BIG DATA MINING AND ANALYTICS
ISSN 2096-0654 06/25 pp637–654
DOI: 10.26599/BDMA.2024.9020010
Volume 7, Number 3, September 2024

Network Diffusion Framework to Simulate Spreading Processes in Complex Networks

Michał Czuba*, Mateusz Nurek, Damian Serwata, Yu-Xuan Qiu, Mingshan Jia, Katarzyna Musial, Radosław Michalski, and Piotr Bródka

Abstract: With the advancement of computational network science, its research scope has significantly expanded beyond static graphs to encompass more complex structures. The introduction of streaming, temporal, multilayer, and hypernetwork approaches has brought new possibilities and imposed additional requirements. For instance, by utilising these advancements, one can model structures such as social networks in a much more refined manner, which is particularly relevant in simulations of the spreading processes. Unfortunately, the pace of advancement is often too rapid for existing computational packages to keep up with the functionality updates. This results in a significant proliferation of tools used by researchers and, consequently, a lack of a universally accepted technological stack that would standardise experimental methods (as seen, e.g., in machine learning). This article addresses that issue by presenting an extended version of the Network Diffusion library. First, a survey of the existing approaches and toolkits for simulating spreading phenomena is shown, and then, an overview of the framework functionalities. Finally, we report four case studies conducted with the package to demonstrate its usefulness: the impact of sanitary measures on the spread of COVID-19, the comparison of information diffusion on two temporal network models, and the effectiveness of seed selection methods in the task of influence maximisation in multilayer networks. We conclude the paper with a critical assessment of the library and the outline of still awaiting challenges to standardise research environments in computational network science.

Key words: computational framework; seed selection; influence maximisation; spreading models; temporal networks; multilayer networks; network science; network control

1 Introduction

With the rapid growth of data generated by humankind in recent decades, the issue of their efficient processing becomes one of the main challenges that need to be addressed to boost experimental sciences. That especially concerns irregular structures, like graphs, that are much harder to enclose in robust frameworks, such as images, audio, or text. Computational network science[1], the primary discipline focusing on applications of graphs, has developed many tools that

• Michał Czuba, Mateusz Nurek, Damian Serwata, Radosław Michalski, and Piotr Bródka are with Department of Artificial Intelligence, Wrocław University of Science and Technology, Wrocław 50-370, Poland. E-mail: michal.czuba@pwr.edu.pl; mateusz.nurek@pwr.edu.pl; damian.serwata@pwr.edu.pl; radoslaw.michalski@pwr.edu.pl; piotr.brodka@pwr.edu.pl.
• Yu-Xuan Qiu, Mingshan Jia, and Katarzyna Musial are with Data Science Institute, University of Technology Sydney, Ultimo, NSW 2007, Australia. E-mail: yuxuan.qiu@uts.edu.au; mingshan.jia@uts.edu.au; katarzyna.musial-gabrys@uts.edu.au.
∗ To whom correspondence should be addressed.
Manuscript received: 2023-10-01; revised: 2024-01-04; accepted: 2024-02-20

facilitate research-oriented computations. They are used in many tasks, like epidemiology[2], social network analysis[3], recommendation systems[4, 5] or urban planning[6], to name just a few. However, the variety and versatility of those solutions significantly stand out from frameworks designed to deal with other data modalities.

As stands in Ref. [7], the number of spreading models and problems they tackle is overwhelming and exceeds the abovementioned examples. One can add to that various network models and forms of simultaneous coexistence of many processes[8]. When these phenomena are up to be analysed, researchers must first design and perform adequate simulations. Unfortunately, as demonstrated in our previous work[9], there is a big proliferation of environments that can be used for that. Hence, outcomes of such research are often presented only as articles without proper code attached, which could allow reproducing results and easily include proposed methods (e.g., a new centrality measure, spreading model, etc.) in following research conducted by another organisation. Although challenges related to the standardisation of experimental environments and results reproducibility have been addressed a long time ago in areas of computer science, like computer vision, signal processing, or machine learning, they are still valid in network science.

This paper presents an enhanced version of Network Diffusion, a framework originally designed to simulate the coexisting spreading processes in multilayer networks[9]. Since publication, it has been transformed into a comprehensive library that allows the processing of both temporal and multilayer networks with various spreading processes on top of them. Though the tool has been developed during various research activities of the authors, we collect developed methods and models (most of them are referenced in Table 1) into one consistent entity. To the best of our knowledge, there is no such library providing similar functionalities. Therefore, this framework can be considered as complementary to other available solutions. We can summarize a contribution brought with this paper as an introduction of a research environment for simulating spreading processes in complex networks to promote open science and research reproducibility. Moreover, with the framework, we tackle and solve (in a minimal way) still valid problems from a domain of influence maximisation, epidemiology, and network modelling.

The outline of the article is the following. First, we present a review of existing solutions tackling given problems (Section 2), then the general outline of library functionalities (Section 3). In Section 4, we show four use cases of experiments conducted with the package. After that, in Section 5, we describe a study of the computational efficiency of Network Diffusion. The paper is summarized in Section 6. For the reader's convenience, we attach a table with abbreviations used in the article (Table 1).

2 Existing Tools

In this section, we would like to briefly introduce existing tools and software packages with functionalities similar to the Network Diffusion package. The detailed list and comparison are already available[9]. Here, we would like to focus on tools that can be used for research on various spreading processes

Table 1 Abbreviations used in the paper.

Abbreviation	Abbreviation expansion	Note
SIS	Suspected-Infected-Suspected	A basic epidemiological spreading model[10]
SIR	Suspected-Infected-Removed	A basic epidemiological spreading model[10]
LTM	Linear Threshold Model	A spreading model in the sense as in Refs. [11, 12] with extension to multilayer networks as in Ref. [13]
ICM	Independent Cascade Model	A spreading model in the sense as in Ref. [12], with extension to multilayer networks similarly to Ref. [13]
SIR-UA	Suspected-Infected Removed and Unaware-Aware	Two coexisting spreading models in the sense as in Refs. [8, 9]
NEM	Network Epistemology Model	A spreading model in the sense as in Ref. [14], with extension to temporal networks as in Ref. [15]
MDS	Minimal Dominating Set	A set of network's nodes allowing to control its entire structure according to Refs. [16, 17]
CogSNet	Cognition-driven Social Network	A temporal network model according to Ref. [18]

in multilayer and temporal networks.

2.1 GLEaMviz

The first application with functionalities corresponding to the designed software is a GLEaMviz[19]. It works with real data, population density, and migration around the world, combined with stochastic models of disease propagation. As a result, it provides a sophisticated simulation environment. Due to the large scale of the experiments (the whole world), a single node is a population of a given size (defined by the user). A very interesting feature is the manual definition of the epidemiological model. GLEaMviz makes this possible by manipulating the compartments. Allowable transitions between them are also fully definable. Users are also able to choose the geographical origin of the disease, specify the initial proportions of individuals in each compartment, determine its duration, and more. There is also an option to generate various visualisations at the end of the experiment. Despite the interesting functionalities mentioned above, GLEaMviz has a few disadvantages: it is limited to disease spreading, only allows the propagation of one process at a time, and does not support multilayer and temporal networks.

2.2 NDlib

Network Diffusion library (NDlib)[20] is a Python package based on the NetworkX library. It allows performing simulations with many predefined epidemiological (such as SIS, SIR, etc.), influence spread (LTM, ICM, Profile, etc.), or opinion formation (Voter, Sznajd, etc.) models, and even dynamics (models with the capacity to change the topology of network). Moreover, the user can create its own customised models. Results visualisation is also possible via Matplotlib or Bokeh with the flexibility to append a custom graphical engine. NDlib also has some interesting run-time features. First, it includes an option to perform a "multi-execution" of the simulation by parallel computing. As this kind of experiment is generally stochastic, this feature gives a chance to see the general behaviour of the observed phenomena. It also enables running the simulation on a server (as well as locally). For users being unfamiliar with Python, NDlib's authors created NDQL, a query language (based on SQL syntax) that supports elementary commands of the library. For those who cannot programme, they also provided a "visualisation framework" to play with some of the models implemented with a graphical user interface-based tool. NDlib is a very useful library with many features. However, it does not directly support experiments where multiple spreading processes interact with each other, as well as experiments on temporal networks.

2.3 SimInf

The next tool is SimInf — a framework for data-driven stochastic disease spread simulations[21], a package for R language. This tool allows a user to define custom spreading models that are executed following a principle of continuous-time Markov chains and the Gillespie stochastic simulation algorithm. Interestingly, no network is needed to run the simulation, as the algorithms work on the assumption of homogeneous mixing and additional data, such as births, deaths, and population movements at predefined time points. After a successful simulation, it is possible to display a summary graph. Nevertheless, we must note that no support for operations on real, multilayer, and temporal networks, as well as the lack of an option to define interacting processes, are its distinct shortcomings.

2.4 Sispread

Sispread[22] is a simple application implemented in C language. It is a console tool without a graphical interface. It focuses on epidemic models (SI, SIS, SIR) with the possibility of deep analysis of the performed experiments. This tool supports very basic IO operations — the user can upload a custom network that meets certain requirements (without the option to manipulate it), or generate it using one of three available models: Barabási-Albert, Erdös-Renyi, or Kleinberg. As a result of the experiment, numerical data are returned for further analysis. Similarly to previous packages, it does not support multiple spreading processes, as well as multilayer or temporal networks.

2.5 STEM

Another tool is Spatio-Temporal Epidemiological Modeler (STEM)[23]. As an extension of the Eclipse development environment, it uses its graphical layout and the general philosophy of user interaction. It requires the user to specify a medium where the simulation is performed with its discretisation level (i.e., whether the node is a municipality or an entire county). The next step is to attach an appropriate solver propagation model and set the starting parameters of the experiment. Once this is done, it is possible to visualize the spread progress. In addition to the extensive user interface, there is an option to run simulations in headless mode, accessible from the terminal. This software also allows for simulating (to a limited extent) the propagation of coexisting phenomena, such as vaccine vs. disease. However, this package is primarily designed for epidemiologists, and all of its functionalities are related to this subject. Therefore, like other tools, it does not support spreading in multilayer or temporal networks.

2.6 EpiModel

EpiModel[24] is an R tool for simulating infectious disease dynamics. It contains deterministic compartmental models, stochastic individual-contact models, and stochastic network models. Network models use the robust statistical methods of Exponential-family Random Graph Models (ERGMs). As a standard, it includes SI, SIS, and SIR models. EpiModel does not directly support real networks (it focuses on generated ones) and multiple processes spreading at the same time. It also focuses on epidemic models. Unlike previously described tools, thanks to the application of ERGMs, it partially supports multilayer and temporal networks.

2.7 Summary

Based on our review of existing tools, we can conclude that, to the best of our knowledge, none of them supports multiple processes at the same time (like information and disinformation, virus and information about the virus, or multiple diseases), and most of them do not support multilayer and temporal networks. That is why we focus on these functionalities in Section 4, where we present a few examples of experiments using the Network Diffusion library.

3 Feature Overview

After a thorough review of the available conclusions from Section 2 are a principal motivation for investing time and human resources in developing the Network Diffusion package. In formulating its design, we operate under the following four premises:

(1) Compatibility with other tools commonly used in the domain of data science;

(2) Development of a tool in the form of a framework as opposed to a library comprising loosely connected code fragments;

(3) Supporting both multilayer and temporal networks; and

(4) Supporting spreading models with discrete states.

By releasing a Python-based tool, we naturally facilitate access to the entire ecosystem associated with that language. Primarily, we maintain compatibility with the widely-used graph processing package, NetworkX, which is the backbone of Network Diffusion. It is good to note, that some parts of the library are implemented in C language — that allows to speed up computations with high computational cost.

The framework-oriented design approach enables the straightforward extension of the tool with new spreading models. That is important, especially in research on social influence, misinformation, or viral marketing, where nuanced phenomena require modelling tailored to their unique nature. Furthermore, this approach allows for incremental project advancement (e.g., as a side effect of in-depth research).

As outlined in Section 2, only a few tools for simulating spreading phenomena operate on temporal or multilayer networks, despite these graph models offering a more accurate reflection of reality. This fundamental disparity leads us to present our internal research environment as a packaged solution. It is good to note that during the implementation of network models, the library is enriched with centrality methods tailored to these specific structures. These measures can be efficiently used to select seed nodes for diffusion processes.

The assumption regarding the discrete nature of dissemination models undoubtedly constitutes a tool limitation. However, the authors have yet to encounter the need to use a different class of models in their research so far. Therefore, this aspect, being unobjectionable, has been omitted.

While describing the functionality, it is worth mentioning the experiment scheme facilitated by this library. The process entails three fundamental steps: defining the propagation model, specifying the network upon which the model is executed, and determining the simulation parameters (e.g., number of steps). With these three components in place, experimenting becomes feasible, yielding precise data concerning the network's state at each simulation step. Section 4.1

demonstrates this procedure using the appropriate modules within the package, employing a selected research problem.

4 Experiments with Network Diffusion

This section presents four challenges from the domain of spreading models, which are tackled using the Network Diffusion library. Please note that all experiments described below are intended to meet two goals. Firstly, various functionalities of the Network Diffusion library are presented. Secondly, despite their compactness, they are big enough to draw particular conclusions regarding issues like phenomena spreading or network modelling. Regarding the above, we do not provide a comparative analysis of the methods used — this paper is not a place for that, and dwelling on such details could disturb the article's main message.

All experiments are conducted in an environment based on Python 3.10 with Network Diffusion 0.13.0. Though simulations are executed parallelly, we use three computers for that. The first one, which serves in problems from Sections 4.1 and 4.3, has been installed macOS 13 with M2 Pro CPU (arm64 architecture) and 32 GB RAM. The second one is used in experiments from Section 4.2 and has been installed Ubuntu 20.04 with Intel Core i7-6500U (x86_64 architecture) CPU and 16 GB RAM. Simulations presented in Section 4.4 are conducted on the third one on a workstation with Windows 10 Pro (22H2), whose hardware consists of Intel Core i7-6820HQ (x86_64 architecture) CPU and 32 GB RAM. It is good to note that we provide the source code enabling the reproduction of all experiments. It can be found in the repository available on GitHub (https://github.com/anty-filidor/bdma-experiments) with the instructions on how to run it.

4.1 Problem: Simultaneous spreading of disease and awareness of the threat

In this Section, we demonstrate how to define a custom spreading model using pre-existing library interfaces and how to conduct an experiment. Two processes (disease and awareness of its existence) interacting with each other will serve as examples. First, the SIR epidemiological model with parameters specific to COVID-19[25], namely the infection rate coefficient (α) and recovery rate coefficient (β). According to the simulated scenario, the second phenomenon, i.e., awareness of the disease Unaware-Aware (UA), is supposed to reduce the infection rate among agents who are conscious of the contagion. Finally, three cases are considered: minimising social interactions (e.g., lockdowns), wearing masks, and the absence of infection countermeasures. With the experiment, we will determine the effectiveness of the two measures mentioned above in combating the rise of the epidemic.

4.1.1 Problem formulation

That scenario can be depicted as a state graph with transitions between states, as shown in Fig. 1. The horizontal transitions occur within the disease context, while the vertical transitions occur within the awareness. For instance, the transition SA$\rightarrow$IA signifies that an agent's state change from suspected and aware to infected and aware takes place with a probability of α'.

It is important to note that the recovery rate does not depend on the awareness of the disease, i.e., IU$\rightarrow$RU and IA$\rightarrow$RA both have the same weight denoted as β. On the other hand, concerning the infection rate, we have assumed that α' will be a reduced value of α by a factor λ appropriate to the simulated scenario (see Table 2). Thus, for lockdown, we set infection risk reduction to 90% ($\lambda = 0.1$)[26], for wearing masks or 1m social distancing to 65% ($\lambda = 0.35$)[27, 28], and for no countermeasures applied no risk reduction will take place, hence $\lambda = 1$.

In the case of the UA process, we assume that agents initially become aware of the disease with a constant and low probability γ. However, if they get infected,

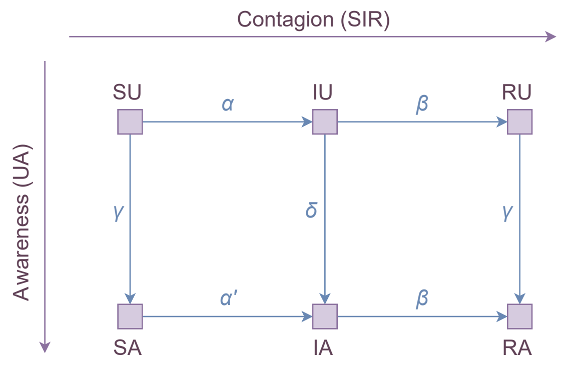


Fig. 1 Graph of states and transitions between them for the spreading model of two processes: contagion (SIR) and awareness of its existence (UA). Each state is represented by two letters indicating the state of both processes, e.g., IU indicates that the node is Infected (I) with the disease and Unaware (U) of its existence. The symbols on the arrows indicates the transition probability from one state to another (for values please see Table 2).

Table 2 Transition probabilities (or weights) and their interpretations in the SIR-UA model.

Symbol	Formula/Value	Description
α	0.19	Probability of infection for unaware agents
α'	$\lambda\alpha$; $\lambda \in \{0.1, 0.35, 1\}$	Probability of infection for aware agents
β	0.10	Probability of recovery
γ	0.01	Probability of awareness for uninfected agents
δ	$\gamma + 1 - 0.3$	Probability of awareness for infected agents

this value increases proportionally to the presence of symptomatic cases in the population observed during the COVID-19 pandemic (70% of people having COVID-19 have symptoms of the disease)[29]. Transition weights between states are also included in Table 2.

Another crucial aspect is the environment where the processes propagate. In this case, we assume propagation occurs in a multilayer network with two layers. In one layer (e.g., a communication network), awareness of the disease spreads, while in the second layer (e.g., a network of physical contacts), the disease itself. Agents can change their states based on interactions with their neighbours, with the probability of adopting the neighbour's state corresponding to the transition weight according to Fig. 1 and Table 2. An exception is the transitions $\mathrm{IU} \rightarrow \mathrm{RU}$, $\mathrm{IA} \rightarrow \mathrm{RA}$, for which no interaction with neighbouring agents is needed for the transitions to occur.

4.1.2 Experiment setup

In the repository attached to the article, within the script utils/models.py, we have included the source code for the `SIR_UAModel` class, which represents the implementation of the problem discussed in this section. It also demonstrates how to extend the Network Diffusion library with custom spreading processes. To create a specific model, one needs to extend the base class (which is `network_diffusion.models.BaseModel`) and concrete the six methods accordingly.

The constructor (`SIR_UAModel._init_` accepts nine parameters, of which six are the transition weights, and the remaining two are the initial percentage of infected agents and aware agents. The class utilises the transition graph introduced in our previous work[9], which helps in determining the potential state change of the evaluated agent during the simulation.

The following method determines the initial state of the network before the start of the simulation (`SIR_UAModel.determine_initial_states`). In this function, seeds are chosen, that is, the infected agents and aware agents from which the process will propagate. The method returns a list of `NetworkUpdateBuffer` structures, which contain the necessary information for updating the network state (specifically, the agent's ID and states in each layer). In this problem, we aim to determine the impact of λ on the number of infections. Therefore, seed nodes are decided to be chosen randomly — that approach is the most transparent in presented circumstances. It is good to add that we will select initially infected and aware actors independently, with budgets equal to 5%.

The function `SIR_UAModel.agent_evaluation_step` is called during each simulation step for each agent individually. It takes the agent's ID, the ID of the layer for which the evaluation takes place, and the network itself. This function determines the state the agent will adopt in the next step of the experiment. That is done by reading the set of possible states the agent can fall in, and then iteratively attempting to adopt the neighbour's state with the probability as depicted in Fig. 1. The output of this function is the new state of an agent in the given network layer.

A function `SIR_UAModel.network_evaluation_step` is responsible for evaluating the entire network in a single experiment step. In this case, it involves iterating over each layer and each agent within that layer to determine its new state according to the method (`SIR_UAModel.agent_evaluation_step`). The output is a list of NetworkUpdateBuffer structures.

The last two methods are needed to generate the experiment report while using this model. These are `_str_`, which provides a description of the model, and `get_allowed_states`, which returns the textual form of the graph presented in Fig. 1.

A model prepared this way can be executed on a given two-layer network with the `network_` `diffusion`.Simulator class[※]. By using the Simulator. `perform_propagation method`, we conduct a simulation that conceptually consists of eight steps[†] — see Algorithm 1.

4.1.3 Results & discussion

As part of the experiment, we utilize three types of networks: AUCS (a real graph obtained from interactions between employees of Department of Computer Science, Aarhus University, Denmark), Scale-Free, and Erdős-Rényi. For each graph, the simulation is repeated 50 times, and the obtained results are averaged. Figures 2–4 depict the results in the form of infection and awareness curves within the population for each evaluated λ value. It can be observed that limiting the infection rate makes sense for every network type. Another conclusion is that the network structure clearly influences the course of the processes. The highest percentage of infected agents at a given time is observed in the random network, accompanied by a rapid saturation of the UA process. We can observe that the propagation dynamics are significantly lower for the Scale-Free graph and AUCS network, where degree distribution is not as uniform as for the Erdős-Rényi model. Consequently, the number of infected agents is lower as well.

Algorithm 1 Simplified simulation procedure in Network Diffusion library

```
1: procedure perform_propagation (network, model, epochs)
2:     states_0 ← model.determine_initial_states ( );
3:     model.update_network (states_0)    ▷Predefined function in the BaseModel class;
4:     for e in [1, 2, …, epochs] do
5:         states_e ← model.network_evaluation_step (network);
6:         model.update_network (network, states_e);
7:     end for
8:     logs ← generate logs from experiment;
9:     return logs;
10: end procedure
```

4.2 Problem: Linear threshold model in temporal networks

4.2.1 Problem formulation

The models of social influence consider different scenarios of the spread. In a binary-LTM that has been introduced in Ref. [11], the adoption of a belief occurs in a situation where a certain fraction of the neighbours in an ego network of a node shares a new belief. In this case, contrary to the stochastic-ICM, we can consider the model deterministic and — given that only unidirectional change is possible — the only stop condition of the model is when no future activations of nodes are possible.

Typically, the LTM model is used in a static network scenario, where the set of nodes and edges is fixed. However, given that this approach is simplistic compared to real-world networks[30], it is worth

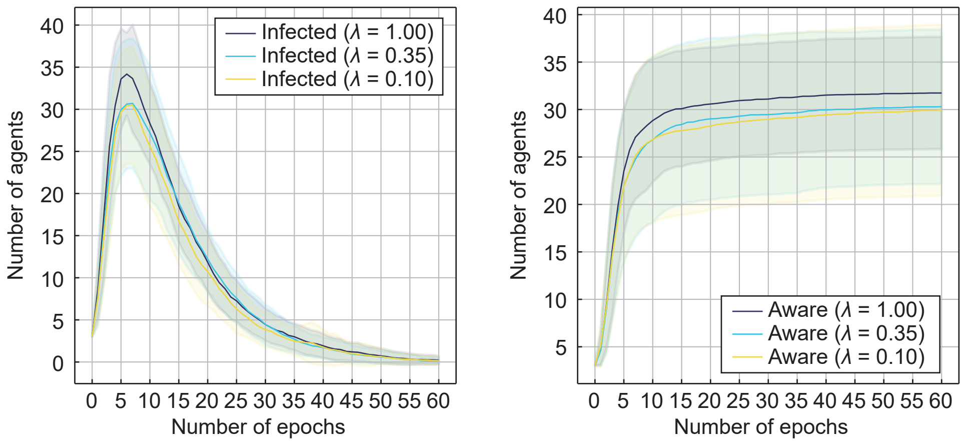


Fig. 2 Infection and awareness curves for spreading of SIR-UA model within aucs-2 network in three different epidemic regimes, where we can observe how different prevention measures lockdown (infection risk reduction by 90% — $\lambda = 0.10$), wearing masks or 1 m social distancing (risk reduction by 65% — $\lambda = 0.35$), and no measures (no risk reduction — $\lambda = 1.00$) affect the number of infected and aware individuals (agents) in the network.

[※] In the initial version of the package, it is called `network_diffusion.MultiSpreading`.
[†] Simulation algorithm is described in a simplified form, and simulations in temporal networks are not covered by the pseudocode.

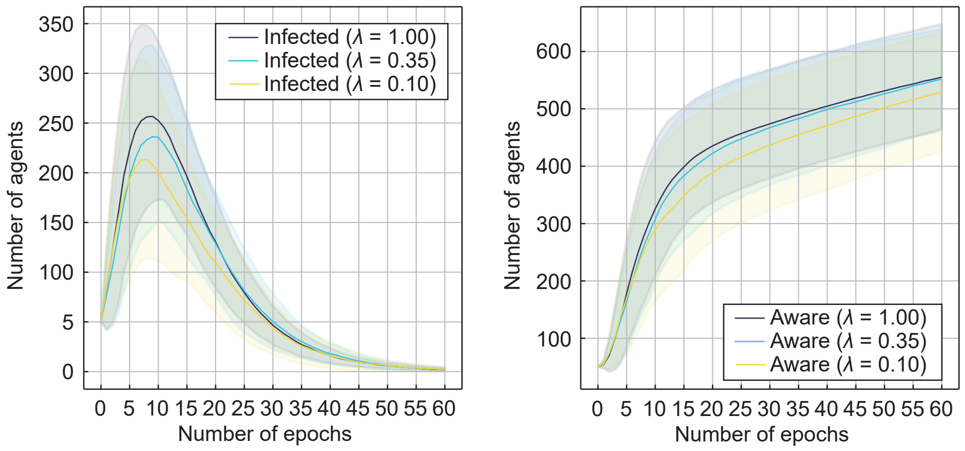


Fig. 3 Infection and awareness curves for spreading of SIR-UA model within sf-2 network in three different epidemic regimes, where we can observe how different prevention measures lockdown ($\lambda = 0.1$), wearing masks or 1 m social distancing ($\lambda = 0.35$), and no measures ($\lambda = 1$) affect the number of infected and aware individuals (agents) in the network.

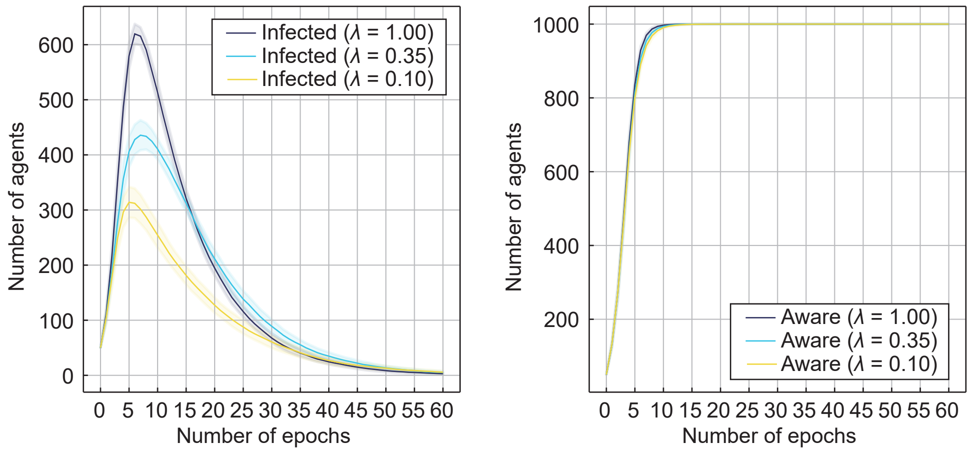


Fig. 4 Infection and awareness curves for spreading of SIR-UA model within er-2 network in three different epidemic regimes, where we can observe how different prevention measures lockdown ($\lambda = 0.1$), wearing masks or 1 m social distancing ($\lambda = 0.35$), and no measures ($\lambda = 1$) affect the number of infected and aware individuals (agents) in the network.

considering the relaxation of the time constraint and moving the problem of seed selection to time-varying networks[31, 32]. Yet, the change of the setting to temporal networks also requires an adequate choice of a temporal network model, which can be considered as a problem by itself. As we already mentioned, for the static scenario the stop condition is when no further activation of nodes is possible. Yet it is worth underlining that for the dynamic setting the activations can still be possible due to the network dynamics, so here the process can evolve for longer.

The goal of this experiment is to answer the question of how the network dynamics change the final spread of the LTM compared to the static scenario. Conceptually one can think of such an experiment in a way that the answer to this question will allow for understanding whether there is a speed-up or slow-down of diffusion dynamics when the network evolves over time[33].

4.2.2 Experiment setup

In the experiment on the spread of influence following the LTM model, the CogSNet model will be used[18]. The properties of this model — contrary to snapshot-based approaches — allow for continuous modelling of temporal networks. Here, each event that happens at a time point either creates an edge between nodes interacting with each other or amplifies it. Next, as time passes, the weights of the edges decay over time and in case of no forthcoming events lead to the vanishing of edges. Otherwise, if events occur, they again lead to an increase in weight. The decaying function is typically a power or exponential function.

This approach, described in more detail in Ref. [18] allows for fine-graded modelling contrary to incremental or snapshot-based temporal network models. However, as there is no continuous variant of the LTM model (the closest to it being introduced in Ref. [34]), to match the temporal network with the LTM, we will be matching the iterations of the spreading model with intervals of time in the CogSNet network. The stop condition for the LTM is when no further activations are possible and — for the temporal case — the process is run for no later than the last edge appeared.

To investigate how much the temporal network speeds up or slows down the spread of influence, the experimental setting is as follows. For experiments, we use the dataset on email exchange in manufacturing company[35]. For the LTM, we focus on evaluating the seeding budget γ and the threshold μ for node activations. For the seed selection strategy, a random seed selection has been applied, thus the experiments are repeated 100 times. As for the CogSNet model parameters, based on the work[18], we use the exponential forgetting function, edge lifetime of seven days, edge removal threshold $\theta = 0.1$, and forgetting function parameter $\lambda = 0.8$, with the units of hours. This set of parameters of the model in the aforementioned work results in an adequate match of human memory imprints and ground truth data. The parameters of the experiments alongside their explanation are shown in Table 3.

4.2.3 Results & discussion

The results of the experiment depicted in Fig. 5 clearly indicate that for the static networks — compared to the temporal one — for combinations of high seeding budget and low threshold, a static network is easily getting penetrated by the LTM diffusion process reaching almost 100% nodes existing in the network. This is not that surprising since in the static network that collapses all the links, there are more paths to reaching other nodes. Contrary to that, in the same regime of threshold values and percentage of initially activated nodes, the spread slows down in the temporal network built upon the CogSNet model. That can be caused by the absence of some potential links that do not appear in the later stage of the network evolution and that could not have been used for activation.

Yet, there is one interesting observation that one can make when looking at the combination of the LTM model parameters that are more challenging for the diffusion process (the central parts of Figs. 5a and 5b). Here, the CogSNet-modelled temporal network leads to a higher number of activations. It is due to the fact that

Table 3 Parameters used in the LTM in temporal networks experiment.

Symbol	Formula/Value	Description
θ	0.1	Edge removal threshold
λ	0.8	Forgetting function parameter
μ	$\mu \in \{0.05, 0.1, 0.15, 0.2, 0.25, 0.5\}$	Threshold of influence for LTM model
γ	$\gamma \in \{1, 5, 15, 25, 50\}$	Seeding budget (in %)

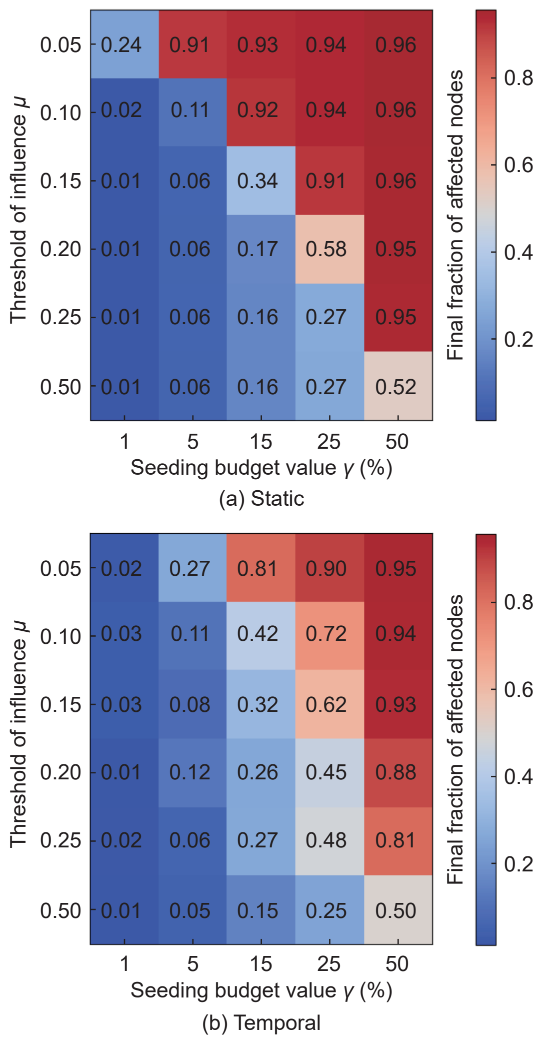


Fig. 5 Comparison of a final spread as a percentage of activated nodes for the LTM model of social influence for two settings: the temporal network based on the CogSNet model and the aggregated static network for the e-mail exchange data in a manufacturing company. The evaluated parameters are the seeding budget (γ) and the LTM threshold (μ).

there are fewer links in the ego networks of activated nodes that had to be activated to change the state of the node. This means that the lower density of the network results in a higher number of activations in this regime. Finally, both approaches perform equally badly in the most difficult scenario (left sides of Figs. 5a and 5b).

4.3 Problem: Multilayer independent cascade model initialised with minimum dominating set

4.3.1 Problem formulation

The influence maximisation problem is well-known because of the work of Ref. [12] and adopts various forms[7]. Studies have shown that the ultimate diffusion effectiveness depends not only on the spreading process itself and its parameters but also significantly on the network structure[36–38]. In general, ranking methods for selecting seed nodes can be categorised into three groups: local, semi-local, and global[39], differing in the scope of the network needed to analyze to determine the metric's value for a given node. Each of these classes varies in the trade-off between accuracy and computational complexity. Therefore, research teams often have to accept lower accuracy in identifying super-spreaders to obtain results within a reasonable time on large networks. Occasionally, one can leverage additional features to select the seed set, e.g., node labels, which are often available when analyzing data derived from social networks, as exemplified in Ref. [40]. However, some studies have shown that it is possible to determine (by analyzing the network's structure) whether computationally expensive methods can surpass basic algorithms, such as degree centrality[41].

Maximising influence problem has long transcended the classical defined graphs as a set of nodes and edges. This experiment will focus on multilayer networks[42]. By utilising the Network Diffusion framework, we will conduct a study to determine (in a minimal extent) whether network control methods are suitable for selecting initial nodes for propagation in graphs of this type. In order to build new seed selection strategies for the spreading processes in networks, we adopt the concept of driver nodes from the control theory. To control a system, one needs first to identify the minimum set of nodes (called driver nodes) that, when driven by distinct signals, can offer full control over the network. We argue that network spreading processes can be viewed as a soft version of control, where control diffuses with a given probability of following certain rule(s). Thus, the intuition is that using driver nodes as seed nodes can result in a more efficient spread over the network, and indeed, it has been successfully employed for classical graphs[17].

Networks with different characteristics feature different numbers of driver nodes. Once the driver nodes are identified, various centrality measures can be used to rank the driver nodes and use this final ranking list in the seed selection process. Experiments show that when driver nodes are used as the basis to build the ranking list, the spreading process is more effective and efficient when compared with the benchmarks. The first study that employs control-theoretic approaches for seed selection utilises the Minimum Dominating Set (MDS) to identify a candidate set of driver nodes[17]. This method has been incorporated into the Network Diffusion framework and extended to work with multilayer networks.

MDS is an optimisation approach that determines the minimum dominating set of nodes in undirected networks[16]. MDS is the smallest subset of nodes, such that every node of a network either belongs to this subset or is adjacent to at least one node in this set. Formally, a dominating set of a graph G is a subset D of the vertices of G, such that every vertex v of G is either in the set D or v has at least one neighbour that is in D.

Each driver node (or in the case of multilayer network: actor) can control its associated nodes (actors) independently, and each non-driver node (actor) is controllable if it is at least adjacent to one driver node (actor). In the MDS method, each node can exert control over its immediate neighbours at the same time, but the control does not propagate further. The driver nodes (actors) are chosen based on a minimal set of nodes (actors) that ensures each node (actor) in the network is either a driver node (actor) itself or directly connected to one. MDS has been applied in various contexts, such as characterising the perturbation of disease genes in the human regulatory network[44] and identifying control variables in protein interaction networks[45].

4.3.2 Experiment setup

In order to demonstrate the use of driver actors as the seeds in the spreading process, we run an experiment using AUCS[46] and Lazega[47] multilayer networks and apply an ICM[12].

In the ICM we start by activating a set of nodes (seeds) which become active. Each active node (in the case of a multilayer network an actor) has one chance to activate each of its neighbours. It succeeds with some probability, known as activation (or propagation) probability. After one iteration, each active node becomes activated and cannot activate anyone anymore. At this point, it is good to note that in the case of multilayer networks actors, not nodes, are the subject of the diffusion (i.e., we care whether the actor is active or not, not nodes that just represent the actor on the particular layers). Therefore, one needs to define how to aggregate impulses from layers of the network influencing the actor. We adopt here the approach proposed originally for LTM extended for multiplex networks (described in Ref. [13]) called "protocol functions" (or strategies). In this study, we use two extreme ones: OR, i.e., an actor gets activated when it gets activated on at least one layer, and AND, i.e., an actor gets activated when it gets activated on all layers it is represented in. It is good to add that between them, there is an entire spectrum of all possible functions that one can invent.

For the first experiment, we use an activation probability of 10% and AND strategy. We test two different values of the seeding budget, i.e., 5% and 10%. To show how the MDS method performs in the context, we compare it with seed selection strategies based on Degree, Betweenness[48], Closeness[49], VoteRank[50], and Berahmand's centrality[51]. They are adapted to work with multilayer networks as follows: Closeness, Betweenness, and Berahmand's centrality, which are computed separately in each layer of the graph, and then final values for actors are obtained by averaging scores from each layer. We modify VoteRank by replacing the Degree of an actor with neighbourhood size[52]. Finally, MDS is obtained as a union of minimal dominating sets calculated on each layer of the network. We transform it into a ranking list by sorting according to the Degrees of actors. Averaged results depicting the efficiency of all evaluated methods can be found in Tables 4 and 5.

In order to show the dynamics of the spreading process triggered by actors belonging to MDS (sorted by degree centrality), we conduct a second simulation. For that, we utilize the AUCS network and multilayer ICM, with a 10% seeding budget, 50% activation probability, and OR strategy. The illustration of the obtained results is depicted in Fig. 6.

4.3.3 Results & discussion

The results for the AND strategy show that the MDS method performs at a similar level as other methods on the two selected datasets regardless of the seeding budget, and it takes a similar number of epochs to reach a stable state where no more new nodes are being activated (Tables 4 and 5). However, it should be noted that the parameters of the spreading model are used for illustration purposes only. The optimal ones are shown to depend on the network structural characteristics[17] and can be optimised separately, but it is out of the scope of this study.

The results of spreading in the AUCS network using the OR strategy are shown in Fig. 6. In each of the layers, the number of activated nodes goes through a phase transition between epoch one and two resulting in a sudden increase in the number of activated nodes. This shows that MDS is a good strategy to activate many nodes early in the spread process. Again, used

Table 4 Average percentage of activated actors for multilayer ICM on AUCS network and average number of epochs needed for the model to reach a stable state when the simulation is stopped. The results for each seed selection method are averaged over 100 runs. The model is executed with 5% and 10% seeding budgets, 50% propagation probability, and AND strategy.

Seed selection method	Seeding budget 5%		Seeding budget 10%	
	Average activated actors ± std	Average number of epochs ± std	Average activated actors ± std	Average number of epochs ± std
Degree	(11.64 ± 3.17)%	2.82 ± 0.77	(20.90 ± 2.66)%	2.79 ± 0.50
Betweenness	(9.79 ± 2.51)%	2.54 ± 0.68	(18.11 ± 3.02)%	2.76 ± 0.55
Closeness	(8.79 ± 1.83)%	2.33 ± 0.60	(16.49 ± 2.43)%	2.46 ± 0.54
VoteRank	(10.98 ± 3.90)%	2.76 ± 0.91	(20.30 ± 3.81)%	2.88 ± 0.52
Berahmand's centrality	(7.72 ± 2.55)%	2.10 ± 0.84	(13.56 ± 1.91)%	2.38 ± 0.58
MDS	(12.31 ± 3.15)%	2.84 ± 0.73	(19.46 ± 3.84)%	2.80 ± 0.51

Table 5 Average percentage of activated actors for multilayer ICM on Lazega network and average number of epochs needed for the model to reach a stable state when the simulation is stopped. The results for each seed selection method are averaged over 100 runs. The model was executed with 5% and 10% seeding budgets, 50% propagation probability, and AND strategy.

Seed selection method	Seeding budget 5%		Seeding budget 10%	
	Average activated actors ± std	Average number of epochs ± std	Average activated actors ± std	Average number of epochs ± std
Degree	(79.17 ± 17.74)%	6.01 ± 0.99	(88.39 ± 3.52)%	4.37 ± 0.52
Betweenness	(78.03 ± 15.53)%	6.02 ± 0.96	(89.32 ± 3.40)%	4.27 ± 0.44
Closeness	(78.55 ± 13.07)%	6.17 ± 0.90	(89.10 ± 3.35)%	4.43 ± 0.57
VoteRank	(82.10 ± 13.30)%	5.84 ± 0.90	(89.03 ± 3.40)%	4.44 ± 0.50
Berahmand’s centrality	(83.10 ± 11.92)%	5.83 ± 1.02	(89.13 ± 3.19)%	4.31 ± 0.50
MDS	(84.90 ± 6.04)%	5.77 ± 0.85	(89.28 ± 3.28)%	4.28 ± 0.45

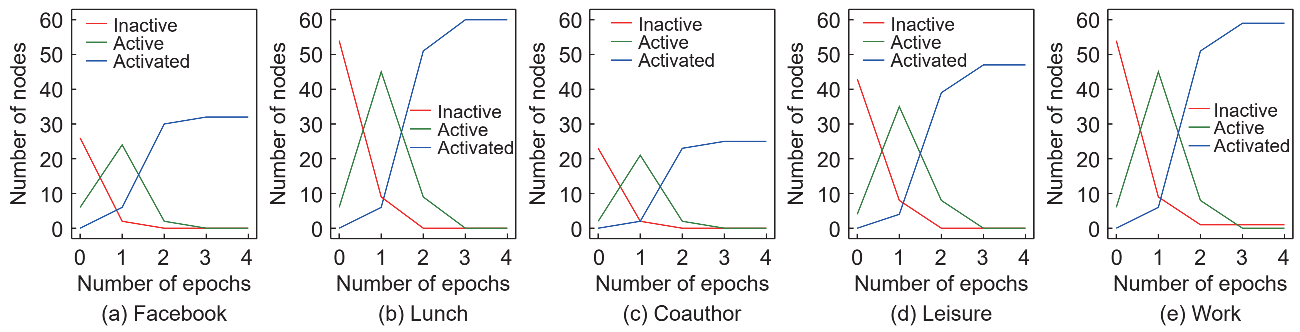


Fig. 6 Dynamics of ICM spreading in each layer of the AUCS network. The model is executed using MDS (degree centrality) as the seed selection method, with a 10% seeding budget, 50% propagation probability, and OR strategy. “Active” means that the node is active and has the potential to activate its neighbours; “Activated” denotes the node that is active but does not have the potential to activate its neighbours any more.

parameters are not optimised and are used for illustrative purposes.

4.4 Problem: Network epistemology model in temporal networks

4.4.1 Problem formulation

NEM is an approach to the formalisation of the social learning process, focusing on belief evolution and dissemination in the social network. Previous research on this model has explored various aspects, including the impact of community structure[14], conformism, and polarisation[53−56] on the effect of the learning process. By adapting the model to a temporal framework, it becomes better equipped to capture the ongoing social network dynamics, aligning it more closely with real-world scenarios[15].

In its basic configuration, the model assumes the existence of a set of K agents organised within a temporal social network, each facing a bandit decision-making problem. At each iteration of the process, agents are presented with a choice between two available actions. The payoff associated with the first action denoted as A, is common knowledge and remains constant at 0.5. In contrast, the payoff associated with the alternative action denoted as B, remains unknown to agents and deviates from the payoff of the action A by a specified value, denoted as ϵ.

Each agent’s decision relies on their individual belief level. The belief assigned to each agent is a value within the range of 0 to 1 interpreted as the probability that the given agent associates with the proposition that action B yields a superior payoff compared to action A.

Subsequent to their decision-making, agents who opt for action B engage in an experimentation phase, collecting evidence related to their actions. They execute the action B a defined N number of times, drawing from a Bernoulli distribution with a probability of success equal to $0.5+\epsilon$.

Following this experimentation phase, agents update their belief levels using Bayes’ rule, calculating posterior probabilities based on their prior beliefs and the evidence they have accumulated. This updating process occurs in a synchronous manner across all agents within the network. The simulation proceeds iteratively through each snapshot of the temporal network.

In the following experiment, we are going to compare the diffusion following NEM between the classical static approach to construct the underlying network topology and the temporal strategy with the CogSNet model[18]. One can notice a similarity with the experiment described in Section 4.2. In fact, it exists, but the spreading model used here is much more complex than LTM. Therefore, the results between them are certainly not redundant.

4.4.2 Experiment setup

Implementation of the model with an extension allowing to run it in the temporal setting is located in `TemporalNetworkEpistemologyModel` class. Apart from previously discussed seeding properties, construction of the model instance requires two specific parameters described in the brief model definition, namely ϵ denoted as `epsilon`, and N denoted as `trials_nr`.

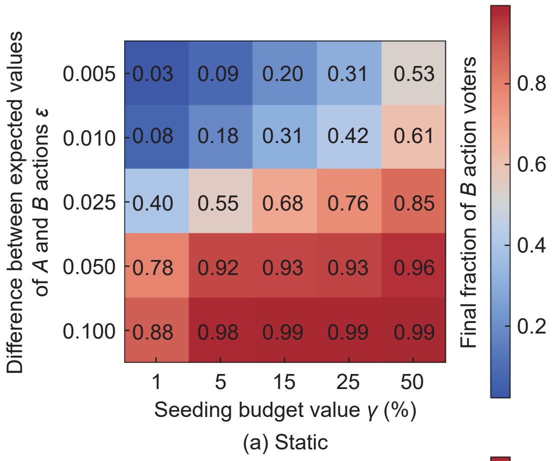
As stated above, in the experiment, we use two types of networks: static and dynamic. Both of them are constructed from a dataset on email exchange in manufacturing company[35]. The number of experiments N that agents perform each iteration is equal to 1. We test different values of the seeding budget γ and values of ϵ, which stands for the difference between expected values of A and B actions. For both cases, we use the random seeding method. The value we compare between these configurations is the final number of actors voting for superior action B. The snapshots constituting the temporal network in this experiment are created with one day interval, resulting in 55 long sequences, with one snapshot for each following day of the underlying data. The parameters of NEM can be established based on the expert knowledge related to a specific sociological phenomenon or by a direct measurement of the underlying social learning process characteristic. In this example the fixed parameter $N = 1$ corresponds to one decision made daily by each agent during the simulation process. The configuration of CogSNet parameters is based on the work in Ref. [18], in which it is demonstrated to effectively match the ground truth data on subjects relations. The values of the experiments' outcomes are averaged across 100 iterations to reduce the random effects. Table 6 outlines the experiment parameters and their description.

Table 6 Parameters used in the Network Epistemology Model in temporal networks experiment.

Symbol	Formula/Value	Description
θ	0.1	Edge removal threshold
λ	0.8	Forgetting function parameter
N	1	Number of experiments (trials) performed by an agent per iteration
ϵ	$\epsilon \in \{0.005, 0.01, 0.025, 0.05, 0.1\}$	Difference between expected values of A and B actions
γ	$\gamma \in \{1, 5, 15, 25, 50\}$	Seeding budget (in %)

4.4.3 Results & discussion

The outcomes of the experiment are shown in Fig. 7. The measure of efficiency of the explored configurations is expressed as the ratio of better, B



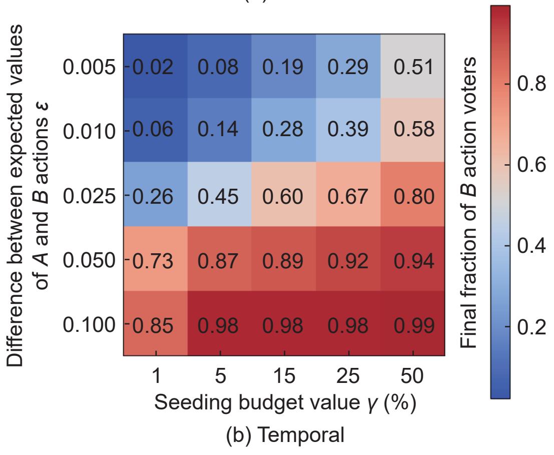


Fig. 7 Number of *B* action voters at the end of the spreading process compared for different levels of problem difficulty and various initial seeding budgets, that in this experiment corresponds to an initial number of *B* voters. The process outcomes are compared for two different approaches for network topology construction — a static network and a temporal network built using the CogSNet method.

action voters to all individuals. The observed results reveal a subtle difference between the two networks in the effectiveness of B action propagation. On average, in the temporal network, diffusion for each configuration reached a smaller or equal range compared to the static network. The most noticeable differences are in the test configurations, where the ϵ value was 0.025, where propagation in the static network covers a few to several percentage points more nodes.

4.5 Data used in experiments

As part of the experiments, we utilize several types of networks. Their basic parameters have been listed in Table 7. It is worth mentioning that the aucs-2 network is created with "lunch" and "facebook" layers to simulate the SIR and UA processes, respectively. The er-2 and sf-2 graphs are created using the NetworkX[59] library, so that the layer responsible for disease transmission is sparser than the one responsible for spreading awareness about the epidemic. Implementation details can be found in the accompanying repository.

5 Limitations & Performance Study

In Section 4 of the paper, we briefly discuss limitations of the library and complement it with the performance analysis. Apart from the package functionalities described in Section 3, it is essential to address its constraints critically. They primarily stem from the adopted design assumptions and the relatively small size of the team involved in the implementation.

The most significant drawback of Network Diffusion is also its greatest advantage — implementation in Python. This results in significantly slower performance compared to if it had been coded in another compiled language. Although a part of the code has been provided in C language and is accessible as bindings to Python, we cannot adopt this approach in the entire project due to the small team. Another package limitation is its support solely for discrete spreading processes (see Section 3). Though the framework is designed more like a set of interfaces to be used in the implementation of custom experiments, it does not include many concrete spreading models that work "out of the box". Comparing that to NDlib, which consists of dozens of pre-defined models, one can find that state to be a limitation. Finally, it is worth noting the absence of a user interface, which may prove impossible for individuals lacking programming skills, leaving them compelled to resort to tools like NetLogo[60].

In order to assess the preformance of the Network Diffusion, we utilize all four models described in the article.The domain chosen to express complexity is the experiment duration as a function of the network size, expressed in the number of actors. We evaluate the following set of parameters:

- Number of simulation steps — 200 (regardless of whether a stable state is achieved earlier or not);
- Evaluation is performed on two-layer multiplex Erdős-Rényi networks with an edge creation probability of 0.1 and the number of actors from range 10 to 1000; and
- Each experiment is repeated 10 times.

The results are presented in Fig. 8. As can be seen, the most computationally expensive models are NEM and LTM, which significantly differ from ICM and SIR-UA. It is also worth noting that the computation time is stable — a standard deviation (a blurry region surrounding curves) is insignificant.

Considering the future of the package, we undoubtedly are going to develop and maintain it. Having defined the framework, it is natural to enhance its contents by new spreading models and functions regarding both temporal and multilayer networks. We are also going to implement machine learning based methods for the identification of the most influencing

Table 7 Networks used in experiments with their basic parameters shortlisted.

Name	Number of layers	Number of actors	Number of nodes	Number of edges	Note
aucs	5	61	224	620	AUCS network[46]
aucs-2	2	60	120	317	AUCS network[46]
lazega	3	71	212	1659	Lazega Law Firm network[47]
er-2	2	1000	2000	27 451	Erdős-Rényi network[57]
sf-2	2	1000	2000	3357	Scale-free network[58]
manuf	1	151	151	20 000	E-mail exchange[35]

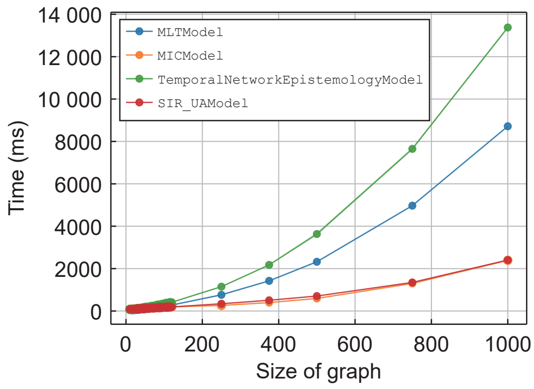


Fig. 8 Complexity of spreading models used in experiments; evaluation is performed on two-layer multiplex Erdős-Rényi networks.

actors. Nonetheless, there is another path that is especially interesting to us — matching the theoretical spreading model to the observed phenomena "in the wild". That will be addressed within the Network Diffusion or at least with its support.

6 Conclusion

Developing and making publicly available comprehensive software frameworks that support replicability and dissemination of research effort is a key and very much required component of scientific enquiry. In order to support the research community, we show in this paper an extended version of the Network Diffusion framework. Its initial concept is presented in Ref. [9] and has now been significantly enriched to encompass both temporal and multilayer networks, as well as various methods and spreading processes that can propagate over different types of graphs.

The models covered in this study include multilayer networks and temporal networks. Additionally, we adopt the node centrality measures for multilayer networks. Spreading models covered in this paper include SIR-UA, temporal LTM, temporal NEM, and multilayer ICM. Presented spreading models, which have been included in the framework's extension, are accompanied by experiments to show how the newly developed parts of the library can be utilized. They also show the interesting future direction of our research that we would like to investigate in the future fully.

As discussed above, the Network Diffusion framework is complementary to other software solutions available in the field of data science. We believe that its comprehensiveness and ease of use make it accessible not only for computer scientists, but also for researchers from other domains who are interested in understanding spread in the systems they work on.

Authors Contribution

Conceptualization (Michał Czuba and Piotr Bródka), data curation (Michał Czuba, Yu-Xuan Qiu, Mateusz Nurek, and Radosław Michalski), formal analysis (All), funding acquisition (Piotr Bródka and Radosław Michalski), investigation (All), methodology (Michał Czuba, Piotr Bródka, Radosław Michalski, and Katarzyna Musial), project administration (Michał Czuba), resources (All), software (Michał Czuba, Damian Serwata, Mateusz Nurek, Mingshan Jia, and Yu-Xuan Qiu), supervision (Piotr Bródka), validation (All), visualisation (Michał Czuba, Damian Serwata, Mateusz Nurek, Mingshan Jia, and Yu-Xuan Qiu), writing – original draft (All), writing – review & editing (Michał Czuba and Piotr Bródka).

Acknowledgment

This research was partially supported by the National Science Centre, Poland (Nos. 2022/45/B/ST6/04145 and 2021/41/B/HS6/02798), and the EU under the Horizon Europe (No.101086321 OMINO). Views and opinions expressed are, however, those of the authors only and do not necessarily reflect those of the National Science Centre, EU, or the European Research Executive Agency.

References

[1] F. Menczer, S. Fortunato, and C. A. Davis, *A First Course in Network Science*. Cambridge, UK: Cambridge University Press, 2020.

[2] R. Pastor-Satorras and A. Vespignani, Epidemic spreading in scale-free networks, *Phys. Rev. Lett.*, vol. 86, no. 14, pp. 3200–3203, 2001.

[3] M. Nurek, R. Michalski, O. Lizardo, and M. A. Rizoiu, Predicting relationship labels and individual personality traits from telecommunication history in social networks using hawkes processes, *IEEE Access*, vol. 11, pp. 8492–8503, 2023.

[4] S. Forouzandeh, K. Berahmand, R. Sheikhpour, and Y. Li, A new method for recommendation based on embedding spectral clustering in heterogeneous networks (RESCHet), *Expert Syst. Appl.*, vol. 231, p. 120699, 2023.

[5] M. Rostami, U. Muhammad, S. Forouzandeh, K. Berahmand, V. Farrahi, and M. Oussalah, An effective explainable food recommendation using deep image clustering and community detection, *Intell. Syst. Appl.*, vol. 16, p. 200157, 2022.

[6] C. Zhong, S. M. Arisona, X. Huang, M. Batty, and G. Schmitt, Detecting the dynamics of urban structure through spatial network analysis, *Int. J. Geogr. Inf. Sci.*, vol. 28, no. 11, pp. 2178–2199, 2014.

[7] S. S. Singh, D. Srivastva, M. Verma, and J. Singh, Influence maximization frameworks, performance, challenges and directions on social network: A theoretical study, *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7570–7603, 2022.

[8] P. Brodka, K. Musial, and J. Jankowski, Interacting spreading processes in multilayer networks: A systematic review, *IEEE Access*, vol. 8, pp. 10316–10341, 2020.

[9] M. Czuba and P. Brodka, Simulating spreading of multiple interacting processes in complex networks, in *Proc. 2022 IEEE 9th Int. Conf. Data Science and Advanced Analytics* (*DSAA*), Shenzhen, China, 2022, pp. 1–10.

[10] A. L. Barabási, *Network Science*. Cambridge, UK: Cambridge University Press, 2016.

[11] M. Granovetter, Threshold models of collective behavior, *Am J Sociol*, vol. 83, no. 6, pp. 1420–1443, 1978.

[12] D. Kempe, J. Kleinberg, and É. Tardos, Maximizing the spread of influence through a social network, in *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery and Data Mining*, Washington, DC, USA, 2003, pp. 137–146.

[13] Y. D. Zhong, V. Srivastava, and N. E. Leonard, Influence spread in the heterogeneous multiplex linear threshold model, *IEEE Trans. Control Netw. Syst.*, vol. 9, no. 3, pp. 1080–1091, 2022.

[14] K. J. S. Zollman, The communication structure of epistemic communities, *Philos. Sci.*, vol. 74, no. 5, pp. 574–587, 2007.

[15] R. Michalski, D. Serwata, M. Nurek, B. K. Szymanski, P. Kazienko, and T. Jia, Temporal network epistemology: On reaching consensus in a real-world setting, *Chaos*, vol. 32, no. 6, p. 063135, 2022.

[16] J. C. Nacher and T. Akutsu, Dominating scale-free networks with variable scaling exponent: Heterogeneous networks are not difficult to control, *New J. Phys.*, vol. 14, no. 7, p. 073005, 2012.

[17] A. Sadaf, L. Mathieson, P. Bródka, and K. Musial, Maximising influence spread in complex networks by utilising community-based driver nodes as seeds, in *Information Management and Big Data*, J. A. Lossio-Ventura, J. Valverde-Rebaza, E. Díaz, and H. Alatrista-Salas, eds. Lima, Peru: Springer, 2023, pp. 126–141.

[18] R. Michalski, B. K. Szymanski, P. Kazienko, C. Lebiere, O. Lizardo, and M. Kulisiewicz, Social networks through the prism of cognition, *Complexity*, vol. 2021, p. 4963903, 2021.

[19] W. Van den Broeck, C. Gioannini, B. Goncalves, M. Quaggiotto, V. Colizza, and A. Vespignani, The GLEaMviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale, *BMC Infect Dis*, vol. 11, no. 1, p. 37, 2011.

[20] G. Rossetti, L. Milli, S. Rinzivillo, A. Sîrbu, D. Pedreschi, and F. Giannotti, NDlib: A python library to model and analyze diffusion processes over complex networks, *Int. J. Data Sci. Anal.*, vol. 5, no. 1, pp. 61–79, 2018.

[21] S. Widgren, P. Bauer, R. Eriksson, and S. Engblom, SimInf: An R package for data-driven stochastic disease spread simulations, *J. Stat. Softw.*, vol. 91, no. 12, pp. 1–42, 2019.

[22] F. P. Alvarez, P. Crépey, M. Barthélemy, and A. J. Valleron, Sispread: A software to simulate infectious diseases spreading on contact networks, *Methods Inf. Med.*, vol. 46, no. 1, pp. 19–26, 2007.

[23] J. V. Douglas, S. Bianco, S. Edlund, T. Engelhardt, M. Filter, T. Günther, K. Hu, E. J. Nixon, N. L. Sevilla, A. Swaid, et al., STEM: An open source tool for disease modeling, *Health Secur.*, vol. 17, no. 4, pp. 291–306, 2019.

[24] S. M. Jenness, S. M. Goodreau, and M. Morris, EpiModel: An R package for mathematical modeling of infectious disease over networks, *J. Stat. Softw.*, vol. 84, p. 8, 2018.

[25] A. A. Toda, Susceptible-Infected-Recovered (SIR) dynamics of COVID-19 and economic impact, arXiv preprint arXiv: 2003.11221, 2020.

[26] P. Wątroba and P. Bródka, Influence of information blocking on the spread of virus in multilayer networks, *Entropy*, vol. 25, no. 2, p. 231, 2023.

[27] D. K. Chu, E. A. Akl, S. Duda, K. Solo, S. Yaacoub, H. J. Schunemann, Schunemann, and COVID-19 Systematic Urgent Review Group Effort (SURGE) study authors, physical distancing, face masks, and eye protection to prevent person-to-person transmission of SARS-CoV-2 and COVID-19: A systematic review and meta-analysis, *Lancet*, vol. 395, no. 10242, pp. 1973–1987, 2020.

[28] D. J. McGrail, J. Dai, K. M. McAndrews, and R. Kalluri, Enacting national social distancing policies corresponds with dramatic reduction in COVID19 infection rates, *PLoS One*, vol. 15, no. 7, p. e0236619, 2020.

[29] M. Alene, L. Yismaw, M. A. Assemie, D. B. Ketema, B. Mengist, B. Kassie, and T. Y. Birhan, Magnitude of asymptomatic COVID-19 cases throughout the course of infection: A systematic review and meta-analysis, *PLoS One*, vol. 16, no. 3, p. e0249090, 2021.

[30] P. Holme and J. Saramaki, Temporal networks, *Phys. Rep.*, vol. 519, no. 3, pp. 97–125, 2012.

[31] R. Michalski, T. Kajdanowicz, P. Bródka, and P. Kazienko, Seed selection for spread of influence in social networks: Temporal vs. static approach, *New Gener. Comput.*, vol. 32, nos. 3&4, pp. 213–235, 2014.

[32] R. Michalski and P. Kazienko, Maximizing social influence in real-world networks — the state of the art and current challenges, in *Propagation Phenomena in Real World Networks*, D. Król, D. Fay, and B. Gabryś, eds. Cham, Switzerland: Springer, 2015, pp. 329–359.

[33] I. Scholtes, N. Wider, R. Pfitzner, A. Garas, C. J. Tessone, and F. Schweitzer, Causality-driven slow-down and speed-up of diffusion in non-Markovian temporal networks, *Nat. Commun.*, vol. 5, no. 1, p. 5024, 2014.

[34] F. Karimi and P. Holme, Threshold model of cascades in empirical temporal networks, *Physica A*, vol. 392, no. 16, p. 3476–3483, 2013.
[35] M. Nurek and R. Michalski, Combining machine learning and social network analysis to reveal the organizational structures, *Appl. Sci.*, vol. 10, no. 5, p. 1699, 2020.
[36] R. Albert and A. L. Barabasi, Statistical mechanics of complex networks, *Rev. Mod. Phys.*, vol. 74, no. 1, pp. 47–97, 2002.
[37] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D. U. Hwang, Complex networks: Structure and dynamics, *Phys. Rep.*, vol. 424, nos. 4&5, pp. 175–308, 2006.
[38] Z. Liu, X. Wu, and P. M. Hui, An alternative approach to characterize the topology of complex networks and its application in epidemic spreading, *Front. Comput. Sci. China*, vol. 3, no. 3, pp. 324–334, 2009.
[39] L. Lü, D. Chen, X. L. Ren, Q. M. Zhang, Y. C. Zhang, and T. Zhou, Vital nodes identification in complex networks, *Phys. Rep.*, vol. 650, pp. 1–63, 2016.
[40] S. Forouzandeh, A. Sheikhahmadi, A. R. Aghdam, and S. Xu, New centrality measure for nodes based on user social status and behavior on Facebook, *Int. J. Web Inf. Syst.*, vol. 14, no. 2, pp. 158–176, 2018.
[41] K. Berahmand, N. Samadi, and S. M. Sheikholeslami, Effect of rich-club on diffusion in complex networks, *Int. J. Mod. Phys. B*, vol. 32, no. 12, p. 1850142, 2018.
[42] M. E. Dickison, M. Magnani, and L. Rossi, *Multilayer Social Networks*. Cambridge, UK: Cambridge University Press, 2016.
[43] A. Sadaf, L. Mathieson, and K. Musial, Effects of global and local network structure on number of driver nodes in complex networks, in *Cyber Security and Social Media Applications*, S. T. Özyer and B. Kaya, eds. Cham, Switzerland: Springer, 2023, pp. 81–98.
[44] B. Wang, L. Gao, Q. Zhang, A. Li, Y. Deng, and X. Guo, Diversified control paths: A significant way disease genes perturb the human regulatory network, *PLoS One*, vol. 10, no. 8, p. e0135491, 2015.
[45] S. Wuchty, Controllability in protein interaction networks, *Proc. Natl. Acad. Sci. USA*, vol. 111, no. 19, pp. 7156–7160, 2014.
[46] L. Rossi and M. Magnani, Towards effective visual analytics on multiplex and multilayer networks, *Chaos Solitons Fractals*, vol. 72, pp. 68–76, 2015.
[47] T. A. B. Snijders, P. E. Pattison, G. L. Robins, and M. S. Handcock, New specifications for exponential random graph models, *Sociological Methodology*, vol. 36, no. 1, pp. 99–153, 2006.
[48] U. Brandes, A faster algorithm for betweenness centrality, *J. Math. Sociol.*, vol. 25, no. 2, pp. 163–177, 2001.
[49] L. C. Freeman, Centrality in social networks conceptual clarification, *Soc. Netw.*, vol. 1, no. 3, pp. 215–239, 1978.
[50] J. X. Zhang, D. B. Chen, Q. Dong, and Z. D. Zhao, Identifying a set of influential spreaders in complex networks, *Sci. Rep.*, vol. 6, p. 27823, 2016.
[51] K. Berahmand, A. Bouyer, and N. Samadi, A new centrality measure based on the negative and positive effects of clustering coefficient for identifying influential spreaders in complex networks, *Chaos Solitons Fractals*, vol. 110, p. 41–54, 2018.
[52] P. Bródka, J. Jankowski, and R. Michalski, Sequential seeding in multilayer networks, *Chaos*, vol. 31, no. 3, p. 033130, 2021.
[53] C. O'Connor and J. Weatherall, Scientific polarization, *Eur. J. Philos. Sci.*, vol. 8, no. 3, p. 855–875, 2018.
[54] J. Weatherall and C. O'Connor, Do as I say, not as I do, or, conformity in scientific networks, arXiv preprint arXiv: 1803.09905, 2019.
[55] J. Weatherall and C. O'Connor, Endogenous epistemic factionalization: A network epistemology approach, arXiv preprint arXiv: 1812.08131**.**
[56] J. O. Weatherall, C. O'Connor, and J. Bruner, How to beat science and influence people: Policymakers and propaganda in epistemic networks, *Br. J. Philos. Sci.*, vol. 71, no. 4, pp. 1157–1186, 2020.
[57] P. Erdös and A. Rényi, On the evolution of random graphs, *Publ. Math. Inst. Hung. Acad. Sci.*, vol. 5, no. 1, pp. 17–60, 1960.
[58] B. Bollobás, C. Borgs, J. Chayes, and O. Riordan, Directed scale-free graphs, in *Proc. 14th Annu. ACM-SIAM Symp. Discrete Algorithms*, Baltimore, MD, USA, 2003, pp. 132–139.
[59] A. A. Hagberg, D. A. Schult, and P. J. Swart, Exploring network structure, dynamics, and function using networkx, in *Proc. 7th Python in Science Conf.*, Pasadena, CA, USA, 2008, pp. 11–15.
[60] U. Wilensky, NetLogo: Center for connected learning and computer-based modeling, http://ccl.northwestern.edu/netlogo, 2023.

Michał Czuba is a PhD candidate at Wrocław University of Science and Technology, Poland, in the discipline of information and communication technology with a focus on computational network science, namely problems of influence maximisation and phenomena spreading in multilayer networks. He also has industrial expertise in machine learning, computer vision systems, and MLOps engineering in projects concerning the commercialisation of research outcomes.

Mateusz Nurek is a PhD candidate at Wrocław University of Science and Technology, Poland. His research area includes network science and machine learning. He is primarily interested in using computational intelligence to study social aspects; therefore, his current research focuses on problems, such as the classification of human relationships or predicting personality traits based on communication patterns.

Damian Serwata is a PhD candidate at Wrocław University of Science and Technology, Poland. His research focus is concentrated on leveraging network science models and tools to study complex social systems and their adaptive abilities. Currently, he is devoted to research on the social learning process, with regards to the investigation of different approaches to represent social structures, bringing the models closer to reality.

Yu-Xuan Qiu presently serves as a postdoctoral research assistant at University of Technology Sydney, Australia. He received the BEng and Meng degrees in computer science from Shenzhen University, China in 2015 and 2018, respectively, and the PhD degree from University of Technology Sydney, Australia in 2023. His primary research interest is graph data mining and management.

Mingshan Jia is a lecturer at University of Technology Sydney, Australia. He received the BEng degree in information engineering from Xi’an Jiaotong University, China in 2008, the MEng degree in information and telecommunication systems from University of Technology of Troyes, France in 2011, and the PhD degree from University of Technology Sydney, Australia in 2022. Controllability of networks, data mining, and machine learning applications in social networks belong to his main research interests.

Radosław Michalski is an associate professor at Department of Artificial Intelligence, Wrocław University of Science and Technology, Poland, where he co-leads the Network Science Lab and leads the Blockchain Exploration Research Group (BERG). His research interests include social influence, diffusion processes in complex networks, and machine learning. He has co-authored more than 50 publications in these areas.

Katarzyna Musial received the MEng degree in computer science from Wrocław University of Science and Technology, Poland in 2006, the another MEng degree in software engineering from the Blekinge Institute of Technology, Sweden in 2006, and the PhD degree from Wrocław University of Science and Technology, Poland in 2009. In the same year, she was appointed as a senior visiting research fellow at Bournemouth University, UK, where she has been a lecturer in informatics since 2010. She joined King’s College London as a lecturer in computer science in 2011. In 2015, she returned to Bournemouth University, UK, where she was an associate professor of computing, as well as the head of the SMART Technology Research Group and a member of the Data Science Initiative. In 2017, she moved to Australia and started working as a professor of network science at Data Science Institute, University of Technology Sydney, Australia, where she co-leads the Complex Adaptive Systems Lab. She is particularly interested in social networks, graph controllability, and complex systems.

Piotr Bródka is an associate professor at Department of Artificial Intelligence, Wrocław University of Science and Technology, Poland. He received the MEng degree in computer science from Wrocław University of Technology, Poland in 2008, and the PhD degree in late 2012. In 2012, he also received another MEng degree in computer science from Blekinge Institute of Technology, Sweden. In 2020, he received a Habilitation (DSc) in information and communication technology. He was a visiting scholar at Stanford University, Australia in 2013, and a visiting professor at University of Technology Sydney, Australia in 2018 and 2019. He has authored over 100 scholarly and research articles on a variety of areas related to complex networks and computational network science, focusing on the extraction and dynamics of communities within social networks, spreading processes in complex networks, and the analysis of multilayer networks.