19th Workshop on Modelling and Mining Networks

Network Diffusion — Framework to Simulate Spreading Processes in Complex Networks

Michał Czuba [1], Mateusz Nurek [1], Damian Serwata [1], Yu-Xuan Qi [2], Mingshan Jia [2], Katarzyna Musial [2], Radosław Michalski [1], Piotr Bródka [1]

[1]Wrocław University of Science and Technology
[2]University of Technology Sydney

05.06.2024

Network Science Lab

Agenda

In this talk, I will introduce a computational library — `network-diffusion`. Here is the agenda:

1. Motivation
2. Key Features
3. Example I - a Predefined Model
4. Example II - a Custom Model
5. Resources and References
6. Limitations of `network-diffusion`

Network Science Lab

Motivation

Spreading phenomena are one of the issues considered by a network science. They can be obeserved in various areas like: dynamics of political opinions, marketing campaigns, spread of epidemics, computer viruses, etc.

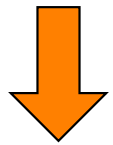
Figure: Artistic representation of a social network.[1]

[1] Source: www.uniroma3.it/articoli/seminario-biased-opinion-dynamics-when-the-devil-is-in-the-details-138122

Network Science Lab

Motivation

Analytical approaches are often insufficient for large graphs, prompting researchers to use computational methods, i.e. simulations.



Thus, like in other branches of computer science, there have been developed tools which addres that issue, allowing to avoid starting from scratch and enhancing the reproducibility of results.[2]

[2] Although network science still ingloriously stands out in this matter.

Network Science Lab

Motivation

There is a bunch of tools that helps in sumulating diffusion processes in networks:

- **NDlib**[RMR+18],
- GLEaMviz[BGG+11],
- SimInf[WBEE19],
- STEM[DBE+19],
- EpiModel[JGM18],
- Sispread[ACBV07],
- ...

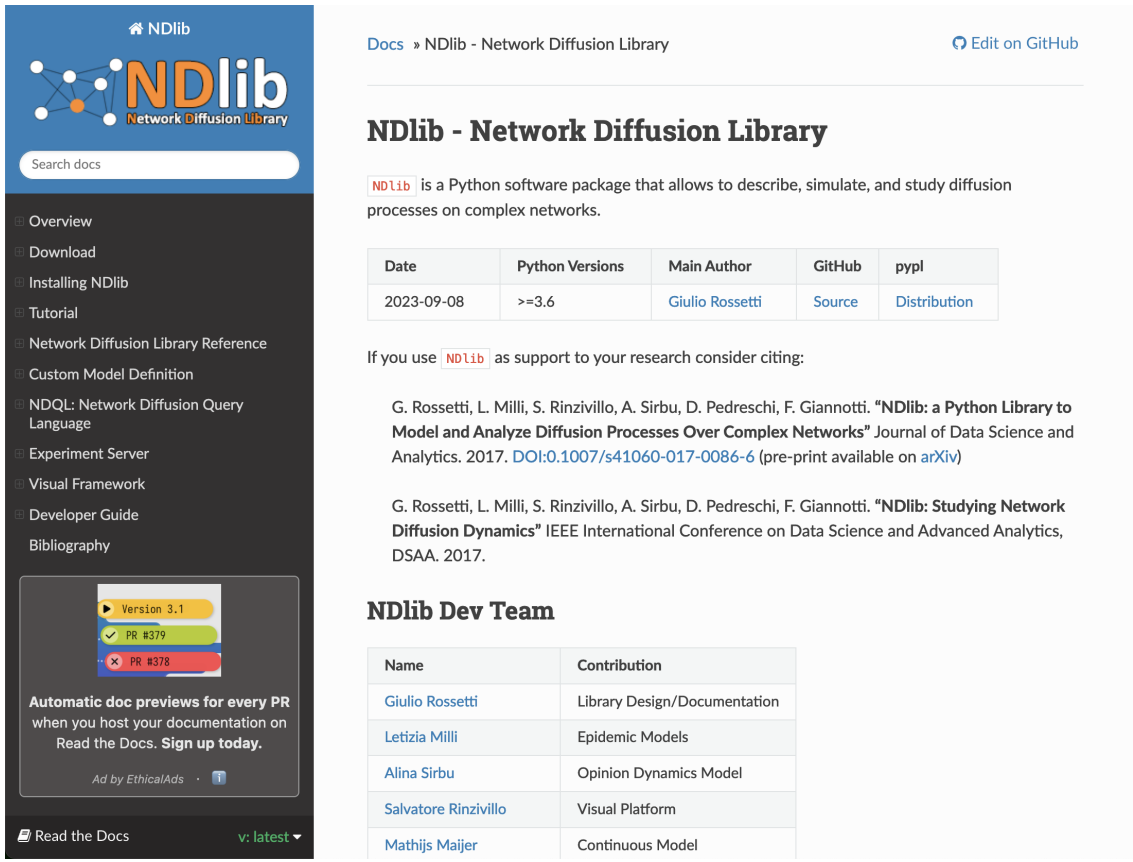


Figure: NDlib's website.

Network Science Lab

Motivation

However, if we consider...

...more complex network models...

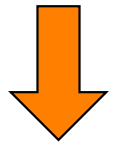
...spreading multiple processes at the same time...

...**a gap among the available toolkits** emerges.

Network Science Lab

Motivation

A focus of our research group is oriented i.a. to: multilayer networks, temporal networks, spreading phenomena, data streams.



As a result of our recent activities, we decided to merge and wrap up a code we developed into a reusable library. We also decided to to share it with the community in an attempt of filling the gap in.

The main operating principles that we determined were:

1. compatibility with other tools commonly used in data science,
2. development of a tool as a framework with open interfaces,
3. supporting both multilayer and temporal networks,
4. supporting spreading models with discrete states.

Network Science Lab

Key Features

Functionalities of `network-diffusion`:

- end-to-end simulation workflow,
- predefined spreading models,
- an interface for implementing custom spreading models,
- support for the temporal network models (CogSNet + discrete windows),
- support for the multilayer networks,
- centrality measures for multilayer networks.

Network Science Lab

Key Features

Environmental requirements for `network-diffusion`:

- support for Linux, macOS, and Windows[3],
- Python (preferred 3.10) compatibility,
- C snippets in the CogSNet module to speed-up computations,
- NetworkX compatibility.

[3] Although we develop and use it mostly on Unix OSs

Network Science Lab

Key Features

To prepare the experiment we have to provide a network, a spreading model and auxiliary parameters. Then, the simulation unfolds as follows:

```
1: procedure PERFORM_PROPAGATION(network, model, epochs)
2:     states_0 ← model.determine_initial_states()
3:     model.update_network(states_0)
4:     for e in [1, ..., 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
```

Network Science Lab

Example I - a Predefined Model

In this example, we will see how to trigger spread under the Linear Threshold Model within a simple, multilayer network.

Linear Threshold Model

Each node:

- can fall in two states: *active* and *inactive*,
- becomes *active* if the fraction of its *active* neighbors to all neighbours exceeds certain threshold.

In case of multilayer networks the actors not the nodes[4] are a subject of the diffusion. Thus, we have to define how to aggregate impulses from the layers. In this example we will consider "OR" strategy, which says that the actor can be activated if any of nodes representing it in the network gets activated.

[4] which can be considered as avatars of the actors on the network's layers

Network Science Lab

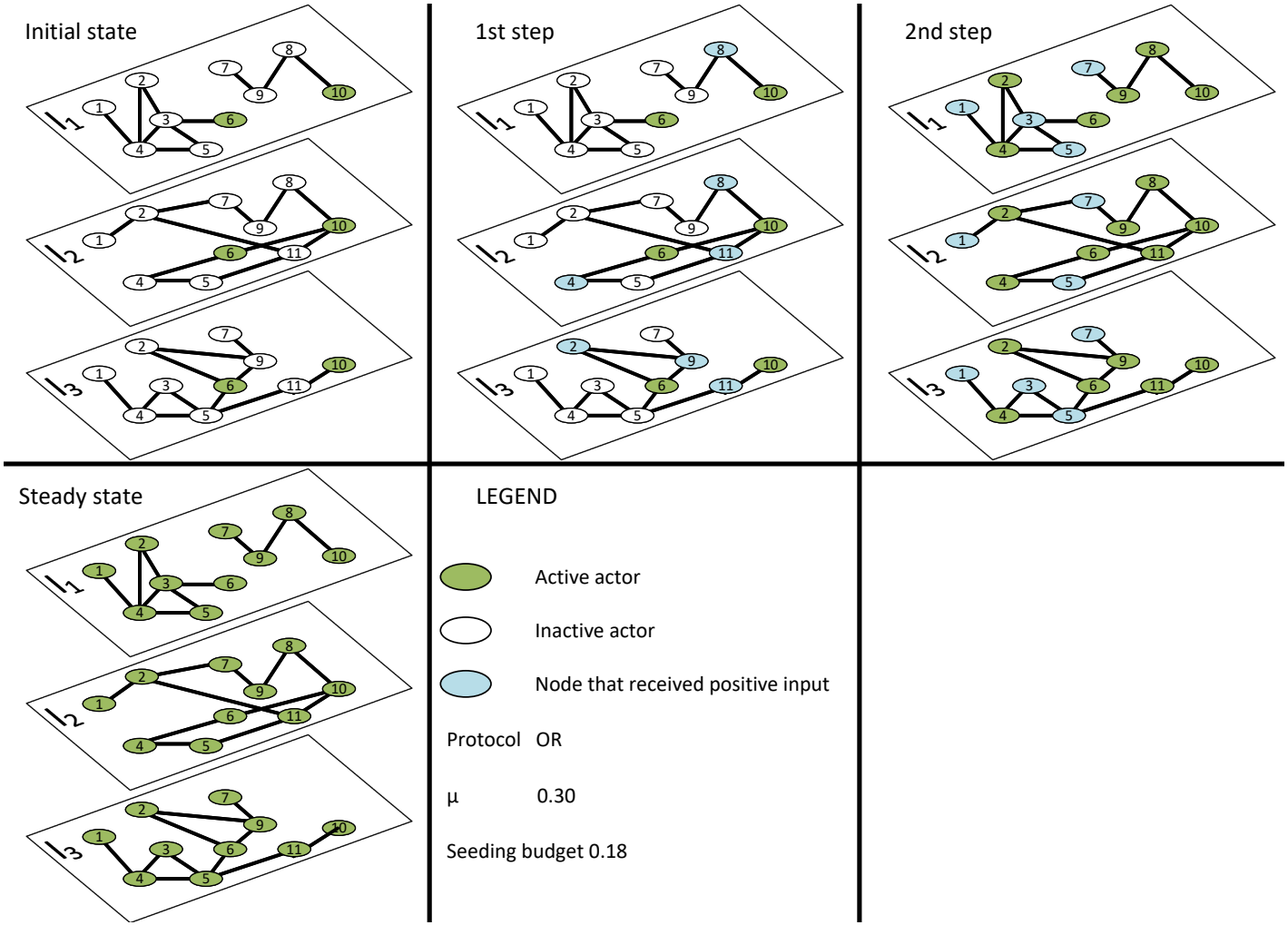


Figure: Propagation according to LTM with the *OR* protocol in a multilayer network - active actors: seeds $\{6, 10\}$, in a stable state: all of them.

Network Science Lab

Example I - a Predefined Model

Let's model this problem with `network-diffusion`!

Example II - a Custom Model

In this example we will consider a joint disease-awareness model (SIR-UA) that can be used e.g. to assess the effectiveness of various countermeasures against the spread of COVID-19 (**see the next presentation for details**):

Table: Transition weights with explanation.

Symbol	Formula / Value	Description
α	0.19	probability of infection for unaware agents
α'	$0.35\alpha = 0.07$	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

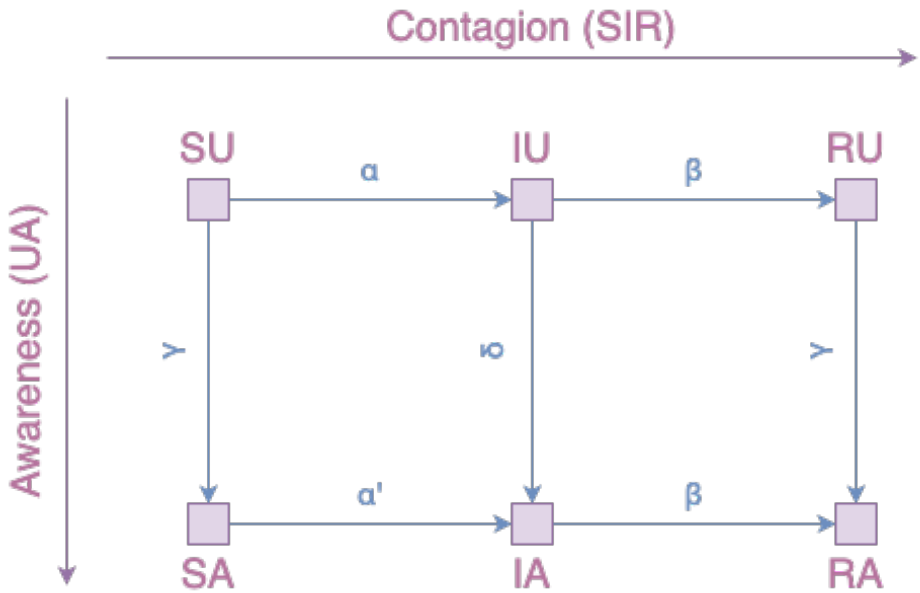


Figure: State and transition graph for SIR-UA.

Network Science Lab

Example II - a Custom Model

Let's model this problem with `network-diffusion`!

Resources and References

The library can be installed via:

```
pip install network-diffusion
```

Other useful resources have been also published:

- PyPI website: `pypi.org/project/network-diffusion`
- GitHub page: `github.com/anty-filidor/network_diffusion`
- Reference guide: `network-diffusion.readthedocs.io`
- A preprint of the paper: `arxiv.org/abs/2405.18085`

Network Science Lab

Limitations of `network-diffusion`

- **Python Implementation**
  - Advantage and drawback due to slower performance.
- **Partial C Code**
  - Some code in C with Python bindings, limited by small team.
- **Support for Discrete Spreading**
  - Limited to discrete spreading processes.
- **Framework Design**
  - Set of interfaces for custom experiments.
  - Few pre-defined spreading models compared to NDlib.
- **No User Interface**
  - Not user-friendly for non-programmers.
  - Alternatives like NetLogo required.

Network Science Lab

References I


Fabian Alvarez, Pascal Crépey, Marc Barthelemy, and Alain-Jacques Valleron, Sispread: A software to simulate infectious diseases spreading on contact networks, Methods of information in medicine **46** (2007), 19.


Wouter Van den Broeck, Corrado Gioannini, Bruno Gonçalves, Marco Quaggiotto, Vittoria Colizza, and Alessandro Vespignani, The gleamviz computational tool, a publicly available software to explore realistic epidemic spreading scenarios at the global scale, BMC Infectious Diseases **11** (2011), no. 1, 37.


Judith Douglas, Simone Bianco, Stefan Edlund, Tekla Engelhardt, Matthias Filter, Taras Günther, Maggie HuKun, Emily Nixon, Nereyda Sevilla, Ahmad Swaid, and James Kaufman, Stem: An open source tool for disease modeling, Health security (2019).

Network Science Lab

References II


Samuel M Jenness, Steven M Goodreau, and Martina Morris, Epimodel: an r package for mathematical modeling of infectious disease over networks, Journal of statistical software **84** (2018).


Giulio Rossetti, Letizia Milli, Salvatore Rinzivillo, Alina Sîrbu, Dino Pedreschi, and Fosca Giannotti, Ndlib: A python library to model and analyze diffusion processes over complex networks, Int. J. Data Sci. Anal. **5** (2018), no. 1, 61–79.


Stefan Widgren, Pavol Bauer, Robin Eriksson, and Stefan Engblom, SimInf: An R package for data-driven stochastic disease spread simulations, Journal of Statistical Software **91** (2019), no. 12, 1–42.

Network Science Lab

Thank you for your attention!

Network Science Lab