Graph Neural Networks: A New Frontier in Network Optimization

Pr. Yassine Hadjadj-Aoul

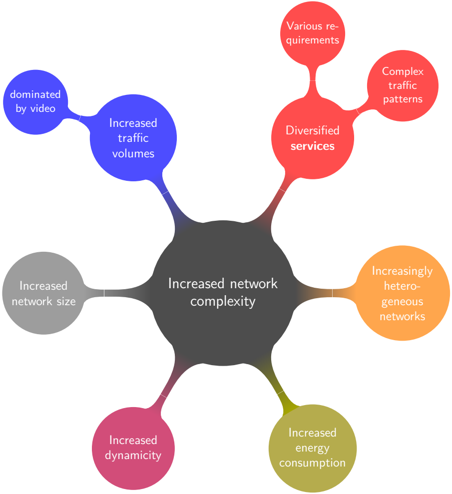
University of Rennes
IRISA Lab.
Ermine INRIA Team-project (previously Dionysos Team-project)

Presentation outline

Presentation outline

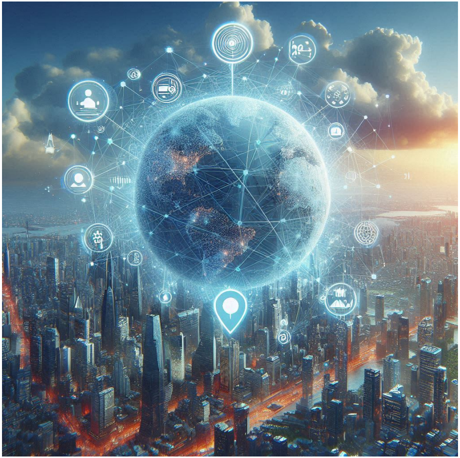
The Challenge of Network Optimization I

- **Increased network complexity**
  - Networks are constantly **evolving** (i.e., xG)
  - Network **conditions change** dynamically (traffic patterns fluctuate, failures, ...)
  - **High dimensionality**, with numerous factors to consider (traffic volume, latency, bandwidth allocation, other services, etc.)
  - **Heterogeneity**



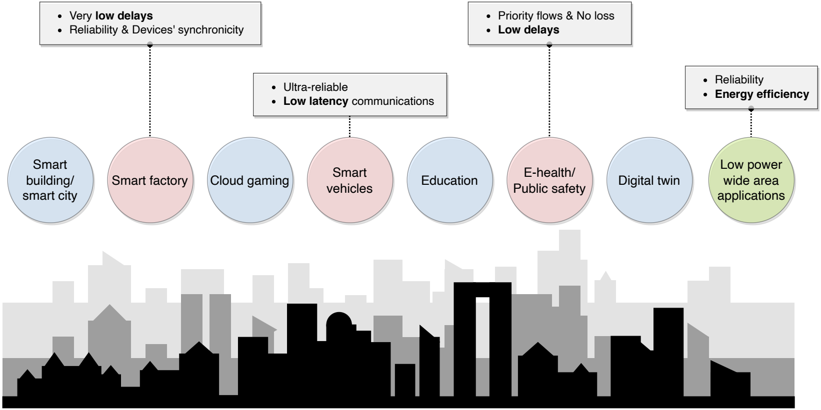
The Challenge of Network Optimization II

- **Distributed Nature of Networks**
  - Networks span **vast distances** (i.e., edge-cloud continuum)
  - Local decisions can have a **global impact**
  - Increased **difficulty** in monitoring and managing network performance



The Challenge of Network Optimization III

- **Multi-objective optimization required**
  - **Trade-offs** are inevitable: e.g. minimizing costs might involve reducing bandwidth allocation, potentially leading to degraded service quality.



Network optimization problems - Combinatorial challenges

- Many **network optimization problems are combinatorial** in nature.
    - These problems involve finding the optimal configuration or solution from a finite set of discrete options (i.e., generally formulated using MILP).
    - While some **MILP problems** can be solved efficiently, others are **known to be NP-hard**.
- Two examples of problems in networking :
    - **Network Tomography** (not necessarily combinatorial)
    - **Network Slicing** (combinatorial)

Limitations of Conventional Methods[1]

- **Mathematical optimization** have the limitation of not always being applicable in a real context
  - The latency of resolution
  - Unsuitability in a real context
    - For service placement problems, the resulting latency and loss are placement-induced measures and cannot be properly included in an optimization problem.
- **Heuristics** are very fast but present some difficulties in finding good solutions
  - stuck on local minimums
- **Meta-heuristics** are slow[2] and they require a realistic simulation environment

[1] PTA Quang, Y. Hadjadj-Aoul, et al., "A deep reinforcement learning approach for VNF-FG Embedding", TNSM, 2019

[2] PTA Quang, …Y. Hadjadj-Aoul, "VNF-FG Embedding: A genetic algorithm approach". Communication Systems, 2019

Limitations of Conventional Methods[1]

- **Mathematical optimization** have the limitation of not always being applicable in a real context
  - The latency of resolution
  - Unsuitability in a real context
    - For service placement problems, the resulting latency and loss are placement-induced measures and cannot be properly included in an optimization problem.
- **Heuristics** are very fast but present some difficulties in finding good solutions
  - stuck on local minimums
- **Meta-heuristics** are slow[2] and they require a realistic simulation environment

In these approaches, past experiences yield no benefit to solve new problems . . . no learning

[1] PTA Quang, Y. Hadjadj-Aoul, et al., "A deep reinforcement learning approach for VNF-FG Embedding", TNSM, 2019

[2] PTA Quang, ...Y. Hadjadj-Aoul, "VNF-FG Embedding: A genetic algorithm approach". Communication Systems, 2019

The Rise of Machine Learning for Network Optimization

Machine Learning offers new possibilities

- Potential benefits:
  - **Learn** from data and adapt to changing network conditions
  - **Handle complex**, multi-objective optimization **problems**
  - Offer potential for **automation** and faster decision-making

AI-based resolution of networking problems

Several classes of resolution methods:

Supervised learning	Unsupervised learning	Reinforcement learning
Knowing input X and output y (labels), we try to find f, $y = f(X)$ **mapping**	Knowing input X, we **classify** it regarding some cost function	We learn how to take actions (**policy**) to maximize a reward function
Possible only when labelled data is available	Our past attempts (using constrained GANs) have not been successful	Designed to solve decision problems (even combinatorial)

Presentation outline

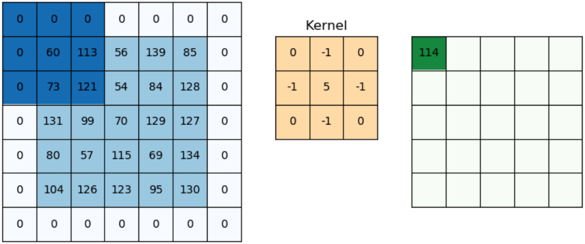
Motivations

- Many **network optimization problems** are inherently based on **graph** structures
  - Network optimization tasks like routing, network slicing, and network tomography rely on understanding the graph topology.
- **Traditional machine learning** techniques **struggle to effectively capture** the rich relational structure and dependencies in **graph data**.
  - Convolutional Neural Networks (**CNNs**) are efficient with a **grid**-like structure (e.g. images).
  - Recurrent Neural Networks (**RNNs**) are well-suited for **sequences** (e.g. Time series prediction).
- Graph Neural Networks (**GNNs**) are **designed for graph data**:
  - Can **learn** from features of nodes and edges in a graph.
  - Could *potentially* lead to **generalize the learning**.

From CNNs to GNNs I

Inspired by animal vision systems, CNNs have made significant contributions to the field of deep learning.

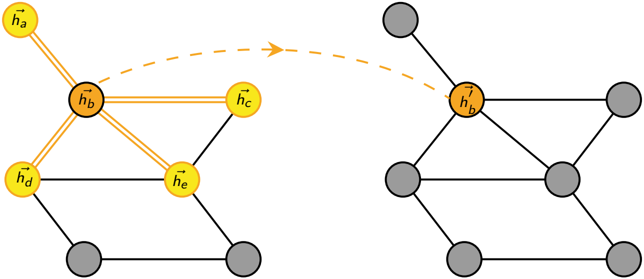
- **Key features : layered structure** with **convolutional layers** and **pooling layers** that are **effective in handling grid-like data** such as images.
  - The operator (kernel) is **applied everywhere in the same way** $\rightarrow$ allow capturing patterns.



From CNNs to GNNs II

With graphs we want something similar:

- Considering **immediate local neighborhood**.
  - **Message-passing** neural network
- Using that information to **update further nodes features**.



$$\vec{h'_i} = g\left(\vec{h_a},\ \vec{h_b},\ \vec{h_c},\ \vec{h_d},\ \ldots\right)$$

Desirable properties of GNNs

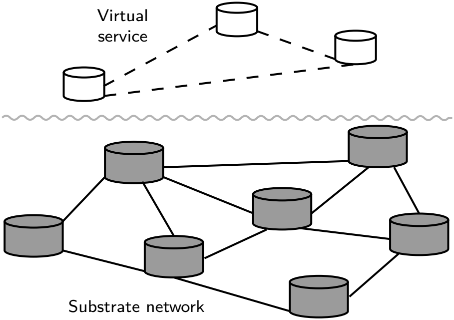
- Fixed number of parameters (independent from input size)
  - Applying a graph convolution layer to graphs of arbitrary sizes.
- Specifying **different importances** to different neighbours.
  - Through **learnable parameters**
- Aggregation function should be permutation invariant (e.g. sum)
  - **Graphs are unordered data structures**: the order of a node's neighbors is arbitrary and does not carry any meaningful information
  - **Consistency in representations:** If the aggregation function is not permutation invariant, different orderings of the same set of neighbors would result in different aggregated representations for the same node.

Presentation outline

Network slicing

Key function: Placement of services in a **VNF-FG** form

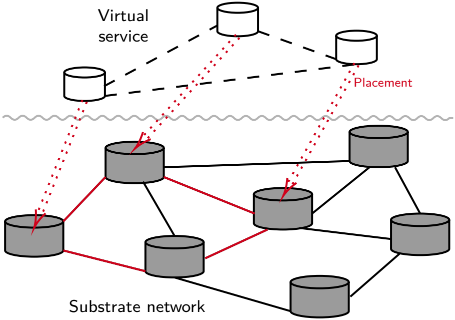
- Involves not only the **placement of VNFs** or CNFs[1] but also addressing a **routing** problem
  - either sequentially or simultaneously
- Need to consider several requirements
  - QoS + system requirements + energy + services' scalability, ...



Network slicing

Key function: Placement of services in a **VNF-FG** form

- Involves not only the **placement of VNFs** or CNFs[1] but also addressing a **routing** problem
  - either sequentially or simultaneously
- Need to consider several requirements
  - QoS + system requirements + energy + services' scalability, ...



Extremely large number of possibilities of placement (very large action space)

- Difficulty in finding an optimal placement, except for very small network instances (**NP-hard problem**)

[1] CNF: Cloud Native Function

Safe DRL-based Network slicing (Traditional)

Using vanilla DDPG[1] for the placement:

- **not suitable** for very large-scale discrete action space
- **no guarantees**

How to ensure a safe placement?

- **Idea:** Knowing an optimal solution, one could find weights for the placement of nodes (using First-Fit) and links (using Dijkstra) to be optimal.
- **Solution:** Learn to find such weights using DDPG (combining DDPG with a Heuristic Fitting Alg.[2])
  - Ensures that you have at least the performance of the heuristic

[1] T.P. Lillicrap et al., "Continuous control with deep reinforcement learning", CoRR, vol. abs/1509.02971, DeepMind, 2015

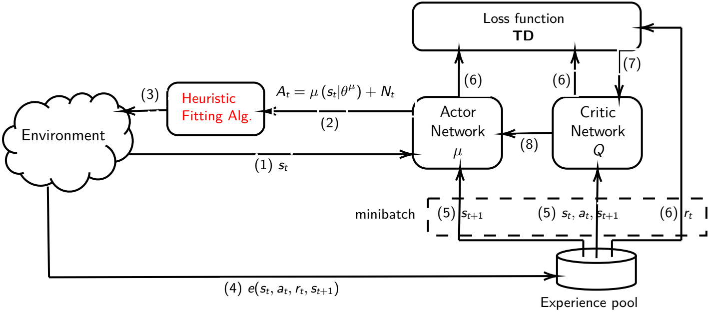
[2] PTA Quang, Y. Hadjadj-Aoul, et al., "A deep reinforcement learning approach for VNF-FG Embedding", TNSM, 2019

HFA-DDPG[1]

State: K VNR

Action: Weights for the placement (for all nodes and links)

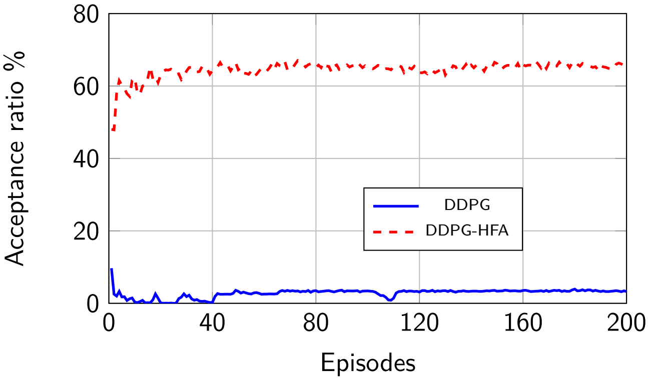
Reward: Acceptance ratio $= \frac{\#\text{ deployed VNR}}{N} \times 100$



[1] PTA Quang, Y. Hadjadj-Aoul, et al., "On Using Deep Reinforcement Learning for VNF-FG Placement", NoF, 2020

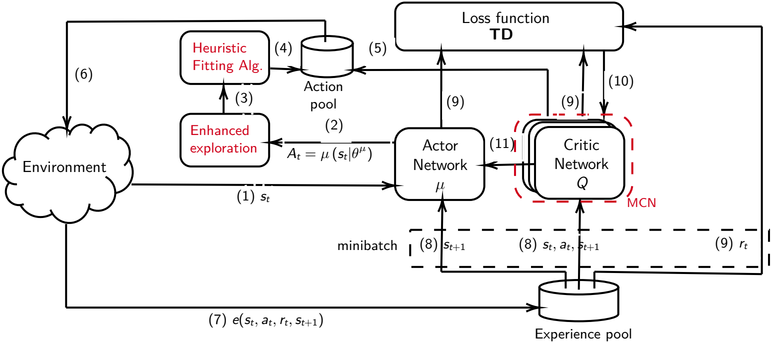
DDPG vs DDPG-HFA

Each point represents the placement of a randomly generated set of VNF-FG.



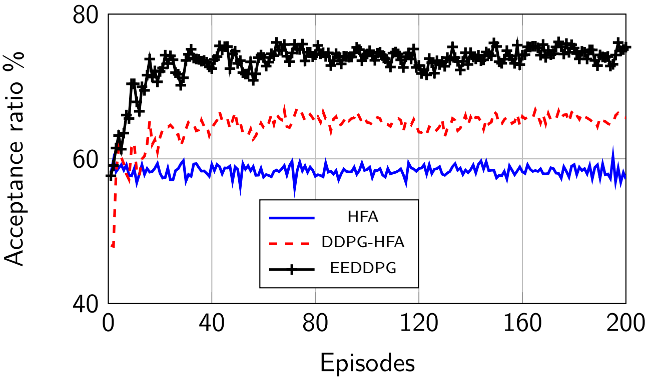
The convergence of the proposed strategy almost immediately with very few episodes.

Enhanced Exploration DDPG Model[1]

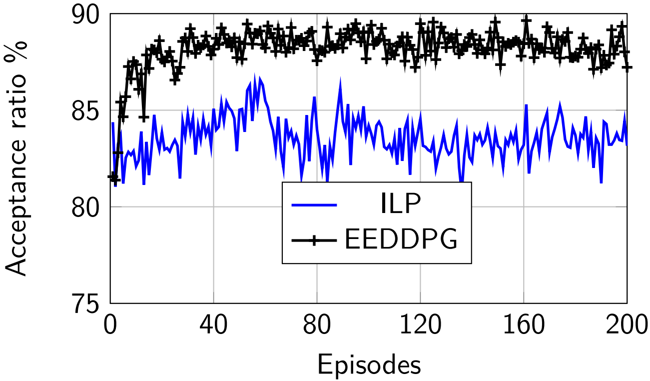


[1] PTA Quang, Y. Hadjadj-Aoul, ..., "On Using Deep Reinforcement Learning for VNF ...Placement". Demo NoF, 2020

EEDDPG - Efficiency



EEDDPG vs ILP



Advantages and limits of the proposal

Advantages:

- **Safe** strategy
  - Allowing to have in the **worst case** the performance of the considered **heuristic**.
- Can **beat many** existing **approaches** (*not always true*)

Limitations

- A very **costly** learning process
- Any topological change implies the **need to learn again** from scratch

Learn to improve policies (GNNs-based)

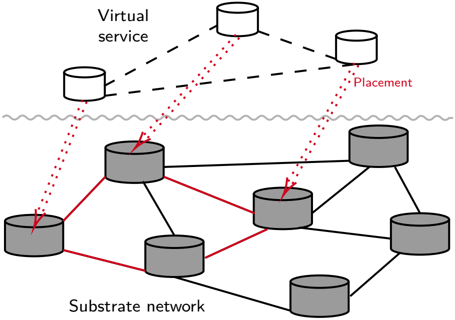
Heuristics are not that efficient, and learning a solution from scratch, may result in an **unsafe** learning

- Having a baseline (e.g., using a heuristic) of the placement's performance is important
- Ensure that the **worst-case** result is equal to the one obtained with the **heuristic**

How to improve the placement of the heuristics ?

- **Idea:** Training an agent to **reduce the optimality gap** of VNE heuristics.
- **Solution:** Modeling the process of improving the quality of the heuristics as a reinforcement learning problem.

[1] A. Rkhami, Y. Hadjadj-Aoul, ..., "Learn to improve: A novel deep reinforcement learning approach ...". CCNC, 2021



The solution is not a homogeneous graph,

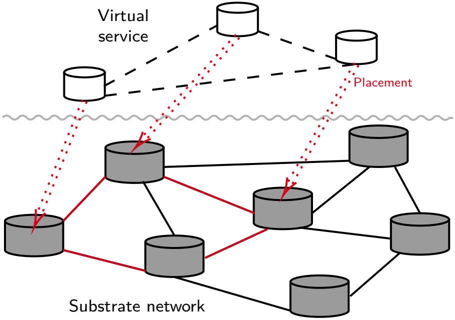
- 2 types of nodes, and 3 types of links

The solution is a heterogeneous graph,

- Graph Convolutional neural Networks (GCN) can deal only with homogeneous graphs
- Relational Graph Convolutional Neural Networks (**RGCN**) was defined as an extension of GCN to **extract features from heterographs**

[1] M. Schlichtkrull, et al., "Modeling relational data with graph convolutional networks". Springer, 2018

RGCN-based state representation



The solution is not a homogeneous graph,

- 2 types of nodes, and 3 types of links

The solution is a heterogeneous graph,

- Graph Convolutional neural Networks (GCN) can deal only with homogeneous graphs
- Relational Graph Convolutional Neural Networks (**RGCN**) was defined as an extension of GCN to **extract features from heterographs**

The main objective is to extract semantic

[1] M. Schlichtkrull, et al., "Modeling relational data with graph convolutional networks". Springer, 2018

Model description I

State (features): Heterogtaph stucture with nodes features:

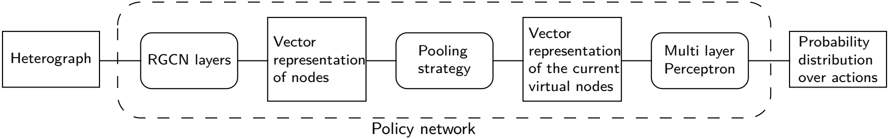
- For each virtual node:
  1. the CPU required by the VNF
  2. **total bandwidth** requested by the virtual links to which the node is attached
  3. a **flag** indicating if the VNF is the current VNF to process
- For each substrate node:
  1. the remaining amount of CPU
  2. **remaining bandwidth** of links to which the substrate node is attached
  3. the number of its **neighbors**

Model description II

Action: applied for the **current** virtual **node** (randomly selected)

- Keep the same placement
- Modify it into another substrate that does not host any other VNF from the same request

Learn a probability distribution over actions

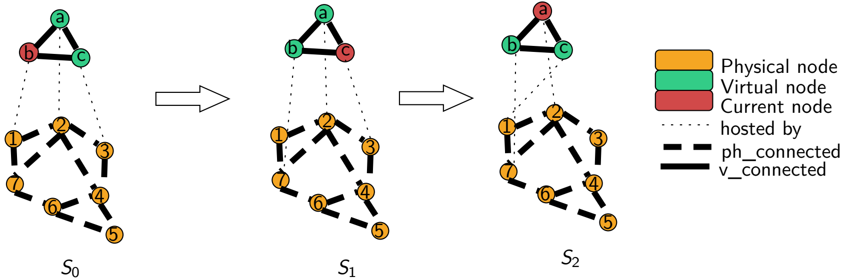


Model description III

Reward:

```
function getReward(bp, r2c)
    reward ← 0
    if r2c = 0 then
        reward ← −100                    ▷ unfeasable solution
    else
        reward ← (r2c − bp)
    end if
    if r2c > bp then
        bp ← r2c                         ▷ new best score
    end if
    return reward, bp
end function
```

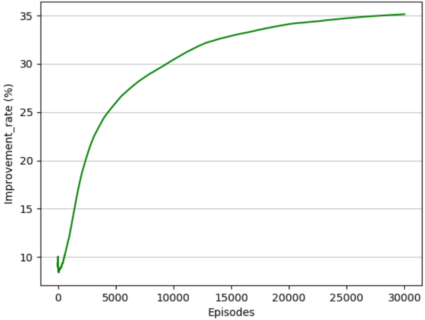
Sequential process of Improvement



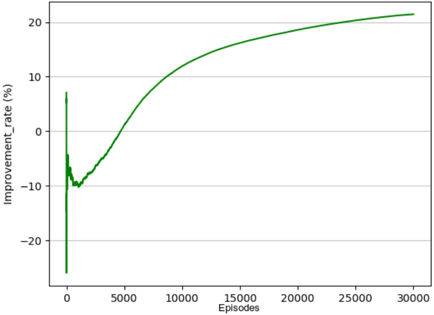
- The GNNs allow here to process any graph in the input.
- The output represents the targeted node.
  - The learning is therefore **not dependent on the input** → changing the topology **do not require relearning**.

First-Fit & Best-Fit improvement

First-Fit improvement



Best-Fit improvement



GNNs for Network Slicing: Promises and Challenges

Promises:

- Ability to capture network graph structure and node dependencies.
- Generic and transferable approach across different topologies.
- Superior performance over traditional heuristics.

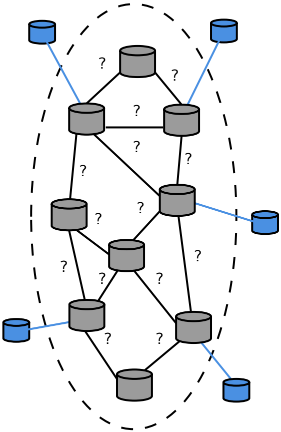
Challenges:

- Lack of interpretability of learned representations.
- Risk of over-smoothing and loss of local information
  - Choosing the right representation is not straightforward due to the **aggregation process, which can cause information to vanish** (we are still grappling with this issue).

Presentation outline

Network tomography I

- A technique used to **diagnose and troubleshoot network** performance issues.
  - By **analyzing data** collected from various points within the network
  - It helps **reconstructing key performance metrics**.
- **General idea:** is to deduce what is happening inside a network from measurements taken from the outside.
  - The ultimate goal is to ensure **complete observability** of the network, enabling informed decisions to be made and performance to be optimised.



Network tomography II

Main problem:

- Identifying links X from paths P measurements (inverse problem)

$$AX = Y \tag{1}$$

where $A(i,j) = 1$ if j belongs to path p_i.

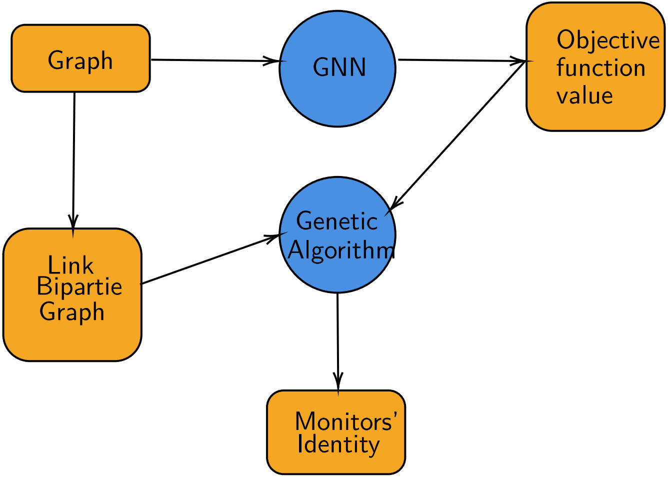
- Typical situation : undetermined system (number of paths smaller than the number of variables).

Network tomography III

Sub-problems:

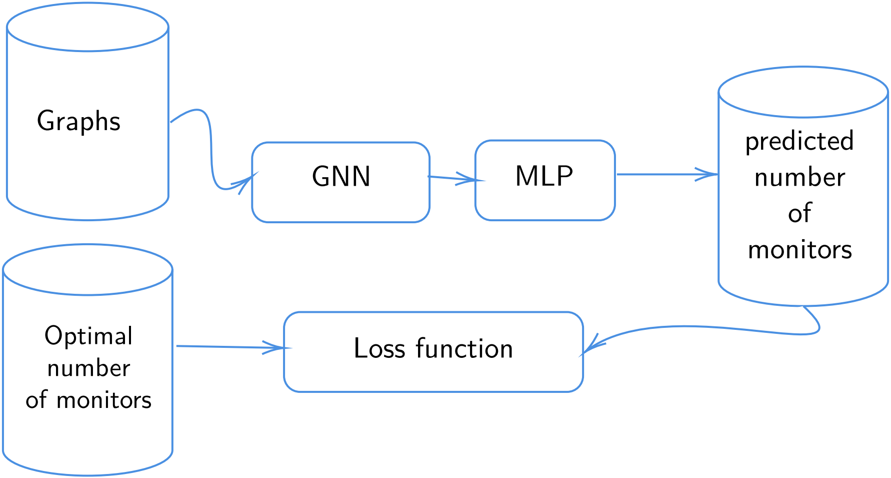
- Determining the optimal number of monitors
- Identifying the best monitors' location.
- Determining the minimal set of paths (or cycles) required to estimate accurately links (or a subset of links in case of network slicing)

Determining the optimal number of monitors using GNNs



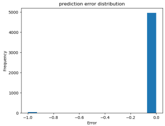
[1] A. Rkhami, Yassine Hadjadj Aoul, . . .: MonGNN: A neuroevolutionary-based solution …slices monitoring. LCN 2021

Determining the optimal number of monitors I

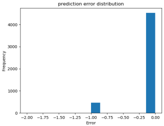


Learning is generalized to any graph structure (any graph as input, the number of monitors as output).

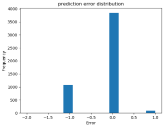
Determining the optimal number of monitors II



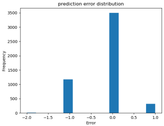
(a) $n = 20$



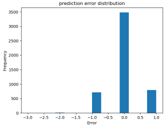
(b) $n = 30$



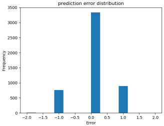
(c) $n = 40$



(d) $n = 50$



(e) $n = 60$

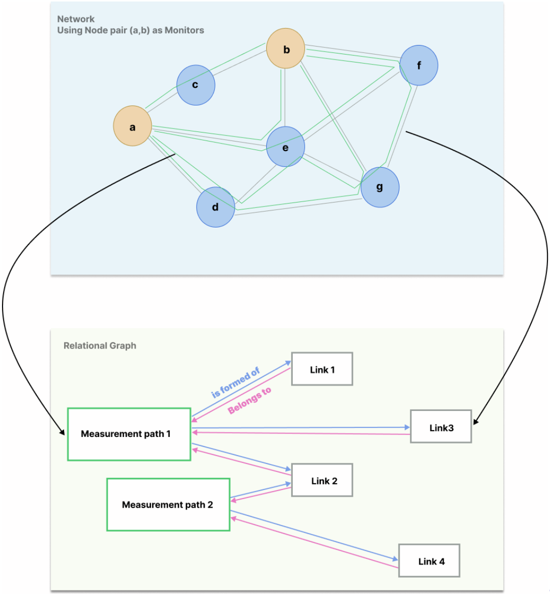


(f) $n = 70$

Figure: Error prediction of number of monitors with Barbasi-Albert graphs

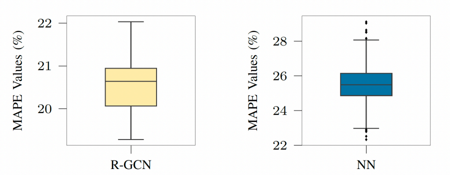
[1] A. Rkhami, Yassine Hadjadj Aoul, . . .: MonGNN: A neuroevolutionary-based solution ...slices monitoring. LCN 2021

Generalizing Monitors Selection in Network Tomography

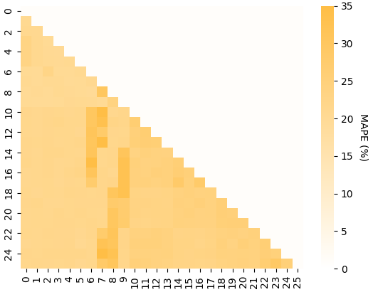


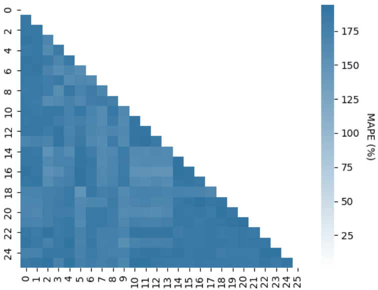
Testing with Mandala topology with 26 nodes and 48 links

- **Small advantage** for R-GCN.
- But not always the case, as for some use cases NN are superior !



Changing the monitors without relearning . . .





R-GCN NN

GNNs for Network tomography: Promises and Challenges

Promises:

- Generic and transferable approach across different topologies.
  - Predicting the number of monitors
- Links identification
  - complicated (solved by removing the MLP to remove the dependency to the output size)
- Superior performance over traditional approaches (SVD, NN).

Challenges:

- Learning links prediction with small network topologies, and predicting links values for bigger topologies without relearning.

Presentation outline

Conclusion

- Several ongoing contributions to progress towards more efficient strategies.
- The question addressed remains open issues.
- A special thanks to my **partners** (Nokia Bell Labs, Orange, TDF, and EXFO), my **colleagues** and my **students**, thanks to whom I have been able to go further than I would have done on my own.

> "We can only see a short distance ahead, but we can see plenty there that needs to be done."
>
> Alan Turing
> Computing machinery and intelligence, 1950