

# Optimizing Task Allocation in the Cloud-Edge Continuum: A Deep Reinforcement Learning and Graph Neural Network Approach

Thanasis Kotsiopoulos<sup>\*†</sup>, Alexandros Nizamis<sup>\*</sup>, Jaime Flor<sup>‡</sup>, Matilde Julián<sup>‡</sup>, Carlos E. Palau<sup>‡</sup>, Konstantinos Votis<sup>\*</sup>,  
Dimitrios Tzovaras<sup>\*</sup> and Panagiotis Sarigiannidis<sup>†</sup>

<sup>\*</sup>Information Technologies Institute, Centre for Research & Technology Hellas  
57001 Thermi, Greece Email: kotsiopoulos@iti.gr

<sup>†</sup>Dept. of Electrical and Computer Engineering, University of Western Macedonia, Campus ZEP Kozani,  
50100 Kozani, Greece

<sup>‡</sup> Communications Department, Universitat Politècnica de València, 46022, Valencia, Spain

**Abstract**—Task placement optimization in cloud-edge-fog environments is a challenging problem that requires balancing multiple objectives, such as minimizing latency and energy consumption, while adhering to resource constraints. This paper proposes a framework that integrates Deep Reinforcement Learning with Graph Neural Networks to address these challenges. Specifically, we explore the effectiveness of such architectures, including Graph Convolutional Networks and Message Parsing Neural Networks, within a DRL agent for task allocation. The framework is evaluated on synthetic task flow graphs, representing parallel workflows of varying complexities (10 and 100 tasks), and benchmarked against traditional methods such as Genetic Algorithms and a Random Agent. Our results demonstrate that the RL Agent with GCN layers outperforms the MPNN-based RL Agent, GA, and Random Agent in small-scale scenarios while it performs equivalently with the MPNN-based RL Agent in large-scale scenarios in which both surpass the heuristic approach and the Random agent.

**Index Terms**—Deep Reinforcement Learning, Graph Neural Networks, Optimization, Heuristics, Cloud-Edge Continuum

## I. INTRODUCTION

Cognitive Computing, built on AI techniques for data-driven decision-making, has gained prominence in recent years [1]. Its foundation lies in IoT, Big Data, and Cloud Computing [2], leading to the concept of the Cognitive Cloud, a cloud-based system that perceives, learns, and adapts dynamically [3]. With the proliferation of IoT devices, this concept evolved into the Cognitive Computing Continuum (CCC), which integrates IoT, Edge, and Cloud to eliminate silos and enable seamless workflows [4]. By combining cloud power with edge responsiveness, the CCC supports workload distribution, real-time detection, scalability, and resource efficiency [5]. European technology roadmaps further highlight the CCC's role in next-generation hyper-connected services, including DestinE, the Metaverse/Web3, holographic telepresence, and autonomous mobility, reshaping digital interactions across entertainment,

cultural heritage, and green mobility [6]. Despite its benefits, CCC faces challenges such as dynamic environments, synchronization, heterogeneity, massive data handling, and strict application constraints [5], [7]. Advances in meshed networking partly address these by enabling dynamic scaling and infrastructure-free communication, yet robust and scalable solutions for automated deployment and runtime management across cloud-edge layers remain immature, especially under stringent latency, bandwidth, and resource constraints [8], [9].

Having all these in mind and building on the work of Almasan et al. [10], who integrate Graph Neural Networks (GNNs) with Deep Reinforcement Learning (DRL) agents to mitigate generalization errors in unseen topologies for optimal network resource allocation, we propose an extended and comparative approach. Our work adapts this paradigm to the dynamic and heterogeneous environment of the cloud-fog-edge continuum, specifically focusing on task placement optimization. We aim to evaluate the performance and generalization capabilities of DRL agents that utilize different GNN architectures, such as Graph Convolutional Networks (GCNs) and Message Passing Neural Networks (MPNNs) as presented in [10]. These approaches are benchmarked against a heuristic algorithm (Multi-Objective MOAED), originally introduced in our previous work [11] and extended in this study, as well as a baseline Random Agent in this complex multi-layer infrastructure. The main contributions of this work are:

- Adaptation to Cloud-Fog-Edge Continuum: Extending the DRL-GNN framework to address task placement in a highly dynamic and heterogeneous environment, considering latency and energy efficiency.
- Comparative Evaluation of GNN Architectures: Assessing the effectiveness of GCN-based and MPNN-based DRL agents in improving generalization and decision-making across unseen topologies.
- Benchmarking Against Alternative Methods: Providing

a comprehensive comparison of DRL-GNN approaches with a heuristic method (Multi-Objective MOAED) and a Random agent to highlight the strengths and limitations of each strategy.

- Insights into Generalization and Scalability: Investigating how different GNN architectures influence the agent's ability to generalize across various cloud-fog-edge configurations and handle increasing system complexity.

## II. RELATED WORK

Recent research explores diverse strategies for orchestration intelligence, including search-based algorithms, mathematical programming, and game-theoretic or deep learning frameworks. Examples include AI4DL, which manages deep learning workloads [12], and Theta-Scan, which optimizes container scaling via behavioral analysis. Reinforcement learning (RL) has emerged as a particularly promising approach for task orchestration across cloud-edge-fog systems [13]. Resource allocation is inherently multi-objective, balancing latency, energy, deadlines, cost, and execution volume [14]. Heuristic methods remain widely used; e.g., Pasiadis et al. [11] combined genetic algorithms with MILP for SDN resource allocation, while [15] minimized transmission times but overlooked deadlines and latency constraints. Recent advances highlight a shift from heuristics toward deep reinforcement learning (DRL), valued for handling high-dimensional data, adapting to dynamic environments, and allocating resources in real time. However, current DRL implementations often neglect application-specific constraints and security concerns [16]. Graph Neural Networks (GNNs) further enhance resource allocation by modeling heterogeneous computing nodes and their interconnections [10], [17], [18]. They dynamically adapt to demand fluctuations, mitigate bottlenecks, and support predictive resource management by forecasting workload needs [19]. Importantly, GNNs enable decentralized, lightweight decision-making at edge nodes, reducing latency, bandwidth usage, and privacy risks [20]. Their message-passing architecture allows nodes to iteratively exchange and aggregate information, capturing complex dependencies across the continuum.

## III. METHODOLOGY

In this section, we present our approach to optimizing resource allocation and task offloading within the cloud-edge-fog continuum of ENACT project [21]. Our methodology integrates DRL with a GNN to address the dynamic and heterogeneous characteristics of such environments, so to deliver the first version of ENACT CCC's task scheduler.

### A. Problem Formulation

The task placement problem is modeled as a Markov Decision Process (MDP), defined by the tuple  $\langle S, A, P, R, \gamma \rangle$ :

- State ( $S$ ): Each state  $s \in S$  captures the current system configuration, latency, energy consumption, and task requirements.

- Action ( $A$ ): Actions  $a \in A$  assign tasks to specific nodes in the continuum.
- Reward ( $R$ ): The reward function evaluates task placement decisions, considering latency, energy consumption, and penalties
- Transition ( $P$ ): Defines the likelihood of moving to state  $s'$  after  $a$  in  $s$ .
- Discount Factor ( $\gamma$ ): A factor  $\gamma \in [0, 1]$  discounts future rewards, balancing immediate and long-term optimization objectives. In our case  $\gamma$  is equal to 0.95

Analytically, the reward function is computed as:

$$R(s, a) = -\text{latency}_{\text{norm}}(t, d) - \text{power}_{\text{norm}}(t, d) - P(t, d), \quad (1)$$

where:

$$\text{latency}_{\text{norm}}(t, d) = \frac{\text{latency}(t, d)}{\max\_latency}, \quad (2)$$

$$\text{power}_{\text{norm}}(t, d) = \frac{\text{power}(t, d)}{\max\_power}. \quad (3)$$

The penalty term  $P(t, d)$  is defined as:

$$P(t, d) = \lambda \cdot \max(0, \text{latency}_{\text{norm}}(t, d) - \text{device\_latency}_{\text{norm}}(d)) + \mu \cdot \max(0, \text{power}_{\text{norm}}(t, d) - \text{device\_power}_{\text{norm}}(d)), \quad (4)$$

where:

$$\text{device\_latency}_{\text{norm}}(d) = \frac{\text{current\_latency}(d)}{\max\_latency}, \quad (5)$$

$$\text{device\_power}_{\text{norm}}(d) = \frac{\text{current\_power}(d)}{\max\_power}. \quad (6)$$

Here,  $\lambda, \mu > 0$  are penalty coefficients, ensuring tasks are discouraged from overloading resources. In our case both values are set to 0.5

### B. Graph Representation

To effectively model the computing continuum, we represent it as a graph  $G = (V, E)$ :

- Nodes ( $V$ ): Each node  $v \in V$  represents either a computing resource (e.g., cloud server, fog node, or edge device) or a task, characterized by attributes such as processing capacity ( $C_v$ ), energy cost ( $E_v$ ), and current workload ( $W_v$ ).
- Edges ( $E$ ): Each edge  $e \in E$  represents a communication link between nodes.

The graph structure dynamically changes based on resource availability and workload demands, allowing adaptive task placement decisions. An indicative figure of the tasks and topologies is presented in Figure 1.

### C. Graph Neural Network Model

The GNN leverages a multi-layer architecture to extract high-level representations from the graph. The architecture consists of three GCN layers followed by a fully connected output layer:

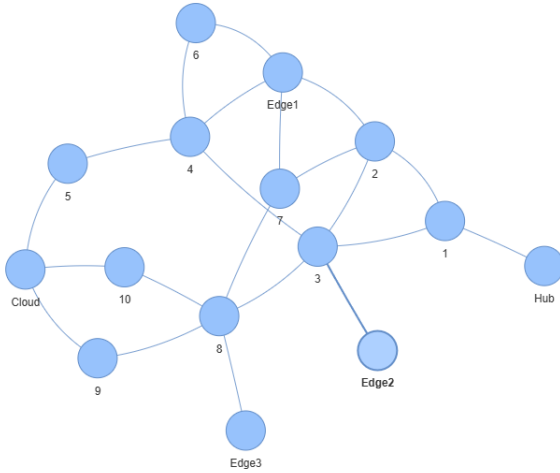


Fig. 1: Methodology

- **Graph Convolutional Layers:** The GCN layers iteratively update node embeddings by aggregating features from neighboring nodes. For the  $k$ -th layer:

$$h_v^{(k)} = \sigma \left( \sum_{u \in \mathcal{N}(v)} \frac{1}{\sqrt{\deg(v) \cdot \deg(u)}} W^{(k)} h_u^{(k-1)} \right),$$

where:

- $h_v^{(k)}$  is the embedding of node  $v$  at layer  $k$ ,
- $\mathcal{N}(v)$  denotes the set of neighbors of node  $v$ ,
- $W^{(k)}$  is the weight matrix for layer  $k$ ,
- $\sigma$  is the ReLU activation function,
- $\deg(v)$  is the degree of node  $v$ .
- **Fully Connected Output Layer:** The output embeddings from the GCN layers are processed through a dense layer to predict Q-values for actions. This layer maps the graph-structured data to task placement decisions.

A leaky ReLU activation is applied after every GCN layer. Also, at every step, excluding the last GCN layer, dropout is applied with a probability equal to 0.3.

#### D. Deep Reinforcement Learning Framework

The DRL agent employs the aforementioned GNN as Deep Q-Network (DQN) which utilizes the embeddings generated by the GNN to estimate the optimal action-value function:

$$Q^*(s, a) = \max_{\pi} \mathbb{E} \left[ \sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \middle| s_0 = s, a_0 = a, \pi \right].$$

The training process minimizes the temporal difference error:

$$\mathcal{L}(\theta) = \mathbb{E}_{(s, a, r, s') \sim \mathcal{D}} \left[ \left( r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta) \right)^2 \right],$$

where:

- $\theta$  are the parameters of the current network,
- $\theta^-$  are the parameters of the target network,
- $\mathcal{D}$  is the replay buffer storing past experiences.

By combining GNNs with DRL, our methodology effectively captures the structural dependencies in the computing continuum, enabling intelligent and adaptive resource management strategies. The optimizer of our framework is the ADAM optimizer and we used the Pytorch Geometric package [22].

#### E. Multi-Objective Genetic Algorithm

This section outlines our Multi-Objective Genetic Algorithm (GA), a heuristic strategy for the task-placement problem in the cloud–edge–fog continuum. Building upon the formulation in [11], we introduce a revised fitness function and integrate crossover and mutation operators previously omitted. Emulating natural selection, the GA iteratively refines a population of candidate assignments to approach optimal or near-optimal allocations.

Each candidate solution (individual) is encoded as a chromosome:

$$\mathbf{x} = [x_1, x_2, \dots, x_n], \quad (7)$$

where  $x_i$  specifies the device (edge, fog, or cloud) hosting task  $i$ . Thus, every chromosome represents a complete mapping of tasks to resources.

The evaluation of a chromosome  $\mathbf{x}$  considers two objectives—latency and power consumption—by computing their normalized, negated sums:

$$\text{normalized\_latency} = - \sum_{i=1}^n \frac{\text{latency}(t_i, x_i)}{\text{max\_latency}}, \quad (8)$$

$$\text{normalized\_power} = - \sum_{i=1}^n \frac{\text{power}(t_i, x_i)}{\text{max\_power}}, \quad (9)$$

with  $\text{latency}(t_i, x_i)$  and  $\text{power}(t_i, x_i)$  denoting the delay and energy use of task  $t_i$  on device  $x_i$ , and  $\text{max\_latency}$ ,  $\text{max\_power}$  being the respective maxima over all devices.

We also enforce feasibility via a resource-constraint check:

$$\text{is\_valid}(\mathbf{x}) = \begin{cases} 1, & \text{if all tasks fit within device capacities,} \\ 0, & \text{otherwise.} \end{cases} \quad (10)$$

Invalid assignments incur a prohibitive penalty:

$$\text{fitness}(\mathbf{x}) = [\infty, \infty], \quad (11)$$

whereas valid solutions receive their objective values:

$$\text{fitness}(\mathbf{x}) = [\text{normalized\_latency}, \text{normalized\_power}]. \quad (12)$$

Subsequently, non-dominated solutions are extracted to construct the Pareto front. We say  $\mathbf{x}_i$  dominates  $\mathbf{x}_j$  if:

$$f_1(\mathbf{x}_i) \leq f_1(\mathbf{x}_j) \quad \text{and} \quad f_2(\mathbf{x}_i) \leq f_2(\mathbf{x}_j), \quad (13)$$

with at least one inequality strict. The resulting Pareto set thus captures the trade-off frontier between latency and power consumption.

Two genetic operators drive the evolution of the population:

- **Crossover:** Parents exchange genetic material to create an offspring. We perform single-point crossover by selecting a random index  $c$  and forming

$$\mathbf{x}_{\text{child}} = [x_1, \dots, x_c]_{\text{parent}_1} \parallel [x_{c+1}, \dots, x_n]_{\text{parent}_2}.$$

Here, the child inherits the prefix from one parent and the suffix from the other, fostering solution diversity.

- **Mutation:** To introduce variation, we randomly pick a gene  $x_i$  and reassign it:

$$x_i \rightarrow x'_i,$$

where  $x'_i$  is selected uniformly from the set of devices. This randomness helps the search escape local optima.

Each generation consists of selection, crossover, mutation, and fitness evaluation. We then combine the current and newly created individuals using elitism:

$$P \leftarrow \text{elitism}(P_{\text{current}}, P_{\text{offspring}}),$$

ensuring top performers carry over. The loop repeats until either the Pareto front ceases to improve or a predefined maximum generation count  $G_{\text{max}}$  is reached. The end result is the set of Pareto-optimal task assignments balancing latency and energy use.

#### F. Random Agent

The Random Agent serves as a simple benchmark for task placement. Rather than optimizing based on system or task characteristics, it assigns each task to a device by chance, providing a baseline against which to compare more advanced algorithms.

For every assignment, the agent picks a device  $d$  from the pool  $D$  uniformly at random:

$$d = \text{random.choice}(D),$$

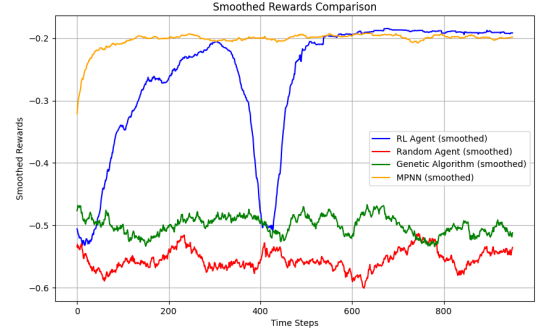
with  $D = \{d_1, d_2, \dots, d_m\}$  denoting the  $m$  available devices. Because `random.choice` draws each element with equal likelihood, the probability of selecting any particular device  $d_i$  is

$$P(d = d_i) = \frac{1}{m}, \quad \forall d_i \in D.$$

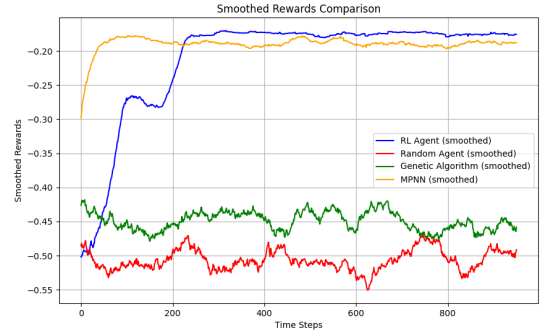
#### IV. EVALUATION

We evaluate our methods on the dataset from [23], which includes 18 synthetic task flow graphs for benchmarking allocation algorithms in edge-hub-cloud systems. The graphs follow three topologies (parallel, serial, hybrid) and three scales (10, 100, 1,000 nodes). In this study, we focus on parallel graphs with 10 and 100 tasks. Performance is measured using a reward function (see Subsection III-A) that balances latency and energy; maximizing it reduces both delay and power consumption, while capacity violations incur heavy penalties. We compare our DRL agent with GNNs against the MPNN baseline [10], a Genetic Algorithm (GA), and a Random Agent, reporting reward trajectories over 1,000 training episodes. Our initial experiments use the 10-task dataset, where the only varied hyperparameter is the GCN hidden dimension. Figure 2 shows the resulting reward evolution.

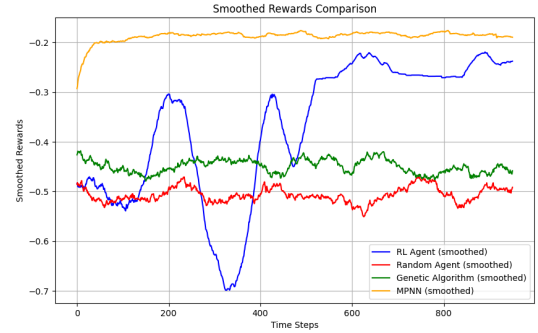
Increasing the GCN hidden dimension from 32 to 64 yields a marked uplift in the RL agent's reward, implying that the GCN layers harness the additional representational capacity to capture richer structural features. However, pushing the



(a) hidden\_dim = 32



(b) hidden\_dim = 64



(c) hidden\_dim = 128

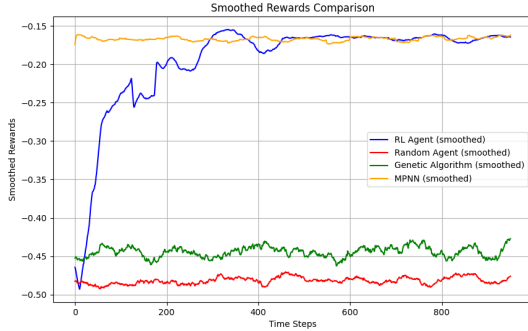
Fig. 2: Smoothed rewards for GCNs with hidden sizes 32, 64, and 128 on the 10-task dataset. RL-based agents outperform GA and Random, with hidden size 64 providing the best tradeoff.

hidden size to 128 produces only slight gains, accompanied by slower convergence and a flatter reward trajectory. Across all configurations, the RL agent (whether equipped with our GCN enhancement or with the MPNN setup from [10]) maintains a clear advantage over both the Random Agent and the Genetic Algorithm, underscoring its superior task-placement optimization.

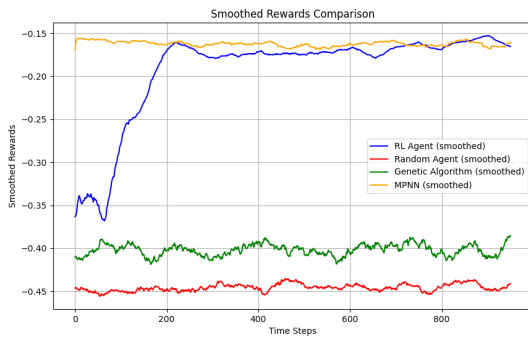
A hidden dimension of 64 emerges as the optimal trade-off between expressive power and training efficiency for our RL agent; larger hidden sizes offer minimal reward improvements

while risking overfitting and extended training times.

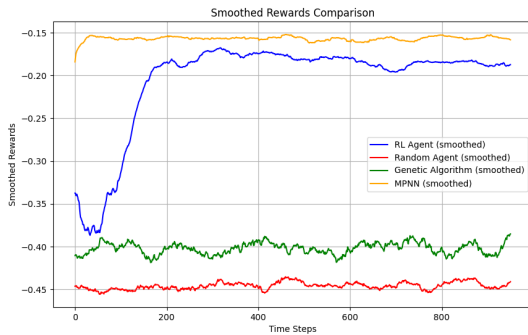
Figure 3 displays the reward curves for each model and algorithm on the 100-task parallel graphs, where once again only the GCN hidden size was varied and the MPNN hyperparameters were kept as in [10].



(a) GCN hidden\_dim = 32



(b) GCN hidden\_dim = 64



(c) GCN hidden\_dim = 128

Fig. 3: Comparison of Smoothed Rewards for Different GCN Hidden Sizes (32, 64, 128) Across Task Placement Algorithms for optimal placement of 100 parallel tasks. The RL Agent and the MPNN agent (RL based) again performs better than the Genetic Algorithm and Random Agent, with hidden size 64 for the GCN layers of the RL agent achieving the best balance between reward maximization and stability.

The findings show that the RL agent, whether enhanced with GCN layers or using the MPNN architecture, achieves notably superior task-to-device matching compared to both the GA and

the Random Agent. The GA delivers only moderate results and falls well behind the RL approaches, while the Random Agent’s performance remains unchanged as the number of tasks grows, reflecting its lack of any adaptive optimization mechanism.

It is identified that scaling the task count from 10 to 100 accelerates convergence for the RL agent across all tested hidden dimensions (32, 64, and 128). A hidden dimension of 64 still offers the optimal trade-off between rapid learning and stability, while increasing to 128 yields only marginal additional gains.

Overall, the GCN-augmented RL agent gains substantial advantages from larger task graphs, converging more quickly and securing higher rewards on the 100-task dataset.

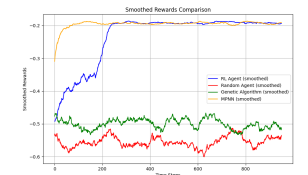
A hidden size of 64 offers the best trade-off between training efficiency and reward maximization for the RL Agent. Larger hidden sizes (e.g., 128) yield only marginal improvements while introducing slower convergence.

The DRL approach incorporating the MPNN with its original hyperparameters performs reliably across all scenarios, whereas the GA and Random Agent fall further behind as task volume grows. Additionally, in smaller-scale settings, the GCN-based DRL demonstrates greater efficiency than its MPNN-based counterpart when the hidden\_dim is 64.

To further examine the effect of GNN design choices, we ran two additional experiments on the 10-task dataset. First, we reduced the GCN hidden dimension to 4 while keeping other settings constant. Despite its reduced capacity, the GCN achieved reward trajectories similar to the MPNN agent, suggesting that in compact graphs, low-dimensional embeddings can capture sufficient structure and serve as lightweight yet effective alternatives. Second, we tested GAT layers with 2 attention heads and hidden size 16. While initially slower to improve, the GAT agent eventually matched the MPNN in reward quality, indicating that attention combined with dropout may support more stable generalization in low-complexity settings.



(a) GCN vs. MPNN (hidden dim 4).



(b) GAT vs. MPNN (2 heads, hidden dim 16).

Fig. 4: Smoothed rewards of GNN variants compared to MPNN on the 10-task dataset.

## V. CONCLUSION

This study evaluated task placement optimization within the cloud-edge-fog continuum using learning-based and heuristic approaches. The RL Agent with GCN layers along with the RL

agent with the MPNN network, consistently achieved the highest reward across all experiments, demonstrating their ability to optimize task placement by minimizing latency and energy consumption. A hidden size of 64 for the GCN layers proved to be the optimal configuration, balancing convergence speed, stability, and performance. Increasing the hidden size to 128 yielded diminishing returns, with only marginal performance improvements and slower convergence. The GA demonstrated moderate performance, but failed to scale effectively as task complexity increased, underscoring the limitations of heuristic approaches in dynamic environments. In future work, we intend to extend the evaluation by incorporating larger datasets and real-world task flow graphs and device configurations to validate the performance of the proposed methods in practical scenarios. Furthermore, we will compare the proposed approaches with other state-of-the-art methods, such as federated reinforcement learning or evolutionary deep learning, to identify areas of further improvement and deliver the next version of the introduced task scheduler.

#### ACKNOWLEDGMENT

This project has received funding from the EU Horizon EUROPE RIA programme under GA No. 101135423 - ENACT. This paper reflects only the authors' views.

#### REFERENCES

- [1] S. Gupta, A. K. Kar, A. Baabdullah, and W. A. Al-Khowaiter, "Big data with cognitive computing: A review for the future," *International Journal of Information Management*, vol. 42, pp. 78–89, 2018.
- [2] M. Chen, F. Herrera, and K. Hwang, "Cognitive computing: architecture, technologies and intelligent applications," *Ieee Access*, vol. 6, pp. 19 774–19 783, 2018.
- [3] S. Moreschini, F. Pecorelli, X. Li, S. Naz, M. Albano, D. Hästbacka, and D. Taibi, "Cognitive cloud: The definition," in *International Symposium on Distributed Computing and Artificial Intelligence*. Springer, 2022, pp. 219–229.
- [4] T. F. 3, "3: Architecture. developing a reference architecture for the continuum-concept, taxonomy and building blocks. zenodo (2023)."
- [5] H. Kokkonen, L. Lovén, N. H. Motlagh, A. Kumar, J. Partala, T. Nguyen, V. C. Pujol, P. Kostakos, T. Leppänen, A. González-Gil *et al.*, "Autonomy and intelligence in the computing continuum: Challenges, enablers, and future directions for orchestration," *arXiv preprint arXiv:2205.01423*, 2022.
- [6] E. Commission, "European industrial technology roadmap for the next generation cloud-edge offering," 2021, accessed: 2025-01-20. [Online]. Available: [https://ec.europa.eu/newsroom/repository/document/2021-18/European\\_CloudEdge\\_Technology\\_Investment\\_Roadmap\\_for\\_publication\\_pMdz85DSw6nqPppq8hE9S9RbB8\\_76223.pdf](https://ec.europa.eu/newsroom/repository/document/2021-18/European_CloudEdge_Technology_Investment_Roadmap_for_publication_pMdz85DSw6nqPppq8hE9S9RbB8_76223.pdf)
- [7] T. Adame, E. Amri, G. Antonopoulos, S. Azaiez, A. Berne, J. S. Camargo, H. Kakoulidis, S. Kleisarchaki, A. Llamedo, M. Prasinos *et al.*, "Presenting the cognifog framework: Architecture, building blocks and road toward cognitive connectivity," *Sensors*, vol. 24, no. 16, p. 5283, 2024.
- [8] V. Casamayor Pujol, A. Morichetta, I. Murturi, P. Kumar Donta, and S. Dustdar, "Fundamental research challenges for distributed computing continuum systems," *Information*, vol. 14, no. 3, p. 198, 2023.
- [9] A. Morichetta, V. C. Pujol, and S. Dustdar, "A roadmap on learning and reasoning for distributed computing continuum ecosystems," in *2021 IEEE International Conference on Edge Computing (EDGE)*. IEEE, 2021, pp. 25–31.
- [10] P. Almasan, J. Suárez-Varela, K. Rusek, P. Barlet-Ros, and A. Cabellos-Aparicio, "Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case," *Computer Communications*, vol. 196, pp. 184–194, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0140366422003784>
- [11] A. Pasiadis, T. Kotsiopoulos, G. Lazaridis, A. Drosou, D. Tzovaras, and P. Sarigiannidis, "Enabling cyber-attack mitigation techniques in a software defined network," in *2021 IEEE International Conference on Cyber Security and Resilience (CSR)*, 2021, pp. 497–502.
- [12] J. Berral, C. Wang, and A. Youssef, "AI4DL: Mining Behaviors of Deep Learning Workloads for Resource Management," in *HotCloud'20: Proceedings of the 12th USENIX Conference on Hot Topics in Cloud Computing*, July 2020, available online. [Online]. Available: [https://www.usenix.org/system/files/hotcloud20\\_paper\\_berral.pdf](https://www.usenix.org/system/files/hotcloud20_paper_berral.pdf)
- [13] G. P. Mattia and R. Beraldi, "Leveraging Reinforcement Learning for Online Scheduling of Real-Time Tasks in the Edge/Fog-to-Cloud Computing Continuum," in *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)*. IEEE, November 2021, pp. 1–9.
- [14] M. R. Rezaee, N. A. W. Abdul Hamid, M. Hussin, and Z. Zukarnain, "Fog Offloading and Task Management in IoT-Fog-Cloud Environment: Review of Algorithms, Networks, and SDN Application," *IEEE Access*, vol. 12, pp. 39 058–39 080, 2024.
- [15] M. Rzepka, P. Boryło, M. D. Assunção, A. Lasoń, and L. Lefèvre, "SDN-Based Fog and Cloud Interplay for Stream Processing," *Future Generation Computer Systems*, vol. 131, pp. 1–17, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X22000140>
- [16] G. P. Mattia and R. Beraldi, "Leveraging Reinforcement Learning for Online Scheduling of Real-Time Tasks in the Edge/Fog-to-Cloud Computing Continuum," in *2021 IEEE 20th International Symposium on Network Computing and Applications (NCA)*. Boston, MA, USA: IEEE, 2021, pp. 1–9.
- [17] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini, "The graph neural network model," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 61–80, 2009.
- [18] A. Robles Enciso and A. Skarmeta, *Task Offloading in Computing Continuum Using Collaborative Reinforcement Learning*, 01 2023, pp. 82–95.
- [19] M. Ma, "Multi-task offloading via graph neural networks in heterogeneous multi-access edge computing," 2024. [Online]. Available: <https://arxiv.org/abs/2306.10232>
- [20] R. Sarkar, S. Abi-Karam, Y. He, L. Sathidevi, and C. Hao, "Flowgnn: A dataflow architecture for real-time workload-agnostic graph neural network inference," 2022. [Online]. Available: <https://arxiv.org/abs/2204.13103>
- [21] A. Nizamis, J. Neises, D. Ospina, U. Wajid, C. I. V. López, P. Trakadas, K. Votis, O. Lazaro, S. Nechifor, and C. Palau, "Enact-a framework for adaptive scheduling and deployments of data intensive workloads on energy efficient edge to cloud continuum," in *2024 IEEE International Conference on Engineering, Technology, and Innovation (ICE/ITMC)*. IEEE, 2024, pp. 1–9.
- [22] M. Fey and J. E. Lenssen, "Fast graph representation learning with pytorch geometric," May 2019, released under the MIT License. [Online]. Available: [https://github.com/pyg-team/pytorch\\_geometric](https://github.com/pyg-team/pytorch_geometric)
- [23] A. Kouloumpris, G. L. Stavrinides, M. K. Michael, and T. Theodoridis, "Datasets of synthetic task flow graphs for evaluating a latency/energy optimization task allocation framework," Feb. 2024. [Online]. Available: <https://doi.org/10.5281/zenodo.10654551>