (continued)

Notions	MATLAB representations
Speed factor for connection weight adaptation (used for Hebbian learning, homophily, and state-connection modulation)	`eta`
Hebbian learning principle	`hebb, mu`
Homophily principle (simple linear homophily)	`slhomo, htau`
Homophily principle (advanced linear homophily)	`alhomo, htau`
Homophily principle (simple quadratic homophily)	`sqhomo, htau`
Homophily principle (advanced quadratic homophily)	`aqhomo, htau`
State-connection modulation	`Adcon, amp`
Length of time for simulation	`time=0:dt:398;` `L=length(time);`
Δt	`dt`
Plotting (figures)	`plot(time(1:230),STDX(1:230,i),'lineWidth',3)`
RMS (root mean square)	`RMS =` `sqrt (nansum ((Output - emp_data) .^2)) / (col * row)`

Due to simplicity of M**ATLAB** and also providing many functions, the M**ATLAB** software became one of the often-used software environments for engineers and computer science developers. It has been used in different aspects of sciences from image processing, due to providing many toolboxes, and also machine learning, analyze and simulates the behavioral dynamics of agents in cognitive science, social science, and in artificial intelligence.

The initialization of the matrices for doing actions, calculating based on the combination functions (identity, advance logistic, advance advanced logistic, scaled sum…), MATLAB representation of functions based on notations of states, relations, and all principles are shown in Table 4 in the appendix.

In Fig. 1, the functional view of the MATLAB code is shown. The process starts with the inputs named number of states (nodes), connection weights, speed factors, and initial values. In the next step, all parameters are allocated in matrices with $1 \times N$ dimensions, where N is the number of the states in the model.

The next phase is allocating the primitive values of the states in the first row of the matrix (STDX) and then specifying the time period for having simulation. If there is a Hebbian learning in the model, then parameters of Hebbian learning, η, hebb, and μ are allocated in three different matrices, similarly for the homophily principle, simple homophily (slhom), advanced homophily (alhom) simple quadratic homophily (sqhom) and advanced quadratic homophily (qhom), threshold (hthau), and finally for state-connection modulation (scm_a) and the number of the states which have modulation ability. Figure 1 illustrates the functional view of written MATLAB code.

Then in the next step, if there is any above-mentioned principle in the model they will multiply by the STDX matrix formed already with time. Meanwhile, the matrix called condy with the weights of states formed in a column-wise order and then make a new row based on any principle existed and then add the influence of them in the matrix. Finally, all matrices with STDX and condy (if there was any state-connection modulation) result in generating the simulation.

The human interaction flowchart is depicted in Fig. 2. As it can be seen, a first step is initialization as providing inputs, for instance, number of states, weights among states, speed factors, and combination functions. In the second step, it is needed to be decided in which system the user wants to work. In this phase, there are two systems, multi-agent system and single-agent system. The former offers simple linear, advanced linear, simple quadratic, advanced quadratic homophily principles, and for latter, there are Hebbian learning and state-connection suppression, and finally, the plotting of simulation occurs.

And for more specification, the flowchart of the code is presented in Fig. 3.

To enable easy use, one can also use a user interface between Excel and MATLAB. As MATLAB works with matrices it might be easier to use an Excel interface of matrices and just read the matrices from the Excel file and then do the execution in MATLAB. Such a matrix expresses the parameters, combination functions, identity function, sum function, scaled sum with scale factor, normalized with normalizing factor, adaptive normalized sum, simple logistic, advanced logistic, advanced logistic and adaptive advanced logistic function with steepness, and threshold and other

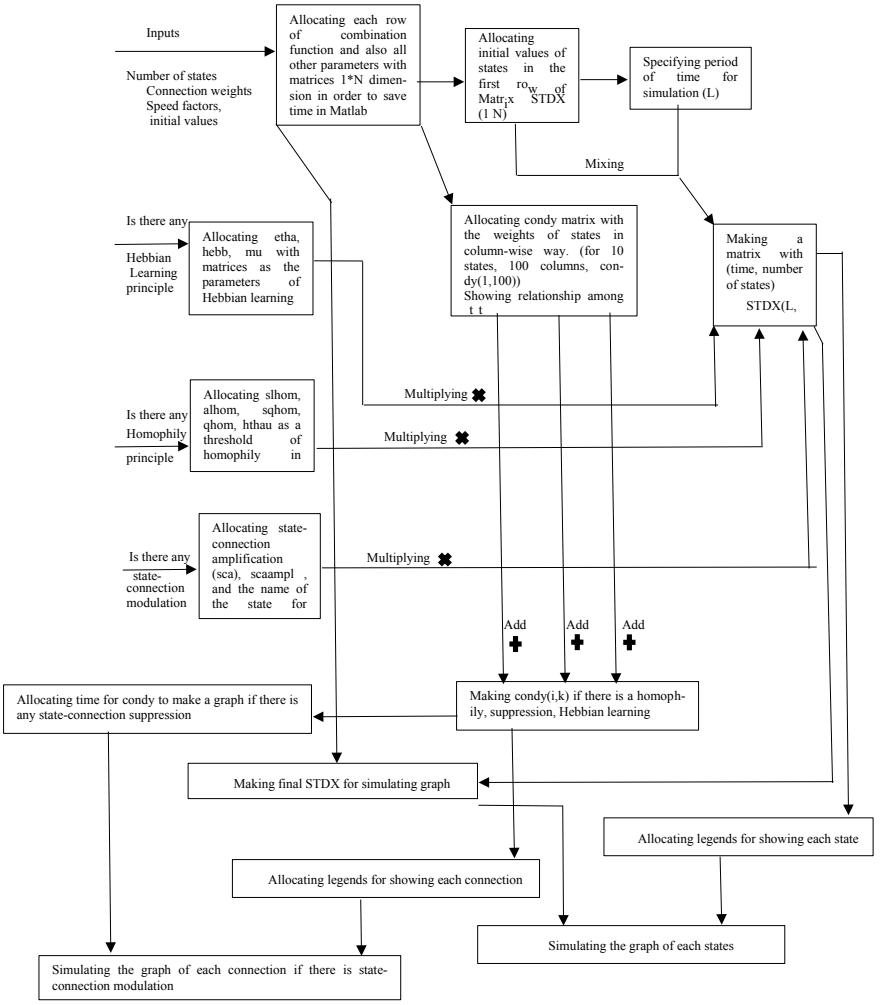


Fig. 1 Functional view of the MATLAB process

principles. As can be seen from Figs. 4 and 5, the MATLAB code reads the matrices based on the matrix representation in Excel; for instance, in these figures, it would be from column C1 to L32 to read matrix for all weights of states, speed factors, deltaT, maxt, combination functions, and finally initial values from the first sheet and if there is any principle combined with the model using the second sheet to read from Excel for Hebbain learning principle and homophily for their principles. Figures 4 and 5 show this option in Excel sheets.

Parameter tuning using the sum of squared residuals and root mean square

For comparison between empirical data and simulation results and optimization of

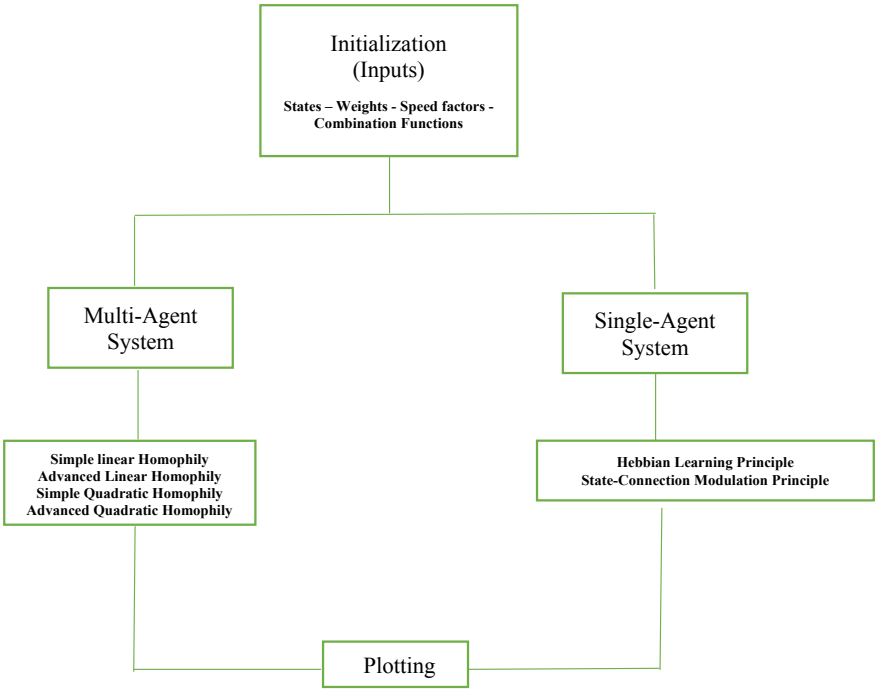


Fig. 2 Human interaction flowchart

parameters, MATLAB components are available; the sum of squared residuals (SSR) has been implemented to calculate the difference.

$$\mathrm{SSR} = ((X(t_1) - Y(t_1))^2 + \cdots + (X(t_N) - Y(t_N))^2)$$

$$\mathrm{RMS} = \sqrt{\frac{\mathrm{SSR}}{N}} = \sqrt{\frac{(X(t_1) - Y(t_1))^2 + \cdots + (X(t_N) - Y(t_N))^2}{N}}$$

```
% loading empirical data

load('Data1.mat', 'Data1');
[row, col]=size(Data1);

% Calculating Root Mean Square
RMS = sqrt (nansum ((Output - emp_data) .^2 )) / (col * row)
```

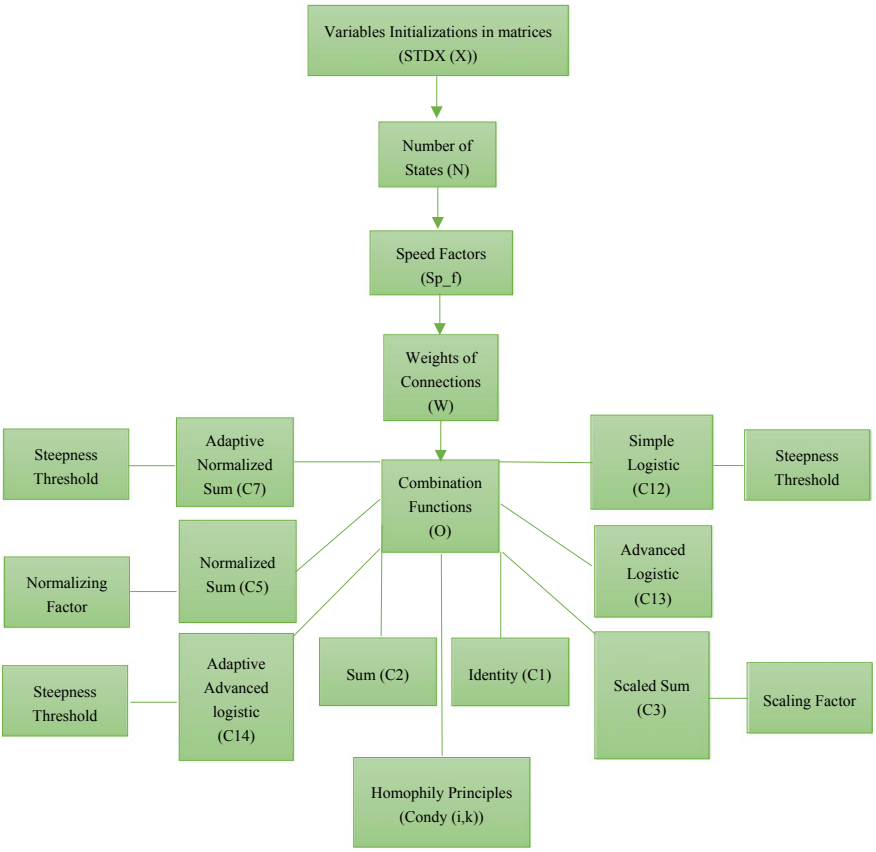
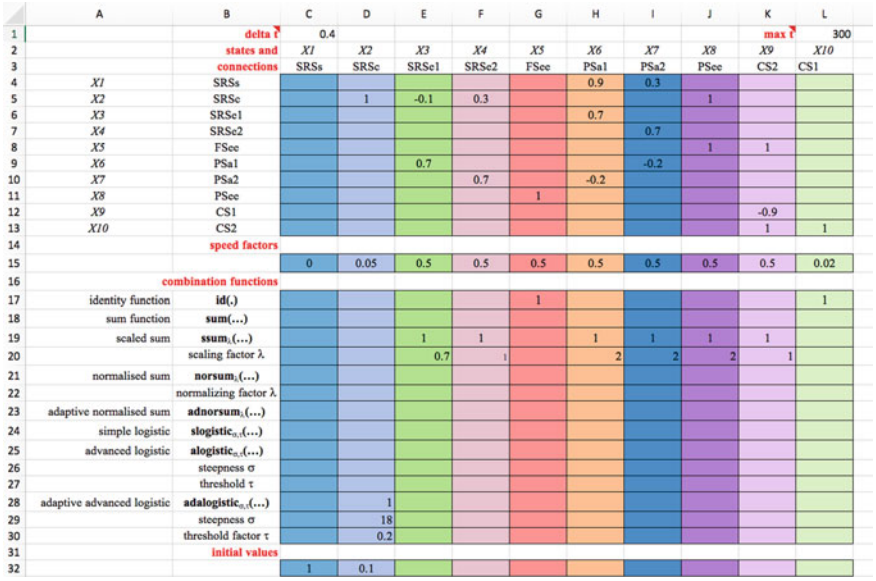


Fig. 3 Structural view of the code

4 Illustration for an Example Network Model

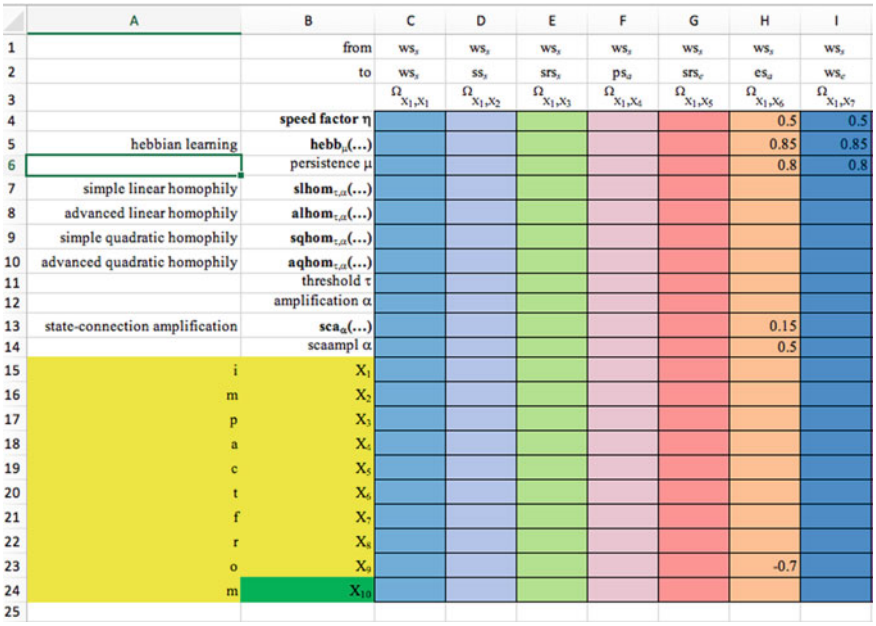
It is shown how the model presented in [5] can be executed in MATLAB. The conceptual representation of temporal-causal network of the model used in [5] is illustrated in Fig. 5, and the explanation of states has been shown in Table 3. As can been seen from Fig. 5, here, both the Hebbian learning principle and state-connection modulation were used. The Hebbian learning principle used between states srs_s and (ps_{a1} and ps_{a2}) and also state-connection suppression between state cs_2 and the connections with Hebbian learning principle. The number of states considered to be 10. Figure 7 shows simulation result of weights of states and Fig. 8 shows state-connection suppression (Fig. 6).

In Fig. 6 shows temporal-causal network model. An overview of explanation of the states is illustrated in Table 3.



	A	B	C	D	E	F	G	H	I	J	K	L
1		delta t	0.4								max t	300
2		states and	X1	X2	X3	X4	X5	X6	X7	X8	X9	X10
3		connections	SRSs	SRSc	SRSe1	SRSe2	FSee	PSa1	PSa2	PSee	CS2	CS1
4	X1	SRSs						0.9	0.3			
5	X2	SRSc		1	-0.1	0.3				1		
6	X3	SRSe1						0.7				
7	X4	SRSe2							0.7			
8	X5	FSee								1	1	
9	X6	PSa1			0.7				-0.2			
10	X7	PSa2				0.7		-0.2				
11	X8	PSee					1					
12	X9	CS1									-0.9	
13	X10	CS2									1	1
14		speed factors										
15			0	0.05	0.5	0.5	0.5	0.5	0.5	0.5	0.5	0.02
16		combination functions										
17	identity function	id(.)					1					1
18	sum function	sum(…)										
19	scaled sum	$ssum_{\lambda}(…)$			1	1		1	1	1	1	
20		scaling factor λ			0.7	1		2	2	2	1	
21	normalised sum	$norsum_{\lambda}(…)$										
22		normalizing factor λ										
23	adaptive normalised sum	$adnorsum_{\lambda}(…)$										
24	simple logistic	$slogistic_{\sigma,\tau}(…)$										
25	advanced logistic	$alogistic_{\sigma,\tau}(…)$										
26		steepness σ										
27		threshold τ										
28	adaptive advanced logistic	$adalogistic_{\sigma,\tau}(…)$		1								
29		steepness σ		18								
30		threshold factor τ		0.2								
31		initial values										
32			1	0.1								

Fig. 4 Excel interface to read for MATLAB programming (parameters, speed factor, combination function, and initial values)



	A	B	C	D	E	F	G	H	I
1		from	ws_s	ws_s	ws_s	ws_s	ws_s	ws_s	ws_s
2		to	ws_s	ss_s	srs_s	ps_a	srs_e	es_a	ws_e
3			$\Omega_{X1,X1}$	$\Omega_{X1,X2}$	$\Omega_{X1,X3}$	$\Omega_{X1,X4}$	$\Omega_{X1,X5}$	$\Omega_{X1,X6}$	$\Omega_{X1,X7}$
4		speed factor η						0.5	0.5
5	hebbian learning	$hebb_{\mu}(…)$						0.85	0.85
6		persistence μ						0.8	0.8
7	simple linear homophily	$slhom_{\tau,\alpha}(…)$							
8	advanced linear homophily	$alhom_{\tau,\alpha}(…)$							
9	simple quadratic homophily	$sqhom_{\tau,\alpha}(…)$							
10	advanced quadratic homophily	$aqhom_{\tau,\alpha}(…)$							
11		threshold τ							
12		amplification α							
13	state-connection amplification	$sca_{\alpha}(…)$						0.15	
14		scaampl α						0.5	
15	i	X_1							
16	m	X_2							
17	p	X_3							
18	a	X_4							
19	c	X_5							
20	t	X_6							
21	f	X_7							
22	r	X_8							
23	o	X_9						-0.7	
24	m	X_{10}							
25									

Fig. 5 Excel interface to read for MATLAB programming (Hebbian, homophily, and suppression principles)

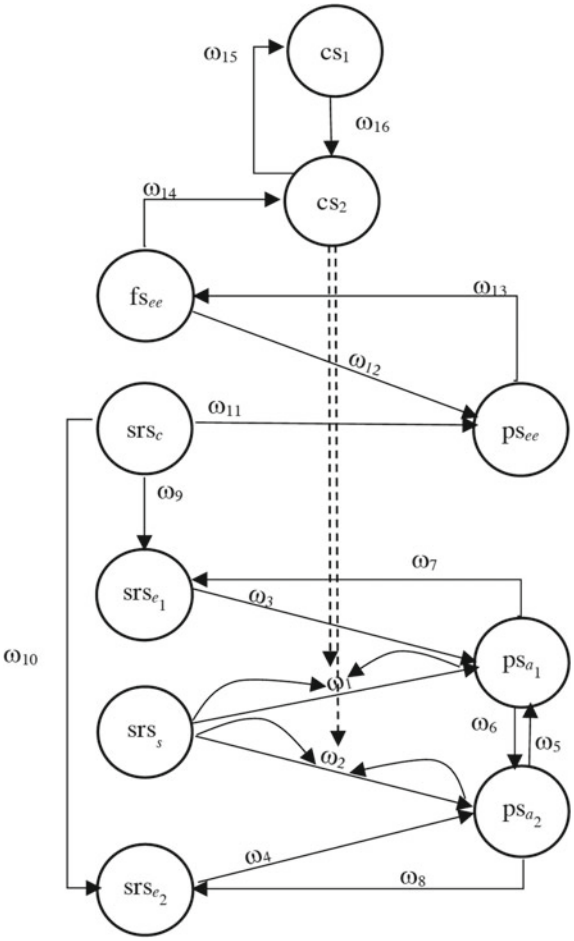


Fig. 6 Adaptive temporal-causal network model's conceptual representation [5]

Table 3 States explanations in the model [5]

X_1	Sensory representation of stimulus s
X_2	Sensory representation of context c
X_3	Sensory representation of action effect e_1
X_4	Sensory representation of action effect e_2
X_5	Feeling state for extreme emotion ee
X_6	Preparation state for action a_1
X_7	Preparation state for action a_2
X_8	Preparation state for response of extreme emotion ee
X_9	Control state for timing of suppression of connections
X_{10}	Control state for suppression of connections

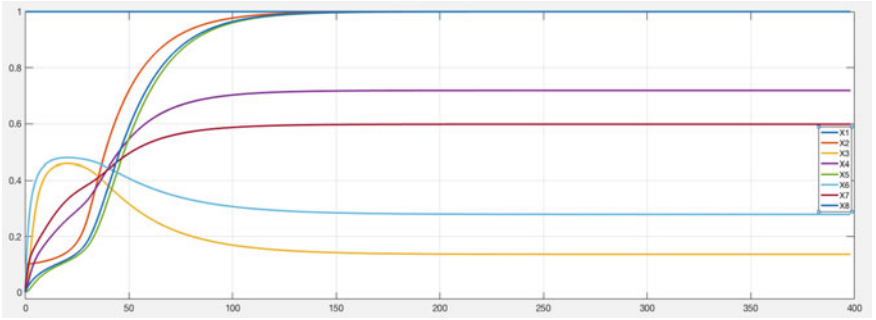


Fig. 7 Simulation outcome of presented model: states

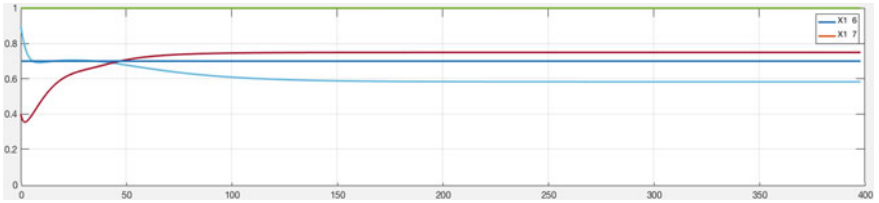


Fig. 8 Simulation outcome for suppression and Hebbian learning for ω_1 (connection X_1–X_6) and ω_2 (connection X_1–X_7)

```
%   SRSs  SRSc   SRSe1   SRSe2  FSee    PSa1    PSa2   PSee    CS1   CS2
%    X1    X2      X3      X4     X5      X6      X7     X8      X9    X10
W=[  0     0       0       0      0       0.9     0.4    0       0     0    %X1  SRSs
     0     1      -0.1     0.3    0       0       0      1       0     0     %X2  SRSc
     0     0       0       0      0       0.7     0      0       0     0     %X3  SRSe1
     0     0       0       0      0       0       0.7    0       0     0     %X4  SRSe2
     0     0       0       0      0       0       0      1       0     0     %X5  FSee
     0     0       0.7     0      0       0       -0.2   0       0     0     %X6  PSa1
     0     0       0       0.7    0       -0.2    0      0       0     0     %X7  PSa2
     0     0       0       0      1       0       0      0       0     0     %X8  PSee
     0     0       0       0      0       0       0      0       0     0     %X9  CS1
     0     0       0       0      0       0       0      0       0     -0.9  %X10 CS2
     ];
```

```
Sp_f=[ 0    0.05    0.5    0.5    0.5    0.5    0.5    0.4    0.02   0.6];

O=[    0     0       0      0      1      0      0      0      1      0   % identity function id(.)

       0     0       0      0      0      0      0      0      0      0   % sum function sum (...)

       0     0       1      1      0      1      1      1      0      1   % Scaled sum sum (...)

       0     0     0.7      1      0      2      2      2      0      1   % Scaling factor

       0     0       0      0      0      0      0      0      0      0   % normalised sun norsum(...)

       0     0       0      0      0      0      0      0      0      0   % normalizing factor

       0     0       0      0      0      0      0      0      0      0   % adnorsum

       0     0       0      0      0      0      0      0      0      0   % slogistic(...)

       0     0       0      0      0      0      0      0      0      0   % alogistic(...)

       0     0       0      0      0      0      0      0      0      0   % steepness

       0     0       0      0      0      0      0      0      0      0   % threshold

       0     1       0      0      0      0      0      0      0      0 % adaptive advanced logistic (...)

       0    18       0      0      0      0      0      0      0      0   % steepness

       0   0.2       0      0      0      0      0      0      0      0   % threshold factor
```

```
% Suppression among connections in Hebbian learning
adcon(13,6)=0.15;
adcon(13,7)=0.15;
adcon(14,6)=0.5;
adcon(14,7)=0.5;
adcon(24,6)=-0.7;
adcon(24,7)=-0.7;


% Hebbian learning among states 1,6
eta(1,6)=0.5;
eta(1,7)=0.8;

hebb(1,6)=0.85;
hebb(1,7)=0.85;

mu(1,6)=0.8;
mu(1,7)=0.8;

% Assign time for plotting
dt=0.25;
time=0:dt:398;
L=length(time);
STDX=zeros(L,N);

% Assign the initialization for states 1 and 2 (can be for any states)
STDX(1,1)=1;
STDX(1,2)=0.1;
```

Figure 9 shows a difference between simulation result and empirical data provided. And the result is 0.01309.

5 Discussion

The implementation of a dedicated MATLAB-based software environment for Network-Oriented Modeling has been described. The modeling approach covered can be found in [12]; see also [1]. This implementation has been used in dynamic and adaptive network-oriented modeling. The environment was illustrated for the example model described in [5] for which previously only an Excel-based model was available.

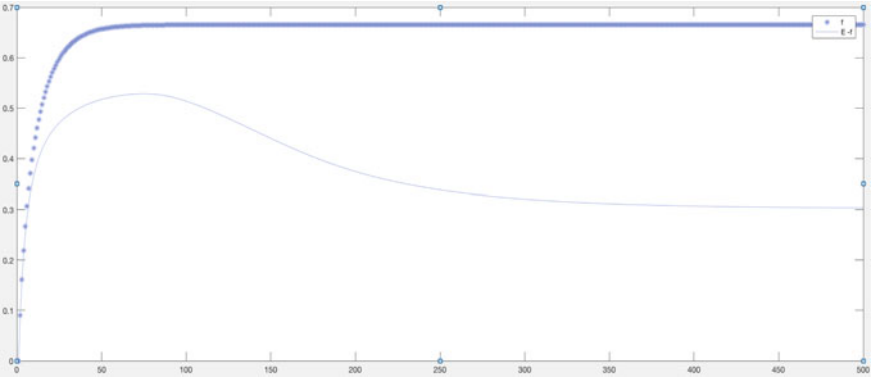


Fig. 9 Simulation result for empirical data (X_1) and simulation result (X_2)

An important advantage of the software environment is that modeling can take place at the level of conceptual representations expressed as labeled graphs or matrices. Therefore, it is suitable in a multidisciplinary context where different disciplines play a role, also disciplines where technical knowledge from computer science or AI is minimal. The more technical numerical representations and the actual execution are taken care of by the software environment, and therefore for users, no programming skills are needed.

Acknowledgements We would like to thank our colleague Fakhra Jabeen, Ph.D. candidate at Vrije Universiteit Amsterdam, for her assistance with making possible to have an Excel interface with current MATLAB code.

Appendix: Inputs Description of the MATLAB Code

```
% Initializing the matrices with zeroe in order to get less time in operating the main codes
   id=zeros(1,N);
   sum=zeros(1,N);
   ssum=zeros(1,N);
   lambda=zeros(1,N);
   norsum=zeros(1,N);
   norlambda=zeros(1,N);
   adnorsum=zeros(1,N);
   slog=zeros(1,N);
   alog=zeros(1,N);
   s=zeros(1,N);
   t=zeros(1,N);
   adalog=zeros(1,N);
```

```
            adas=zeros(1,N);
            adat=zeros(1,N);
            id=O(1,:);
            sum=O(2,:);
            ssum=O(3,:);
            lambda=O(4,:);
            norsum=O(5,:);
            norlambda=O(6,:);
            adnorsum=O(7,:);
            slog=O(8,:);
            alog=O(9,:);
            s=O(10,:);
            t=O(11,:);
            adalog=O(12,:);
            adas=O(13,:);
            adat=O(14,:);
            eta=zeros(N);
            hebb=zeros(N);
            mu=zeros(N);
            slhomo=zeros(N);
            alhomo=zeros(N);
            sqhomo=zeros(N);
            aqhomo=zeros(N);
            htau=zeros(N);
            amp=zeros(N);
            adcon=zeros(14+N, N^2);
clc
clear
close all
format long
N=10;
```

See Table 4.

Table 4 Notions of states and MATLAB representations of functions

Notions of states	MATLAB representations
Combination function (identity)	`C1=condy(i-1,k);`
Combination function (sum)	`C3=htau(ii,jj)-abs(STDX(i-1,ii)-STDX(i-1,jj));`
Combination function (scaled sum)	`C8=ssum(j)*C7/lambda(j);`
Combination function (normalized sum)	`C9=norsum(j)*C7/norlambda`
Combination function (adaptive normalized)	`C7=C7+condy(i-1,(jj-1)*N+j)*STDX(i-1,jj);`
Combination function (simple logistic)	`C12=slog(j)*(1/(1+exp(-s(j)*(C7-t(j)))));`
Combination function (advance logistic)	`C13=alog(j)*(((1/(1+exp(-s(j)*(C7-t(j)))))` `-(1/(1+exp(s(j)*t(j)))))*(1+exp(-s(j)*t(j))));`
Combination function (adaptive advance logistic)	`C14=adalog(j)*(((1/(1+exp(-adas(j)*(C7-adat(j)*C11))))-`
Hebbian learning principle	`condy(i-1,k)+eta(ii,jj)*(hebb(ii,jj)*(STDX(i-1,ii)` `*STDX(i-1,jj)*C2+mu(ii,jj)*C1)`
Homophily principles (simple linear homophily)	`slhomo(ii,jj)*(C1+amp(ii,jj)*C1*C2*C3)`
Homophily principles (advanced linear homophily)	`alhmo(ii,jj)*(C1+amp(ii,jj)*C2*((C4+C3)/2)+C1*((C4-C3)/2))`
Homophily principles (simple quadratic homophily)	`sqhomo(ii,jj)*(C1+amp(ii,jj)*C1*C2*C5)`
Homophily principles (advanced quadratic homophily)	`aqhmo(ii,jj)*(C1*amp(ii,jj)*C2*((C6+C5)/2)+C1*((C6-C5)/2))-C1)*dt;`
State-connection suppression	`adcon(13,k)*(C1+adcon(14,k)*(CC)*(C2)* C1)`

(continued)

Table 4 (continued)

Notions of states	MATLAB representations
All values for states	`STDX(i,j)=STDX(i-1,j)+Sp_f(j)*(aggimpact(i-1,j)-STDX(i-1,j))*dt;`

All values for connection weights	`condy(i,k)=condy(i-1,k)+eta(ii,jj)*(hebb(ii,jj)*(…`
	`STDX(i-1,ii)*STDX(i-1,jj)*C2+mu(ii,jj)*C1)+…`
	`adcon(13,k)*(C1+adcon(14,k)*(CC)*(C2)*…`
	`C1)+slhomo(ii,jj)*(C1+amp(ii,jj)*C1*C2*…`
	`C3)+alhomo(ii,jj)*(C1+amp(ii,jj)*C2*((C4+C3)/2)+C1*…`
	`((C4-C3)/2))+sqhomo(ii,jj)*(C1+amp(ii,jj)*C1*C2*C5)+…`
	`aqhomo(ii,jj)*(C1*amp(ii,jj)*C2*((C6+C5)/2)+C1*((C6-C5)/2))-…`
	`C1)*dt;`

RMS (root mean square)	% loading empirical data
	`load('Data1.mat', 'Data1');`
	`[row, col]=size(Data1);`
	% calculating root mean square
	`RMS = sqrt (nansum ((Output - emp_data) .^2)) / (col * row)`

References

1. J. Treur, The Ins and Outs of Network-Oriented Modeling: from biological networks and mental networks to social networks and beyond. *Transactions on Computational Collective Intelligence*. Paper for Keynote Lecture at ICCCI'18 (2018)
2. M. McPherson, L. Smith-Lovin, J.M. Cook, Birds of a feather homophily in social networks. Ann. Rev. Sociol. **27**, 415–444 (2001)
3. A.L. Barabási, R. Albert, Emergence of scaling in random networks. Science **286**, 509–512 (1999)
4. D. Hebb, *The Organization of Behavior* (Wiley, 1949)
5. J. Treur, S.S.M. Ziabari, An adaptive temporal-causal network model for decision making underacute stress, in *Proceedings of the 10th International Conference on Computational Collective Intelligence, ICCCI'18*, vol. 2, ed. by N.T. Nguyen. Lecture Notes in Computer Science, vol. 11056 (Springer, Berlin, 2018), pp. 13–25
6. S.S.M. Ziabari, J. Treur, Cognitive modelling of mindfulness therapy by autogenic training, in *Proceedings of the 5th International Conference on Information System Design and Intelligent Applications, INDIA'18*. Advances in Intelligent Systems and Computing (Springer, Berlin, 2018)
7. S.S.M. Ziabari, J. Treur, Integrative Biological, Cognitive and affective modeling of a drug-therapy for a post-traumatic stress disorder, in *Proceedings of the 7th International Conference on Theory and Practice of Natural Computing, TPNC'18* (Springer, Berlin, 2018)
8. S.S.M. Ziabari, J. Treur, Computational analysis of gender differences in coping with extreme stressful emotions, in *Proceedings of the 9th International Conference on Biologically Inspired Cognitive Architecture (BICA2018)* (Elsevier, Czech Republic, 2018)
9. S.S.M. Ziabari, Integrative cognitive and affective modeling of deep brain stimulation, in *Proceedings of the 32nd International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems (IEA/AIE 2019)* (submitted for publication)
10. S.S.M. Ziabari, J. Treur, An adaptive cognitive temporal-causal network model of a mindfulness therapy based on humor, in *International Conference on Computational Science (ICCS 2019)* (submitted for publication)
11. S.S.M. Ziabari, J. Treur, An adaptive cognitive temporal-causal network model of a mindfulness therapy based on music, in *Proceedings of the 10th International Conference on Intelligent Human Computer Interaction (IHCI2018)* (Springer, India, 2018)
12. J. Treur, *Network-Oriented Modeling: Addressing Complexity of Cognitive, Affective and Social Interactions* (Springer, 2016)
13. B.J. Kuipers, J.P. Kassier, How to discover a knowledge representation for causal reasoning by studying an expert physician, in *Proceedings English International Joint Conference on Artificial Intelligence, IJCAI'83*, ed. by F.R.G. Karlsruhe (William Kaufman, Los Altos, CA, 1983)
14. J. Treur, Verification of temporal-causal network models by mathematical analysis. Vietnam J. Comput. Sci. **3**, 207–221 (2016)
15. E. Pearce, J. Launay, R.I.M. Dunbar, The ice-breaker effect: singing together mediates fast social bonding. R. Soc. Open Sci. (2015). https://doi.org/10.1098/rsos.150221
16. D. Byrne, The attraction hypothesis: do similar attitudes affect anything? J. Pers. Soc. Psychol. **51**(6), 1167–1170 (1986)
17. A. Mislove, B. Viswanath, K.P. Gummadi, P. Druschel, You are who you know: inferring user profiles in online social networks, in *Proceedings of the WSDM'10*, New York City, New York, USA, 4–6 Feb 2010 (2010), pp. 251–260
18. M.E.J. Newman, The structure and function of complex networks. SIAM Rev. **45**, 167–256 (2003)
19. S. Aral, L. Muchnik, A. Sundararajan, Distinguishing influence-based contagion from Homophily driven diffusion in dynamic networks. Proc. Natl. Acad. Sci. USA **106**(2), 1544–1549 (2009)

20. C.R. Shalizi, A.C. Thomas, Homophily and contagion are generically confounded in observational social network studies. Sociol. Methods Res. **40**(2), 211–239 (2011)
21. C.E.G. Steglich, T.A.B. Snijders, M. Pearson, Dynamic networks and behavior: separating selection from influence. Sociol. Methodol. **40**, 329–393 (2010)
22. M.P. Mundt, L. Mercken, L.I. Zakletskaia, Peer selection and influence effects on adolescent alcohol use: a stochastic actor-based model. BMC Pediatr. **12**, 115 (2012)