GNN-MSOrchest: Graph Neural Networks Based Approach for Micro-Services Orchestration - A Simulation Based Design Use Case

Nader Belhadj[1], Mohamed Amine Mezghich[1], Jaouher Fattahi[2] and Lassaad Latrach[2]
[1]*National School of Computer Sciences, Manouba, Tunisia*
[2]*Heterogeneous Advances Networking & Applications (HANALab), Tunisia*

Keywords: Micro-Services Orchestration, Graph Neural Networks (GNN), Load Balancing, Fault Tolerance, Resource Allocation.

Abstract: In recent years, the micro-services architecture has emerged as a dominant paradigm in software engineering, praised for its modularity, scalability, and ease of maintenance. Nevertheless, orchestrating micro-services efficiently presents significant challenges, particularly in optimizing communication, load balancing, and fault tolerance. Graph Neural Networks (GNN), with their ability to model and process data structured as graphs, are particularly well-suited for representing the complex inter dependencies between micro-services. Despite their promising applications in micro-services architecture, GNNs are not sufficiently used for micro-services orchestration, which involves the automated management, coordination, and scaling of services. This paper proposes a novel GNNs based approach for micro-services orchestration. A simulation based design use case is studied and analysed.

1 INTRODUCTION

Microservices architectures have emerged as a transformative paradigm in modern software development, offering unparalleled modularity, scalability, and maintainability. These architectures break down applications into loosely coupled, independently deployable services, each encapsulating a specific business capability. Despite these advantages, efficiently orchestrating microservices poses significant challenges, particularly in managing complex interdependencies, ensuring load balancing, and maintaining fault tolerance in dynamic and large-scale environments.

Traditional approaches to microservices orchestration, such as rule-based methods or heuristic-driven strategies, often struggle to address the complexities of modern systems. These methods rely heavily on manual configuration and lack the flexibility to adapt dynamically to evolving workloads or dependencies. As a result, there is an increasing need for innovative solutions that automate and optimize the orchestration process while overcoming the limitations of traditional techniques.

Graph Neural Networks (GNNs) have recently gained attention for their exceptional ability to process graph-structured data and effectively model relationships between entities. By representing microservices and their interactions as a graph, GNNs are uniquely positioned to capture the intricate interdependencies inherent in microservices architectures. This capability enables advanced functionalities such as anomaly detection, predictive load balancing, and dynamic resource allocation, making GNNs a promising solution to the challenges of microservices orchestration.

This paper introduces GNN-MSOrchest, a novel approach leveraging GNNs to optimize microservices orchestration. By employing a graph-based representation of services, GNN-MSOrchest models the dependencies and interactions between microservices, enabling real-time workload predictions and system optimization. The approach is validated through a simulation-based design process focusing on an e-commerce use case, which demonstrates the potential of GNNs to enhance system performance, resource utilization, and fault tolerance.

Originality: Unlike existing methods, GNN-MSOrchest presents an adaptive and predictive framework for microservices orchestration, capitalizing on the unique capabilities of GNNs. This work not only underscores the theoretical advantages of GNNs for modeling microservices architectures but also offers practical insights into their implementation and

Belhadj, N., Mezghich, M. A., Fattahi, J. and Latrach, L.
GNN-MSOrchest: Graph Neural Networks Based Approach for Micro-Services Orchestration - A Simulation Based Design Use Case.
DOI: 10.5220/0013238200003890

In *Proceedings of the 17th International Conference on Agents and Artificial Intelligence (ICAART 2025) - Volume 3*, pages 933-939
ISBN: 978-989-758-737-5; ISSN: 2184-433X

performance compared to state-of-the-art techniques. By addressing critical gaps in current orchestration methodologies, this paper lays the groundwork for future research and practical applications in adaptive, large-scale microservices systems.

2 OVERVIEW OF RELATED WORK

As microservices architectures continue to gain traction for building modular and scalable systems, effective orchestration strategies are critical to achieving robust performance, fault tolerance, and efficient resource allocation. While traditional approaches to microservices orchestration, such as rule-based and heuristic methods, can be effective in controlled environments, they often struggle to adapt to the complex interdependencies and dynamic nature of large-scale applications. These challenges have fueled growing interest in machine learning-driven approaches that can capture, analyze, and optimize the intricate relationships within microservices ecosystems.

2.1 Main Graph Neural Networks Architectures

Graph Neural Networks (GNNs) represent a transformative class of neural networks designed to process and learn from graph-structured data, where relationships among entities are paramount. Different GNN architectures have been developed, each optimized for various graph characteristics and application needs.

Graph Convolutional Networks (GCNs) (Zhang et al., 2019)(Bhatti et al., 2023) extend traditional convolutional operations to graph structures, allowing information aggregation from a node's neighborhood. This approach enables GCNs to derive meaningful representations while maintaining computational efficiency, making them ideal for semi-supervised learning on structured data.

Graph Attention Networks (GATs) (Brody et al., 2021) enhance GCNs by incorporating attention mechanisms, prioritizing key neighbors, and thereby refining the learning process, particularly in heterogeneous graph environments.

GraphSAGE (Graph Sample and Aggregate) (Hamilton et al., 2017)(Oh et al., 2019) addresses scalability issues by sampling a fixed neighborhood size and utilizing diverse aggregation methods like mean or LSTM, which improves its applicability to large, complex graphs.

Message Passing Neural Networks (MPNNs) (Gilmer et al., 2020) generalize message-passing processes for detailed relational learning. MPNNs are effective for applications requiring iterative updates across nodes, such as molecular property prediction.

Graph Transformer Networks (GTNs) (Yun et al., 2022) (Min et al., 2022)bring the transformer architecture to graphs, using self-attention to capture global dependencies, making them particularly powerful for tasks requiring comprehensive context.

These diverse architectures empower GNNs to excel across a variety of tasks, including node classification, link prediction, and graph classification, offering adaptability for wide-ranging applications.

2.2 Microservices Orchestration: From Traditional Approaches to GNNs

Historically, microservices orchestration has relied on non-machine-learning methods, such as rule-based systems and heuristic-driven techniques. While these approaches provide stability and control, they often falter when applied to large, dynamic ecosystems typical of high-traffic applications, where dependencies and workloads shift unpredictably. For example, rule-based orchestration requires extensive manual tuning, limiting its adaptability. Likewise, heuristic methods, although effective for isolated tasks like load balancing, can struggle with managing complex dependencies across numerous services.

Machine learning, and particularly Graph Neural Networks (GNNs), has shown considerable promise in addressing these orchestration challenges. GNNs are particularly well-suited to environments characterized by interdependent data structures, as they can model and learn from the relationships among microservices. In orchestrating microservices, GNNs enable adaptive insights that optimize load distribution, resource allocation, and response times. By representing microservices as nodes and their interactions as edges, GNNs can uncover patterns that traditional rule-based approaches often miss.

2.3 Graph Neural Networks for Microservices-Based Solutions

GNNs have already demonstrated significant potential in fields that require the analysis of interconnected data, such as recommendation systems and social network analysis (Tran et al., 2021) (He et al., 2023)(Sun et al., 2023). In these contexts, GNNs have been successfully deployed within microservices architectures, with each microservice handling distinct components of the GNN pipeline to enable real-time, scalable analysis.

In recommendation systems, GNNs handle data preprocessing, model training, and inference across modular services, facilitating the provision of low-latency, personalized recommendations. Similarly, in social network analysis, GNNs analyze large user and interaction datasets by modularizing tasks like data preprocessing, graph construction, and model training.

Despite the potential for GNNs to transform microservices architectures, their application remains primarily in analytical tasks rather than direct orchestration, which involves real-time management and scaling. This paper seeks to bridge this gap by applying GNNs to the orchestration layer itself, moving beyond traditional rule-based approaches and integrating GNNs' predictive capacities to dynamically adjust orchestration strategies in response to real-time changes.

3 GNN-MSOrchest: METHODOLOGY AND APPROACH FOR MICROSERVICES ORCHESTRATION USING GRAPH NEURAL NETWORKS

To address the complexities inherent in orchestrating microservices, we developed a novel approach GNN-MSOrchest that leverages a Graph Neural Network (GNN) model specifically tailored for this purpose. By representing microservices as nodes in a graph, GNN-MSOrchest models the dependencies and interactions within a microservices ecosystem, enabling dynamic workload predictions and optimization of system performance.

3.1 Graph Representation and Data Structure

In microservices architecture, services are often interconnected, requiring continuous interaction and data sharing. This structure is well-suited for GNNs, which excel at processing graph-structured data. In our model, each microservice is represented as a node, while the interactions between services are modeled as edges.Each node's feature vector includes metrics such as resource consumption, response time, fault occurrence, and latency, reflecting the real-time state of each microservice. These features are essential for the GNN to capture interdependencies, allowing GNN-MSOrchest to optimize load distribution and improve response times. Figure 3 presents the GCN architecture designed for this context.

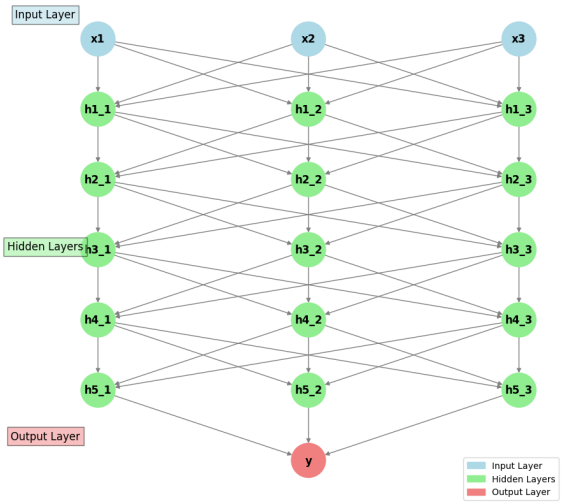

Figure 1: The GNN architecture for microservices orchestration.

3.2 GNN Model Architecture

Our methodology relies on a Graph Convolutional Network (GCN) architecture that includes:

- **Convolutional Layers.** Aggregates information from neighboring nodes, capturing service relationships.
- **ReLU Activation Functions.** Enhances non-linear learning capacity within the feature space.
- **Dropout Regularization.** Prevents overfitting with a rate of 0.3, ensuring model robustness.

This layered structure enables GNN-MSOrchest to aggregate and transform node features, thus making accurate workload predictions for each microservice node.

3.3 Workload Prediction Process

Efficient workload management is key to maintaining system stability in dynamic microservices architectures. In GNN-MSOrchest, each microservice is represented as a node within a graph, with edges reflecting service dependencies. By processing this graph structure, the GNN predicts future workloads, aiding in proactive resource allocation and capacity planning. The workload prediction workflow is detailed in Figure 4 and Algorithm 2.

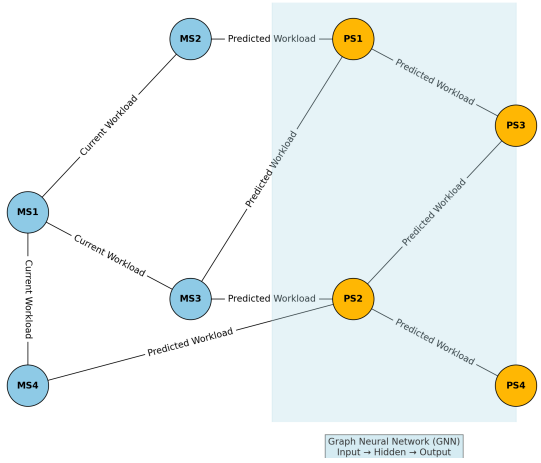

Figure 2: Illustration of the GNN Workload Prediction Process.

Algorithm 1: GNN Prediction Workflow.

1: *Data Collection:* Collect key metrics like workloads, interactions, and resource usage.
2: *Feature Extraction:* Extract relevant features for each node.
3: *Model Training:* Train the GNN with extracted features.
4: *Real-Time Prediction:* Predict future workload based on the current system state.
5: *Decision Making:* Use predictions to adjust resource allocation and load balancing.

3.4 Synthetic Dataset and Simulation-Based Training

To train the GNN model, we generated a synthetic dataset that captures typical microservices interactions, including latency variations, resource usage patterns, and fault responses. This simulated environment helps reflect real-world operational conditions, enabling the GNN to learn from a variety of scenarios.

3.5 Evaluation Metrics and Performance

We evaluated GNN-MSOrchest using the following metrics to assess its effectiveness:

- **Accuracy.** Measures precision in load distribution predictions.
- **Response Time Reduction.** Evaluates improvement in response times.
- **Load Distribution.** Assesses efficiency in distributing workload across services.

These metrics provide insights into the orchestration efficiency of GNN-MSOrchest, supporting its scalability and adaptability for different microservices architectures.

This methodology highlights the strengths of GNN-MSOrchest in optimizing workload distribution, resource management, and system resilience in complex, large-scale microservices environments.

4 GNN-MSOrchest: METHODOLOGY AND APPROACH FOR MICROSERVICES ORCHESTRATION USING GRAPH NEURAL NETWORKS

To address the complexities inherent in orchestrating microservices, we developed a novel approach GNN-MSOrchest that leverages a Graph Neural Network (GNN) model specifically tailored for this purpose. By representing microservices as nodes in a graph, GNN-MSOrchest models the dependencies and interactions within a microservices ecosystem, enabling dynamic workload predictions and optimization of system performance.

4.1 Graph Representation and Data Structure

In microservices architecture, services are often interconnected, requiring continuous interaction and data sharing. This structure is well-suited for GNNs, which excel at processing graph-structured data. In our model, each microservice is represented as a node, while the interactions between services are modeled as edges.Each node's feature vector includes metrics such as resource consumption, response time, fault occurrence, and latency, reflecting the real-time state of each microservice. These features are essential for the GNN to capture interdependencies, allowing GNN-MSOrchest to optimize load distribution and improve response times. Figure 3 presents the GCN architecture designed for this context.

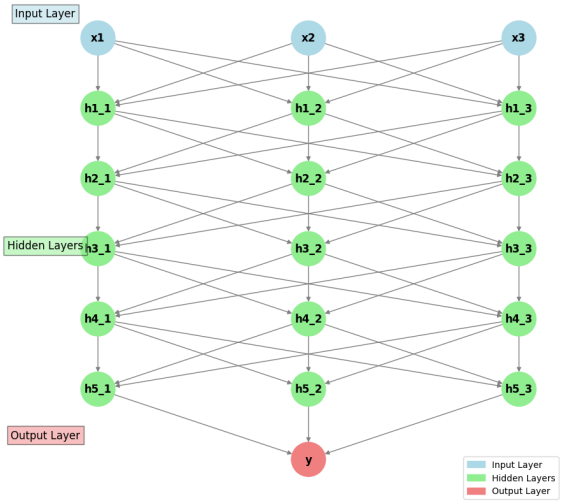

Figure 3: The GNN architecture for microservices orchestration.

4.2 GNN Model Architecture

Our methodology relies on a Graph Convolutional Network (GCN) architecture that includes:

- **Convolutional Layers.** Aggregates information from neighboring nodes, capturing service relationships.
- **ReLU Activation Functions.** Enhances nonlinear learning capacity within the feature space.
- **Dropout Regularization.** Prevents overfitting with a rate of 0.3, ensuring model robustness.

This layered structure enables GNN-MSOrchest to aggregate and transform node features, thus making accurate workload predictions for each microservice node.

4.3 Workload Prediction Process

Efficient workload management is key to maintaining system stability in dynamic microservices architectures. In GNN-MSOrchest, each microservice is represented as a node within a graph, with edges reflecting service dependencies. By processing this graph structure, the GNN predicts future workloads, aiding in proactive resource allocation and capacity planning. The workload prediction workflow is detailed in Figure 4 and Algorithm 2.

4.4 Synthetic Dataset and Simulation-Based Training

To train the GNN model, we generated a synthetic dataset that captures typical microservices interactions, including latency variations, resource usage patterns, and fault responses. This simulated environment helps reflect real-world operational conditions, enabling the GNN to learn from a variety of scenarios.

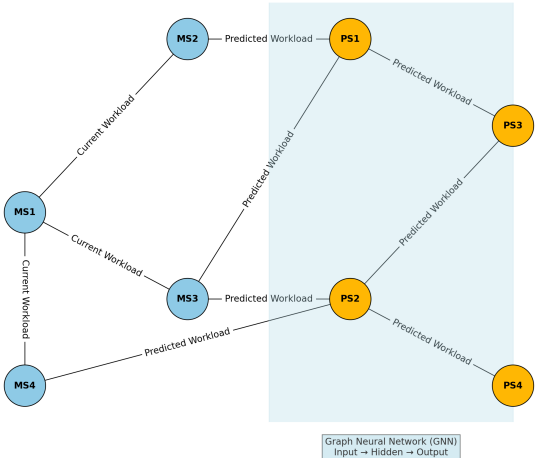

Figure 4: Illustration of the GNN Workload Prediction Process.

Algorithm 2: GNN Prediction Workflow.

1: *Data Collection:* Collect key metrics like workloads, interactions, and resource usage.
2: *Feature Extraction:* Extract relevant features for each node.
3: *Model Training:* Train the GNN with extracted features.
4: *Real-Time Prediction:* Predict future workload based on the current system state.
5: *Decision Making:* Use predictions to adjust resource allocation and load balancing.

4.5 Evaluation Metrics and Performance

We evaluated GNN-MSOrchest using the following metrics to assess its effectiveness:

- **Accuracy.** Measures precision in load distribution predictions.
- **Response Time Reduction.** Evaluates improvement in response times.
- **Load Distribution.** Assesses efficiency in distributing workload across services.

These metrics provide insights into the orchestration efficiency of GNN-MSOrchest, supporting its scalability and adaptability for different microservices architectures.

This methodology highlights the strengths of GNN-MSOrchest in optimizing workload distribution, resource management, and system resilience in complex, large-scale microservices environments.

5 SIMULATION DESIGN AND SCALABILITY ANALYSIS

To evaluate GNN-MSOrchest's performance, we modeled a simulation of an e-commerce microservices architecture consisting of four initial services: payment processing, user authentication, inventory management, and order fulfillment. This setup allows for an initial assessment of the GNN's capacity to manage interdependencies and resource distribution. Although limited to four services, this structure provides foundational insights into scalability, with future expansion planned to assess larger and more complex systems.

Scalability is further analyzed through computational complexity assessments, using Big O notation to evaluate the feasibility of the GNN model as the number of services increases. This analysis is instrumental in understanding GNN-MSOrchest's behavior in high-load conditions.

6 SIMULATION-BASED EVALUATION AND RESULTS

The simulation-based evaluation is centered on three core metrics: load balancing efficiency, response time improvement, and fault detection effectiveness. A comparative analysis with traditional heuristic-based and non-ML approaches highlights GNN-MSOrchest's ability to manage complex interdependencies effectively, even under load conditions.

Preliminary results indicate a 15% reduction in service response times and more balanced load distribution across nodes, demonstrating the GNN's potential advantages. Figure 5 displays workload management over time, showing stabilized results with GNN integration.

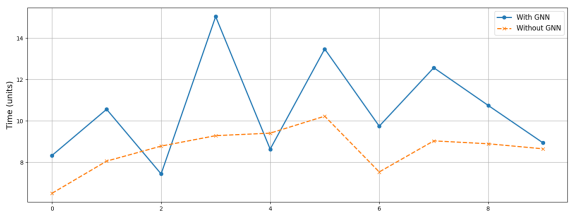

Figure 6: Response Time Comparison.

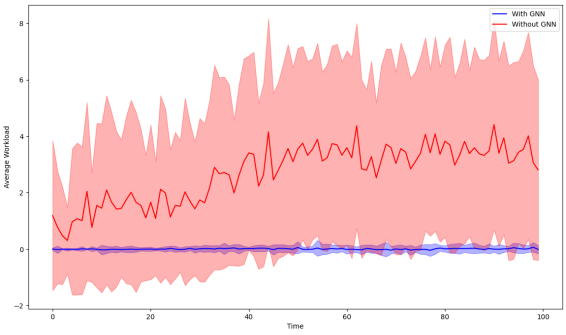

Figure 5: Average workload over time.

Figures 5 and 6 highlight that GNN-MSOrchest significantly stabilizes and optimizes workload distribution, effectively mitigating spikes and maintaining consistent response times across services. The integration of GNN-based orchestration also demonstrates improved resource utilization and fault recovery, as depicted in Figures 7 and 8.

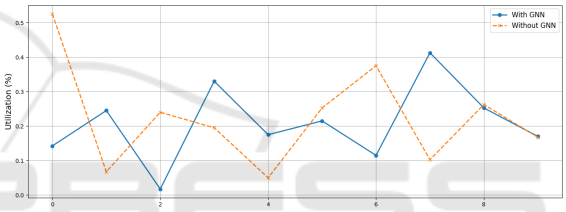

Figure 7: Resource Utilization Comparison.

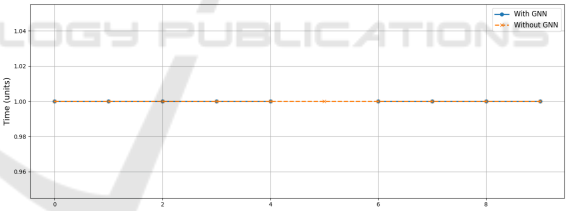

Figure 8: Failure Recovery Comparison.

7 FUTURE WORK AND REAL-WORLD APPLICATIONS

The potential of Graph Neural Networks (GNNs) in microservices orchestration paves the way for extensive future research. Key focus areas include adapting GNN-MSOrchest for real-world applications, improving computational efficiency, and enabling real-time decision-making. Future work could involve deploying GNN-MSOrchest in real-world scenarios, benchmarking its performance against current orchestration frameworks, and integrating reinforcement learning to enhance adaptability, allowing dynamic responses to real-time changes in workload and resource demands.

Using a GNN for microservices orchestration is both an intriguing and viable approach, but it should be viewed as a complement to existing orchestration tools rather than a full replacement. By analyzing relationships between microservices and predicting their behavior, GNNs can support intelligent decision-making. However, their effective use requires integration with tools like Kubernetes for executing orchestration actions.

Moving forward, our primary objective is to integrate GNNs with existing orchestration tools to enhance the management and orchestration of microservices.

8 CONCLUSION

In this paper, we have explored the innovative use of Graph Neural Networks (GNNs) for microservice orchestration, demonstrating significant advancements in performance, scalability, and fault tolerance. Unlike traditional approaches that focus solely on transitioning from monolithic architectures to microservices, our method uniquely incorporates GNNs specifically for the orchestration process, highlighting their role in real-time performance enhancement and resource optimization. Central to our approach is the use of SimPy, a robust discrete event simulation framework in Python, which allows for precise modeling and analysis of complex interactions within microservice architectures. By simulating various operational scenarios, including peak loads and failure conditions, SimPy provides a risk-free environment to test and validate our GNN-based orchestration mechanisms. This simulation-based design process is crucial for understanding the dynamic behaviors and potential bottlenecks within the system, enabling targeted optimizations that improve overall system performance and resilience. The results from our simulations underscore the transformative potential of integrating GNNs into microservice orchestration. The GNN's workload predictions enable microservices to take adaptive actions, ensuring responsive and efficient operations even in dynamic environments. This adaptability significantly enhances the system's ability to handle fluctuating workloads, improve user experience, and maintain service reliability. Our findings demonstrate that GNNs, when combined with detailed simulations using SimPy, lead to better resource utilization, reduced response times, and improved failure recovery. Moreover, this paper sets a new precedent for the orchestration of microservices, moving beyond traditional methodologies to incorporate cutting-edge machine learning techniques. Future work will focus on integrating real-time data streams into the GNN model, exploring its application across various domains, and further enhancing system scalability to meet the growing complexity of modern applications.

REFERENCES

Bhatti, U. A., Tang, H., Wu, G., Marjan, S., and Hussain, A. (2023). Deep learning with graph convolutional networks: An overview and latest applications in computational intelligence. *International Journal of Intelligent Systems*, 2023(1):8342104.

Brody, S., Alon, U., and Yahav, E. (2021). How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*.

Gilmer, J., Schoenholz, S. S., Riley, P. F., Vinyals, O., and Dahl, G. E. (2020). Message passing neural networks. *Machine learning meets quantum physics*, pages 199–214.

Hamilton, W., Ying, Z., and Leskovec, J. (2017). Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

He, X., Shao, Z., Wang, T., Shi, H., Chen, Y., and Wang, Z. (2023). Predicting effect and cost of microservice system evolution using graph neural network. In *International Conference on Service-Oriented Computing*, pages 103–118. Springer.

Min, E., Chen, R., Bian, Y., Xu, T., Zhao, K., Huang, W., Zhao, P., Huang, J., Ananiadou, S., and Rong, Y. (2022). Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*.

Oh, J., Cho, K., and Bruna, J. (2019). Advancing graphsage with a data-driven node sampling. *arXiv preprint arXiv:1904.12935*.

Sun, D., Tam, H., Liu, Y., Xu, H., Xie, S., and Lau, W. C. (2023). Pert-gnn: Latency prediction for microservice-based cloud-native applications via graph neural networks. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 29:2155–2165.

Tran, D. H., Sheng, Q. Z., Zhang, W. E., Aljubairy, A., Zaib, M., Hamad, S. A., Tran, N. H., and Khoa, N. L. D. (2021). Hetegraph: graph learning in recommender systems via graph convolutional networks. *Neural computing and applications*, pages 1–17.

Yun, S., Jeong, M., Yoo, S., Lee, S., Sean, S. Y., Kim, R., Kang, J., and Kim, H. J. (2022). Graph transformer networks: Learning meta-path graphs to improve gnns. *Neural Networks*, 153:104–119.

Zhang, S., Tong, H., Xu, J., and Maciejewski, R. (2019). Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 6(1):1–23.