



## UvA-DARE (Digital Academic Repository)

### LLM-based Optimization Algorithm Selection for High-Performance Networks Orchestration

Dalgkitsis, Anestis; Hsu, Cyril; Papagianni, Chrysa; Grosso, Paola; de Laat, Cees

**DOI**

[10.1145/3731599.3767458](https://doi.org/10.1145/3731599.3767458)

**Publication date**

2025

**Document Version**

Final published version

**Published in**

Proceedings of 2025 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis (SC 2025 Workshops)

**License**

CC BY

[Link to publication](#)

**Citation for published version (APA):**

Dalgkitsis, A., Hsu, C., Papagianni, C., Grosso, P., & de Laat, C. (2025). LLM-based Optimization Algorithm Selection for High-Performance Networks Orchestration. In *Proceedings of 2025 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis (SC 2025 Workshops): 16-21 Nov 2025, St. Louis, MO, USA* (pp. 854-859). Association for Computing Machinery. <https://doi.org/10.1145/3731599.3767458>

**General rights**

It is not permitted to download or to forward/distribute the text or part of it without the consent of the author(s) and/or copyright holder(s), other than for strictly personal, individual use, unless the work is under an open content license (like Creative Commons).

**Disclaimer/Complaints regulations**

If you believe that digital publication of certain material infringes any of your rights or (privacy) interests, please let the Library know, stating your reasons. In case of a legitimate complaint, the Library will make the material inaccessible and/or remove it from the website. Please Ask the Library: <https://uba.uva.nl/en/contact>, or a letter to: Library of the University of Amsterdam, Secretariat, P.O. Box 19185, 1000 GD Amsterdam, The Netherlands. You will be contacted as soon as possible.

UvA-DARE is a service provided by the library of the University of Amsterdam (<https://dare.uva.nl>)

# LLM-based Optimization Algorithm Selection for High-Performance Networks Orchestration

Anestis Dalgkitis  
Multiscale Networked Systems  
University of Amsterdam  
Amsterdam, North Holland  
Netherlands  
a.dalgkitis@uva.nl

Cyril Hsu  
Multiscale Networked Systems  
University of Amsterdam  
Amsterdam, North Holland  
Netherlands  
s.h.hsu@uva.nl

Chrysa Papagianni  
Multiscale Networked Systems  
University of Amsterdam  
Amsterdam, North Holland  
Netherlands  
c.papagianni@uva.nl

Paola Grosso  
Informatics Institute  
University of Amsterdam  
Amsterdam, North Holland  
Netherlands  
p.grosso@uva.nl

Cees de Laat  
Informatics Institute  
University of Amsterdam  
Amsterdam, North-Holland  
Netherlands  
Lawrence Berkeley National Lab  
Berkeley, California, USA  
c.t.a.m.delaat@uva.nl

## Abstract

The rapid growth of Artificial Intelligence (AI) applications, particularly through the widespread adoption of Large Language Models (LLMs), has caused an unprecedented growth in computing and network infrastructures. Current infrastructure expansion cannot keep pace, resulting in suboptimal performance. This creates an urgent need for network automation capable of dynamically orchestrating services and exploiting all available resources. Manual optimization processes are slow, error-prone, and unable to meet the requirements of complex, multi-domain, and data-intensive networks. A fundamental challenge is the absence of a universal optimization algorithm that performs effectively across all scenarios. In this paper, we present preliminary work on an LLM-based optimization algorithm selection framework for multi-domain, high-performance networks orchestration. The proposed framework utilizes LLM-generated descriptive embeddings of algorithms, network state logs, and service requests to identify the most suitable optimization method from a pool of algorithms, curating optimization to the current scenario.

## CCS Concepts

• **Networks** → Network management; • **Computing methodologies** → Natural language processing; • **Theory of computation** → Mathematical optimization; • **Software and its engineering** → Search-based software engineering.

## Keywords

Network Optimization, Large Language Models (LLMs), High-Performance Networks, Zero-Touch Network Orchestration, Algorithm Selection

## ACM Reference Format:

Anestis Dalgkitis, Cyril Hsu, Chrysa Papagianni, Paola Grosso, and Cees de Laat. 2025. LLM-based Optimization Algorithm Selection for High-Performance Networks Orchestration. In *Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis (SC Workshops '25)*, November 16–21, 2025, St Louis, MO, USA. ACM, New York, NY, USA, 6 pages. <https://doi.org/10.1145/3731599.3767458>

## 1 Introduction

The widespread adoption of Artificial Intelligence (AI) workflows, especially those powered by Large Language Models (LLMs) has significantly increased the computational and networking demands on high-performance infrastructures. Orchestrating complex workflows across distributed, heterogeneous networks has become increasingly challenging as the resources stagnate and complexity grows exponentially. Traditional optimization approaches designed for predictable workloads are now obsolete and unable to adapt under these conditions. In a counter-intuitive way, LLMs which contribute to this growing complexity, can also be leveraged to address it by enabling for the first time context-aware decision-making in orchestration processes, similar to human decision making, but at a fraction of the time.

The algorithm selection problem is well established in the literature since the 70s [10], with early studies supporting that no single optimization algorithm performs optimally across all problem instances. There is no global optimization algorithm. In multi-domain, data-intensive networks, the simultaneous use of multiple algorithms with conflicting objectives can degrade performance and reduce efficiency in resource allocation tasks or service orchestration [12]. Static selection strategies, or triggers based solely on thresholds and events, lack the contextual awareness necessary



This work is licensed under a Creative Commons Attribution 4.0 International License. *SC Workshops '25, St Louis, MO, USA*  
© 2025 Copyright held by the owner/author(s).  
ACM ISBN 979-8-4007-1871-7/25/11  
<https://doi.org/10.1145/3731599.3767458>

for robust and adaptive decision-making, leading to suboptimal optimization strategies.

Hence, it is evident that there is a need for a global intelligent optimization strategy that is able to distinguish the best optimization strategy needed based on the context. Human intervention, while context-aware, is too slow and error-prone for real-time operation. LLMs, on the other hand, can offer a scalable alternative by processing extensive logs, service descriptions, and identifying performance objectives based on text, to identify an optimal strategy and select the most suitable optimization algorithm from a set. Introducing another layer of abstraction in network service optimization based on algorithm selection could potentially revolutionize the industry and create a new type of architecture that drastically outperforms the status quo.

In this work, we propose an LLM-based Algorithm Selection Framework for high-performance network orchestration that introduces a context-driven abstraction layer prior to optimization. The framework utilizes LLM-generated descriptions and reasoning to interpret large-scale textual and numerical operational data, dynamically selecting the optimal optimization algorithm for the current service optimization request. We present a preliminary evaluation of our framework conducted with both simulations and a realistic demo prototype deployed on the experimental testbed FABRIC, spanning multiple continents and data-centers.

Our contributions are threefold by:

- Introducing a global optimization framework that utilizes algorithm selection based on the available data, optimization objective and conditions, to curate the optimization strategy;
- Designing an additional layer to optimization that is able to understand and process heterogeneous text-based context, and make an informed selection based on it; and
- Validating the experiment with both simulations and a testbed demo in FABRIC, on a multi-domain slice with preliminary results.

The remainder of this paper is organized as follows: Section II provides a brief background, explores related works and core projects that our work is based upon. Section III presents the framework architecture and components. Section IV outlines the live demonstration prototype that showcases the proposed solution in real-time. Section V, presents the simulation setup, evaluation methods and preliminary results of our LLM-based optimization algorithm selection feasibility study. Finally, Section VI concludes this work and highlights the future research directions.

## 2 Background & Related Work

In this section we introduce the core concepts that our Algorithm Selection framework is based upon. First we expand on the background of Algorithm Selection and then present the related works of the studied optimization scenario of Service Partitioning, also referred to simply as partitioning.

### 2.1 Algorithm Selection

Algorithm Selection was first presented in the 70s [10]. It is defined as the problem of selecting the most appropriate algorithm for a given problem instance from a set of available algorithms [12].

Traditional approaches in algorithm selection utilize methods such as sorting, ranking and score systems. Specifically, Cunha et al. in [1], present a ranking strategy to determine the suitability of an algorithm for a given problem instance. Authors in [7] presented an algorithm selection approach based on similarity, clustering and algorithm instance features. Specifically, Tornede et al. in [11] argue that considering only problem instance features, can lead to performance degradation and provide a related study.

Recently, algorithm selection has emerged again alongside the surge in the use of LLMs, as their reasoning abilities enable them to interpret context from vast textual datasets and have proven to be particularly effective for feature extraction, that can be matched with other techniques to provide a selection.

The authors in [12] propose an LLM-based algorithm selection framework that focuses on extracting features as representations from both algorithm and problem instances, and use them to determine the most suitable algorithm. The authors demonstrated that including algorithm representations significantly enhances selection accuracy compared to traditional methods relying only on problem features, especially when training data is limited.

### 2.2 Network Service Partitioning

Network Service Partitioning is the practice of dividing network services into distinct, well-defined parts or domains to improve scalability, performance, and resource management. The objective is to enable independent, parallel operation and optimization of each partition across the network.

We opt for service partitioning as our preliminary use case scenario study, as it provides a generic high-level orchestration example that has diverse optimization objectives and long term planning is required [6]. Service partitioning is prominent in decentralized computing and 5G/6G zero-touch service management and orchestration research for multi-site topologies.

Traditional approaches are based on classical optimization or heuristics for partitioning or similar problems such as Virtual Network Function (VNF) placement. The authors in [2] use Integer Linear Programming and Gurobi Solver to minimize the End-to-End (E2E) latency between users and service VNFs in network services. On a similar note, Quang et al. in [9] use Integer Linear Programming for dynamic resource allocation to solve VNF embedding problem.

Most common machine learning approaches use deep neural networks for network orchestration. The authors in [4] propose a global scalable Reinforcement Learning (RL)-based solution for VNF embedding and migration.

In more recent approaches, Pantelas et al. in [8] devised a deep multi-agent RL scheme for decentralized partitioning in multi-domain networks. Their solution uses the double deep Q-Network learning (DDQN). Similarly, the authors in [3] present a multi-agent RL decentralized service function chain orchestration framework that can significantly reduce service latency compared to centralized solutions.

In this preliminary work we develop several service partitioning baselines based on the presented related works. We design and deploy an optimization algorithm selection framework to study a

diverse set of LLM models to assess the feasibility and performance of the proposed work.

### 3 Framework Architecture

In this section, we outline the proposed architecture and functional components.

#### 3.1 Optimization Algorithm Selection Framework Architecture

Fig. 1 presents the architecture of the proposed framework. We

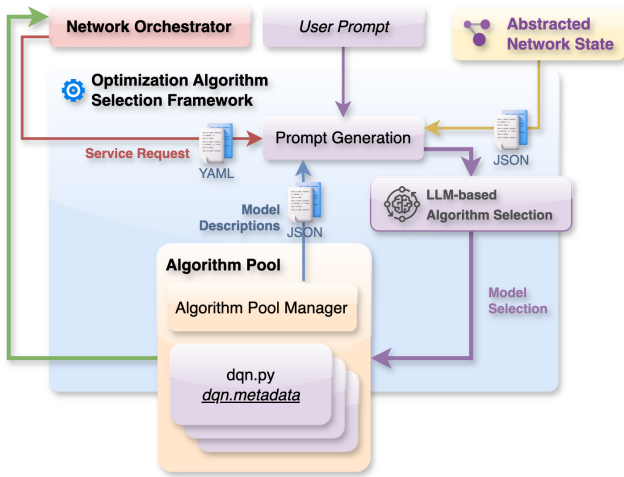


Figure 1: LLM-based optimization algorithm selection framework architecture and components.

present the main components and their functionality below:

- **Algorithm Pool:** a module that contains and manages a set of network service optimization algorithms and models. There are metadata tied to the algorithms that offer additional information such as textual descriptions, Abstract Syntax Tree (AST) and complexity analyses, as well as past performance metrics. A management sub-module is responsible for read-write operations.
- **Prompt Generator:** a module that concentrates all textual information required to build the algorithm selection prompt. It extracts the requested optimization objective, custom user prompt, abstract network representation and an extensive description of the available algorithm pool set.
- **LLM Algorithm Selector:** It is responsible for LLM-inference based on the generated prompt. It provides post-processing and verification of the output and handles possible errors during selection text generation.

The framework interacts with the managing Network Orchestrator via an API.

## 4 Demonstration Prototype & Preliminary Implementation

In this section we provide information about the prototype that we have designed to demonstrate the proposed framework, interacting with a real network to complement the simulation results.

### 4.1 FABRIC International Testbed

The Framework for Accelerated Built-In Resilience Infrastructure for Computing, or FABRIC for short, is a high-performance international research infrastructure testbed, created to support research between institutions and companies [5]. It offers a scalable, distributed environment equipped with high-performance computing resources available to researchers. The testbed also features advanced networking capabilities and FABRIC Across Borders, also known as FAB, an extension that connects the core North American network with global institutions, to encourage international collaboration and speed up scientific discovery across the globe.

### 4.2 Experimental FABRIC Slice Setup

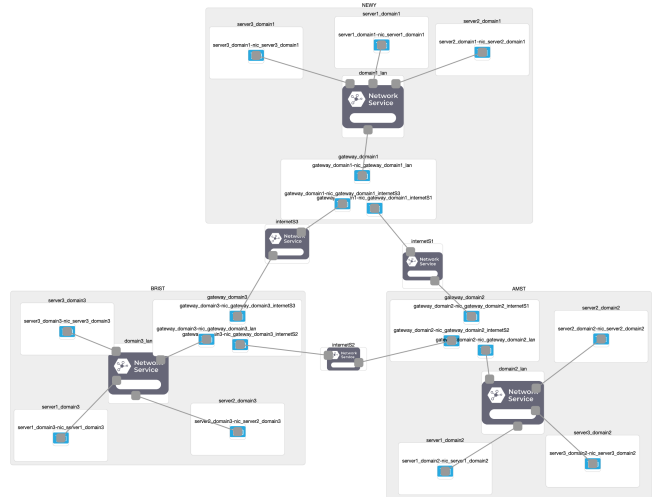


Figure 2: Proof of FABRIC slice topology as exported from the portal slice dashboard.

The demo slice is deployed between north-eastern America and north-western Europe, interconnecting data-centers from the University of New York (NEWY), The University of Amsterdam (AMST), and the University of Bristol (BRIST). Fig. 2 shows the topology of prototype slice that is used to showcase LLM-based optimization algorithm selection in real-time. Each datacenter represents a domain and has 3 Virtual Machines (VMs) attached that act as intra-domain servers able to host service VNFs.

### 4.3 Code Setup & Prototype Deployment

Fig. 3 shows the architecture of the live demo prototype. Each slice has a rudimentary domain manager that monitors the available local resources and can either accept or reject the partitioned slice requests. The optimization algorithm selection module acts as an

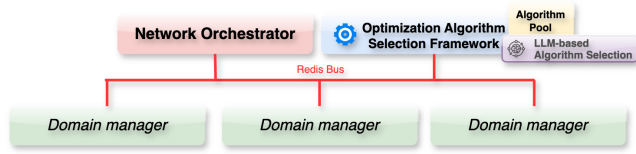


Figure 3: Prototype implementation architecture.

orchestration, and resides in the AMST domain. All modules, developed in Python, communicate via Redis publish-subscribe topics. For demo purposes the LLM-based algorithm selection is based on OpenAI GPT-4o-mini and Llama3.2:1b hosted locally with Ollama.



Figure 4: Preliminary interactive demo dashboard.

In addition Fig. 4 shows a preliminary dashboard that is used to showcase the system operation in real-time. Part of the demo includes letting the user interact with the prototype by typing a custom prompt or altering the network state.

## 5 Preliminary Framework Evaluation

In this section we present a preliminary evaluation of the LLM-based optimization algorithm selection. We study the feasibility of the proposed solution for the scenario of service partitioning via simulation. We also explain the baselines and evaluation methods before proceeding to the presentation of the preliminary results.

### 5.1 Services

In order to assess the performance of the proposed framework, we define network services inspired by the 5G, beyond 5G and 6G operator services, as they express a diverse set of requirements for optimization tasks. We define the service requirements as presented in the Table 1. The columns present the service type category, average E2E service latency denoted as  $\lambda$  in ms, average E2E service throughput denoted as  $\tau$  in Mbps, and expected reliability expressed in percentages.

### 5.2 Baselines

We use the following baselines to assess the robustness and adaptability of optimization algorithm selection across various generated scenarios.

Table 1: Service Types

| Service Type | $\lambda$ (ms) | $\tau$ (Mbps) | Reliability |
|--------------|----------------|---------------|-------------|
| eMBB         | 1–10           | 100–1000      | 99.9%       |
| URLLC        | 0.5–5          | 10–100        | 99.999%     |
| mMTC         | 10–100         | 1–10          | 99%         |
| V2X          | 1–5            | 50–200        | 99.99%      |
| AR/VR        | 5–20           | 200–500       | 99.9%       |

Regarding the algorithm selection, we are utilizing the following models for our study:

- **Llama3.2:1b**: Lightweight local model to test selection under tight compute/latency and strict data-privacy constraints.
- **Llama3.2:3b**: Stronger on-device baseline to study sensitivity to model size while keeping reproducibility and security.
- **GPT-4o**: High-capability reference to serve as an approximate upper bound and assess cross-provider robustness by using a 3rd party option.
- **GPT-4o-mini**: Cost/latency-efficient 3rd party option to evaluate budget-aware selection and trade-offs.

The partitioning methods are based on the presented background and related works. Specifically we use the following

- **Single DDQN RL Agent**: An RL agent with global network visibility that learns an optimal partitioning policy by minimizing overestimation bias in Q-value updates [4].
- **Multi-Agent DDQN RL**: An extension of DDQN where multiple RL agents operate in parallel per domain, each responsible for local operations. The agents act cooperatively or competitively, learning to improve network-wide service partitioning [3, 8].
- **Greedy Partitioning**: A baseline heuristic approach that iteratively selects the locally optimal partitioning choice step by step, without considering long-term consequences, simulating models that aim for simplicity and low computational cost.
- **SLA-Aware Partitioning**: A heuristic baseline that partitions services by optimizing contractual constraints first.
- **Logic-Based Partitioning**: A rule-driven baseline strategy where partitioning decisions follow predefined logical conditions, simulating simple approaches.
- **Resource-Based (CPU) Partitioning**: Resource-centric greedy method where partitioning is determined primarily by CPU utilization and availability.
- **Random Partitioning**: Non-deterministic baseline where partitions are decided randomly. Used to evaluate the relative advantage of more sophisticated algorithms.

### 5.3 Evaluation methods

We describe different evaluation methods used to assess the feasibility and performance of the proposed solution.

Starting with Success Rate, we define the **Partitioning Success Rate PSR** as the successful partitions  $S_{success}$  per total services  $S_{total}$ , multiplied by 100%:

$$PSR = |S_{success}|/|S_{total}| \tag{1}$$

A successful partition  $S_{success}$  requires the following state:

- **Resource Constraints Met:** Each partition fits within domain capacity limits;
- **Valid VNF Placement:** All VNFs are successfully assigned to domains after partition admission;
- **Network Reachability:** Inter-domain links can support the service traffic requirements;
- **SLA Feasibility:** The resulting partition can meet the service latency and throughput requirements.

SLA Compliance Rate is calculated as follows:

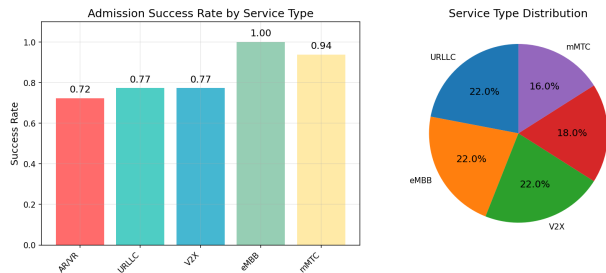
$$SCR = |S_{compliant}|/|S_{success}| \tag{2}$$

An SLA compliant service partitioning  $S_{compliant}$  requires the following state:

- **Latency constraints met:** Maximum acceptable E2E service delay.
- **Throughput constraints met:** Minimum required data transfer rate achieved by the admitted service.
- **Reliability:** Service admission probability threshold met without failed instantiations.
- **Availability:** Service uptime percentage requirement met.

### 5.4 Preliminary Results

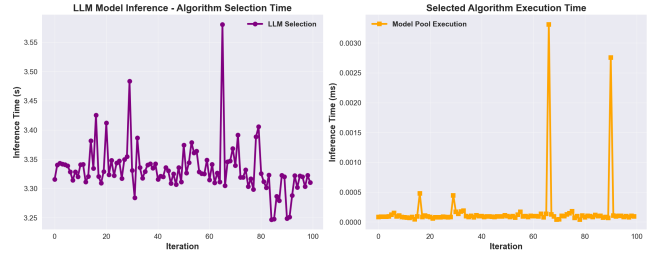
First, we explore a live prototype study for 100 iterations. We are using the open source llama3.2:3b for LLM-based algorithm selection for the demo slice topology.



**Figure 5: (a) Partitioning admission success rate per service type. (b) Service Type Distribution.**

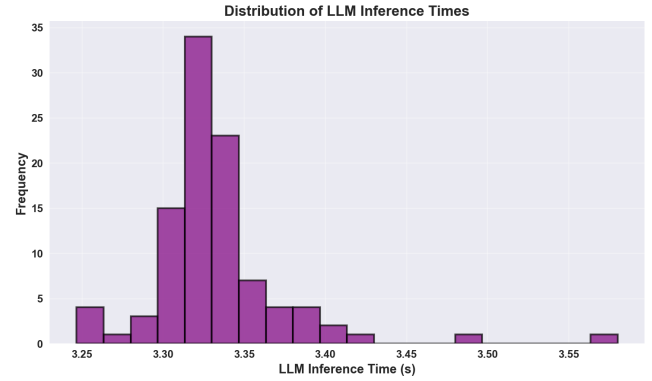
Fig. 5a presents the admission success rate per service type, meaning the rate of successfully deployed partitioned services on all underlined domains, without violating the SLA. We can see that AR/VR, URLLC and V2X services have the lowest success rate as they require low latency and most partitions violate the service SLA. Fig. 5b shows the service type distributions that were used for this study.

Moreover, Fig. 6a shows the total LLM inference time for 100 iterations for the model llama3.2:3b. Similarly, Fig. 6b shows the total Model pool model inference time of the selected algorithm. In this particular instance the selection fluctuated between the Greedy solution and Single DDQN due to the simplicity of the test scenario.



**Figure 6: (a) LLM inference time with Llama:3b local model. (b) Selected partitioning algorithm total execution time. Both plots show a snapshot of 100 iteration simulation.**

It is easy to distinguish when the algorithm selection opted for the Single DDQN solution.



**Figure 7: LLM inference time distribution in milliseconds.**

To conclude this study, Fig. 7 shows the distribution of local LLM inference times for the model Llama3.2:3b. The LLM inference time is easily affected by the current available resources when a local model is used. Under normal load, especially when sufficient unified or GPU memory is available, the algorithm selection lasts a predictable and reliable amount of time, as expressed by the plot approximately 3.322 seconds.

In this part of the preliminary results section, we present an extensive simulation that compares various aspects of algorithm selection with different LLMs, including both local and 3rd party options.

In Fig. 8a we show a partitioning success rate comparison per LLM-based algorithm selection model. Fig. 8b presents the average inference time per selection model. We can see that the small local model Llama3.2:1b offers similar performance to the rest of the models and a brief inference time. The 3rd party gpt-4o model requires more inference time as it uses reasoning to generate a more accurate selection. Similarly, the online gpt-4o-mini model offers quick inference with a tradeoff in accuracy.

Table 2 highlights a numerical performance comparison of the optimization algorithm selection models. The columns show the partitioning success rate in percentage, average E2E service latency denoted as  $\lambda$  in ms, average E2E service throughput denoted as  $\tau$  in Mbps, and the average execution time in ms.



Figure 8: (a) Partitioning success rate by LLM-based algorithm selection Model. (b) Average inference time per model.

Table 2: Comparison Table

| Model       | Success % | $\lambda$ (ms) | $\tau$ (Mbps) | Time (ms) |
|-------------|-----------|----------------|---------------|-----------|
| llama3.2:1b | 84.18%    | 15.45          | 176.29        | 207.360   |
| llama3.2:3b | 92.15%    | 15.85          | 178.14        | 756.445   |
| gpt-4o      | 88.32%    | 14.87          | 208.06        | 447.103   |
| gpt-4o-mini | 68.45%    | 15.08          | 209.85        | 158.993   |

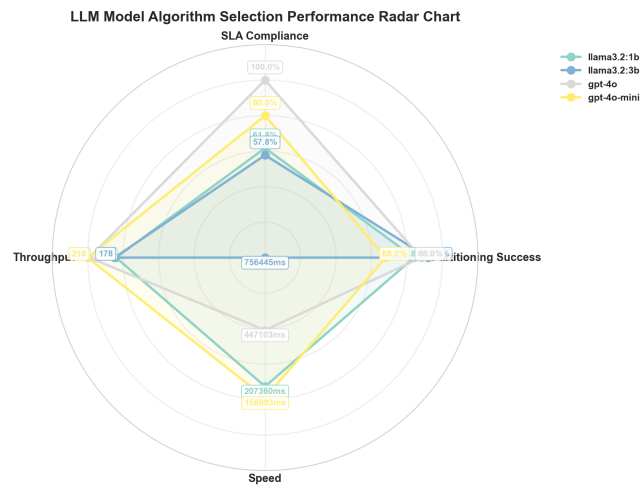


Figure 9: Selection performance radar chart.

Finally, Fig. 9 shows a radar plot of the current study comparing the optimization algorithm selection per model.

## 6 Conclusion

In this work we have presented a preliminary study on an LLM-based optimization algorithm selection framework for high-performance, multi-domain network orchestration scenarios. The proposed approach dynamically curates the selection of optimization algorithms best suited to the current conditions and scenarios, potentially outperforming static or single-algorithm strategies. Preliminary experiments on the FABRIC testbed demonstrate feasibility and the potential of this method to improve orchestration efficiency by introducing one abstracted context-aware layer in the optimization process.

Our future work will focus on expanding to a diverse LLM model comparison with intermediate-sized models that was skipped, due to lack of available inference hardware during the preliminary results. We will introduce comparisons between different prompting methods such as chain-of-thought, retrieval-augmented generation, and instruction fine-tuning. Finally to provide a more generalized use case that includes several different network service optimization scenarios, not confined only to service partitioning.

## Acknowledgments

This work is supported by CIENA, the European Commission H2020 project DESIRE6G (101096466). It was made possible by FABRIC and specifically FABRIC Across Borders.

## References

- [1] Tiago Cunha, Carlos Soares, and André CPLF de Carvalho. 2018. A label ranking approach for selecting rankings of collaborative filtering algorithms. In *Proceedings of the 33rd Annual ACM Symposium on Applied Computing*. ACM, 1393–1395.
- [2] Richard Cziva and Dimitrios P. Pazaros. 2017. On the Latency Benefits of Edge NFV. *2017 ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS) (2017)*, 105–106.
- [3] Anestis Dalgkitsis, Luis Garrido, Kostas Ramantas, Luis Alonso, and Christos Verikoukis. 2021. SCHEMA: Service Chain Elastic Management with Distributed Reinforcement Learning. In *GLOBECOM 2021 - 2021 IEEE Global Communications Conference*. 1–6.
- [4] Anestis Dalgkitsis, Prodromos-Vasileios Mekikis, Angelos Antonopoulos, Georgios Korkentzas, and Christos Verikoukis. 2020. Dynamic Resource Aware VNF Placement with Deep Reinforcement Learning for 5G Networks. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*. 1–6. doi:10.1109/GLOBECOM42002.2020.9322512
- [5] FABRIC Testbed Project. 2024. FABRIC: Adaptive Programmable Research Infrastructure for Computer Science and Science Applications. <https://portal.fabric-testbed.net/about/fabric>. Accessed: 2025-09-11.
- [6] Cyril Shih-Huan Hsu, Anestis Dalgkitsis, Chrysa Papagianni, and Paola Grosso. 2025. Transformer-Empowered Actor-Critic Reinforcement Learning for Sequence-Aware Service Function Chain Partitioning. arXiv:2504.18902 [cs.NI] <https://arxiv.org/abs/2504.18902>
- [7] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. 2010. ISAC – Instance-Specific Algorithm Configuration. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*. IOS Press, 751–756. doi:10.3233/978-1-60750-606-5-751
- [8] Angelos Pentelas, Danny De Vleeschauwer, Chia-Yu Chang, Koen De Schepper, and Panagiotis Papadimitriou. 2023. Deep Multi-Agent Reinforcement Learning With Minimal Cross-Agent Communication for SFC Partitioning. *IEEE Access* 11 (2023), 40384–40398. doi:10.1109/ACCESS.2023.3269576
- [9] P. T. A. Quang, A. Bradai, K. D. Singh, G. Picard, and R. Riggio. 2019. Single and Multi-Domain Adaptive Allocation Algorithms for VNF Forwarding Graph Embedding. *IEEE Transactions on Network and Service Management* 16, 1 (2019), 98–112. doi:10.1109/TNSM.2018.2876623
- [10] John R. Rice. 1976. The Algorithm Selection Problem\*\*This work was partially supported by the National Science Foundation through Grant GP-32940X. This chapter was presented as the George E. Forsythe Memorial Lecture at the Computer Science Conference, February 19, 1975, Washington, D. C. *Advances in Computers*, Vol. 15. Elsevier, 65–118. doi:10.1016/S0065-2458(08)60520-3
- [11] Alexander Tornede, Marcel Wever, and Eyke Hüllermeier. 2020. Extreme Algorithm Selection with Dyadic Feature Representation. In *Discovery Science - 23rd International Conference, DS 2020, Thessaloniki, Greece, October 19–21, 2020, Proceedings (Lecture Notes in Computer Science, Vol. 12323)*. Springer, 309–324. doi:10.1007/978-3-030-61527-7\_21
- [12] Xingyu Wu, Yan Zhong, Jibin Wu, and KC Tan. 2024. AS-LLM: When Algorithm Selection Meets Large Language Model. <https://openreview.net/forum?id=17aD9VMQUq>