Joint Communication and Inference User Allocation in LLM Native Networks

Benedetta Picano
Dept. of Information Engineering
University of Florence, Italy
benedetta.picano@unifi.it

Alessandro Buratto
Dept. of Information Engineering
University of Padova, Italy
alessandro.buratto.1@phd.unipd.it

Leonardo Badia
Dept. of Information Engineering
University of Padova, Italy
leonardo.badia@unipd.it

***Abstract*—Large language models (LLMs) are game changers for future next-generation networks, unlocking new opportunities for disruptive and interactive services and applications. Edge computing enables deployment of LLMs closer to the users, allowing for the implementation of highly responsive intelligent systems. This paper proposes a matching theory-based algorithm to optimize the user-LLM association and considers both the communication and inference delay, in the presence of capacity-constrained edge nodes. The objective is to minimize end-to-end user delay, that is, the time elapsed between when a user submits a request and when the response is sent back. Therefore, a matching game is formulated between the users and the LLMs, assuming heterogeneous LLMs, specialized in different types of learning tasks. The scenario is modeled as a matching game with externalities and incomplete lists, which terminates in a stable configuration, leveraging monotonic user preference list metric, within the algorithm execution. A comparative performance evaluation against different state-of-the-art techniques confirms the advantages of adopting a joint communication and inference aware approach to orchestrate the user-LLM assignments.**

***Index Terms*—Distributed computing, large language models, generative AI as a service, matching game, edge computing.**

I. Introduction

In recent times, large language models (LLMs) have drawn significant attention for their exceptional ability to understand and produce natural language and media. From a technological point of view, LLMs are based on deep neural networks and self-supervised learning, both of which have been established for decades. However, what distinguishes recent LLMs is their unprecedented scale and versatility, which have reshaped their applications and broadened our understanding of their potential [1], [2]. Thanks to their extensive training process, based on billions of parameters and Internet-scale datasets, these models can handle previously unseen data without additional training, adopting a zero-shot learning approach [3], or can be further refined using smaller, task-specific datasets to improve their performance for particular applications, thus adhering to the fine-tuning paradigm [4]. Prominent examples, such as ChatGPT and Bard, have transformed the way artificial intelligence (AI) approaches natural language processing. These models demonstrate advanced understanding, ability to generate human-like text, contextual adaptability, and strong problem-solving capabilities. These features, together with advances in communication technologies that provide reliable high-rate links [5], permit, in perspective, LLMs to proactively assist and engage users, providing support for decision-making or facilitating interactive applications.

Nonetheless, current LLMs are highly dependent on cloud computing, which introduces challenges such as long service latency, significant bandwidth usage, and privacy risks [6], [7]. The intrinsic centralized nature of cloud computing limits the ability to deliver rapid inferences required for real-time applications, e.g., in robotics control or exploration, where instant responses are key [8]. Moreover, the need to transmit vast amounts of multi-modal data to cloud servers results in substantial bandwidth consumption, cost, and network congestion [9]. Edge computing solutions, i.e., deploying LLM on edge nodes closer to data sources, represent a promising way to overcome these limitations [10], [11]. In this picture, the limited resources typical of edge infrastructures must be efficiently managed to ensure low-latency responses and seamless interaction between users and LLMs [12]. Moreover, designing a resource-aware strategy for user-edge node assignments, which is instrumental in establishing appropriate user-LLM pairings, becomes essential [13], [14].

We propose a joint communication-inference allocation strategy to assign users to LLMs. Our framework applies matching theory [15], establishing a matching game with externalities to provide the allocation. The dual structure of matching theory is well suited to formalize the problem, where each user demands a specific learning task to a specialized LLM. The objective is to minimize the user's end-to-end (e2e) delay, i.e., the time elapsed since the submission of the service request until the LLM generates the result and sends it back to the user. Each user can be served by only one LLM, and that LLM must be specialized in the task required by the user. This leads to the formulation of a matching game with incomplete lists [16]. Furthermore, due to the dependencies between the users' preference lists, the resulting matching involves externalities; thus, its stability is not guaranteed applying the algorithms available in the literature [17].

Our contributions can be summarized as follows. (i) Modeling the user-LLM assignment problem, considering both the communication and inference delay, where LMMs are

This work was supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) Mission 4, Component 2, Investment 1.3, CUP C93C22005250001, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART")

deployed at the edge of the network but are specialized in different learning tasks. With respect to the literature, accounting for the LLM inference delays that impact responsiveness and decision-making represents a further original element. (ii) Design and implementation of a matching game with incomplete lists and externalities to provide an assignment with the objective of minimizing the mean e2e user delay. The proposed matching algorithm is stable despite the presence of externalities, thanks to the monotonicity of the user preference list metric over algorithm iterations. (iii) Comparison of the proposed framework with different approaches in terms of e2e delay and dropping rate, i.e., the probability of receiving the service after the deadline associated with each user.

Our comparative analysis shows superior performance over other matching strategies, thus providing efficient and timely task allocation. The remainder of the paper is organized as follows. In Sec. II the related literature is presented, and in Sec. III the problem statement is detailed. Sec. IV describes the matching game designed, and the results are illustrated in Sec. V. Conclusions are drawn in Sec. VI.

II. Related works

In [18], the authors investigate the potential of large-scale AI models in the context of sixth-generation (6G) wireless networks, highlighting how these models can enable innovative applications and enhance network capabilities while addressing critical technical and operational challenges. Another framework in [19] exploits LLM on the autonomous edge for connected intelligence, focusing on how LLMs can enable edge devices to handle complex tasks with minimal human intervention, emphasizing their applicability in distributed and resource-constrained environments.

LLMind is introduced in [20] as a system to integrate LLM with IoT systems, with the objective of solving complex tasks. Orchestration to align LLM capabilities with IoT data is investigated, highlighting challenges in terms of scalability, real-time performance, and task adaptation. In [10], the authors propose the application of LLMs to enhance multi-agent systems for 6G communications, so as to improve task coordination, decision-making, and dynamic adaptability in multi-agent environments. Similarly, the LAMBO framework [21] integrates LLMs into distributed edge intelligence, addressing the deployment of LLMs on resource-constrained edge devices. Therefore, the paper discusses challenges in resource management, latency reduction, and task-specific optimizations. The application of LLMs in 6G edge environments is then explored in [22]. The paper in-depth reviews the vision, challenges, and opportunities arising from pushing LLMs to the edge, addressing key issues such as computational constraints, data privacy, and real-time processing in decentralized networks. In [11], the authors propose a novel Cached Model-as-a-Resource paradigm to provision LLM agents for edge intelligence in space-air-ground integrated networks. The study focuses on leveraging cached LLMs to improve computational efficiency and resource utilization in highly dynamic and heterogeneous environments. The framework addresses critical challenges in latency, scalability, and task-specific adaptability, paving the way for more robust and intelligent edge applications.

Conversely, the problem of assigning edge computing resources efficiently, yet avoiding a full-scale optimization, e.g., with some meta-heuristic approach, is also found in the literature. For instance, [13] considers a distributed Markov approximation algorithm with linear complexity to optimize task allocation in real-time. In [23], a similar problem is solved through particle swarm optimization. Other works solve this distributed optimization assignment task through the use of game theory tools. In [24], the authors use a static game of complete information to assign task on the edge or the cloud.

Instead, we represent the assignment of edge computing resources as a *matching game*. This idea can be found in other papers such as [15], which uses it for resource allocation accounting for capacity constraints and user preferences to optimize edge server selection. However, the main aspect coexisting with the server selection in that paper is the existence of inter-cell interference, which the matching game tries to avoid, by selecting the best association that avoids causing interference, whereas the content of the task itself is atomic. Conversely, in our analysis the main parameter to match is the suitability of the LLM to the specific task, especially including the inference delay, which is harder to characterize.

Similar approaches are also considered in [14] and [25]. The former considers the objective of the matching to be the social welfare, under a general utility function that includes also economic aspects. The latter instead includes, among other terms, the computation time, as we do here, but the main focus is about the uncertainty of the evaluation, which makes the game Bayesian in nature. Instead, our approach assumes knowledge about the e2e user delay, but the main distinguishing aspect, not considered by any of the previous paper, is the inference delay, resulting in different suitability scores of the LLM models to the task requested by the user.

Previous research on LLM deployment at the edge has largely concentrated on model compression, task offloading, or heuristic allocation strategies, often treating communication and inference latency as separate concerns rather than optimizing them jointly. In contrast, our approach introduces multiple innovations, listed below.

Joint Optimization of Communication and Computation. While existing methods focus on either communication or computation efficiency, our matching framework integrates both, ensuring an allocation that minimizes end-to-end delay.

Matching Game with Externalities and Incomplete Preferences. Unlike centralized optimization approaches, our method models the user-LLM association as a matching problem, incorporating externalities and incomplete preference lists to enable a scalable and distributed solution.

Heterogeneous LLM Specialization with Stability Guarantees. Our framework considers LLMs specialized for different tasks, ensuring that the allocation remains stable, meaning that neither users nor edge nodes have an incentive to switch their assignments. This enhances robustness, fairness, and adaptability in dynamic edge environments.

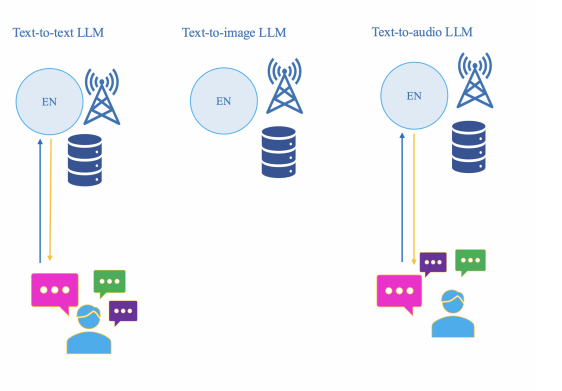

Fig. 1. System Model

III. Problem Statement

As illustrated in Fig. 1, the scenario considers a set of users $\mathcal{U} = \{1, \ldots, u, \ldots, U\}$ and a set of edge nodes $\mathcal{E} = \{1, \ldots, e, \ldots, E\}$. Each user demands learning capabilities γ_u (in tokens) to a set of LLMs deployed on edge nodes. The processing of γ_u is considered successful if completed within the corresponding deadline d_u. As a first step in the user-LLM assignment problem, we assume that each edge node e hosts a single LLM instance. This simplifies the formulation and allows us to focus on optimizing the matching process, while considering both communication and inference latency. In practical deployments, edge nodes may host multiple LLMs, each specialized in different tasks. In such a scenario, users can be served through separate queues for each LLM, parallelizing the inference process and reducing congestion at the edge. Consequently, the set of LLMs is $\mathcal{L} = \{1, ..., e, ..., E\}$. In line with the literature on this topic, we assume that the edge computing nodes coincide with small base stations [26]. Moreover, we consider that each user requests task processing accessing to a synchronous dedicated channel that does not interfere with the access of other users within the same area. Each edge node $e \in \mathcal{E}$, disposes of a total storage capacity Δ_e, such that the residual capacity can be expressed as

$$\mathcal{S}_e = \Delta_e - \sum_{u \in \mathcal{U}} s_u \alpha_{u,e}, \tag{1}$$

where $\alpha_{u,e}$ is a binary variable equal to 1 when u is served by the LLM placed on e, zero otherwise. Each user u occupies s_u storage to be served by the LLM placed on e.

Once γ_u arrives to e, each token is transformed into a vector embedding, i.e., a numerical representation that the model can process to make inferences. These embeddings are then fed into the LLM, which processes them to produce an appropriate output for the user. During the decoding phase, the LLM generates a sequence of vector embeddings that represent its response to the input prompt. These are subsequently converted into output tokens, generated one by one until a stopping condition (e.g., hitting a token limit or encountering a stop word) is met. Then, the model generates an end token to notify the conclusion of token generation. Since LLMs generate one token per iteration, the number of iterations to complete a response corresponds to that of generated output tokens. Therefore, the inference delay is

$$I_{e,u} = \tau_u \iota_e, \tag{2}$$

where τ_u is the number of tokens generated by u and ι_e is the inter-token latency. Similarly, the e2e delay of user u is

$$D_{e,u} = 2 \cdot C_{e,u} + I_{e,u} + Q_{e,u}, \tag{3}$$

where $C_{e,u}$ is the time spent to send the input tokens γ_u to the LLM placed on edge node e. Since we assume an interference-free scenario without contention, the transmission rate is constant, and the communication delay is

$$C_{e,u} = \frac{\gamma_u}{B_e}, \tag{4}$$

where B_e is the allocated bandwidth to edge node e.

In highly congested network environments, simultaneous access of multiple users to the same edge node can lead to fluctuations in latency and increased communication delays. These effects may influence the efficiency of LLM deployment strategies and the overall system responsiveness. Nonetheless, we assume an ideal scenario without contention or interference, allowing us to focus on the core optimization of the user-LLM assignment process. Also, note that we assumed that the output token has the same size as the input token.[1] Furthermore, $Q_{e,u}$ is the queuing delay experienced by u due to users previously allocated on e, which depends on the number of users assigned to the edge node. A higher number of allocated users at node e results in longer queuing times, which in turn affects the overall end-to-end delay. Queues follow a first-in-first-out policy.

Our goal is to produce allocation matrix $\boldsymbol{A} = \{\alpha_{u,e}\}_{u,e} \in \{0,1\}^{U \times E}$, minimizing the mean e2e user delay. Formally,

$$\min_{\boldsymbol{A}} \frac{1}{|\mathcal{U}|} \sum_{e \in \mathcal{E}} \sum_{u \in \mathcal{U}} D_{e,u} \alpha_{u,e}, \tag{5}$$

$$\text{s.t.} \quad \sum_{e \in \mathcal{E}} \alpha_{u,e} = 1, \quad \forall u \in \mathcal{U}, \tag{6}$$

$$\sum_{u \in \mathcal{U}} s_u \alpha_{u,e} \leq \Delta_e \quad \forall e \in \mathcal{E} \tag{7}$$

IV. Matching Theory for Communication-Inference User Allocation

We establish a matching game between the set of users $\mathcal{U}$ and the edge nodes $\mathcal{E}$. Matching theory is a quantitative model of choices whose objective is to build mutually beneficial relationships between the elements belonging to two opposite sets, considering the individual preferences of each element. Matching theory [28] is based on preference lists that describe the level of satisfaction of each element in being matched with each element of the opposite set. Each element of $\mathcal{U}$ ranks

[1] While output token lengths can vary depending on the task, many applications exhibit an input:output ratio of approximately 1. Examples include, but are not limited to, machine translation, and structured NLP tasks like Named Entity Recognition [27]. Thus, assuming an equal number of input and output tokens provides a reasonable approximation for delay estimation.

Algorithm 1 Inference-Aware Matching

Input: Set of devices $\mathcal{U}$, set of edge nodes $\mathcal{E}$
Output: Stable matching $\mathcal{M}$ of devices and edge nodes
// Step 1: Build preference lists
for each device $u \in \mathcal{U}$ **do**
    Construct preference list based on communication and inference latency
for each edge node $e \in \mathcal{E}$ **do**
    Construct preference list based on request deadline
// Step 2: Proposal phase
for each device $u \in \mathcal{U}$ **do**
    Send a proposal to the most preferred edge node
// Step 3: Edge nodes accept proposals
for each edge node $e \in \mathcal{E}$ getting at least one proposal **do**
    Accept the most preferred user among the proposals
    Reject the remaining proposals
// Step 4: Iterate
while some devices remain unassigned **do**
    Unmatched users update preferences
    Unmatched users propose to the preferred edge node
    Edge nodes update assignments based on preferences
return Stable matching $\mathcal{M}$

the elements of $\mathcal{E}$ to denote its individual preference in being computed on each LLM, and vice-versa.

Each user u creates its preference list $\mathcal{V}_u(\cdot)$ ranking in increasing order each edge node e, accordingly to

$$\mathcal{V}_u(e) = D_{e,u}. \tag{8}$$

As a consequence, the first choice LLM for user u is the node inducing the lowest e2e delay. The edge node preference lists, denoted with $\mathcal{W}_e(u)$, are built in accordance with

$$\mathcal{W}_e(u) = d_u, \tag{9}$$

sorting the users in descending order. The proposed matching algorithm works as follows

1. Each unallocated user creates its preference list in accordance with (8), ranking edge nodes having the LLM specialized on the learning task requested by the user.
2. Each unallocated user proposes to be computed on its favorite edge node.
3. Each edge node builds its preference list.
4. Each edge node selects the best proposal among those received, and rejects the others.
5. Repeat 2)-5) until all the users are allocated or resources are available and coherent with the learning tasks.

The proposed algorithm, whose pseudocode is detailed in Algorithm 1, subtends that preference lists are updated after each algorithm round, to keep consistency between the algorithm choices, the residual available resources, and the queue size on each edge node. This implies the existence of dependencies among players' preferences and impacts the stability of the game, which is not trivial. To validate the convergence of the proposed approach in a stable outcome matching $\mathcal{M}$, the definition of a two-sided exchange stable matching (S2ES) is recalled [29].

Def. 1: An outcome matching $\mathcal{M}$ is an S2ES matching if no pair of users (u_1, u_2) exists s.t.:

1) $\mathcal{V}_{u_1}(\mathcal{M}(u_2)) \leq \mathcal{V}_{u_1}(\mathcal{M}(u_1))$ and
2) $\mathcal{V}_{u_2}(\mathcal{M}(u_1)) \leq \mathcal{V}_{u_2}(\mathcal{M}(u_2))$ and
3) $\mathcal{W}_{\mathcal{M}(u_1)}(u_2) \leq \mathcal{W}_{\mathcal{M}(u_1)}(u_1)$ and
4) $\mathcal{W}_{\mathcal{M}(u_2)}(u_1) \leq \mathcal{W}_{\mathcal{M}(u_2)}(u_2)$ and
5) $\exists \psi \in \{u_1, u_2, \mathcal{M}(u_1), \mathcal{M}(u_2)\}$ s.t. at least one of the conditions $1) - 4)$ is strictly verified.

Based on Def. 1, the proposed matching algorithm reaches a final S2ES configuration, since the resources of edge nodes iteratively decrease as the algorithm proceeds. Similarly, since the system does not drop already allocated requests, Q_u can only remain the same or become worse. This means that, considering a target user u allocated on edge node e during the kth iteration of the algorithm, it cannot increase its condition by changing partner, since the available resources and the queue delay experienced of remaining edge nodes will remain unchanged at best. The same reasoning can be extended to edge nodes, proving that condition 5) of Def. 1 is not verified, concluding that the outcome matching $\mathcal{M}$ is stable.

The computational complexity of the algorithm can consider the worst-case scenario where all users prefer the same edge node. In this case the algorithm terminates in $|U|$ steps, thus the computational complexity is $\mathcal{O}(UE \log E)$.

V. Simulation Results

In this section, we present results obtained by resorting to extensive numerical simulations. The algorithm behavior is analyzed in terms of mean e2e delay and dropping probability, i.e., the probability of having users experiencing a completion time greater than the corresponding deadline. The experiments focus on the decision-making ability of the matching game proposed, in solving the problem of assignment, in comparison with alternative schemes. Results averaged over the number of 1000 independent runs. Experimentation was performed using an Apple M1 Pro CPU equipped with 32 GB of RAM. As a reference scenario, we considered three types of LMMs: text-to-text LLM, text-to-audio LLM, and text-to-image LLM. Then, we set a number of edge node uniformly distributed within the interval $[13, 17]$. For each edge node, we deployed the type of LLM by uniformly drawing a number from 1 to 3, which indicates the type of LLM hosted in that edge node. Parameter d_u is assumed to be uniformly distributed within the interval $[1, 3.5]$, and s_u and Δ_e were uniformly distributed in $[1.1, 3.1]$ and $[3.3, 5.5]$, respectively. Similarly, we considered an integer number of users between 95 and 105 with uniform distribution. Furthermore, the number of tokens per user was generated between 240 and 248 with uniform distribution. Finally, ι_e was randomly selected within $[0.01, 0.03]$ ms, and $C_{e,u}$ in $[0.60, 0.80]$ ms. To assess the proposed matching approach, indicated as Inference-Aware Matching (IAM), we implemented the following schemes from the literature.

Gale-Shapley Matching: the Gale-Shapley algorithm is applied in its traditional form, without updating preference lists after each assignment [28].

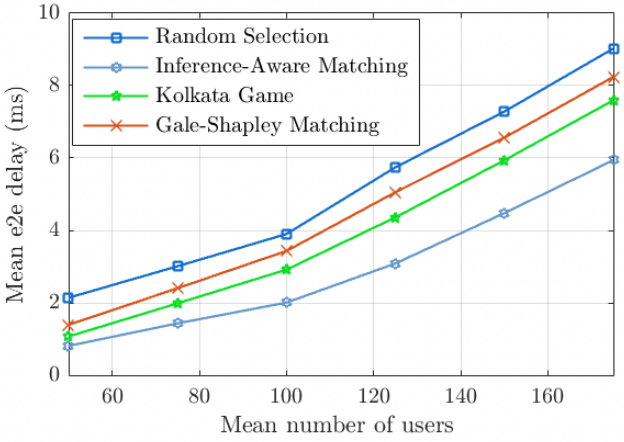

Fig. 2. Mean user e2e delay as a function of the number of users

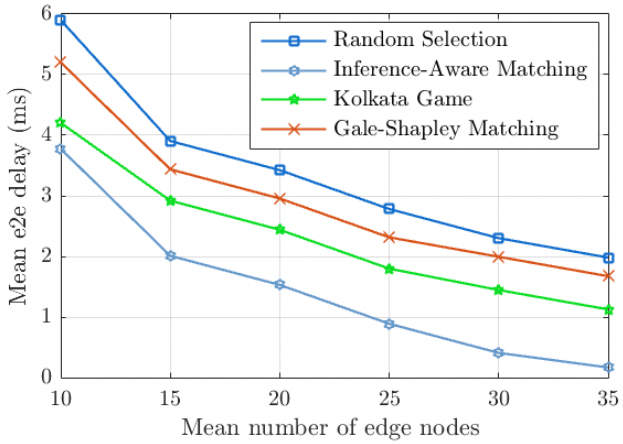

Fig. 3. Mean user e2e delay as a function of the number of edge nodes

Kolkata Game: this repeated game, detailed in [30], is a matching game based on one-side preferences, i.e., where only one of the two parties involved in the game expresses preferences toward the other. Similarly to our approach, in the Kolkata game users express preferences according to (8). Once users have submitted the allocation request to edge nodes, each node randomly selects the preferred user among the proposing ones. The random choice represents the main difference compared to IAM, which selects the preferred user based on (9) among users proposing allocation.

Random Selection: each user chooses the edge node for computation at random, according to a uniform distribution.

Fig. 2 depicts the mean e2e delay in ms, as a function of the mean number of users demanding learning tasks. As it is evident from the figure, IAM reduces the mean e2e delay, in comparison to the considered alternatives, also for system configurations where the number of users increases. Given the close similarity between the Kolkata Game and the Proposed Selection, the Kolkata curve is the second best. The performance degradation compared to the proposed selection is due to the random choice component introduced by the edge nodes toward the users. What the curves have in common is the increasing trend as the number of tasks grows. This is due to the rising queue delays in the LLMs, leading to an overall increase in the load of the system under study.

Similarly, Fig. 3 shows the mean e2e delay as a function of the mean number of edge nodes in the network. Also in this case, IAM outperforms the alternative schemes. For example, by setting a quality of service constraint in terms of mean e2e delay, and fixing it to 2 ms, we observe that the proposed approach meets the constraint with 15 edge nodes, whereas the Kolkata game requires approximately 23 edge nodes. This highlights how IAS make more efficient use of the system resources. The superior performance achieved by IAM is also confirmed in terms of dropping probability, as shown in both Figs. 4 and 5. In particular, Fig. 4 shows the probability of dropping as a function of the mean deadline value associated with the LMM learning task requested by users. IAM reaches a lower dropping probability compared with the benchmarks. The reason is that in (9), the edge nodes prefer learning tasks with larger deadlines, i.e., those further away in time. This choice increases the likelihood of completing the learning task within the required deadline and improves dropping probability of the system. The dropping probability as a function of the mean storage per user is illustrated in Fig. 5. Once again, the joint communication and inference-aware approach ensures greater control over the e2e delay, which, in turn, impacts the dropping probability. In this case, increasing the value of s_u alters the combinatorial space of the problem. As s_u increases, while keeping the total capacity of the edge nodes constant, the number of learning tasks that can be allocated decreases, leading to a corresponding rise in the dropping probability.

VI. Conclusions

This paper investigated the problem of the association between users and LLMs within a native-generative artificial intelligent network, where edge nodes have heterogeneous and limited capabilities, and LLMs have different learning specializations. The paper proposes a matching game to solve the user-LLM assignment problem, considering both the communication and the inference delay. The main objective is the minimization of the user end-to-end delay and the game formulated is a matching with externalities and incomplete lists, whose stability is proved by construction. Numerical results exhibit the validity of the communication and inference-aware approach compared to alternative decision-making schemes.

Future works may include the extension to multi-task and multi-model scenarios, allowing inferences from multiple LLMs. This would require redefining the framework to incorporate multi-dimensional preferences and resource-sharing constraints, enabling a multi-dimensional allocation of resources. Another important direction is the adaptation to dynamic edge environments, exploring reallocation mechanisms to evolve user-LLM associations based on real-time conditions and changing workloads. Optimizing further critical metrics, such as energy efficiency and operational costs, would enhance the practicality of the framework. Incorporating power-aware resource allocation that balances performance with energy consumption would make the system more sustainable, while

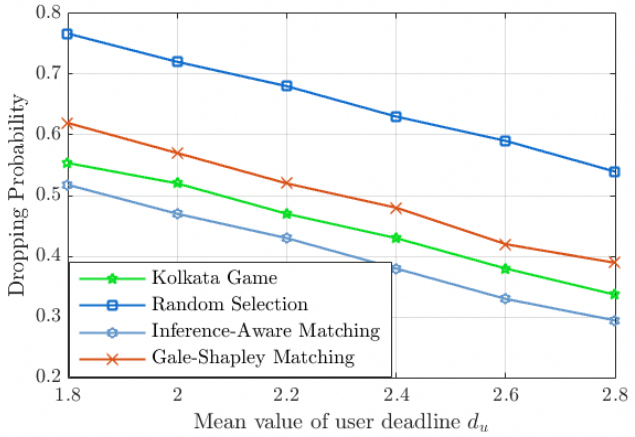

Fig. 4. Mean dropping probability delay as a function of the user deadline

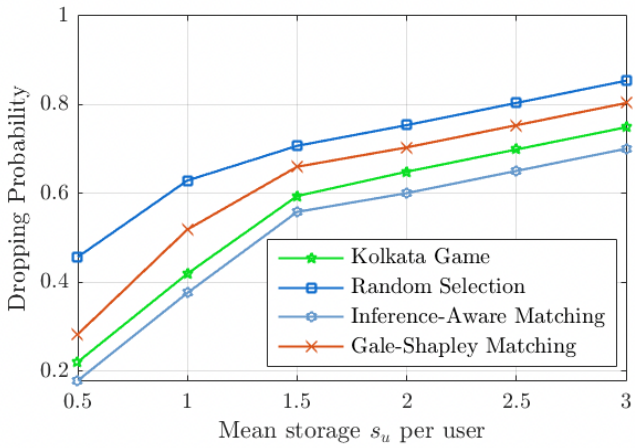

Fig. 5. Mean dropping probability as a function of the user storage demand

cost-aware optimization would help reducing the cost of large-scale LLM deployment.

REFERENCES

[1] T. B. Brown, "Language models are few-shot learners," *arXiv preprint arXiv:2005.14165*, 2020.

[2] R. Bommasani, D. A. Hudson, E. Adeli, R. Altman, S. Arora, S. von Arx, M. S. Bernstein, J. Bohg, A. Bosselut, E. Brunskill *et al.*, "On the opportunities and risks of foundation models," *arXiv preprint arXiv:2108.07258*, 2021.

[3] T. Kojima, S. S. Gu, M. Reid, Y. Matsuo, and Y. Iwasawa, "Large language models are zero-shot reasoners," *NeurIPS*, vol. 35, pp. 22 199–22 213, 2022.

[4] Y. Lin, H. Lin, W. Xiong, S. Diao, J. Liu, J. Zhang, R. Pan, H. Wang, W. Hu, H. Zhang *et al.*, "Mitigating the alignment tax of RLHF," in *Proc. Conf. Empirical Methods Nat. Lang. Process.*, 2024, pp. 580–606.

[5] C. Chaccour, M. N. Soorki, W. Saad, M. Bennis, and P. Popovski, "Can terahertz provide high-rate reliable low-latency communications for wireless VR?" *IEEE Internet Things J.*, vol. 9, no. 12, pp. 9712–9729, 2022.

[6] 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-Confidence Computing*, p. 100211, 2024.

[7] 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.

[8] Y. Guan, D. Wang, Z. Chu, S. Wang, F. Ni, R. Song, and C. Zhuang, "Intelligent agents with LLM-based process automation," in *Proc. ACM SIGKDD*, 2024, pp. 5018–5027.

[9] Y. Huang, H. Du, X. Zhang, D. Niyato, J. Kang, Z. Xiong, S. Wang, and T. Huang, "Large language models for networking: Applications, enabling techniques, and challenges," *IEEE Network*, vol. 39, no. 1, pp. 235–242, 2025.

[10] F. Jiang, Y. Peng, L. Dong, K. Wang, K. Yang, C. Pan, D. Niyato, and O. A. Dobre, "Large language model enhanced multi-agent systems for 6G communications," *IEEE Wireless Commun.*, 2024, early Access.

[11] M. Xu, D. Niyato, H. Zhang, J. Kang, Z. Xiong, S. Mao, and Z. Han, "Cached model-as-a-resource: Provisioning large language model agents for edge intelligence in space-air-ground integrated networks," *arXiv preprint arXiv:2403.05826*, 2024.

[12] A. Buratto, M. Levorato, and L. Badia, "DCP: a TCP-inspired method for online domain adaptation under dynamic data drift," in *Proc. IEEE Int. Symp. World Wirel. Mobile Multim. Netw. (WoWMoM)*, 2024, pp. 269–278.

[13] Z. Ning, P. Dong, X. Wang, S. Wang, X. Hu, S. Guo, T. Qiu, B. Hu, and R. Y. Kwok, "Distributed and dynamic service placement in pervasive edge computing networks," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 6, pp. 1277–1292, 2020.

[14] Y. Du, J. Li, L. Shi, T. Liu, F. Shu, and Z. Han, "Two-tier matching game in small cell networks for mobile edge computing," *IEEE Trans. Serv. Comp.*, vol. 15, no. 1, pp. 254–265, 2019.

[15] Q.-V. Pham, T. Leanh, N. H. Tran, B. J. Park, and C. S. Hong, "Decentralized computation offloading and resource allocation for mobile-edge computing: A matching game approach," *IEEE Access*, vol. 6, pp. 75 868–75 885, 2018.

[16] B. Picano, E. Vicario, and R. Fantacci, "An efficient flows dispatching scheme for tardiness minimization of data-intensive applications in heterogeneous systems," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 6, pp. 3232–3241, 2023.

[17] Y.-C. Chen and G. Hu, "A theory of stability in matching with incomplete information," *Am. J. Econ.: Microeconomics*, vol. 15, no. 1, pp. 288–322, 2023.

[18] 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, 2024.

[19] Y. Shen, J. Shao, X. Zhang, Z. Lin, H. Pan, D. Li, J. Zhang, and K. B. Letaief, "Large language models empowered autonomous edge AI for connected intelligence," *IEEE Commun. Mag.*, vol. 62, no. 10, pp. 140–146, 2024.

[20] 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.*, 2025, early Access.

[21] L. Dong, F. Jiang, Y. Peng, K. Wang, K. Yang, C. Pan, and R. Schober, "LAMBO: Large AI model empowered edge intelligence," *IEEE Commun. Mag.*, 2025, arXiv preprint arXiv:2308.15078.

[22] 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," *arXiv preprint arXiv:2309.16739*, 2023.

[23] S. Ma, S. Song, J. Zhao, L. Zhai, and F. Yang, "Joint network selection and service placement based on particle swarm optimization for multi-access edge computing," *IEEE Access*, vol. 8, pp. 160 871–160 881, 2020.

[24] V. Mancuso, L. Badia, P. Castagno, M. Sereno, and M. A. Marsan, "Effectiveness of distributed stateless network server selection under strict latency constraints," *Computer Netw.*, p. 110558, 2024.

[25] C. Su, F. Ye, T. Liu, Y. Tian, and Z. Han, "Computation offloading in hierarchical multi-access edge computing based on contract theory and Bayesian matching game," *IEEE Trans. Veh. Tech.*, vol. 69, no. 11, pp. 13 686–13 701, 2020.

[26] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: A survey," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 450–465, 2017.

[27] J. Li, A. Sun, J. Han, and C. Li, "A survey on deep learning for named entity recognition," *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 1, pp. 50–70, 2020.

[28] D. Manlove, *Algorithmics of matching under preferences*. World Scientific, 2013, vol. 2.

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

[30] B. K. Chakrabarti, "Kolkata restaurant problem as a generalised El Farol bar problem," in *Econophys. Markets Business Netw.* Springer, 2007, pp. 239–246.