MonGNN: A neuroevolutionary-based solution for 5G network slices monitoring

Anouar Rkhami, Yassine Hadjadj-Aoul, Gerardo Rubino, Abdelkader Outtagarts

MonGNN: A neuroevolutionary-based solution for 5G network slices monitoring

Anouar Rkhami
Inria, Univ. Rennes, IRISA

Yassine Hadjadj-Aoul
Univ. Rennes, Inria, IRISA

Gerardo Rubino
Inria, Univ. Rennes, IRISA

Abdelkader Outtagarts
Nokia-Bell Labs, France

***Abstract*—Monitoring the status of network slices is a priority for network operators to ensure that SLAs are not violated. To overcome the limitations of direct slices' monitoring, network tomography (NT) is seen as a promising solution. NT-based solutions require constraining monitoring traffic to follow specific paths, which we can achieve by using segment-based routing (SR). This allows deploying customized probing scheme, such as cycles' probing. A major challenge with SR is, however, the limited length of the monitoring path. In this paper, we focus on the complexity of that task and propose MonGNN, a standalone solution based on Graph Neural Networks (GNNs) and genetic algorithms to find a trade-off between the quality of monitors' placement and the cost to achieve it. Simulation results show the efficiency of our approach compared to existing methods.**

I. Introduction

By leveraging the latest advances in Network Functions Virtualization (NFV) and Software-Defined Networking (SDN), Network Slicing (NS) enables an efficient support of constrained and diverse services, in terms of Quality of Service (QoS) needs, on top of a single infrastructure. This capability is considered to be one of the most important features of 5G networks and beyond [1]. Indeed, NS opens up new business opportunities for network operators by allowing them to automatically lease customized network slices to third parties.

Despite the advantages it brings, NS poses a set of challenges. Once the slices are leased, network operators will have to ensure that the negotiated Service-Level Agreements (SLAs) with slices' tenants are not violated. As a result, monitoring the status of the slices will be a priority for them to perform troubleshooting and traffic engineering.

Several approaches in the literature focus on the problem of network monitoring, considering in particular end-to-end measurements. These approaches are generally referred to as network tomography (NT) [10]. Some recent works focus on a specific approach of source routing for probing, where only cycles are used [2]. The probing cycle schemes have a significant advantage compared to regular paths. Since the source probe is actually the destination, this reduces synchronization problems. Despite the flexibility offered by source routing in monitoring, this technology presents many challenges. Indeed, most commercial switches only support a limited number of SIDs in packet headers [8]. This adds a constraint on the length of the probed cycles.

The construction of an NT based solution is done through three main steps: placement of monitors, deployment of the probing strategy (paths to follow) to collect end-to end measurements and then the inference of network metrics. In this work we focus on finding the optimal number of monitors and their placement such that we cover all the slices by probing cycles of limited length. The task becomes then equivalent to the minimum set covering problem, which is known as an NP-Complete problem [4].

Several solutions have been proposed to solve the problem of monitors' placement, that can be classified into two classes: exact and heuristics. Exact methods ensure optimality but they are applicable only on small instances. On the other hand, heuristics can be applied to large instances but they are suboptimal.

To find a trade-off between the quality of the solution and the cost to obtain it, we propose MonGNN, an autonomic solution based on Graph Neural Networks (GNN) to learn to predict the optimal number of monitors and then use genetic algorithms to reveal their identity.

The main contributions of our paper can be summarized as follows:

- We propose an Integer Linear Programming (ILP) formulation of the following problem: given a set of slices running some services, determine the optimal placement of monitors in order to cover all these slices using cycles whose length is less than a certain amount k.
- We propose a learning-based approach to find a trade-off between the quality and the cost of the solution.
- We unveil the potential of Graph Neural Networks to predict the optimal number of monitors.
- We define a Genetic Algorithm (GA) to identify the monitors, based on the predicted value of the objective function.
- We enhance the GA by a local search method to deal with the estimation error that can occur in predicting the number of monitors.
- We conduct extensive simulations and we compare our approach against benchmark solutions. The results show the superiority of our technique in terms of optimality gap as well as computational complexity.

The remainder of this paper is organized as follows. We introduce background information in Sec. II. Sec. III presents the main related work on network tomography. In Sec. IV, we provide an overview of the context of the paper and the notation we have used as well a formal definition of the

problem we aim to solve. Then, in Sec. V, we delve into the details of our proposed solution. In Sec. VI, we describe both the simulation setup and the results analysis. Finally, in Sec. VII, we conclude and give an overview of our future work.

II. Background

A. Graph Neural Networks (GNN)

Graph Neural Networks (GNNs), are a type of neural networks dedicated to graph structured data. They were introduced in [18] and several variants of it have been proposed [21]. Graph Convolutional Networks (GCNs) and Graph Attention Networks (GATs) are among the most known variants.

GCNs [12] generalize the convolution operation from euclidean data (images and grid shaped data) to non-euclidean data (graphs). The key objective of GCNs is to learn a function that for each node in a graph generates a vector representation of it by aggregating its own features and its neighbors' features.

Let $\mathcal{G} = (V, E)$ be a graph, where V is the set of nodes and E the set of edges. We assume that each node $n \in V$ is characterized by an input feature vector $\boldsymbol{in}_n$. The graph convolution operation takes as input the feature vector $\boldsymbol{in}_n$ and outputs a new vector $\boldsymbol{out}_n$ according to the following equation:

$$\boldsymbol{out}_n = f\Big(\sum_{m \in \mathcal{N}(n)} \frac{1}{\sqrt{d_m d_n}} \boldsymbol{in}_m W^l\Big) \tag{1}$$

where $\mathcal{N}(n)$ represents the set composed of the node n and its neighbours, d_m is the degree of the node m plus one, W^l is the weight matrix of the l^{th} GCN layer, and f is an activation function.

GATs [20] have been proposed as an improvement of GCNs. Instead of doing a normalized sum of the features of the neighbors, GATs include the attention mechanism in the propagation step to generate the nodes' encoding, which is computed using the following equations:

$$\boldsymbol{out}_n = f\Big(\sum_{m \in \mathcal{N}(n)} \alpha_{nm} \boldsymbol{in}_m W^l\Big), \tag{2}$$

$$\alpha_{nm} = \frac{\exp\left(\text{LeakyReLU}\left(a^T \left[\boldsymbol{in}_m W^l || \boldsymbol{in}_n W^l\right]\right)\right)}{\sum_{j \in \mathcal{N}(n)} \exp\left(\text{LeakyReLU}\left(a^T \left[\boldsymbol{in}_n W^l || \boldsymbol{in}_j W^l\right]\right)\right)}, \tag{3}$$

where $||$ denotes the concatenation operator, a is a fully connected neural network and $.^T$ the transposition.

To stabilize the training process, the authors of GAT proposed to use multi head attention. It applies H independent attention head matrices to compute the hidden states and then concatenates their features. With the multi-head attention mechanism the node encoding becomes governed by the following equations:

$$\boldsymbol{out}_n = \|_{h=1}^{h=H} f\Big(\sum_{m \in \mathcal{N}(n)} \alpha_{nm}^k \boldsymbol{in}_m W^l\Big), \tag{4}$$

$$\alpha_{nm}^h = \frac{\exp\left(\text{LeakyReLU}\left(a_h^T \left[\boldsymbol{in}_m W^l || \boldsymbol{in}_n W^l\right]\right)\right)}{\sum_{j \in \mathcal{N}(n)} \exp\left(\text{LeakyReLU}\left(a_h^T \left[\boldsymbol{in}_n W^l || \boldsymbol{in}_j W^l\right]\right)\right)}. \tag{5}$$

B. Segment routing

In segment routing, each probing cycle is encoded by a list of segment IDs (SID). There are two types of SIDs: nodes-SIDs and adjacency-SIDs. The former identify a shortest-path while the latter identify a link segment. The use of nodes-SIDs is feasible; however it may lead to an ambiguous definition of the probing cycle and accordingly introduce a bias in the measurement process [16]. In fact, when using nodes-SIDs to identify a path between a source s and a destination d and due to the load balancing mechanism deployed, we will not be sure which path the traffic followed exactly.

In this work, we consider the adjacency-SIDs and we assume that each probing cycle is encoded only using a list of adjacency-SIDs of a maximum length of k.

III. Related Work

The emergence of networks and services virtualization brings new issues and challenges. Among these, the monitoring of network slices is certainly among of the most important ones given the multiplicity of services and their diversity in terms of QoS.

We generally distinguish between two categories of monitoring: direct and indirect methods. Direct methods rely on protocol-based tools such as SNMP or traceroute. This type of methods incurs a significant monitoring traffic overhead and is not always feasible due to the dependence on cooperation between internal nodes. Indirect methods overcome these limitations and infer the state of the network based on end-to-end measurements between nodes with monitoring capabilities (i.e., monitors). This is known in the literature as Network Tomography (NT) [19]. Another motivation for using NT approaches for the slices' monitoring is their perfect fit with the SDN and the NFV paradigms. Indeed, NT methods require a holistic view of the state of the network, which is guaranteed through SDN technologies. Alongside the holistic view, NT methods also require that monitoring traffic follow a specific path. To achieve this, segment routing (SR) is considered as a promising candidate. It provides, indeed, fast and efficient tools to deploy traffic forwarding strategies along arbitrary paths without any additional protocol or signaling procedures. It enables to constrain the traffic to pass through particular places and thus, to build specific probing schemes. Besides, SDNs allow arbitrary routing, but SR is more scalable and practical [5]. In SR, the path is encoded in the packet header as a list of segment IDs (SIDs). Then, the packet is forwarded according to the instructions corresponding to the stacked SIDs.

Thanks to the advantages it brings, SR routing was proposed as a key tool for network monitoring in several works. In [2], the authors introduce an SR-based monitoring solution where

only a single monitor is deployed, and monitoring probes follow the cycles starting and ending at that monitor. They propose *Scmon*, a framework that minimizes the number of cycles covering the physical topology and encodes efficiently the probing cycles with segment IDs. The use of a single monitor reduces the monitoring cost and avoids the synchronization issues. However, this requires the use of very long cycles to cover all the resources which can introduce a bias in the measurements. Similarly, X.Li and al. [14], focused on using a single monitor and proposed an improvement of the Scmon framework. They put the accent on minimizing the total length of all probing cycles that can be generated from a single monitor. Furthermore, the authors of [13] proposed several ILP formulations of the minimum cycles cover problem. The common points between our proposal and these works is the use of segment routing and cycles probing. However, what distinguishes us is that we focus on finding a subset of nodes as monitors and not a single monitor.

In [7], the authors studied theoretically the problem of monitor placement to identify additive links metrics. They gave necessary and sufficient conditions to identify an additive metric on all the network links for a given topology. Then, they proposed a solution that minimizes the number of monitors to exchange probes with regular paths or cycles. However, no constraint on the length of the path is considered. Similarly, in [9], authors built necessary and sufficient conditions on identifying additive metrics but using only regular paths. They followed an algebraic approach, and they proposed an algorithm based on the graph decomposition into strongly connected components to identify the minimum number of monitors that guarantee the identifiability of all links. Another work [15] focuses on a different vision of the problem of monitoring placement, where the number of monitors is fixed first and then they aim to find their optimal placement to identify as many links as possible.

More recently, M.Rahali and al. [16] focused on the same problem we consider here. They propose a Greedy solution to which we compare our learning based method.

IV. Context & Problem formulation

A. Context

As mentioned earlier, in this work we aim to monitor the state of a set of slices deployed on top of a physical infrastructure and running different services using a network tomography-based method.

In general building a network tomography monitoring solution consists of three main steps: the placement of monitors, the establishment of a probing strategy and finally inferring internal metrics based on external measurements.

In this work, we focus on the optimal monitor placement problem, but under certain constraints: the probing scheme will consist of cycles that we deploy using segment routing. The use of probing cycles eliminates the synchronization problem that one might have when using regular monitoring paths. It should be noted, however, that the use of segment routing adds a constraint to the length of the probing cycles.

B. Problem Formulation

The notation used in this subsection are summarized in Table I. We assume that the topology of the physical network

Notation	Description
$\mathcal{G} = (\mathbb{V}, \mathbb{L})$	The physical topology
$\mathbb{V}, \mathbb{L}$	Set of physical nodes and links
$\mathbb{C}$	Set of measurement cycles
$\mathbb{S}$	Set of slices to monitor
k	Maximum length of probing cycles
$\mathcal{C}_v^k$	Set of cycles starting at node v
$\mathcal{L}_v^k$	Links covered by $\mathcal{C}_v^k$

TABLE I: Main notations

is known, and we model it as an undirected graph $\mathcal{G} = (\mathbb{V}, \mathbb{L})$, where $\mathbb{V}$ and $\mathbb{L}$ represent the sets of physical nodes and of the physical links, respectively. Let $\mathbb{C}$ denote the set of the probed cycles and $\mathbb{S}$ the set of slices we want to monitor and which are deployed on top of the physical network $\mathcal{G}$. The cardinals of the sets $\mathbb{V}, \mathbb{L}, \mathbb{S}$ and $\mathbb{C}$ are respectively : $n = |\mathbb{V}|$, $m = |\mathbb{L}|$, $p = |\mathbb{C}|$ and $s = |\mathbb{S}|$

The objective of collecting the end-to end measurements is to detect failures and to monitor the state of the links on top of which slices are deployed. Therefore, the state of a link is defined by the fact that the monitored metric (e.g. delay, loss rate) is above or below a threshold. Hence, the metric of interest becomes non additive (its value on a link is 1 iff there is an anomaly on that link).

Covering all the physical links on top of which slices are deployed, will allow then the detection of any anomaly. In this work, we focus on covering all physical links, not just the subset of links hosting slices, as this will be more robust in the event of slice migration and will not necessitate recalculating the monitors' placement. From this observation we focus on finding the minimum number of monitors to cover the maximum number of physical links using cycles.

Given the physical network, we denote the cycles formed by at most k links that can be generated from each node v as $\mathcal{C}_v^k$. Similarly, for each vertex v the set of links covered by $\mathcal{C}_v^k$ is denoted as $\mathcal{L}_k^v$.

The first step in solving the monitor placement is to compute for each node v, the set of cycles whose length is less than k: $\mathbb{C} = \{\mathcal{C}_{v_1}^k, \mathcal{C}_{v_2}^k, \cdots, \mathcal{C}_{v_n}^k\}$. Algorithm 1 summarizes this process.

Once the list of cycles is obtained, the next step is to determine the minimum set of monitors $\mathcal{M} \subset \mathbb{V}$ such that: $\bigcup_{v \in \mathcal{M}} \mathcal{L}_v^k = \mathbb{L}$.

In other words, the task of finding the optimal set of monitors which cover all links can be considered as a particular case of the Minimum set cover problem, which is known to be NP-Complete.

To summarize, to determine the optimal set of monitors we

Algorithm 1 Compute Cycles whose length less or equal to k [16]

Input: a graph $\mathcal{G} = (\mathbb{V}, \mathcal{L})$, maximum length of cycles k
Output: The set of cycles $\mathbb{C}$

for all node v in $\mathbb{V}$ **do**
  $v\_start \leftarrow v$
  $\mathcal{C}_v^k \leftarrow \emptyset$
  visited $\leftarrow \emptyset$
  COMPUTECYCLES(v, *visited*, v_start)
  $\mathbb{C} \leftarrow \mathbb{C} \cup \mathcal{C}_v^k$
return $\mathbb{C}$
function COMPUTECYCLES($v, visited, vstart$)
  if $|$*visited*$| > k$ **then**
    return
  if $v = vstart$ **then**
    $\mathcal{C}_v^k.add(visited)$
    return
  for $u \in \mathcal{N}(v)$ **do**
    if $u \notin$ *visited* **then**
      COMPUTECYCLES(v, *visited*, v_start)

need to solve the following integer linear program (ILP):

$$\begin{aligned} \min \quad & \sum_{v \in \mathbb{V}} x_v \\ s.t \quad & \sum_{v: l \in \mathcal{L}_k^v} x_v \geq 1, \qquad \forall l \in \mathbb{L} \\ & x_v \in \{0, 1\} \end{aligned}$$

V. PROPOSED METHOD

To solve the problem presented in previous section and due to its NP-completeness, we introduce MonGNN, a GNN based framework that divides the problem into two tasks: determining the number of monitors and then identifying them.

A. Learn to predict the number of monitors

The main process of the first task is depicted in Figure 1. The objective of this task is to learn a function f that maps a graph and a maximum cycle length to the needed number of monitors. It is defined as follows:

$$f(\mathcal{G}, k) = m_n \tag{6}$$

To learn this function we model it using neural networks (NNs). In fact, NNs can approximate any function as long as the hidden layers contain sufficient numbers of hidden neurons.

Since f takes as input a graph, we used a Graph Neural Network (GNN). GNNs take as input a graph and outputs a node-level representation: a matrix, where each row is a vector that encodes both the attributes and the structure of a vertex in the input graph. Then, to obtain a graph-level representation (a vector that encodes the whole graph), we choose to apply a simple average of the node-level representation. Figure 2 summarizes the main parts of the GNN module represented in Figure 1.

Next this graph encoding is fed into a Multilayer Perceptron (MLP) [17] to obtain the number of monitors.

To train this neural architecture, we consider a supervised setting. We start by the data collection process, which is summarized below:

- We generate N graphs.
- For each graph we compute the cycles map using Algorithm 1.
- Based on the cycles map, we generate a link map that models the relation between the nodes and the links they can cover.
- The problem becomes a set covering problem that we solve using the ILP defined in previous section (using Gurobi[1]) to obtain the optimal labels.

At the end of this process, we obtained a dataset of pairs (graphs, optimal number of monitors needed to cover all links). This dataset was, then, used for the training of the neural architecture minimizing the mean square error (MSE) between the true labels and predicted ones.

Once the training step was done, we got a model that, given a graph, predicts the optimal number of monitors that we need to cover all links.

Apart from the fast prediction, the obtained model doesn't need to compute neither the cycles map nor the link cover bipartite graph in order to determine the number of monitors. It learns how to directly map the raw physical topology to that optimal number of monitors.

B. Monitors identification

Until now, our model is able just to predict the number of monitors we need. In this step we aim to answer the following question: given the optimal number of monitors, how can we determine their identity?

The answer to the question above is depicted in Figure 3. The GNN module refers to a trained model.

The process of revealing the identity of the optimal monitors is done as follows: we feed a graph to the trained module to obtain a prediction of the optimal number of monitors. Then we compute the link bipartite graph, and we feed them with the graph to the genetic algorithm in charge of revealing the identity of monitors. The genetic algorithm (GA) have 5 phases: initialization, fitness score, selection, crossover and mutation. The general workflow of the GA is presented in Figure 4.

- **Initialization step**: In the initialization step, we generate a random generation of solutions (individuals). Each solution is a boolean vector whose size equals the number of nodes in the graph. The j^{th} node is selected as a monitor if and only if the j^{th} of the boolean vector equals one. The generation of this initial population is based on the predicted number of monitors. For example let's suppose that the number of nodes in the graph is set to N and the predicted number is m; in this case, all random

[1] https://www.gurobi.com/

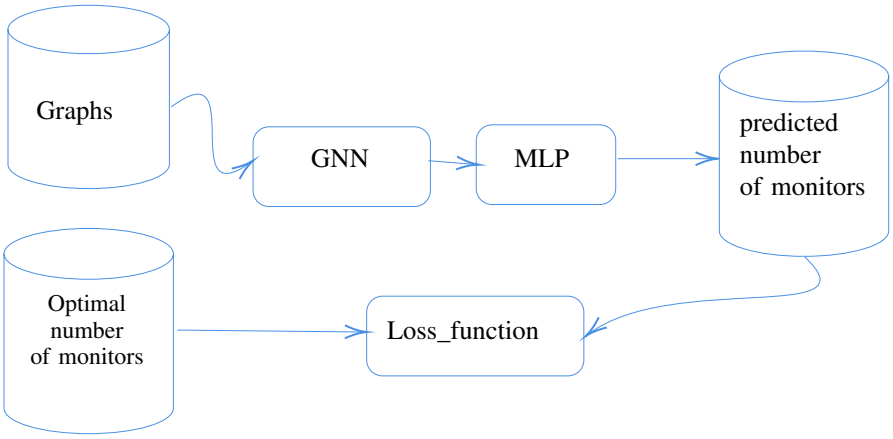


Fig. 1: The training process to learn to predict the number of monitors.

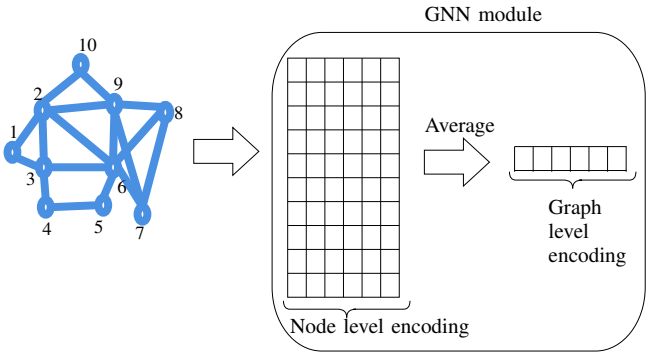


Fig. 2: GNN Module

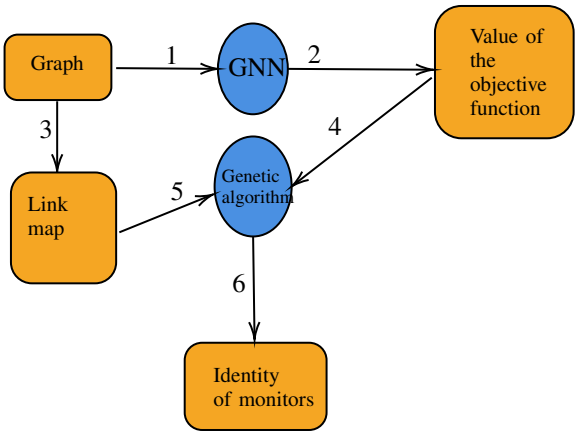


Fig. 3: Process to determine the identity of the monitors

vectors will be of size N and they all contain m ones and $N-m$ zeroes.

- **Selection step and fitness score**: The objective of the selection step is to select individuals (solutions) that pass their genes to the next generation. We consider a simple selection process, only keeping the best elements and killing the rest.
- **Fitness score**: To quantify the fitness of an individual we define a fitness score using the following equation: $s(\text{solution})$ is equal to the number of links covered by the selected elements of the solution.
- **Mutation and cross over:** In the cross-over, the aim is to combine the genes of two individuals in the hope of obtaining a better solution. In the context of our problem, the cross-over is done by taking two individuals and swapping a single different element between them. Finally, the mutation is done by randomly changing the state of two nodes while keeping the number of ones in each individual equal to n

As we will show in the performance evaluation part, the trained model may underestimate or overestimate the optimal number of monitors. In spite of this misestimation, the prediction remains close to the optimal solution because it will add one node at most to the number of monitors. To handle this error we propose a local search approach to enhance the process of determination of the identity of the optimal monitors. The local search process checks if the solution with n nodes cover all links. If it does, it will try to find a solution with $n-1$, otherwise it will increase the number to search with $n+1$ nodes using the genetic algorithm explained before.

VI. Performance Evaluation

In this section, we assess the performance of our approach based on extensive simulations. First we describe the simulation setup and the baselines, to which we compare the performance of our model. Then, we present and discuss the results.

A. Simulation setup

In order to measure the performance of our method, we tested it using different topologies. We consider two families of random graphs: Erdos Reyni(ER) [6] and Barbasi-Albert (BA) [3].

For ER topologies they are defined by two parameters, the number of nodes n and the probability pr of edge creation. To ensure that the generated graph is connected we selected $pr =$

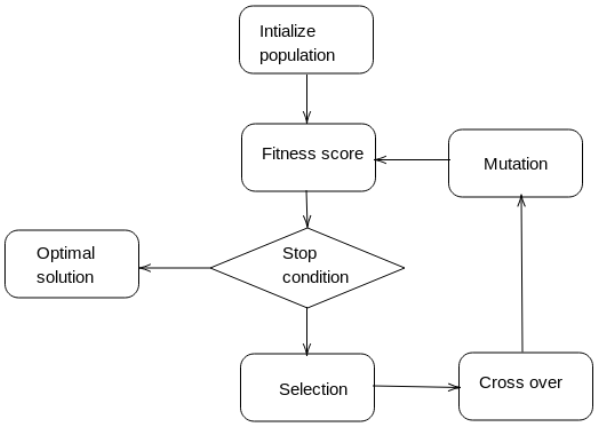


Fig. 4: Genetic algorithm workflow

$2\times\frac{\ln(n)}{n}$. On the other hand, the BA networks are characterized by the number of nodes n and the number of edges to attach from a new node to existing nodes *m_ba* we select *m_ba* $= 2$. For each family we consider six values for the number of nodes n : 20, 30, 40, 50, 60 and 70. For space constraint, and without loss of generality, in these tests we fix the maximal length of cycles k to 6. Other values of k gave similar results.

For each type of graph and for each number of nodes, we generate 20000 graphs that we split into 15000 graphs for training and 5000 for test.

For the neural architecture, in the GNN part we use a two layers GAT network with 3 heads and use the Exponential Linear Unit (ELU) as an activation function. Then, we consider a seven layer MLP where each hidden layer contains respectively 256, 128, 64, 32, 16, 8 hidden units and the output layer with a single neuron. For the training, we set the batch size to 8 and to learn, we use Adam optimizer [11] with a learning rate of 0.0001.

B. Baselines

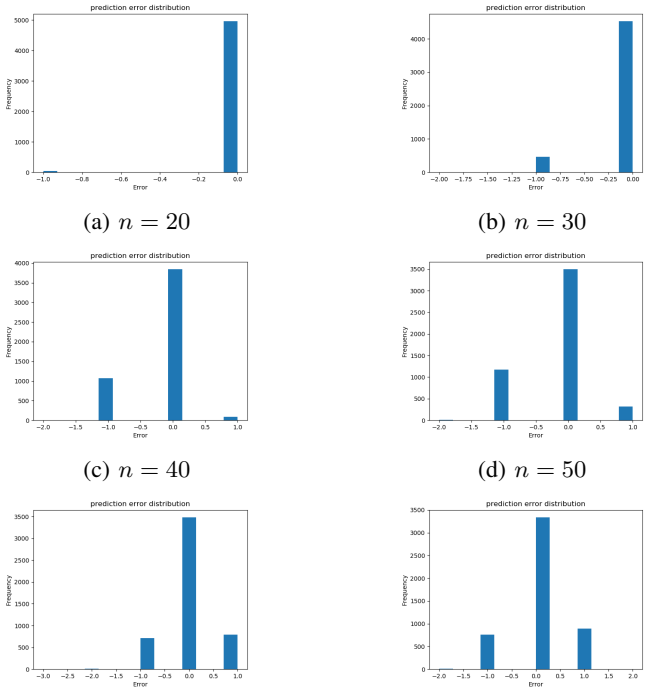
We compare our approach, with two methods:

- A greedy solution that takes as input the link maps and selects iteratively the nodes that cover the highest number of links uncovered.
- A genetic algorithm combined with a binary search (GABS): this is a variant of our proposal but without the GNN part. We directly run the GA to find the optimal solution.

C. Simulation results

1) Prediction of the number of monitors: Fig. 5 and Fig. 6 illustrate the performance of the neural architecture in predicting the number of monitors. We put the accent on the distribution of the prediction error. The prediction was conducted on the test graphs, that the model has never seen during training. In both types of graphs and in all sizes, we observe that the model predicts correctly the optimal number of monitors with a high frequency. Furthermore, when it misses the optimal values, it stays close to it. The absolute value of the error is either zero or one.

Note that, once the model is trained, our model can reach this performance without recalculating neither the cycles of the graphs nor the link map. The model learnt how to directly map the topology of the network into the number of required monitors.



(a) $n = 20$ (b) $n = 30$

(c) $n = 40$ (d) $n = 50$

(e) $n = 60$ (f) $n = 70$

Fig. 5: Error prediction of number of monitors with BA graphs

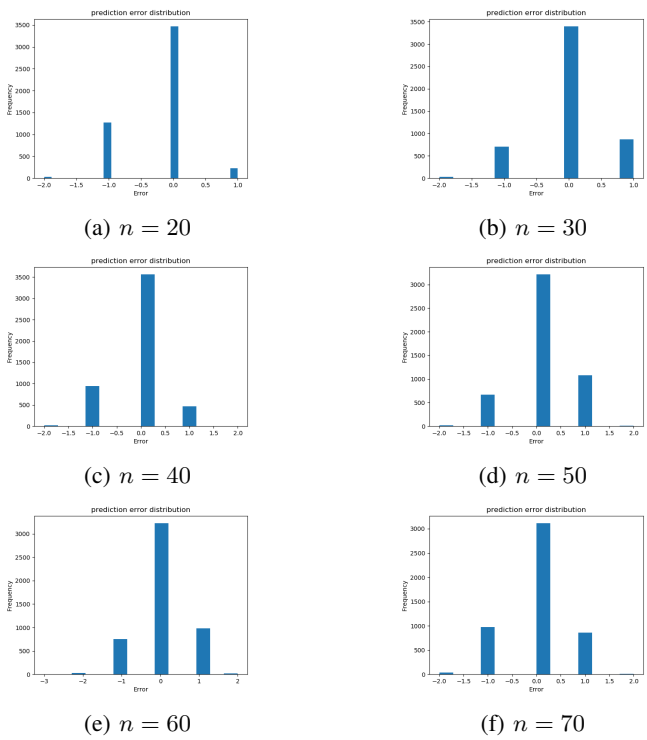
2) Optimality gap: Hereafter, we propose to benchmark the performance of MonGNN against the baselines mentioned earlier. We compare the three methods in terms of the optimality gap, which is defined as follows:

$$optimal\_gap(\%) = 100 \times \frac{optimal\_value - method\_value}{optimal\_value} \tag{7}$$

where *optimal_value* represents the optimal number of monitors and $method\_value$ refers to the number of monitors given by a certain method.

The results for ER and BA graphs are shown, respectively, in Fig. 7 and Fig. 8. We can observe that our methods outperform the greedy solution in all cases, especially with the ER graphs. In the BA graphs, all solutions have the same performance on small graphs. Our method conserve the same behaviour on large graphs, while the optimality gap of the greedy method starts to increase.

MonGNN and GABS have the same performance. We then compare them in terms of execution time. Fig. 9 and Fig. 10

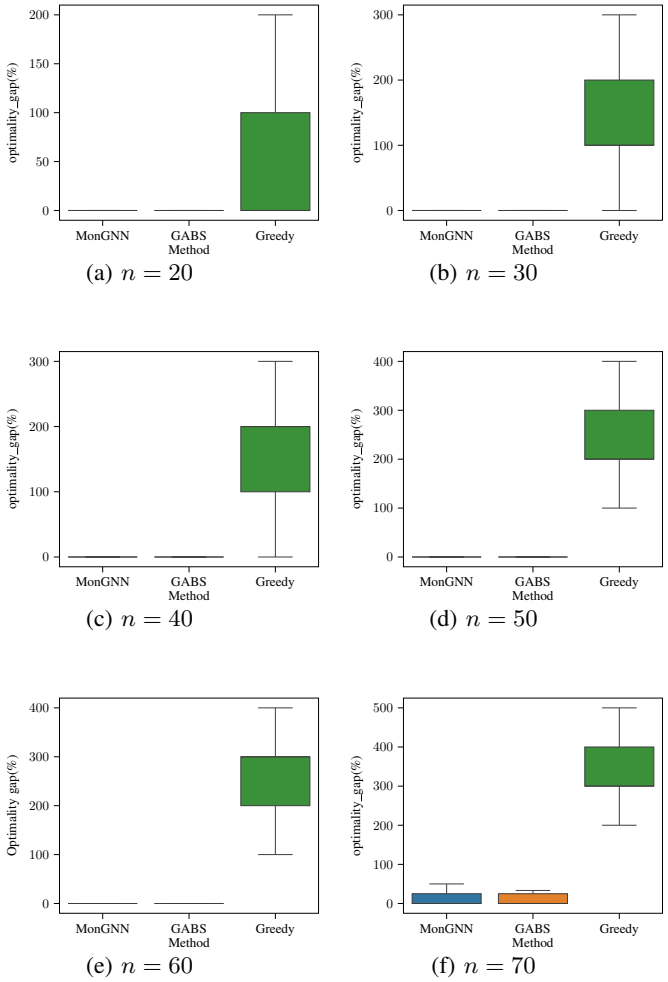


(a) $n = 20$ (b) $n = 30$

(c) $n = 40$ (d) $n = 50$

(e) $n = 60$ (f) $n = 70$

Fig. 6: Error prediction of number of monitors with ER graphs



(a) $n = 20$ (b) $n = 30$

(c) $n = 40$ (d) $n = 50$

(e) $n = 60$ (f) $n = 70$

Fig. 7: Optimality gap : ER graphs

illustrate, respectively, the obtained results for the ER and BA graphs.

Although both methods have quasi the same performance we can see that MonGNN is faster than GABS. In fact, with the learning part, our method can predict efficiently the number of monitors, hence, it reduces effectively the search space and then reduce the time wasted in exploring trivial and unfeasible solutions.

VII. Conclusion

In this work, we proposed a new solution to find the optimal monitor placement to cover all slices with probing cycles. We formulated the task as a set cover problem, and we proposed a learning based approach. We divided the task into two sub-tasks: the prediction of the number of monitors, and their identification. We unveiled the potential of GNNs in solving the first sub-task. Then, we proposed a genetic algorithm to deal with the second one.

There are lot of directions to go for the future work:(1) Even we avoid the process of computing cycles in the prediction step, we are obliged to do it to reveal the identity of monitors, so it is useful to find a solution that avoids this; (2) Given that the optimal solution for large instances can not be determined, it would be good to propose methods to enhance the generalization on large graphs of the model that was trained on small instances.

References

[1] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck. Network slicing and softwarization: A survey on principles, enabling technologies, and solutions. *IEEE Communications Surveys Tutorials*, 20(3):2429–2453, 2018.

[2] F. Aubry, D. Lebrun, S. Vissicchio, M. T. Khong, Y. Deville, and O. Bonaventure. Scmon: Leveraging segment routing to improve network monitoring. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, 2016.

[3] A.-L. Barabási. Network science. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1987):20120375, 2013.

[4] J. Beasley. An algorithm for set covering problem. *European Journal of Operational Research*, 31(1):85–93, 1987.

[5] Cisco. Making networks sdn-ready with segment routing. https://www.segment-routing.net/images/lightreading$_r eport.pdf$, 2017.

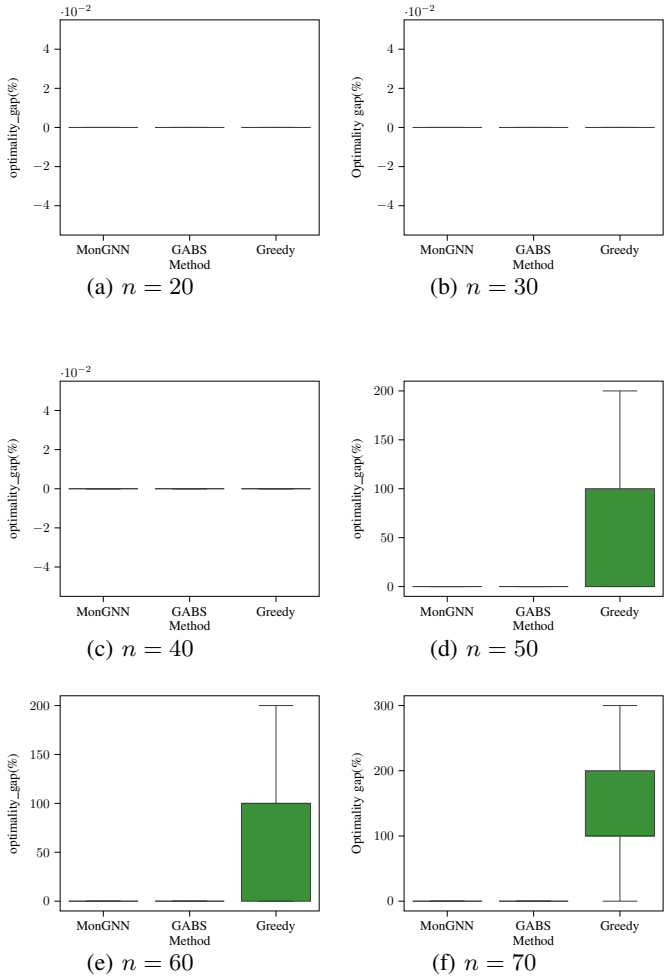
[6] P. Erdős and A. Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[7] A. Gopalan and S. Ramasubramanian. On identifying additive link metrics using linearly independent cycles and paths. *IEEE/ACM Transactions on Networking*, 20(3):906–916, 2012.

[8] R. Guedrez, O. Dugeon, S. Lahoud, and G. Texier. Label encoding algorithm for mpls segment routing. In *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, pages 113–117, 2016.

[9] T. He, A. Gkelias, L. Ma, K. K. Leung, A. Swami, and D. Towsley. Robust and efficient monitor placement for network tomography in dynamic networks. *IEEE/ACM Transactions on Networking*, 25(3):1732–1745, 2017.

[10] T. He, L. Ma, A. Swami, and D. Towsley. *Network Tomography*. Cambridge University Press, Cambridge, England, UK, May 2021.



(a) $n = 20$ (b) $n = 30$

(c) $n = 40$ (d) $n = 50$

(e) $n = 60$ (f) $n = 70$

Fig. 8: Optimality gap : BA graphs

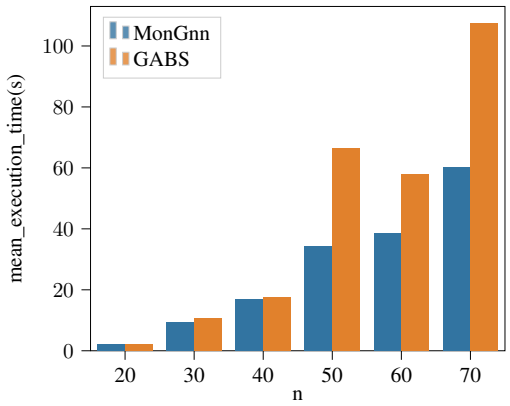


Fig. 9: ER graphs: Mean execution time(s)

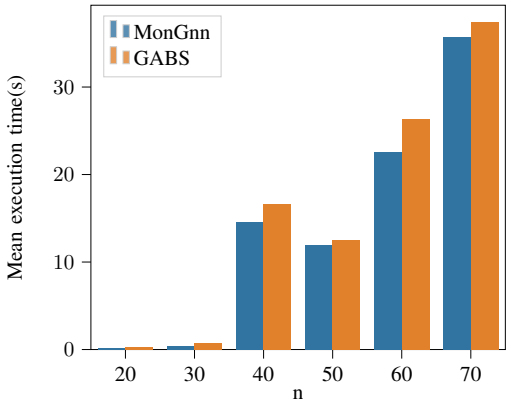


Fig. 10: BA graphs: Mean execution time (s)

[11] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[12] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks, 2017.

[13] X. Li and K. L.Yeung. Ilp formulation for monitoring-cycle construction using segment routing. In *2018 IEEE 43rd Conference on Local Computer Networks (LCN)*, pages 485–492, 2018.

[14] X. Li and K. L. Yeung. Designing network monitoring schemes based on segment routing. In *GLOBECOM 2017 - 2017 IEEE Global Communications Conference*, pages 1–6, 2017.

[15] L. Ma, T. He, K. K. Leung, A. Swami, and D. Towsley. Link identifiability in communication networks with two monitors. In *2013 IEEE Global Communications Conference (GLOBECOM)*, pages 1513–1518, 2013.

[16] M. Rahali, J.-M. Sanner, and G. Rubino. Feal: A source routing framework for efficient anomaly localization. In *ICC 2020 - 2020 IEEE International Conference on Communications (ICC)*, pages 1–7, 2020.

[17] H. Ramchoun, M. A. J. Idrissi, Y. Ghanou, and M. Ettaouil. Multilayer perceptron: Architecture optimization and training. *IJIMAI*, 4(1):26–30, 2016.

[18] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[19] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American statistical association*, 91(433):365–377, 1996.

[20] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[21] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and P. S. Yu. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, Jan 2021.