Optimising network efficiency in the epidemic scenario

With Deep Epidemic Efficiency Network (DEEN)

Magdalena Proszewska, Michał Bujak,
Rafał Kucharski, Marek Śmieja, Jacek Tabor

Jagiellonian University, Kraków, Poland

18.04.2024

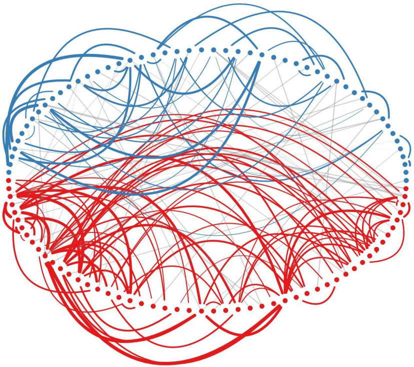
Aim of the study

We aim to **reduce virus spreading** in a system represented by the **graph** structure while **maintaining the highest utility** levels. Deep Epidemic Efficiency Network (DEEN) model:

- based on Graph Convolutional Neural Network with a novel loss function;
- outputs a graph partition maximising utility at a set epidemic threshold;
- applicable to real-life problems, validated against three scenarios;
- capable of maintaining close to the original utility with a great reduction in the spreading potential.

Method showcase

Ride-pooling service with 150 travellers in NYC. **Decomposition** of graph into **two clusters** resulted in **decrease of the performance 3%** and **increase in the epidemic threshold by 170%**.



Notation

- $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ - system: weighted, directed graph;
- $\mathbf{A} = (\mathbf{A}_{ij}) \in \mathrm{R}^{|\mathcal{V}| \times |\mathcal{V}|}$ - adjacency matrix with weights;
- $\Delta = (\Delta_{ij}) \in \{0, 1\}^{|\mathcal{V}| \times |\mathcal{V}|}$:

$$\Delta_{ij} = \begin{cases} 1, & \mathbf{A}_{ij} > 0 \\ 0, & \mathbf{A}_{ij} \leq 0. \end{cases}$$

- Decomposition $\mathcal{H}$ for $\mathcal{G}$ is a subgraph (i.e. $\mathcal{V}_{\mathcal{H}} = \mathcal{V}, \mathcal{E}_{\mathcal{H}} \subseteq \mathcal{E}$) consisting of disconnected components, i.e. $\mathcal{H} = \bigcup_{j \leq k} \mathcal{H}_j$ such that:
  - $\mathcal{V}_{\mathcal{H}} = \bigcup_{j \leq k} \mathcal{V}_{\mathcal{H}_j}$,
  - $\mathcal{E}_{\mathcal{H}} = \bigcup_{j \leq k} \mathcal{E}_{\mathcal{H}_j}$,
  - $\mathcal{V}_{\mathcal{H}_i} \cap \mathcal{V}_{\mathcal{H}_j} = \emptyset$ for $i \neq j$.

  The set of decompositions of graph $\mathcal{G}$ we denote as $\mathcal{D}(\mathcal{G})$.

Utility Function

Utility function **effectiveness of the** network **system** given by the graph. To ensure **applicability** of the algorithm to a variety of scenarios, we make the following assumption of the utility function:

1. may not have a closed-form analytical solution;
2. can be approximated with link weights;
3. is evaluated with an external (black-box) algorithm.

Following postulates by Dawar[1], we assume that the utility function $U : \mathcal{G} \to [0, +\infty)$ is **non-increasing** with respect to the **edge removal**, i.e.

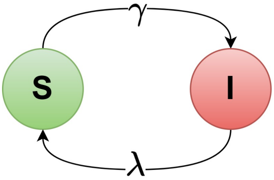
$$\mathcal{H} \subseteq \mathcal{G} \implies U(\mathcal{H}) \leq U(\mathcal{G}).$$

In particular,

$$\mathcal{H} \in \mathcal{D}(\mathcal{G}) \implies U(\mathcal{H}) \leq U(\mathcal{G}).$$

[1] Dawar, Bera, and Goyal, *High-utility itemset mining for subadditive monotone utility functions*

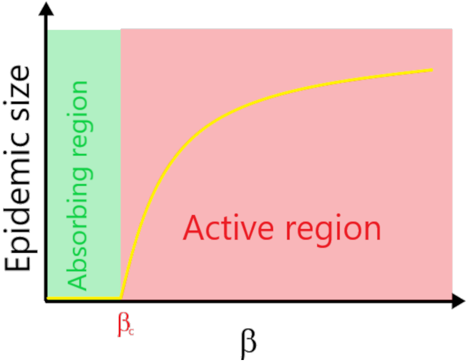
SIS model



- **S** - Susceptible
- **I** - Infected
- γ - Infection rate
- $\boldsymbol{\lambda}$ - Recovery rate

$\boldsymbol{\beta} = \gamma/\lambda$ - Effective transmission rate

We seek the critical effective transmission β_c rate where the epidemic is absorbed with time.



Epidemic threshold

The critical effective transmission depends on the topology of the graph. Hence, we denote it as $ET(\mathcal{G})$.
To account for the different nodes' degrees, we apply the heterogeneous mean-field approach. Following results by Wang[2], for a connected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$,

$$ET(\mathcal{G}) = \frac{\sum_{v \in \mathcal{V}} \deg(v)}{\sum_{v \in \mathcal{V}} \deg(v)^2}.$$

The considered graph $\mathcal{G}$ is not always connected. For the not connected graph $\mathcal{G}$, let $\mathcal{G} = \bigcup_{i \leq K} \mathcal{G}_i$, where $G_i = (\mathcal{V}_i, \mathcal{E}_i)$ is connected for $i \leq K$ and $\mathcal{G}_i \cap \mathcal{G}_j = \emptyset$ for $i \neq j$. We denote $C(\mathcal{G}) = \{\mathcal{G}_i : i \leq K\}$. Then,

$$ET(\mathcal{G}) = \sum_{i \leq K} \frac{|\mathcal{V}_i|}{|\mathcal{V}|} ET(\mathcal{G}_i).$$

[2] Wang et al., "Unification of theoretical approaches for epidemic spreading on complex networks"

Optimisation problem

Input:

- weighted graph $\mathcal{G}$;
- target epidemic threshold β_c;
- utility function U (unknown to us).

Output:

Decomposition $\mathcal{H}_{\max} \in \mathcal{D}(\mathcal{G})$ defined as:

$$\begin{aligned} \max_{\mathcal{H} \in \mathcal{D}(\mathcal{G})} \quad & U(\mathcal{H}), \\ \text{s.t.} \quad & ET(\mathcal{H}) \geq \beta_c. \end{aligned} \tag{1}$$

Model architecture

The framework is realised using **Graph Convolutional Neural Networks** (GCNN) (Kipf and Welling[3]) with the **output** later defines by the **softmax**.
To include information about the node features itself, we modify the weight matrix before passing to the GCNN:

$$\widehat{\mathbf{A}} = \mathbf{A} + \delta \mathbf{I},$$

where $\delta \in \mathbb{R}_+$ (as suggested by Lampert[4]). The GCNN returns the assignment matrix $\mathbf{S} \in \mathbb{R}^{|\mathcal{V}| \times K}$:

$$\mathbf{S} = \mathrm{softmax}(\mathrm{GCNN}(\widehat{\mathbf{A}})),$$

where $K \in \mathbb{N}_+$ is the resulting number of clusters.
Unlike Kipf and Welling, we do experience over-smoothing, hence we do not apply Laplacian normalisation.

[3] Kipf and Welling, *Semi-Supervised Classification with Graph Convolutional Networks*

[4] Lampert and Scholtes, *The Self-Loop Paradox: Investigating the Impact of Self-Loops on Graph Neural Networks*

Loss function

We construct the loss function so to incorporate three factors:

- utility maximisation;
- epidemic threshold maximisation;
- prevent degenerate solutions.

Utility loss

Motivation:

1. represent system performance;
2. differentiable;
3. quickly and analytically computable;
4. be generally applicable.

Reasons 2, 3 and 4 encouraged us to propose a formula which does not include the exact utility formulation in a given problem. Furthermore, our algorithm can find a solution in a setting where the true utility function is unknown.

Utility loss formula

Let $S = [s_i]^T_{i \leq |\mathcal{V}|}$. $s_i \in \mathbb{R}^K$ represents cluster assignment of i-th node.

$$\mathcal{L}_u(\mathbf{S}; \mathbf{A}) = \frac{1}{|\mathcal{V}|} \sum_{i=1}^{|\mathcal{V}|} \sum_{j=1}^{|\mathcal{V}|} a_{ij} (1 - s_i s_j^T)$$

$\mathcal{L}_u(\mathbf{S}; \mathbf{A})$ forces nodes connected by an edge of high weight to be in the same cluster.

$$\mathcal{L}_u(\mathbf{S}; \mathbf{A}) = \frac{1}{|\mathcal{V}|} \mathbf{e}^T (\mathbf{A} \odot (\mathbf{e}\mathbf{e}^T - \mathbf{S}\mathbf{S}^T))\mathbf{e},$$

where $\odot$ denotes element-wise multiplication.
Proper (local) manipulation of the graph edges' weights additionally prevents the creation of isolated nodes.

Virus spreading loss

We aim to **maximise the epidemic threshold**. Softmax is a continuous assignment, hence we approximate the nodes' degree in the continuous form too.

$$d_i = \sum_{j=1}^{|\mathcal{V}|} \sum_{k=1}^{K} \Delta_{ij} \cdot s_{ik} \cdot s_{jk}$$

$$\mathbf{d} = \text{diag}(\Delta^T \mathbf{S}\mathbf{S}^T)$$

Then, we define the virus spreading loss for a connected graph as

$$\mathcal{L}_{vs}(\mathbf{S}; \mathbf{A}) = -\frac{\|\mathbf{d} + \mathbf{e}\|_1}{\|\mathbf{d} + \mathbf{e}\|_2^2}.$$

Virus spreading loss

$$\mathcal{L}_{vs}(\mathbf{S};\mathbf{A}) = -\frac{\|\mathbf{d}+\mathbf{e}\|_1}{\|\mathbf{d}+\mathbf{e}\|_2^2}$$

Presence of $\mathbf{e}$ (self-loop with weight 1) prevents $\mathcal{L}_{vs} \xrightarrow{d\to 0} -\infty$.
For graph $\mathcal{G}$ comprised of $C(\mathcal{G})$ connected components,

$$\mathcal{L}_{vs}(\mathbf{S};\mathbf{A}) = \sum_{(\mathcal{V}_i,\mathcal{E}_i)\in C(\mathcal{G})} \frac{|\mathcal{V}_i|}{|\mathcal{V}|}\mathcal{L}_{vs}(\mathbf{S}^{(i)};\mathbf{A}^{(i)}),$$

where $\mathbf{S}^{(i)}$ and $\mathbf{A}^{(i)}$ represent the assignment and weight matrices for the graph $\mathcal{G}_i$, respectively.

Collapse regularisation

Collapse regularisation proposed by Tsitsulin[5]:

- prevents the trivial decomposition;
- otherwise the algorithm finds local minima (empty clusters) that trap the gradient;
- does not dominate optimisation of the main objective.

Let $||A||_F = \sqrt{\sum_{i \leq m, j \leq n} |a_{ij}|^2}$ denote the Frobenius norm.

$$\mathcal{R}_c(\mathbf{S}) = \frac{\sqrt{K}}{|\mathcal{V}|} \left\| \sum_i \mathbf{S}_i^\top \right\|_F - 1$$

[5] Tsitsulin et al., *Graph Clustering with Graph Neural Networks*

Final loss formula

$$\mathcal{L}_{DEEN}(\mathbf{S};\mathbf{A}) = \mathcal{L}_u(\mathbf{S};\mathbf{A}) + \lambda\mathcal{L}_{vs}(\mathbf{S};\mathbf{A}) + \mathcal{R}_c(\mathbf{S})$$

- λ balances virus spreading and performance;
- high λ prioritise epidemic prevention;
- low λ favours performance;
- in all our experiment we recognised $\lambda = 0.4$ as the optimal level.

Solving optimisation problem

To conduct calculation we need to fix the number of clusters:

- large number yield more components, hence lowering the transmission;
- small number helps to maintain connectivity required for better performance.

To find the optimal level for a target epidemic threshold, we conduct a binary search to find the least number of clusters that exceeds the given level. For technical reasons, we also apply the maximum number of clusters which we consider as a hyperparameter.

Modularity

For an adjacency matrix A of an undirected graph $\mathcal{G}$ with n nodes and m edges, cluster assignments $c_1, \ldots, c_n$,

$$\mathcal{Q} = \frac{1}{2m} \sum_{ij} [\mathbf{A}_{ij} - \frac{d_i d_j}{2m}] \delta(c_i, c_j).$$

Greedy algorithms:

- progressive by Clauset-Newman-Moore[6];
- regressive by Girvan-Newman[7].

[6] Clauset, Newman, and Moore, "Finding community structure in very large networks"

[7] Newman, "Fast algorithm for detecting community structure in networks"

Baselines

Another analytical approach is **spectral clustering**. Vectors associated with positive eigenvalues of spectrum of the graph Laplacian point to minimal cuts.

GNN baselines:

- MinCutPool[8]: approximate the minimum K-cut;
- Just Balance GNN[9]: minimise local quadratic variation;
- DMoN[10]: maximise modularity.

[8] Bianchi, Grattarola, and Alippi, *Spectral Clustering with Graph Neural Networks for Graph Pooling*

[9] Bianchi, "Simplifying Clustering with Graph Neural Networks"

[10] Tsitsulin et al., *Graph Clustering with Graph Neural Networks*

Hyperparameters

For each experiment we use the same architecture and nearly the same hyperparameters.

- 3 non-normalised graph convolutional layers, 1 dense layer;
- ReLU activation after each layer;
- Adam optimiser with learning rate 0.001;
- train till convergence (2000 epochs);
- maximum number of clusters: $\frac{n}{2}$ for transportation experiment, 32 otherwise (for larger graphs).

Three cases

In the experimental part, we analyse three potential real-world applications.

1. **Ride-pooling**: We limit potential combinations of travellers who can share a ride. Our goal is to minimise the pandemic risk while maintaining benefits associated with the ride-pooling service.
2. **Country regions**: Given the region map of Poland, we seek an optimal decomposition (cross-regional lockdown) such that the business and educational exchange is unimpeded, while the pandemic risk is reduced.
3. **Peer-to-peer**: For Gnutella P2P file sharing network, we aim to minimise the computer virus infection risk while maintaining high connectivity between peers.

Case I: Ride-pooling

Setting: Ride-pooling is a transportation service similar to standard taxi, with an additional perk that travellers share parts of their trips. The spatio-temporal distribution of travellers' requested trips creates a compatibility graph, where travellers of similar origin-destination paths (and time) are connected.
Aim: Decompose the compatibility graph in a manner such that the vehicle mileage reduction is maximised and the epidemic threshold is exceeded.
Edge weights:

$$w(v,u) = \frac{d(v) + d(u) - d(v,u)}{d(v) + d(u)},$$

$d(v,u)$ - vehicle mileage when u and v travel together, $d(u)$ - distance when u travels alone.
True utility: External black-box[11].

[11] For creation of the compatibility and utility we rely on Kucharski and Cats, "Exact matching of attractive shared rides (ExMAS) for system-wide strategic evaluations".

Case II: Country regions

Setting: We represent a map of 3000 regions of Poland as a graph, where neighbouring regions are connected. Each region is characterised by its population.
Aim: Optimise an optimal cross-regional lockdown such that the work and education exchange between regions remains stable and the risk of pandemic is reduced.
Edge weights:

$$w(v,u) = \left(\frac{p(v)}{2 \cdot \max_{w \in \mathcal{N}(u)} p(w)} + \frac{p(u)}{2 \cdot \max_{w \in \mathcal{N}(v)} p(w)} \right).$$

True utility: We follow the accessibility formulas by Levison[12]:

$$U(v, \mathcal{R}_v) = \sum_{u \in \mathcal{R}_v / \{v\}} \frac{p(v) \cdot p(u)}{d(u,v)},$$

where $p(u)$ is the population, $d(u,v)$ denotes distance between centres of regions and $\mathcal{R}_v$ denotes regions reachable from v.

[12] Levinson, "Accessibility and the journey to work"

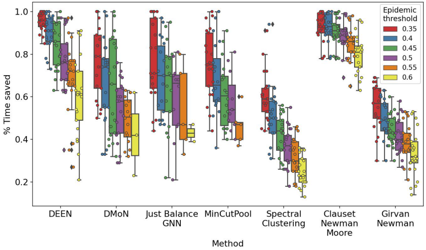
Case III: P2P network

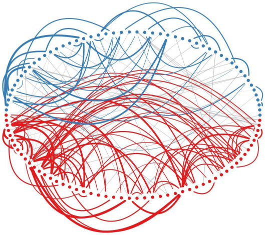
Setting: Gnutella peer-to-peer file sharing snapshots create a vast network. We assume hosts are prone to computer viruses spread via the network.
Aim: Maintain possibly many connections while reducing the risk of computer virus spreading.
Edge weights:

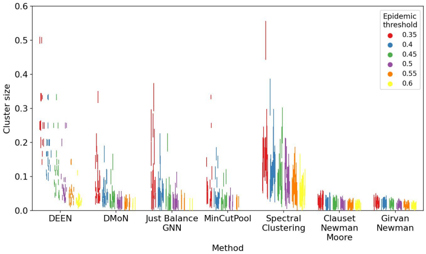
$$w(v,u) = \frac{1}{2 \cdot deg(v)} + \frac{1}{2 \cdot deg(u)}.$$

True utility: The fraction of preserved edges.

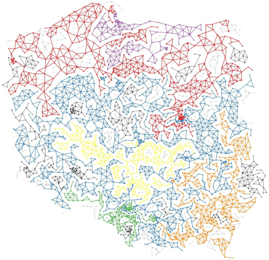
Case I: Ride-pooling



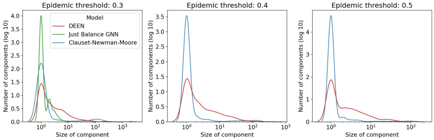




Case II: Regions in Poland

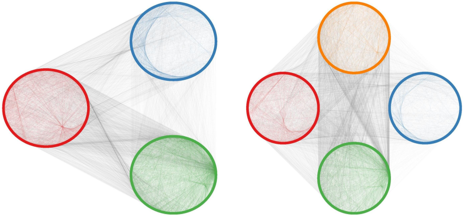


Epidemic threshold	0.3	0.4	0.5
DEEN	0.37	0.36	0.28
DmoN	-	-	-
Just Balance GNN	0.61	-	-
MinCutPool	-	-	-
Clauset-Newman-Moore	0.31	0.29	0.26



Case III: P2P Network

	Utility			Clusters size (AVG $\pm$ SD)		
Epidemic threshold	0.4	0.5	0.6	0.4	0.5	0.6
DEEN	0.38	0.28	0.22	0.25 ± 0.01	0.17 ± 0.01	0.14 ± 0.01
DMoN	0.53	0.43	–	0.36 ± 0.09	0.29 ± 0.08	–
Just Balance GNN	0.59	0.39	–	0.42 ± 0.02	0.28 ± 0.06	–
MinCutPool	0.76	–	–	0.47 ± 0.13	–	–
Clauset-Newman-Moore	0.33	0.26	0.22	0.00 ± 0.00	0.00 ± 0.00	0.00 ± 0.00



Ablation study