Received 27 February 2025; revised 31 March 2025; accepted 10 April 2025. Date of publication 16 April 2025; date of current version 30 April 2025.

Digital Object Identifier 10.1109/OJCOMS.2025.3561605

A Matching Game for LLM Layer Deployment in Heterogeneous Edge Networks

BENEDETTA PICANO[1] (Member, IEEE), DINH THAI HOANG[2] (Member, IEEE), AND DIEP N. NGUYEN[2] (Senior Member, IEEE)

(Special Issue on Generative AI and Large Language Models Enhanced 6G Wireless Communication and Sensing)

[1]School of Electrical and Data Engineering, University of Florence, 50121 Florence, Italy
[2]School of Electrical and Data Engineering, University of Technology Sydney, Ultimo, NSW 2007, Australia

CORRESPONDING AUTHOR: B. PICANO (e-mail: benedetta.picano@unifi.it).

This work was supported in part by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (Program "RESTART") under Grant PE0000001.

ABSTRACT With the growing demand for computational and storage capabilities of modern learning models, performing their computation exclusively in a centralized manner has become increasingly impractical. Executing the inference of foundation models in a distributed manner presents significant challenges, particularly in optimizing both computing and communication resources. This work introduces a novel deployment scheme for large language model (LLM) layers that jointly considers computation and communication efficiency within an edge network environment to address these issues. Specifically, we resort to the matching theory to effectively orchestrate the distributed deployment of the LLM layers across the edge nodes of the networks, where nodes have varying computational capacities and communication speed. This framework is based on a two-sided game, enabling each layer to express its individual preferences for node allocation while allowing nodes to prioritize their preferred layers. This mutual selection process minimizes inference latency in the learning process and models the bubble time as game externalities, assuming a sequential pipeline execution. The algorithmic solution reaches a stable matching outcome. Performance evaluation was conducted considering both simulations and a small-scale testbed to measure the effectiveness of the proposed algorithm compared to state-of-the-art alternatives. In particular, the small-scale testbed was developed to distribute an LLM to support autonomous driving, leveraging the vision-language model paradigm. The results highlight performance improvements of up to around 10% in comparison to the Koklata game alternative.

INDEX TERMS Foundation models, distributed inference, matching theory, edge intelligence.

I. INTRODUCTION

THE RAPID evolution of foundation models has revolutionized the design and functionality of next-generation intelligent networks, unveiling new opportunities and enabling capabilities that were previously beyond reach. A foundation model refers to any model developed through extensive training on large-scale datasets, typically using self-supervised learning paradigms. These models are inherently versatile, as they can be fine-tuned and adapted to perform a wide plethora of downstream tasks with remarkable learning performance [1], [2], [3], [4], [5], [6], [7]. Notable examples include BERT [8] and GPT-3 [9], which exemplify the transformative impact of these models across various domains. From a technological perspective, the core principles of foundation models are deep neural networks and self-supervised learning, which are not novel concepts as they have been integral components of machine learning for decades. However, what distinguishes modern foundation models is their unprecedented scale and the breadth of their applications. By leveraging billions of parameters and datasets encompassing an Internet-scale corpus, foundation models exhibit the remarkable ability to generalize to unseen data without undergoing additional training phases, following the zero-shot learning paradigm. This characteristic significantly enhances their adaptability and opens the door to more dynamic use cases. Furthermore, these models can undergo fine-tuning, a process of retraining on smaller, task-specific datasets to optimize their performance for targeted applications, ensuring high precision and robustness [10].

Alongside these rapid advancements, the development and deployment of foundation models bring significant computational challenges, particularly during both their pre-training and inference phases. The pre-training process for these models demands substantial computational resources, including high-performance hardware such as GPUs or TPUs, and often spans weeks or even months. This results in considerable costs in terms of energy consumption and the infrastructure required to manage such large-scale computations. For instance, training state-of-the-art models like GPT-3 involves processing billions of parameters over extensive datasets, necessitating advanced optimization techniques and parallel processing capabilities [11]. However, the computational demand does not end with pretraining. The inference phase, where these models are deployed to process real-time requests, is equally resource-intensive. Due to their large size and complex architectures, foundation models require significant memory and processing power to perform tasks such as text generation, contextual reasoning, or decision-making. Moreover, the reliance on cloud-based architectures introduces challenges such as latency, bandwidth constraints, and data privacy concerns [12], [13]. This challenge is particularly pronounced in latency-sensitive applications or edge environments, where limited computational resources may struggle to meet the demands of large-scale inference. These dual pressures highlight the necessity for innovations in both training efficiency and inference optimization to make the use of foundation models more sustainable and accessible.

Edge computing has recently emerged as a promising processing paradigm for performing collaborative distributed inference [14]. Accordingly, edge nodes host shards of foundation learning models and exchange intermediate output values to complete the autoregressive inference. During the inference process of foundation models, every input token must sequentially pass through the entire stack of transformer layers that comprise the model. This architectural requirement creates significant challenges when implementing distributed inference across a heterogeneous edge network, i.e., a network composed of devices with differing capacities [11]. Each layer processes intermediate representations of the input data, which must then be communicated to the next processing node in the sequence. Resource-constrained nodes may struggle to handle even a subset of transformer layers, necessitating careful scheduling and load distribution strategies. Additionally, ensuring synchronization between nodes while maintaining the strict sequential flow of data through the model poses additional operational challenges. Therefore, the distributed inference process is tightly coupled to the model architecture, as the placement of transformer layers across different nodes directly influences both performance and resource utilization.

For instance, transferring intermediate activations between layers located on separate nodes incurs communication overhead, which must be minimized to achieve low-latency responses. Simultaneously, computational workloads need to be balanced across nodes with varying processing capabilities to ensure efficient utilization of resources. One additional significant challenge in implementing distributed inference for foundation models is the problem of bubble-time in model parallelism. This phenomenon arises when different portions of the model are allocated to separate devices, and the sequential nature of the transformer architecture creates idle periods on some devices while others are actively processing. Since each token must propagate through the transformer layers in sequence, the devices hosting later layers cannot begin processing until they receive the intermediate results from the preceding layers. This leads to underutilization of computational resources, as certain devices remain idle during these waiting periods, forming bubbles of unproductive time [14], [15]. To overcome these obstacles, it is essential to design strategies that jointly optimize communication and computation resource allocation, taking into account the specific requirements and constraints imposed by the foundation model architecture and network resources. In so doing, distributed inference can enable the deployment of foundation models in resource-constrained environments, unlocking their potential for real-time and scalable applications [11], [14], [15]. Traditional centralized optimization approaches may be computationally expensive and infeasible in dynamic environments, while other game-theoretic models, such as auction-based or bargaining formulations, introduce complexities such as communication overhead, as nodes must continuously exchange bids. To tackle these challenges, we adopt a matching theory approach, which provides a scalable and distributed framework for resource allocation. Unlike conventional optimization methods, matching theory explicitly considers the preferences and constraints of both Large Language Model (LLM) inference tasks and edge nodes, leading to an adaptive allocation mechanism.

This paper aims to develop an innovative matching theory-based algorithm to optimize the LLM layer deployment across a heterogeneous edge network. The theory of matching provides a structured method for associating elements from two distinct groups based on their preferences, aiming to establish mutually advantageous pairings. The objective of a matching game is to reach a stable outcome in which no two participants have an incentive to deviate by exchanging their assigned partners. The proposed game model is established between LLM layers and edge nodes, where communication and inference time contributions are considered as metrics of the two sets involved in the game. The main objective of the matching game is to minimize the inference latency. In order to reduce bubble times, the waiting time experienced by batches is modeled as game externalities, assuming a sequential pipeline execution.

The stability of the designed matching algorithm is proved and discussed. A small-scale testbed is also presented to demonstrate the performance of the proposed approach in a real-world use case.

The contributions of this paper can be summarized as follows.

- Formulation of the distributed inference optimization problem for LLMs to minimize the inference latency. The problem addressed considers a heterogeneous edge network, i.e., a network where edge nodes have different computation and communication facilities;
- Development of a novel matching theory-based algorithm to achieve a sub-optimal solution to the problem formulated. The designed matching models the deployment of LLM layers among edge nodes, and bubble times represent game externalities.
- Proof and analysis of the stability of the matching game, where the preferences of players are influenced by dependencies and relationships within the system. Externalities arise when the ranking of a specific match for one participant depends not only on their direct preferences but also on the assignments of other participants in the system. This interdependence introduces significant challenges to achieving stable matchings, as traditional stability criteria do not account for such complex dynamics.
- Experimental results to provide a critical comparison between the proposed matching solution and alternative algorithmic approaches. Furthermore, a small-scale testbed is discussed to showcase the application of a distributed LLM as a support system for autonomous driving. In this scenario, the LLM is employed to assist in tasks such as contextual decision-making, natural language-based navigation guidance, and real-time environmental analysis. The experimental results highlight the practical feasibility and effectiveness of the proposed framework, emphasizing its potential to perform effective LLM shards deployment.

The rest of the paper is articulated as follows. In Section II, we present an accurate literature review. The problem statement is detailed in Section III, and the proposed approach is described in Section IV. Performance evaluation is discussed in Section V, and conclusions are drawn in Section VI.

II. RELATED WORKS

In recent works, matching theory is widely adopted to perform assignment in distributed computing systems. For example, in paper [16] authors propose a dynamic matching game that jointly optimizes task offloading and resource allocation in fog computing environments. The model incorporates externalities to reflect the interdependence of decisions across users and servers, adapting to changing workloads and link conditions over time. Paper [17] tackles joint optimization of content caching, service placement, and task offloading in UAV-enabled MEC networks. The framework considers latency, energy, and mobility to dynamically allocate tasks across flying and ground servers, optimizing user experience in aerial edge scenarios. A cross-layer matching game for cloud-assisted mobile edge networks is designed in [18], where task scheduling spans both the service and trading layers. The model integrates overbooking strategies and forward contracts to manage uncertainty and enable risk-aware service allocation in dynamic environments. Authors in [19] introduce a tripartite matching model for UAV-assisted covert communications. The framework forms stable associations among UAVs, users, and jammers by leveraging game theory and auction mechanisms, focusing on security, interference mitigation, and low-complexity matching under covert constraints. Similarly, in [20] a matching game-based framework for resource allocation in space communication networks under incomplete information is developed. The system addresses relay selection and downlink scheduling by integrating reinforcement learning with matching theory, achieving robust decisions in uncertain environments.

A. LLMS FOR NETWORKS

In [21], the authors explore the role of large-scale AI models in advancing sixth-generation (6G) wireless networks. The study discusses the opportunities these models present, including their ability to enable novel applications and improve network performance, while also identifying key technical and operational obstacles. Suggestions for future research are provided to enhance the integration of AI models within 6G ecosystems. The work in [22] introduces a framework that leverages large language models (LLMs) to empower autonomous edge computing for connected intelligence. This research highlights the capability of LLMs to enable edge devices to manage complex tasks with minimal human involvement, focusing on their usability in distributed systems and environments with constrained resources. In [23], the authors propose a system called LLMind, designed to merge the power of LLMs with IoT infrastructures to address intricate tasks. The paper examines orchestration strategies that align the abilities of LLMs with IoT data processing needs, emphasizing challenges related to scalability, real-time operations, and task adaptability. The authors of [24] propose a new approach for using LLMs to enhance multi-agent systems within the context of 6G communication networks. This work outlines how LLMs can improve collaboration, decision-making, and dynamic adaptability in multi-agent environments, with a particular focus on optimizing communication efficiency and task coordination. Similarly, [25] presents LAMBO, a framework that incorporates LLMs into distributed edge intelligence systems. The study examines how LLMs can be deployed on edge devices to offer intelligent services despite resource limitations. The authors address challenges such as resource management, latency reduction, and task-specific optimization. The application of LLMs within 6G edge computing environments is further analyzed in [26]. This work offers a comprehensive review of the opportunities, challenges, and vision for deploying LLMs at the edge, tackling issues such as computational limitations, data privacy, and real-time processing within decentralized networks. The Cached Model-as-a-Resource paradigm is introduced in [27], which provisions LLM agents for edge intelligence in space-air-ground integrated networks. The framework leverages cached LLMs to enhance computational efficiency and optimize resource allocation in dynamic and heterogeneous environments. Key challenges addressed include reducing latency, improving scalability, and enabling task-specific adaptability, thereby laying the groundwork for more robust edge applications. The potential and future applications of Large Language Models (LLMs) within the networking domain are thoroughly discussed in [28], where the authors outline innovative solutions and propose directions for future research. Additionally, [29] emphasizes the significance of multi-modal LLMs—foundation models designed to handle diverse downstream tasks. The paper stresses the need to rethink LLM architectures to accommodate distributed systems, particularly for managing computer vision traffic. Distributed training architectures are explored to address the challenges arising from communication-intensive training paradigms. In [30], a systematic review examines the use of LLMs in anomaly detection and forecasting, with particular attention to the challenges posed by time series data, such as seasonal patterns and limited dataset availability. The review also evaluates major LLM models and their capabilities in predicting rare events. Similarly, [31] presents a comprehensive survey on integrating LLMs with edge intelligence, delving into architectures, applications, and issues of security and trustworthiness for LLM-based systems deployed at the edge. Time series forecasting with LLMs is further explored in [32], focusing on designing effective prompts to enable LLMs to determine periodic patterns within datasets. The study uses the LLMTime model with GPT-3.5-Turbo, GPT-4-Turbo, and LLaMA-2, applying these models to open-source datasets such as AirPassengersDataset, AusBeerDataset, GasRateCO2Dataset, and others. In [33], the combination of hybrid Convolutional Neural Networks (CNNs) and LLMs is surveyed for Intrusion Detection Systems (IDS) in sensor-based environments. The paper highlights how CNNs can extract spatial features while LLMs capture contextual relationships, resulting in improved detection performance. A systematic review of LLM applications in cybersecurity is presented in [34], where the authors explore tasks such as threat detection, malware analysis, incident response, and phishing prevention. The review highlights challenges including the need for large, high-quality datasets, enhancing model interpretability, and adapting to evolving threats. The use of LLMs for optical network log analysis is investigated in [35], with a focus on the LLaMA-2 model enhanced through instruction tuning. The study demonstrates how this approach facilitates effective log interpretation, anomaly detection, troubleshooting, and network performance optimization. Finally, [36] examines the application of LLMs for optimizing wireless access point placement and quantity, comparing the results with Ant Colony Optimization. The findings confirm the potential of LLMs to deliver robust and efficient solutions for wireless network design. While the above studies provide valuable insights into the integration of LLMs within networking and edge computing, most of them focus on enabling intelligent behavior at the application level [21], [22], [23], [24], [25], or present high-level architectural visions for future 6G systems [26], [27], [28]. These contributions emphasize the potential of LLMs to enhance network services and intelligent coordination but do not address the concrete problem of how to deploy LLM inference workloads in a distributed manner. In particular, they typically do not model the fine-grained layer-wise execution constraints, nor the communication overhead introduced by model parallelism in heterogeneous networks. Other studies investigate LLMs in contexts such as anomaly detection [30], [34], intrusion detection systems [33], time series forecasting [32], and log analysis [35], often leveraging LLMs as general-purpose predictors or feature extractors. While relevant, these works do not consider the deployment side of LLMs at the system level. A few papers explore distributed training or inference architectures [29], [31], but they do not formalize the problem using optimization or game-theoretic techniques. In contrast, our work introduces a formal matching-based framework to address the deployment of LLM inference layers onto edge devices, taking into account heterogeneity, inter-layer dependencies, and communication latency. To the best of our knowledge, this is the first attempt to model such a problem using matching games with externalities, enabling stable and interpretable allocations that are amenable to practical planning under real-world constraints.

B. DISTRIBUTED LLMS

The work in [11] investigates the decentralized training of foundation models within heterogeneous environments, where computational nodes vary significantly in capacity and connectivity. The proposed framework enables collaborative training by partitioning the training process into manageable segments distributed across nodes with diverse resources. To address the inherent challenges of heterogeneity, the study introduces adaptive resource allocation methods and gradient synchronization techniques that reduce bottlenecks caused by disparities in node performance. The results demonstrate how decentralized training can accelerate model development while lowering the reliance on centralized infrastructures, paving the way for more inclusive and scalable intelligent systems. Paper [15] explores the challenges and solutions for performing distributed inference and fine-tuning of large language models (LLMs) across the Internet. The authors propose a framework that enables collaborative inference, where different segments of the model are processed across geographically distributed nodes. Additionally, the study delves into fine-tuning strategies that incorporate decentralized datasets, allowing for task-specific optimization while preserving data privacy. The work addresses key challenges, including synchronization across nodes, communication efficiency, and adapting LLM architectures to distributed environments, offering insights into scalable LLM deployment for diverse applications. The authors in [14] introduce a novel approach to optimizing LLM inference by leveraging collaborative edge computing. The study focuses on partitioning LLM architectures across multiple edge nodes, enabling efficient utilization of distributed computational resources. By dividing the model into shards and deploying them strategically across a network of edge devices, the framework reduces latency and minimizes bandwidth usage during inference. Key contributions include techniques to balance computational load among nodes and strategies to mitigate communication overhead, making it a viable solution for deploying LLMs in resource-constrained environments while maintaining high performance and scalability. While the above studies present valuable contributions to distributed LLM deployment, their focus is mainly on architectural design and system-level orchestration. For instance, [11] and [15] address collaborative training and inference in heterogeneous or wide-area networks, with emphasis on synchronization protocols and data privacy. The work in [14] proposes a practical sharding mechanism to reduce inference latency by splitting LLMs across multiple edge devices. Although it explicitly considers latency constraints, the allocation of layers do not involve stability guarantees. Furthermore, prior works often assume centralized control or static resource views, which may limit their adaptability in dynamic edge environments with fluctuating loads and heterogeneous capabilities. The deployment of large language models (LLMs) in edge networks is still an emerging research area. Few works address the specific challenges of distributing LLMs across heterogeneous edge nodes. Most existing studies focus on model compression, offloading strategies, or federated learning. However, they often neglect the impact of network constraints, assuming ideal or simplified communication models. This omission leads to solutions that may be computationally efficient but impractical in real-world edge environments where both communication and computation play a crucial role. Additionally, prior works predominantly rely on heuristic-based placement strategies or centralized optimization approaches, which lack flexibility and scalability in dynamic edge network scenarios. They fail to provide a structured, adaptive mechanism to balance computation and communication trade-offs. In contrast, our work introduces a matching-theory-based framework that explicitly incorporates both computational and communication constraints in the LLM deployment process. By formulating the problem as a two-sided matching game, we offer a distributed and scalable solution that dynamically optimizes LLM allocation while ensuring fairness among edge nodes. This approach differentiates our work from existing heuristics and optimization-based methods, making it more adaptable to real-world edge AI deployments.

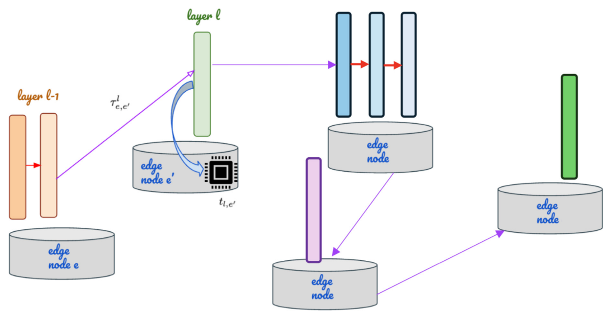


FIGURE 1. System model.

III. PROBLEM STATEMENT

As illustrated in Fig. 1, we consider a set $\mathcal{L} = \{1, \dots, L\}$ of L layers. Each layer l, $\forall l = 1, \dots, L$, consumes m_l memory. We consider a set of edge nodes $\mathcal{E} = \{1, \dots, e, \dots, E\}$, where c_e represents the memory capacity of node e. The residual available capacity of node e can be expressed as

$$\mathcal{S}_e = c_e - \sum_{l \in \{1,\dots,L\}} m_l \alpha_{l,e}, \tag{1}$$

where $\alpha_{l,e}$ is a binary variable equal to 1 when u is served by the LLM placed on e, zero otherwise. Each layer l occupies m_l storage to be processed by the edge node e, and such computation requires a time $t_{l,e}$ to process the layer l on the edge node e. Edge nodes are connected by high-speed links, and the rate of the link between the edge nodes e and e' is denoted with $R_{e,e'}$. Consequently, the transmission time results to be

$$\tau^l_{e,e'} = \frac{s_l}{R_{e,e'}}, \tag{2}$$

in which s_l expresses the size of the intermediate value produced by layer l. Note that when successive layers are hosted on the same edge node, the transmission time is zero.

Therefore, the inference latency results to be [14]

$$\mathcal{I} = \sum_{l \in \mathcal{L}} \sum_{e \in \mathcal{E}} t_{l,e} \alpha_{l,e} + \sum_{l \in \mathcal{L}} \sum_{e \in \mathcal{E}} \sum_{e' \in \mathcal{E}} \alpha_{l,e} \alpha_{l+1,e'} \tau^l_{e,e'}. \tag{3}$$

The proposed planning model assumes that node resources are released upon completion of layer computation. Accordingly, the execution time on each node accumulates the durations of all previously scheduled layers. While this allows resources to be reused, the model also captures potential idle periods, referred to as bubble time [14], arising when a node must wait for data from a preceding layer computed on a different device. These dependencies are modeled through inter-layer externalities. Although this abstraction omits fine-grained tensor-level dynamics, it offers a tractable and effective way to capture key scheduling constraints in distributed inference.

The main objective of this paper is to produce an allocation matrix $\boldsymbol{A} \in \{0, 1\}^{L \times E}$, whose generic element is $\alpha_{l,e}$, capable of minimizing the inference latency of the LLM. In formal terms we have

$$\min_{\boldsymbol{A}} \ \mathcal{I}, \tag{4}$$

$$s.t. \ \sum_{e\in\mathcal{E}} \alpha_{l,e} = 1, \quad \forall l \in \mathcal{L}, \tag{5}$$

$$\sum_{l\in\mathcal{L}} m_l \alpha_{l,e} \le c_e, \quad \forall e \in \mathcal{E}. \tag{6}$$

Constraint (5) imposes that each layer can be hosted by one and only one edge node, and condition (6) expresses that the allocated layers cannot exceed the total memory capacity of the corresponding edge node.

The problem expressed by (4)–(6) has an NP-hard complexity. The theorem can be proved by reduction. In fact, the problem formulated can be cats to the 0-1 knapsack problem [37]. The objective function can be equivalently expressed as $\max \mathcal{I}^{-1}$, subject to $\sum_{l\in\mathcal{L}} m_l\alpha_{l,e} \le c_e$. Since $\alpha_{l,e} \in \{0, 1\}$, by mapping c_e in the knapsack capacity parameter, and $\mathcal{I}^{-1}$ and m_l in the weight and volume of the generic item, respectively, we obtain the formulation of the 0-1 knapsack problem [37].

Consequently, a distributed and sub-optimal strategy must be devised to provide an appropriate solution to the problem formulated.

IV. A MATCHING GAME FOR DISTRIBUTED LLM INFERENCE

A. MATCHING GAME

Unlike centralized optimization, which requires global coordination and can become computationally intractable, matching theory enables a distributed decision-making process where each agent autonomously selects the best option while still ensuring system-wide stability. This approach is particularly suited to LLM inference deployment, as it allows inference tasks to express preferences over edge nodes based on computational power and communication latency, while edge nodes prioritize tasks according to their resource availability. By incorporating these mutual preferences, matching ensures a fair and stable allocation where no inference task or edge node has an incentive to deviate from the assigned deployment, eliminating the need for extensive global coordination. The adoption of a matching game formulation is motivated by the intrinsic two-sided nature of the layer-to-node allocation problem. In this setting, both LLM layers and edge servers express preferences: layers aim to minimize inference latency and avoid resource bottlenecks, while edge nodes seek assignments that align with their current computational and communication capabilities. Matching theory offers a principled and efficient framework to handle these mutual preferences, ensuring stable allocations without the need for global coordination or repeated training. Compared to auction-based or learning-based methods, this approach is lightweight, interpretable, and naturally robust to the heterogeneity and dynamics of edge networks. We formulate a matching game between the set of layers, i.e., $\mathcal{L}$, and the edge nodes, i.e., $\mathcal{E}$. Matching theory provides a quantitative framework for decision-making, aiming to establish mutually advantageous relationships between elements from two distinct sets, based on their respective preferences. The foundation of matching theory lies in preference lists, which quantify the satisfaction level of each element when paired with a counterpart from the opposite set. In this context, each element in $\mathcal{L}$ assigns a ranking to the elements of $\mathcal{E}$, reflecting its preference for being processed on a specific edge node, while the elements in $\mathcal{E}$ similarly rank the layers based on their own preferences.

Each layer l constructs its preference list $\mathcal{V}_l(\cdot)$ by ranking each edge node e in ascending order according to

$$\mathcal{V}_l(e) = t_{l,e} + \beta_{l,e}, \tag{7}$$

where $\beta_{l,e}$ models the bubble time, expressed as the waiting time to receive the intermediate output values of previous layers. Specifically, the term $\beta_{l,e}$ is defined as

$$\beta_{l,e} = \sum_{d\in\mathcal{L}'}\sum_{e\in\mathcal{E}} t_{d,e}\alpha_{d,e} + \sum_{d\in\mathcal{L}'}\sum_{e\in\mathcal{E}}\sum_{e'\in\mathcal{E}} \alpha_{d,e}\alpha_{d+1,e'}\tau^{d}_{e,e'}, \tag{8}$$

where $\mathcal{L}' = \{1, \dots, l-1\}$. As a consequence, the first choice for the l-th layer is the node inducing the lowest processing time. On the other hand, the edge node preference lists, denoted with $\mathcal{W}_e(l)$, are built in accordance with

$$\mathcal{W}_e(l) = \tau^{l-1}_{e^\star,e}, \tag{9}$$

where $e^\star$ is the edge node which hosts the $(l-1)$-th layer. Sorting equation (9) in ascending order, edge nodes select the layer with minimum transmission time.

The proposed matching algorithm operates as follows.

1) Each unallocated layer creates its preference list following (7);
2) Each unallocated layer proposes to be hosted on its favorite edge node.
3) Each edge node creates its preference list accordingly to (9).
4) Each edge node selects its most favorite layer among the proposals received and rejects the others.
5) Repeat 2)-5) until all the layers are allocated or resources are available.

The pseudocode of the matching game is presented in Algorithm 1.

B. GAME STABILITY

The proposed algorithm assumes that preference lists are dynamically updated after each iteration to maintain consistency between the algorithm's decisions and the current state of the system, such as the residual available resources and the queue length on each edge node. These updates introduce dependencies among the players' preferences, significantly complicating the stability of the game, and resulting in a non-trivial challenge in this class of matching problems.

To ensure that the proposed approach converges to a stable matching outcome, denoted as $\mathcal{M}$, the concept of a two-sided exchange stable matching (S2ES) is revisited [38].

Algorithm 1 Matching Game

```
1:  for each layer $l$ in $\mathcal{L}$ do
2:      Create the preference list
3:  end for
4:  for each edge node $e$ in $\mathcal{E}$ do
5:      Create the preference
6:  end for
7:  for each layer $l$ do
8:      Send proposal to the most preferred node
9:  end for
10: for each edge node $e$ receiving at least one proposal do
11:     Accept the favorite layer among those proposing, reject the others
12: end for
```

Definition 1: A final matching $\mathcal{M}$ is an S2ES matching if no pair of layers (l_1, l_2) exists s.t.:

1) $\mathcal{V}_{l_1}(\mathcal{M}(l_2)) \leq \mathcal{V}_{l_1}(\mathcal{M}(l_1))$ and
2) $\mathcal{V}_{l_2}(\mathcal{M}(l_1)) \leq \mathcal{V}_{l_2}(\mathcal{M}(l_2))$ and
3) $\mathcal{W}_{\mathcal{M}(l_1)}(l_2) \leq \mathcal{W}_{\mathcal{M}(l_1)}(l_1)$ and
4) $\mathcal{W}_{\mathcal{M}(l_2)}(l_1) \leq \mathcal{W}_{\mathcal{M}(l_2)}(l_2)$ and
5) $\exists \psi \in \{l_1, l_2, \mathcal{M}(l_1), \mathcal{M}(l_2)\}$ s.t. at least one of the conditions $1) - 4)$ is strictly verified.

Theorem 1: The proposed matching algorithm converges to a final S2ES configuration.

Proof: We prove stability by contradiction. Assume there exists a deviation (l_1, l_2) such that the matching outcome $\mathcal{M}$ is not S2ES. This implies that at least one of the conditions in Definition 1 is strictly violated, meaning that at least one layer or edge node could improve its allocation by switching. However, we observe that the monotonicity of the resource consumption process ensures that at each iteration, the available resources of edge nodes decrease or remain unchanged as layers are allocated step by step. Additionally, since the system does not drop already allocated requests, the waiting delay β_l for any layer l can only remain the same or worsen. Now, consider a target layer l allocated to an edge node e at iteration k. If l wants to deviate to another edge node e', it would need to satisfy conditions 1)-4) in Definition 1, meaning that both l and e' must strictly benefit from the deviation. However, since edge resources are non-increasing and waiting time is non-decreasing, any deviation would lead to at most the same or a worse allocation, contradicting the assumption of a beneficial deviation. A similar argument applies to edge nodes: once an edge node accepts a layer, its available resources are reduced, making it impossible to later regret a previous matching and switch to a different configuration that satisfies S2ES conditions. Therefore, condition 5) in Definition 1 is not satisfied for any deviation, ensuring that the final matching $\mathcal{M}$ is stable. ■

Theorem 1 establishes the stability of the proposed matching algorithm, ensuring that the system reaches a S2ES configuration where no further reallocations occur. This result is fundamental in the context of edge resource allocation, as it guarantees that the system does not experience perpetual oscillations or inefficient reassignment cycles. Stability is crucial for maintaining predictable resource utilization, minimizing communication overhead, and ensuring consistent service quality across edge nodes. This proof formalizes the convergence of the algorithm and highlights how the iterative reduction of available resources enforces stability in a distributed manner. Unlike classical centralized optimization methods that require global coordination, this approach guarantees convergence using only local preference updates, making it highly scalable and well-suited to edge AI deployments.

The same reasoning applies to edge nodes, ensuring that condition 5) of Definition 1 is not satisfied for any deviation. Consequently, the outcome matching $\mathcal{M}$ is proven to be stable.

C. COMPLEXITY ANALYSIS

To analyze the computational complexity of the proposed framework under the worst-case scenario, we focus on the situation where every edge node is capable of supporting all layers. Each layer must sort the elements of the set $\mathcal{E}$ based on the associated preference list metric. Consequently, the time complexity for this sorting operation per layer is $O(E \log E)$. Considering all the layers in $\mathcal{L}$ we have a complexity of $O(LE \log E)$. Moreover, the computational complexity associated with building the preference lists for the edge nodes within the set $\mathcal{E}$ is $O(EL \log L)$. Since typically $E << L$ and the algorithm terminates in L steps, the total computational complexity results to be driven by

$$O\Big(EL \log L\Big). \tag{10}$$

Regarding termination, we observe that the proposed algorithm is guaranteed to reach a fixed point in a finite number of steps. In the worst-case scenario, where all LLM layers initially target the same edge node (assumed to have unbounded capacity), each node accepts at most one proposal per iteration, and the algorithm terminates in at most as many rounds as the number of LLM layers. From a deployment perspective, the main scalability bottleneck is the communication bandwidth required to support the exchange of large model weights and activations. LLM inference involves large model weights and activations, which require high-throughput interconnects to avoid excessive transfer delays. Without sufficient bandwidth, the benefits of layer-wise distributed inference can be offset by communication overheads. Importantly, the primary computational bottleneck arises when scaling to larger LLMs, i.e., increasing the number of layers. In this case, distributed inference becomes more appealing and potentially necessary, but also more challenging to orchestrate efficiently. As model sizes continue to grow, the trade-off between centralized and distributed inference becomes increasingly relevant. While centralized execution may avoid communication overhead, it often lacks the computational scalability and geographic proximity required by latency- sensitive applications. In contrast, distributed inference enables layer-level parallelism and load balancing, but introduces additional challenges in scheduling, synchronization, and inter-node communication. A comprehensive analysis of these trade-offs, especially under varying model sizes and network constraints, represents a promising direction for future research.

V. PERFORMANCE EVALUATION

This section presents both results obtained through extensive numerical simulations and a small-scale testbed. The performance of the solution proposed are first investigated through extensive simulations to assess the decision-making capability of the matching game developed in solving the LLM layer deployment problem, with comparisons to alternative schemes. Then, a small-scale real-world testbed is discussed to illustrate the behavior of the proposed strategy in action.

A. SIMULATED PERFORMANCE ANALYSIS

To perform simulation, we consider an LLM architecture structured as a sequential stack of Transformer layers, where each layer is uniquely identified by its position within the sequence. This sequential organization ensures that each layer maintains its role in the model's hierarchical representation learning process. We further characterize each layer based on computational cost and memory footprint as follows. Based on empirical studies on LLM [14], [39], [40], [41], as a reference scenario, we consider a heterogeneous edge node network, where $\tau^{l}_{e,e'}$ and $t_{l,e}$ are uniformly selected within $[4, 8]$ ms and $[3, 20]$ ms, respectively. Then, we set a number of edge nodes uniformly distributed within the interval $[5, 12]$, considering a number of LLM layers equal to 32. Furthermore, we consider m_l as uniformly distributed within $[1.1, 2.2]$ GB, whereas c_e is uniformly distributed in $[4, 32]$ GB. Each simulation is repeated over 1,000 independent runs, and results are averaged to ensure statistical reliability. All simulations were conducted on a system powered by an Apple M1 Pro CPU with 32 GB of RAM, providing a consistent and reliable computational environment. All experiments are conducted in a controlled environment using Python 3.10 and NumPy, with custom logic implemented for matching dynamics and latency computation.

To analyze the performance of the proposed matching approach (PMA), we implement the following additional decision-making schemes for comparison purposes

- *Kolkata Game:* the Kolkata repeated game, as described in [42], is a matching game that relies on one-sided preferences, meaning only one party in the game specifies preferences for the other. Similar to the PMA, in the Kolkata framework, layers express their preferences based on (7). After layers submit their allocation requests to edge nodes, each node randomly selects its preferred layer from the pool of proposals it receives.
- *Random Selection:* each user selects an edge node for computation by making a random choice based on a uniform distribution.
- *First-bid Auction Game:* layers express their bids based on $1/\mathcal{V}_l(e)$, and nodes accept the highest bid among those received.

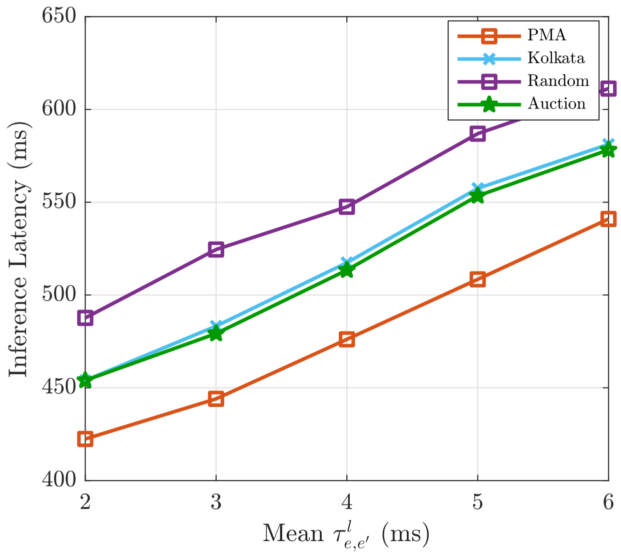


FIGURE 2. Inference Latency as a function of the mean transmission time, assuming a sequential architecture.

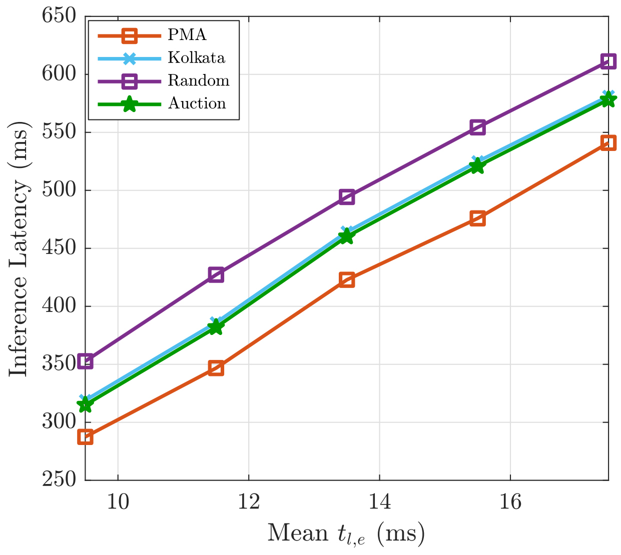


FIGURE 3. Inference Latency as a function of the mean processing time, assuming a sequential architecture.

Assuming a sequential processing architecture, Figure 2 illustrates the inference latency as a function of the mean transmission time per layer. As it is evident, the inference latency increases as the transmission time grows. Furthermore, the PMA outperforms the alternative strategies here considered for comparison. In particular, the Kolkata algorithm does not reach the inference values achieved by

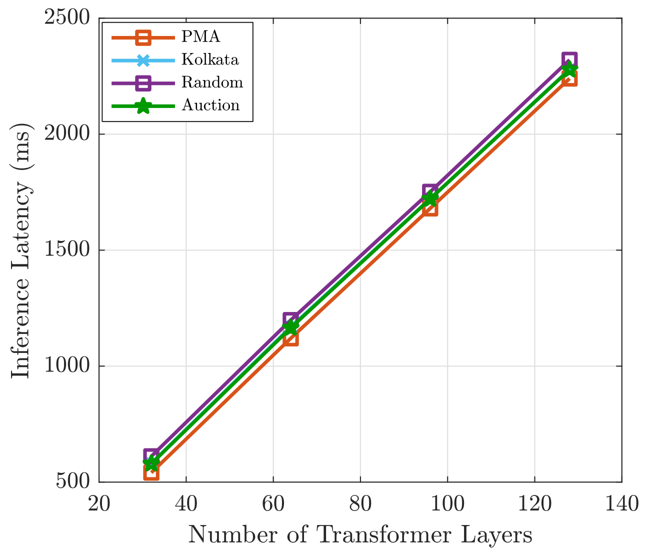


FIGURE 4. Inference Latency as a function of the transformer layer, assuming a sequential architecture.

the PMA due to the random component that rules the choices provided by edge nodes. Similarly, in Figure 3 the inference latency is expressed as a function of the processing time. Comparison with alternative algorithms confirms the superiority of the PMA in performing decision-making, however, we can also appreciate the impact of the processing time on the overall inference. Specifically, reducing the processing time significantly improves the overall inference time. In addition, Figure 4 exhibits the behavior of the inference latency as the number of layers increases. In our experimental setup, we consider a variable number of layers ranging from 30 to 120, reflecting different deployment scenarios in heterogeneous edge networks. This range encompasses both moderate-scale models and deeper architectures, allowing us to evaluate the impact of layer granularity on system performance. For reference, LLaMA-7B, a model widely used in various edge and cloud-based applications, consists of 32 Transformer layers. Our selected range extends beyond this, covering scenarios relevant to scalability analysis in edge infrastructures. By varying the number of layers, we analyze the trade-offs between latency, resource utilization, and inference efficiency, investigating adaptability across different edge computing environments. High numbers of transformer layers clearly lead to a degradation in inference performance, especially in a sequential architecture. Nevertheless, the PMA manages to ensure reduced inference times compared to other approaches.

B. TESTBED

To evaluate our algorithm in a real-world context, we developed a small-scale testbed (Figure 5) consisting of four edge nodes with the following configurations: **A**. an Intel Core i7-12700K paired with an NVIDIA RTX 3090, **B** Ryzen 7 3700X, **C** an Intel Core i7-9700K, and **E** an AMD Ryzen Threadripper 1950X-16. Components **F** and **D** are the network element and the UPS, respectively. We selected

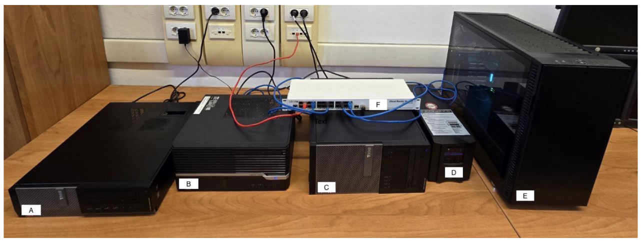


FIGURE 5. Heterogeneous testbed.

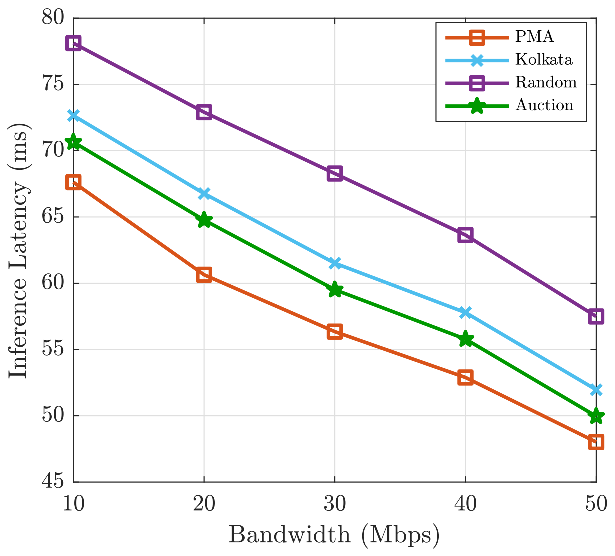


FIGURE 6. Inference Latency as a function of bandwidth, assuming a sequential architecture.

LLaMA-7B as the LLM for our experiments due to its lightweight architecture, making it suitable for deployment in resource-constrained environments. Inspired by recent works [43], [44], we have applied the LLM to perform decision-making actions for autonomous driving support. The LLM interacts with a dynamic environment using multi-modal, multi-view sensor data and natural language instructions. We exploited a dataset of data clips, where each clip includes one navigation instruction, several notice instructions, a sequence of multi-modal, multi-view sensor data, and control signals. Experimentation was performed in CARLA-Leaderboard. The inference was tested distributing LLaMA-7B among four edge nodes. In so doing, the data batches were parallelized, obtaining the performance illustrated in Figure 6. Figure 6 exhibits the inference latency ruling the bandwidth of the connection among edge nodes. Also in this use-case where the computation is parallelized, the PMA achieves better performance in comparison to both the Kolkata and Random alternatives.

VI. CONCLUSION

This paper addressed the problem of the LLM layer deployment across an edge network, heterogeneous in both computational and communication resources. The framework developed is based on the matching theory principles, and it is devoted to minimizing the inference latency of the distributed LLM. The game formulation presents interdependencies among players' preferences, meaning that the stability is not trivial to prove. Performance evaluation is provided in comparison to alternative state-of-the-art algorithmic approaches. Finally, a small-scale testbed is proposed to cast the solution designed to a real-world case study, where the LLM exploits its ability in understanding NLP to support next-generation autonomous driving scenarios.

REFERENCES

[1] R. Bommasani et al., "On the opportunities and risks of foundation models," 2021, *arXiv:2108.07258*.

[2] S. Yang, O. Nachum, Y. Du, J. Wei, P. Abbeel, and D. Schuurmans, "Foundation models for decision making: Problems, methods, and opportunities," 2023, *arXiv:2303.04129*.

[3] B. Xiao, B. Kantarci, J. Kang, D. Niyato, and M. Guizani, "Efficient prompting for LLM-based generative Internet of Things," *IEEE Internet Things J.*, vol. 12, no. 1, pp. 778–791, Jan. 2025.

[4] Y. Qu, M. Ding, N. Sun, K. Thilakarathna, T. Zhu, and D. Niyato, "The frontier of data erasure: A survey on machine unlearning for large language models," *Computer*, vol. 58, no. 1, pp. 45–57, Jan. 2025.

[5] J. Wen et al., "Generative AI for low-carbon artificial intelligence of things with large language models," *IEEE Internet Things Mag.*, vol. 8, no. 1, pp. 82–91, Jan. 2025.

[6] S. Zhang et al., "Large models for aerial edges: An edge-cloud model evolution and communication paradigm," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 1, pp. 21–35, Jan. 2025.

[7] G. Sun et al., "Large language model (LLM)-enabled graphs in dynamic networking," *IEEE Netw.*, early access, Dec. 5, 2024, doi: 10.1109/MNET.2024.3511662.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," in *Proc. North Amer. Chapter Assoc. Comput. Linguist.*, 2019, pp. 1–16. [Online]. Available: https://api.semanticscholar.org/CorpusID:52967399

[9] T. Brown et al., "Language models are few shot learners," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 1877–1901.

[10] Y. Lin et al., "Mitigating the alignment tax of RLHF," 2024, *arXiv:2309.06256*.

[11] B. Yuan et al., "Decentralized training of foundation models in heterogeneous environments," 2023, *arXiv:2206.01288*.

[12] Y. Yao, J. Duan, K. Xu, Y. Cai, Z. Sun, and Y. Zhang, "A survey on large language model (LLM) security and privacy: The good, the bad, and the ugly," *High-Confid. Comput.*, vol. 4, Jun. 2024, Art. no. 100211.

[13] W. Zhao, W. Jing, Z. Lu, and X. Wen, "Edge and terminal cooperation enabled LLM deployment optimization in wireless network," in *Proc. IEEE/CIC ICCC Wkshps*, 2024, pp. 220–225.

[14] M. Zhang, X. Shen, J. Cao, Z. Cui, and S. Jiang, "EdgeShard: Efficient LLM inference via collaborative edge computing," *IEEE Internet Things J.*, early access, Dec. 31, 2024, doi: 10.1109/JIOT.2024.3524255.

[15] A. Borzunov et al., "Distributed inference and fine-tuning of large language models over the Internet," 2023, *arXiv:2312.08361*.

[16] J. Xu, Y. Yao, X. Xu, W. Feng, and P. Li, "Joint optimization of task offloading and resource allocation of fog network by considering matching externalities and dynamics," *IEEE Trans. Mobile Comput.*, vol. 24, no. 4, pp. 2534–2550, Apr. 2025.

[17] Y. Zhao et al., "Joint content caching, service placement, and task offloading in UAV-enabled mobile edge computing networks," *IEEE J. Sel. Areas Commun.*, vol. 43, no. 1, pp. 51–63, Jan. 2025.

[18] H. Qi et al., "Bridge the present and future: A cross-layer matching game in dynamic cloud-aided mobile edge networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 12522–12539, Dec. 2024.

[19] Y. Xu et al., "Tripartite matching game model in UAV-assisted covert communication network," *IEEE Commun. Lett.*, vol. 28, no. 7, pp. 1619–1623, Jul. 2024.

[20] X. Mi, Y. Song, C. Yang, Z. Han, and C. Yuen, "MAGIC: Matching game-based resource allocation with incomplete information in space communication network," *IEEE Trans. Commun.*, vol. 72, no. 6, pp. 3481–3494, Jun. 2024.

[21] Z. Chen, Z. Zhang, and Z. Yang, "Big AI models for 6G wireless networks: Opportunities, challenges, and research directions," *IEEE Wireless Comm.*, vol. 31, no. 5, pp. 164–172, Oct. 2024.

[22] Y. Shen et al., "Large language models empowered autonomous edge AI for connected intelligence," *IEEE Commun. Mag.*, vol. 62, no. 10, pp. 140–146, Oct. 2024.

[23] H. Cui, Y. Du, Q. Yang, Y. Shao, and S. C. Liew, "LLMind: Orchestrating AI and IoT with LLM for complex task execution," *IEEE Commun. Mag.*, vol. 63, no. 4, pp. 214–220, Apr. 2025.

[24] F. Jiang et al., "Large language model enhanced multi-agent systems for 6G communications," *IEEE Wireless Commun.*, vol. 31, no. 6, pp. 48–55, Dec. 2024.

[25] L. Dong et al., "LAMBO: Large AI model empowered edge intelligence," 2023, *arXiv:2308.15078*.

[26] Z. Lin, G. Qu, Q. Chen, X. Chen, Z. Chen, and K. Huang, "Pushing large language models to the 6G edge: Vision, challenges, and opportunities," 2023, *arXiv:2309.16739*.

[27] M. Xu et al., "Cached model-as-a-resource: Provisioning large language model agents for edge intelligence in space-air-ground integrated networks," 2024, *arXiv:2403.05826*.

[28] C. Liu, X. Xie, X. Zhang, and Y. Cui, "Large language models for networking: Workflow, advances and challenges," 2024, *arXiv:2404.12901*.

[29] J. Du, T. Lin, C. Jiang, Q. Yang, C. Bader, and Z. Han, "Distributed foundation models for multi-modal learning in 6G wireless networks," *IEEE Wireless Commun.*, vol. 31, no. 2, pp. 20–30, Jun. 2024.

[30] J. Su et al., "Large language models for forecasting and anomaly detection: A systematic literature review," 2024, *arXiv:2402.10350*.

[31] O. Friha, M. Amine Ferrag, B. Kantarci, B. Cakmak, A. Ozgun, and N. Ghoualmi-Zine, "LLM-based edge intelligence: A comprehensive survey on architectures, applications, security and trustworthiness," *IEEE Open J. Commun. Soc.*, vol. 5, pp. 5799–5856, 2024.

[32] H. Tang et al., "Time series forecasting with LLMs: Understanding and enhancing model capabilities," 2024, *arXiv:2402.10835*.

[33] S. Elouardi, A. Motii, M. Jouhari, A. N. H. Amadou, and M. Hedabou, "A survey on hybrid-CNN and LLMs for intrusion detection systems: Recent IoT datasets," *IEEE Access*, vol. 12, pp. 180009–180033, 2024.

[34] I. Hasanov, S. Virtanen, A. Hakkala, and J. Isoaho, "Application of large language models in cybersecurity: A systematic literature review," *IEEE Access*, vol. 12, pp. 176751–176778, 2024.

[35] Y. Pang et al., "Large language model-based optical network log analysis using LLaMA2 with instruction tuning," *J. Opt. Commun. Netw.*, vol. 16, no. 11, pp. 1116–1132, 2024.

[36] K. Qiu, S. Bakirtzis, I. Wassell, H. Song, J. Zhang, and K. Wang, "Large language model-based wireless network design," *IEEE Wireless Commun. Lett.*, vol. 13, no. 12, pp. 3340–3344, Dec. 2024.

[37] V. Vazirani, *Queueing Systems*, vol. 1. Berlin, Germany, 2013.

[38] E. Bodine-Baron, C. Lee, A. Chong, B. Hassibi, and A. Wierman, "Peer effects and stability in matching markets," in *Proc. SAGT*, 2011, pp. 117–129.

[39] S. Fakhoury, A. Naik, G. Sakkas, S. Chakraborty, and S. K. Lahiri, "LLM-based test-driven interactive code generation: User study and empirical evaluation," *IEEE Trans. Softw. Eng.*, vol. 50, no. 9, pp. 2254–2268, Sep. 2024.

[40] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software testing with large language models: Survey, landscape, and vision," *IEEE Trans. Softw. Eng.*, vol. 50, no. 4, pp. 911–936, Apr. 2024.

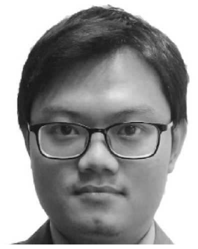
[41] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "An empirical evaluation of using large language models for automated unit test generation," *IEEE Trans. Softw. Eng.*, vol. 50, no. 1, pp. 85–105, Jan. 2024.

[42] B. K. Chakrabarti, "Kolkata restaurant problem as a generalised EL Farol bar problem," in *Econophysics of Markets and Business Networks*. Milan, Italy: Springer, 2007, pp. 239–246.

[43] H. Shao, Y. Hu, L. Wang, S. L. Waslander, Y. Liu, and H. Li, "LMDrive: Closed-loop end-to-end driving with large language models," 2023, *arXiv:2312.07488*.

[44] K. Renz et al., "CarLLaVA: Vision language models for camera-only closed-loop driving," 2024, *arXiv:2406.10165*.

BENEDETTA PICANO (Member, IEEE) received the M.E. degree in computer engineering and the Ph.D. degree in information engineering from the University of Florence, Italy, in 2016 and 2020, respectively. She was a Visiting Scholar with Houston University, Houston, USA. She was a Visiting Researcher at the University of Pisa, Italy. Her research interests include computer networking, wireless communications, and machine learning application, with emphasis on system performance optimization and generation of multimodal trajectories with generative flow networks. She served on the Editorial Boards for the IEEE TRANSACTIONS ON VEHICULAR TECHNOLOGY and *Peer-to-peer Networking and Applications* journal. She was a Guest Editor of the Special Issue: Applications of Internet of Things Networks in 5G and Beyond, in Sensors. Finally, she was a Workshop Program Chair for ISSRE 2023, and a Co-Chair of Track 11: Wireless Networks: Protocols, Security and Services, for IEEE Vehicular Technology Conference 2024.



DINH THAI HOANG (Member, IEEE) received the Ph.D. degree in computer science and engineering from Nanyang Technological University, Singapore, in 2016. He is currently a Faculty Member with the School of Electrical and Data Engineering, University of Technology Sydney, Australia. His research interests include emerging topics in wireless communications and networking, such as ambient backscatter communications, vehicular communications, cybersecurity, the Internet of Things, and 5G networks. He is also an Exemplary Reviewer of the IEEE TRANSACTIONS ON COMMUNICATIONS, in 2018, and the IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, in 2017 and 2018, respectively. He is also an Editor of IEEE WIRELESS COMMUNICATIONS LETTERS and the IEEE TRANSACTIONS ON COGNITIVE COMMUNICATIONS AND NETWORKING.

DIEP N. NGUYEN (Senior Member, IEEE) received the M.E. degree in electrical and computer engineering from the University of California at San Diego (UCSD) and the Ph.D. degree in electrical and computer engineering from The University of Arizona. He was a DECRA Research Fellow with Macquarie University and a member of Technical Staff with Broadcom, CA, USA, ARCON Corporation, Boston, consulting the Federal Administration of Aviation, on turning detection of UAVs and aircraft, and the U.S. Air Force Research Laboratory, on anti-jamming. He is currently a Faculty Member with the Faculty of Engineering and Information Technology, University of Technology Sydney. His recent research interests include computer networking, wireless communications, and machine learning application, with emphasis on systems' performance and security/privacy. He has received several awards from LG Electronics, UCSD, The University of Arizona, the U.S. National Science Foundation, and the Australian Research Council.