NeuralGAP: Deep Learning Evaluation of Networked Avionic Architectures

Rodrigo S. de Moraes
Institute of Computer and Information Science
Linköping University
rodrigo.moraes@liu.se

Simin Nadjm-Tehrani
Institute of Computer and Information Science
Linköping University
simin.nadjm-tehrani@liu.se

***Abstract*—Evaluation of system requirements in the early concept design stage of networked architectures is important since many future choices are restricted after this stage. Evaluating whether a candidate topology meets dependability and timeliness requirements has complex sub-problems, such as routing data exchanges in the network, finding shortest paths, and clustering network elements according to their attributes. In this paper, we build on an earlier work that tackles the generation of network topologies and propose a hybrid approach based on genetic algorithms and graph convolutional networks to address the topology evaluation problem. We apply the proposed evaluation method to a realistic industrial use case and show that it is up to 5 times faster than the previous method.**

***Index Terms*—Graph Neural Networks, Network Topology Generation, Network Topology Evaluation**

I. Introduction

Conceptualizing safety-critical networked systems, particularly those with extra requirements such as timeliness and secure data exchange (e.g., avionic and automotive systems), poses significant challenges. During the early concept stage, domain specialists must investigate the tradeoff between multiple, often conflicting, requirements, and evaluate the merits of different networked platform configurations.

In earlier work [1], we explored the challenges of searching for suitable network topologies and introduced NetGAP, an automated method to support early-stage exploration and concept design of networked systems. In NetGAP, graphs representing candidate network topologies are generated using Monte Carlo tree search (MCTS) and a domain-specific grammar that expresses possible hardware module interconnections.

Assessing the alignment of these candidate topologies with application requirements, however, involves solving a series of hard problems. These include: mapping software processes to hardware modules, routing messages through the links, and searching for paths to estimate communication resource adequacy. Handling these for each candidate solution significantly impacts evaluation efficiency and scalability

In this paper, we extend NetGAP and integrate a graph convolutional network (GCN) regressor into the topology evaluator algorithm with the following two contributions:

1) We propose a hybrid GCN/GA model (NeuralGAP) to calculate the reward of a given network topology against an application model and a set of application and topology requirements.
2) We show that the proposed evaluator accelerates and improves the efficacy of the search for network topologies when applied to a realistic use case: a synthetic application design provided by an industrial partner.

This paper showcases the new model's ability to identify candidate topologies that deviate from the application requirements, achieving faster results than the previous genetic algorithm implementation. In experiments, we show that NeuralGAP consistently improves the quality of synthesized candidate topologies, accelerating the search and broadening the explored space.

II. Related works

Evaluating candidate topologies involves addressing various problems which are challenging to solve with traditional techniques (e.g. network calculus [2], mixed-integer linear programs [3], and graph analysis [4]). Here, we explore works using graph neural networks, particularly graph convolutional networks [5], which successfully handle network-related problems [6], [7], [8], [9] and could offer scalability benefits.

Analyzing a topology involves evaluating communication aspects such as network link bandwidth, message scheduling, and data latency. Typical techniques for that are network calculus for latency estimation [10], and mixed-integer linear programming or dynamic programming for schedulability [3], [11]. However, recent works advocate for machine learning to address similar challenges, e.g. to assess Ethernet network feasibility within time-sensitive networks [12], to predict end-to-end delay in Software Defined Networking [7], to estimate routing congestion in large-scale integrated circuits [13]. Another aspect concerns the reliability and resilience of network links, nodes, and the network as a whole. Traditionally, exact methods and Monte Carlo simulations have been used for this purpose [4], [14], [15]. In the context of machine learning, Davila-Frias et al. [16] use deep neural networks to estimate the reliability degradation function of network components over time. Our work is intended to combine many (potentially conflicting) requirements in the same attempt.

This work was supported by Sweden's Innovation Agency – Vinnova, as part of the national projects on aeronautics, NFFP7, project CLASSICS (NFFP7-04890). The authors wish to thank the industrial partners from Saab AB for their valuable technical input and for the use case used in the paper.

III. Problem statement

This paper focuses on the evaluation of network topologies to determine their alignment with the needs of a specific application. In this section, the relevant terms are defined, and the problem is formalized.

A. Terminology

Here, we present terminology from earlier works [1] that we reuse in the remainder of the paper.

- **Application**: defines functionality and is represented by software **processes** who exchange messages and cooperate to provide the expected behavior.
- **Platform:** the set of conceivable hardware and system software components where processes can be deployed. **Hardware modules** provide **resources** (i.e. communication bandwidth, memory, and computation capacity) for application processes.
- **Platform Configuration** or **Topology**: denotes the arrangement of a selected set of hardware modules based on specific interconnection patterns.

B. The topology evaluation problem

Let $T = (V, E, l_E, l_V)$ be the graph representing a candidate topology, where V represents a set of hardware modules, E is a set of directed edges representing the links between the hardware modules, and l_E and l_V are labeling functions that assign labels to edges and vertices, respectively. Let $V_p \subseteq V$ be the subset of vertices of T labeled as processing modules. Let $P = \{p_1, \ldots, p_m\}$ be a finite set of software processes and $G = \{g_1, \ldots, g_n\}$ is a partition over the set P, i.e. each process p in P is contained and $\sum_i^n |g_i| = m$. Finally, let $w_i : V_p \longrightarrow G$ be a bijective mapping from labeled (processing) vertices in V_p to process groups in G and $W = \{w_1, \ldots, w_h\}$ be the set of all such possible mappings. Note that, in practice, each w_i represents a possible allocation of software processes to processing modules in V_p.

Now, let $f : (T, \mathbb{A}) \times W \longrightarrow [0, 1]$ be a function that evaluates the extent to which a topology T with the mapping w_i is able to host the envisaged application given the requirements expressed by $\mathbb{A}$.

The reward of a topology T based on a grouping G (representing the mapping of processes onto modules) evaluated against the requirements $\mathbb{A}$, is expressed by the function $r(T, G, \mathbb{A})$. Then our goal is to find the maximum reward expressed as follows:

$$r(T, G, \mathbb{A}) = f(T, w_{max}, \mathbb{A}),$$
$$w_{max} = \underset{w_i \in W}{argmax} \quad f(T, w_i, \mathbb{A})$$

where w_{max} is the maximum reward, calculated across all mappings in W.

The last equation shows a significant challenge in topology evaluation: finding w_{max} in W is computationally intensive as there could be up to $n!$ potential mappings, with $n!$ denoting the factorial of the number of process groups.

Another topology evaluation challenge involves the existence of requirements in $\mathbb{A}$ that demand complex analysis of the topology T in f. Some of these are specific to the use case and are hard to predict ahead of time. They include tasks like pathfinding and clique identification. While efficient algorithms exist for some tasks, others are challenging to approximate when assessing thousands of candidate solutions.

IV. Accelerating topology evaluation

In NetGAP, the authors employ a genetic algorithm (GA) to address the topology evaluation problem. While this approach effectively handles the problem's combinatorial nature, it requires solving complex graph sub-problems for each solution considered during its execution. In this paper, we propose a hybrid GA/GCN topology evaluator to address this complexity.

The hybrid evaluator works in two steps: filtering and refinement. During the filtering step, candidate topologies are swiftly analyzed by the neural network. If the neural network's reward exceeds a defined threshold, th_{ref}, the topology is passed to the genetic algorithm for further refinement. If the score falls below the threshold, we exit the evaluation module. This mechanism enables rapid sorting and pinpointing of promising candidates for in-depth analysis while ignoring those that do not offer viable solutions. The choice of threshold th_{ref} is determined empirically by the user and should be chosen considering the reward function and the accuracy of the GCN module. Figure 1 shows the proposed hybrid evaluator.

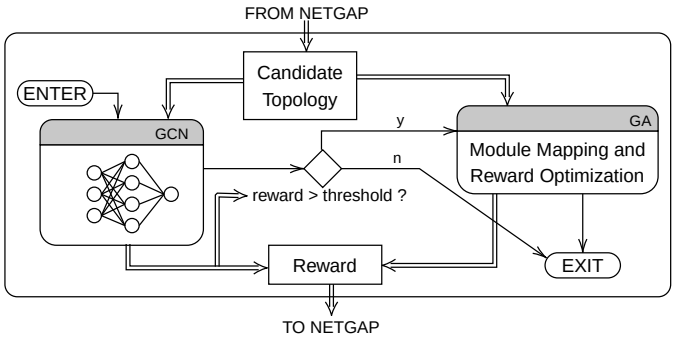

Fig. 1: The hybrid evaluation module of NeuralGAP. Double arrows represent the flow of data, single arrows represent the control flow. Grey areas highlight the different algorithms used for refinement and filtering.

The proposed network is composed of 3 stages. The first stage is based on graph convolution layers. Each layer convolves the nodes in a graph with their neighbors, extracting features from both nodes and edges based on their attributes and those of their neighbors. Each convolution layer is followed by a normalization layer. Our model cascades 4 of these convolution and normalization ensembles.

Next in sequence is the pooling layer. To avoid overfitting, this layer averages the node features learned by the convolution layers and creates a single graph-level embedding from a batch of training inputs. The final stage of the network is the regression stage, which is composed of 3 standard, fully

connected, linear layers. Their objective is to associate a reward in the interval $[0, 1]$ to the graph embeddings received from the pooling stage.

V. Experiments and Results

We assess NeuralGAP by using it in an industrial but synthetic scenario. The use case consists of an application model constructed to mimic the properties of airborne systems in a concept design stage at our industrial partner. It has 91 periodic processes that exchange 629 periodic messages. Of these 629, 50 represent high-priority (HP) flows.

A. Requirement definition

For evaluation purposes, we consider the following requirements:

- **Communication latency:** The application is time-sensitive, therefore the latency of inter-process messages must be minimized.
- **Resiliency:** HP messages should have redundant paths, allowing for resiliency and reconfiguration as a means of fault tolerance if necessary.
- **Resource utilization:** Load on computing modules and network links should not exceed 80%.
- **Computation Adequacy:** A candidate topology should have as many computation modules as needed to host the processes while respecting the resource utilization requirements.

B. Reward Function

The score of a topology can be calculated as the weighted average of the following three terms:

- **Latency score**: characterizes how much latency the messages being exchanged by processes mapped to the modules within a given topology are subjected to. We approximate it as follows:

$$l_s = \frac{2e^{1-x_l/load_{th}-o}}{h}$$

  Where x_l is the maximum communication load observed in any single link; o is the number of overloaded links; and h is the mean number of hops in the path of a message. The negative exponential formulation of l_s guarantees convergence towards networks with lower loads and fewer hops.
- **Cost:** We assume hardware modules cost 10 monetary units (10u) and that links cost 0.1 monetary units (0.1u).
- **Resiliency Score:** Reflects the adherence to the resilience requirements, this score is calculated as the ratio between the number of HP flows with redundant paths in the candidate topology and the total number of HP flows.

Due to the characteristics of this reward function, topology rewards are not expected to exceed the value of 0.85. Therefore, topologies with scores above 0.80 are considered "good".

C. Experiments and Results

With NetGAP we generated a dataset of 58,314 topology graphs annotated according to the requirements above and used it to train the network. Training took about 17min on an Nvidia T4 accelerator.

We compare the NetGAP-based evaluator and the new NeuralGAP hybrid evaluator in two sets of experiments. The first set terminates the search based on a given elapsed search time. This choice lets us explore how the evaluators affect the solutions given the same execution time. The second set of experiments terminates the search based on the number of iterations performed. By choosing this termination parameter, we can compare which evaluator provides better results given the same amount of work (same number of solutions explored). Each experiment was sampled 90 times.

Figures 2 and 3 show the reward scores and the size of the explored space for the time-limited experiments. Each figure shows a boxplot over 90 runs of each experiment for each timeout. The orange lines indicate the median and the boxes show the interquartile range. The whiskers extend to 1.5 times the interquartile range and the crosses indicate the outliers. The green and blue areas show the shape of the data distribution.

From Figure 2, we can see that the hybrid algorithm beats the genetic algorithm by a good margin from the start, at a time limit of 15s, producing better topologies consistently across runs. The hybrid algorithm maintains its edge to the end, at a limit of 300s, when it still produces more consistent and better median results than the genetic algorithm.

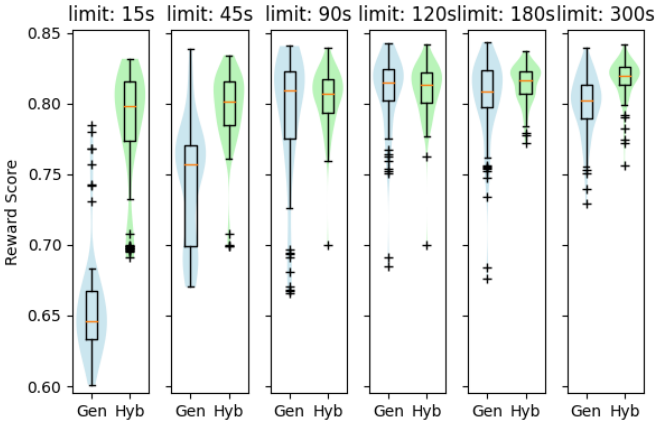

Fig. 2: Distribution of the solution rewards in the time-limited experiments.

Figure 3 provides insight into the underlying reasons for the more convergent behavior of the hybrid evaluator. Given the same time limitation, the hybrid evaluator can consistently explore more of the solution space than the genetic algorithm, helping the main search loop prune out worse regions and focus on looking for solutions in the more promising regions.

The iteration-limited experiment (Figure 4) shows that given a low number of iterations, the original NetGAP approach produces high-reward results more consistently. As the number

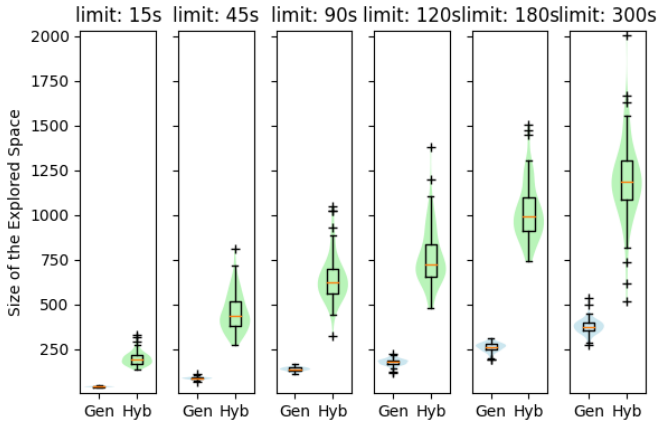

Fig. 3: Distribution of the size of the explored space (number of explored topologies) in the time-limited experiments.

of allowed iterations increases, the hybrid evaluator produces more consistent and better median results, beating the genetic evaluator at 1600 iterations. We hypothesize that this behavior arises because the hybrid evaluator tends to slightly underestimate rewards within the range $[0.55, 0.85]$. We conjecture that, at a low iteration count, underestimating a topology's reward harms the search by prematurely steering it away from regions of interest. Yet, with more iterations, this underestimation appears advantageous as the increased sample size averages out any detrimental effects.

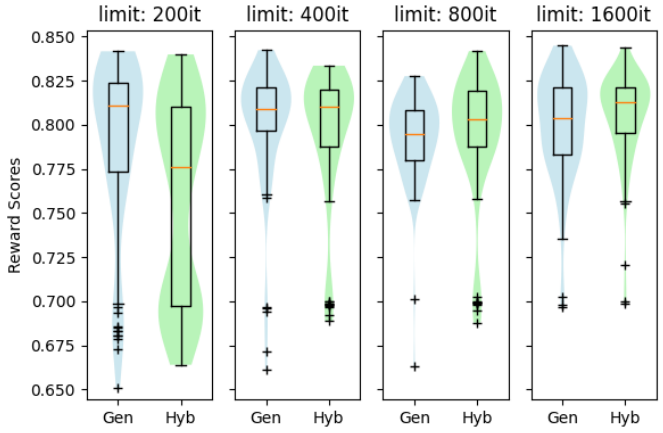

Fig. 4: Distribution of the solution rewards in the iteration-limited experiments.

To study the sensitivity of the approach to the refinement threshold we ran dedicated experiments with different thresholds and found that while the quality of the solutions did not change significantly, higher thresholds increase the size of the explored space (charts omitted due to space limitations).

VI. Conclusion

In this paper, we proposed NeuralGAP to evaluate network topologies in the early concept stage. We have shown that the proposed approach is not only faster than an earlier method, but it also accelerates convergence to a similar high-reward topology. The insights provided during the exploration attest to the capabilities of neural networks to be used in the context of state space exploration for industrial applications.

The main downside of the proposed approach is the overhead of training the network (ca 17 min). Therefore, the evaluator proposed in this paper is more suitable for workflows that require the generation of a large number of topologies with similar features or similar requirements.

Future work may involve adjusting NeuralGAP for online training with new topologies during the concept exploration process and comparing the grammar-based topology generation approach to other methods.

References

[1] R. S. de Moraes and S. Nadjm-Therani, "Netgap: A graph-grammar approach for concept design of networked platforms with extra-functional requirements," *under review*, 2022.

[2] J.-Y. Le Boudec and P. Thiran, eds., *Network Calculus*, pp. 3–81. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[3] R. Zhao, G. Qin, Y. Lyu, and J. Yan, "Security-aware scheduling for TTethernet-based real-time automotive systems," *IEEE Access*, vol. 7, pp. 85971–85984, 2019.

[4] Z. Zhang, W. An, and F. Shao, "Cascading failures on reliability in cyber-physical system," *IEEE Transactions on Reliability*, vol. 65, no. 4, pp. 1745–1754, 2016.

[5] M. Welling and T. N. Kipf, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2016.

[6] Z. Xiangyun, W. Lijun, L. Zhiyuan, and J. Yulin, "Deep reinforcement learning with graph convolutional networks for load balancing in SDN-based data center networks," in *International Computer Conference on Wavelet Active Media Technology and Information Processing ((ICCWAMTIP)*, pp. 344–352, IEEE, 2021.

[7] Z. Ge, J. Hou, and A. Nayak, "Forecasting SDN end-to-end latency using graph neural network," in *International Conference on Information Networking (ICOIN)*, pp. 293–298, IEEE, 2023.

[8] Y. Peng, C. Liu, S. Liu, Y. Liu, and Y. Wu, "Smarttro: Optimizing topology robustness for internet of things via deep reinforcement learning with graph convolutional networks," *Computer Networks*, vol. 218, 2022.

[9] J. Coleman, M. Kiamari, L. Clark, D. D'Souza, and B. Krishnamachari, "Graph convolutional network-based scheduler for distributing computation in the internet of robotic things," in *IEEE Military Communications Conference (MILCOM)*, pp. 1070–1075, IEEE, 2022.

[10] S. Bondorf, P. Nikolaus, and J. B. Schmitt, "Quality and cost of deterministic network calculus: Design and evaluation of an accurate and fast analysis," *Measurement and Analysis of Computing Systems*, vol. 1, jun 2017.

[11] Y. Zhang, F. He, and H. Xiong, "Scheduling rate-constrained flows with dynamic programming priority in time-triggered ethernet," *Chinese Journal of Electronics*, vol. 26, no. 4, pp. 849–855, 2017.

[12] T. L. Mai and N. Navet, "Deep learning to predict the feasibility of priority-based Ethernet network configurations," *ACM Transactions on Cyber-Physical Systems*, vol. 5, sep 2021.

[13] K. Rusek, J. Suárez-Varela, P. Almasan, P. Barlet-Ros, and A. Cabellos-Aparicio, "Routenet: Leveraging graph neural networks for network modeling and optimization in SDN," *IEEE Journal on Selected Areas in Communications*, vol. 38, no. 10, pp. 2260–2270, 2020.

[14] J. E. Ramirez-Marquez and D. W. Coit, "A Monte-Carlo simulation approach for approximating multi-state two-terminal reliability," *Reliability Engineering and System Safety*, vol. 87, no. 2, pp. 253–264, 2005.

[15] J. Ayoub, W. Saafin, and B. Kahhaleh, "k-terminal reliability of communication networks," in *IEEE International Conference on Electronics, Circuits, and Systems (ICECS)*, vol. 1, pp. 374–377 vol.1, 2000.

[16] A. Davila-Frias, N. Yodo, T. Le, and O. P. Yadav, "A deep neural network and Bayesian method based framework for all-terminal network reliability estimation considering degradation," *Reliability Engineering and System Safety*, vol. 229, 2023.