Network Diffusion — Framework to Simulate Spreading Processes in Complex Networks

Wrocław University of Science and Technology

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

Network Science Lab

Introduction

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. With the advancement of computational network science analytical approaches became insufficient for large graphs, prompting researchers to use computational methods. In recent years, the scope of network science 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. Unfortunately, the pace of advancement is often too rapid for existing computational packages to keep up with the functionality updates...

Problem

There is a bunch of very good and robust tools that helps in sumulating diffusion processes in networks, e.g. `ndlib` (which we love).

However, if we consider...

...more complex network models,...

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

...a gap among the available toolkits emerges.

Our Contribution

In order to address the issue, we decided to redesign, polish, and share our internal environment which we are using in the lab. Thus, in our recent work [1], we presented `network-diffusion`. To start using the library just type in your shell:

```
pip install network-diffusion
```

Or scan this code to reach the docs with more examples:

Key Features

- **End-to-End Simulation Workflow**: The library enables users to simulate diffusion processes in complex networks with ease. Whether you are studying information spread, disease propagation, or any other diffusion phenomena, this library has you covered.
- **Support for Temporal Network Models**: You can work with temporal models, allowing you to capture the dynamics of processes over time. These temporal models can be created using regular time windows or leverage CogSnet.
- **Support for Multilayer Network Models**: The library supports multilayer networks, which are essential for modelling real-world systems with interconnected layers of complexity
- **Predefined Spreading Models**: You have the option to use predefined diffusion models such as the Linear Threshold Model, Independent Cascade Model, and more. Those are implemented to simplify the simulation process, allowing users to focus on their specific research questions.
- **An Interface for Implementing Custom Spreading Models**: Additionally, `network-diffusion` allows you to define your own diffusion models using open interfaces, providing flexibility for researchers to tailor simulations to their unique requirements.
- **New Centrality Measures**: The library provides a wide range of centrality measures specifically designed for multilayer networks. These measures can be valuable for selecting influential seed nodes in diffusion processes.
- **NetworkX Compatibility**: The package is built on top of `networkx`, ensuring seamless compatibility with this popular Python library for network analysis.

Example

Extending Linear Threshold Model to multilayer Networks

In a diffusion under Linear Threshold Model [2], 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 (μ).

In case of multilayer networks — **actors are the subject of the process, while the nodes are their auxiliary representation**. Thus, we have to define how to aggregate impulses from the layers. In this example we will consider method introduced by [3] — a protocol functions, namely "OR" variant which says that the actor can be activated if any of nodes representing it gets activated.

Visualisation

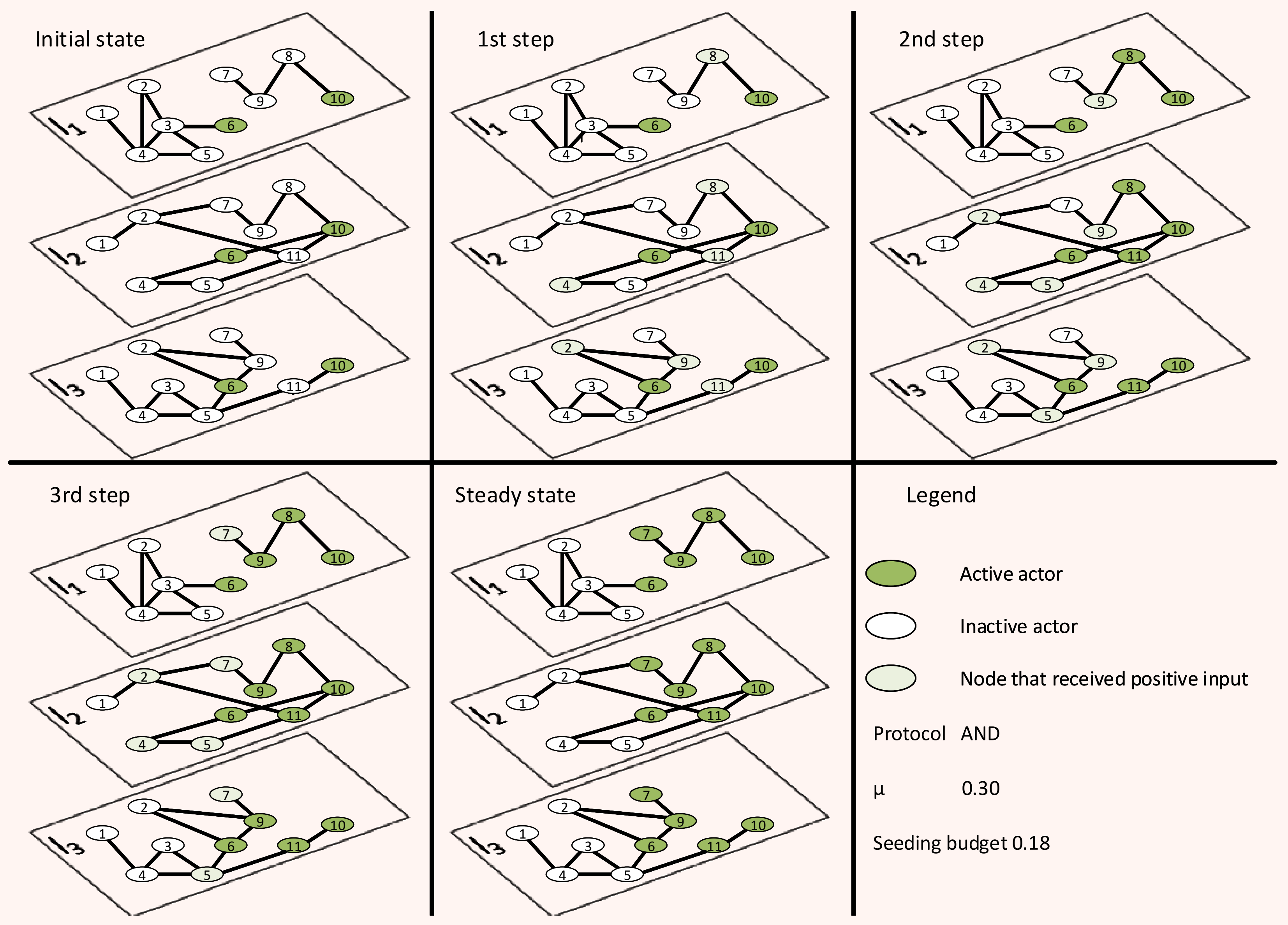

Figure 1. Example of spreading of MLTM in toy network with protocol AND.

Modelling this case with `network-diffusion`

```python
import network_diffusion as nd

# define the model with its internal parameters
spreading_model = nd.models.MICModel(
    seeding_budget=[90, 10, 0],  # 95% act suspected, 10% infected, 0% recovered
    seed_selector=nd.seeding.RandomSeedSelector(),  # pick infected act randomly
    protocol="OR",  # how to aggregate impulses from the network's layers
    probability=0.5,  # probability of infection
)

# get the graph - a medium for spreading
network = nd.mln.functions.get_toy_network_piotr()

# perform the simulation that lasts four epochs
simulator = nd.Simulator(model=spreading_model, network=network)
logs = simulator.perform_propagation(n_epochs=3)

# obtain detailed logs for each actor in the form of JSON
raw_logs_json = logs.get_detailed_logs()

# or obtain aggregated logs for each of the network's layer
aggregated_logs_json = logs.get_aggragated_logs()

# or just save a summary of the experiment with all the experiment's details
logs.report(visualisation=True, path="my_experiment")
```

Outcomes

References

[1] Michał Czuba et al. "Network Diffusion — Framework to Simulate Spreading Processes in Complex Networks". In: *Big Data Mining And Analytics* (2024), pp. 1–13. DOI: `10.26599/BDMA.2024.9020010`. URL: `https://doi.org/10.26599/BDMA.2024.9020010`.

[2] David Kempe, Jon Kleinberg, and Éva Tardos. "Maximizing the Spread of Influence Through a Social Network". In: *9th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2003, p. 137.

[3] Yaofeng Desmond Zhong, Vaibhav Srivastava, and Naomi Ehrich Leonard. "Influence Spread in the Heterogeneous Multiplex Linear Threshold Model". In: *IEEE Transactions on Control of Network Systems* 9.3 (2022), pp. 1080–1091. DOI: `10.1109/TCNS.2021.3088782`.