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

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, we present `network-diffusion`. To start using 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.

Extending the LTM to multilayer networks

Linear Threshold Model in its initial form [1] cannot be directly applied to multilayer networks — **actors are the subject of the process, while the nodes are their auxiliary representation**... Therefore, we need to define:

- what does it mean that an actor is (or is not) active,
- how does it relate to diffusion dynamics taking place within layers, where it is represented.

In our research, we used the approach proposed by [2] with amendments so that a homogeneity among actors has been imposed in the sense of an activation threshold (μ) and a protocol (v.i.).

Protocol function in MLTM

According to [2], state of the actor n of a multilayer network $M = (N, L, V, E)$ in the time step t is determined by a following function:

$$x_n(t) = \begin{cases} 1, & \text{if } \ y_n(t) \geq \delta \text{ or } \ x_n(t-1) = 1 \\ 0, & \text{otherwise} \end{cases}$$

Where:

δ - a parameter of the model, $\delta \in [\frac{1}{|L|}, 1]$,
$y_n(t)$ - a mean input of actor n (represented in K layers) in time t, $y_n(t) = |K|^{-1} \sum_{k \in K} y_v^k(t)$.
$y_v^k(t)$ - an impulse of node v from layer k in time t, $y_v^k(t) \in \{0, 1\}$

Toy example

We decided to examine two extreme cases: $\delta = \mathbf{1}\ \mathbf{(AND)}$ **and** $\mathbf{delta} = \frac{\mathbf{1}}{\mathbf{L}}\ \mathbf{(OR)}$. In the former one, an actor gets activated if it receives sufficient influence on all layers where it is represented, and conversely the latter, where sufficient input in at least one layer is enough for activation.

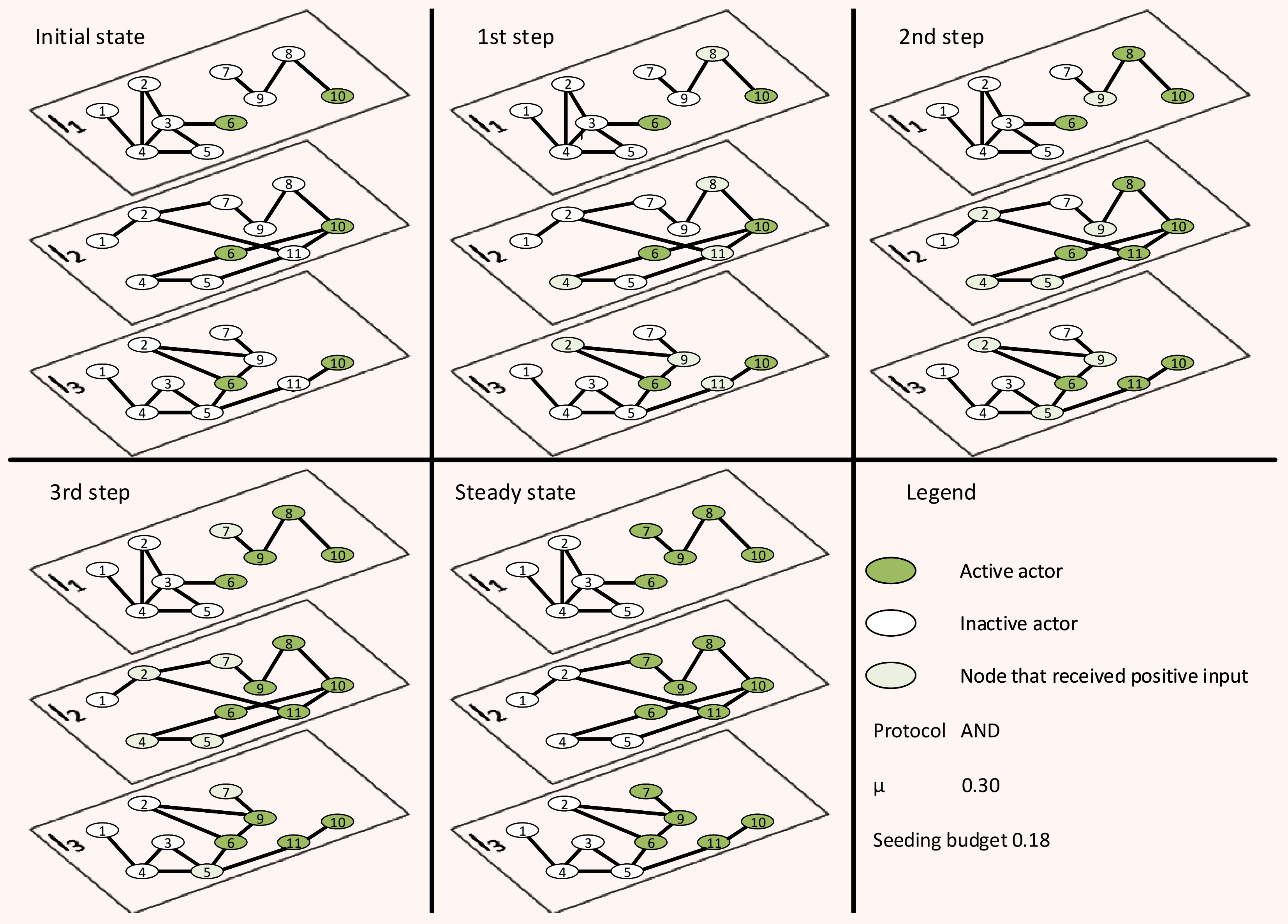

Figure 1. Example of spreading of MLTM in toy network with protocol AND.

A problem we tackled

Budget constrained influence maximisation

Let $\mathcal{S}$ be a family of sets of cardinality s over actors of the multilayer network M in the sense that $\mathcal{S} \subseteq powerset(N)$. Let $\sigma : \mathcal{S} \to \mathbb{R}^+$ b an arbitrary function that maps a set of actors used as a seed set to a number denoting the expected size of activated actors in a binary discrete system. An influence maximisation problem for seeding budget of size s is an issue of finding $S_0 : \arg\max(\sigma) = S_0 \wedge |S_0| \leq s$.

Measuring an efficiency of the diffusion

We used **Gain** metric to assess a performance of the spreading model that bases on a number of seeds, a number of actors that could be activated, and a number of active actors when diffusion faded down:

$$G = 100 \cdot \frac{|S_D - S_0|}{|N - S_0|}$$

Seed selection methods

During the study, we evaluated the following methods to select seed set for MLTM: