A review on Caching for minimizing Latency in Edge-Cloud Continuum

Kaushik Mishra[1], Ganesh Khekare[2]
[1]Lincoln University College (LUC), Malaysia;
[2]School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, 632014, Tamil Nadu, India
E-mail ID: pdf.kaushik@lincoln.edu.my, khekare.123@gmail.com

Abstract: The fast growth of latency-sensitive Internet of Things (IoT) applications has made it even more important for cloud-assisted Mobile Edge Computing (MEC) systems to be able to quickly offload tasks and cache services. A lot of research has been done on offloading and caching strategies through joint optimization. However, most of these methods depend on centralized control, static assumptions about service popularity, or computationally intensive learning frameworks, which make them less scalable and adaptable when traffic changes. Furthermore, the coordination of offloading, caching, and service replacement across various temporal scales is inadequately explored in existing literature. This paper provides a thorough review and redefinition of the joint task offloading and service caching problem in cloud-assisted MEC environments, driven by the identified gaps. The proposed solutions are highlighted in future work.

Keywords: Mobile Edge Computing; Service caching; Service Replacement; Task offloading; Cloud computing

---

Introduction

The rapid growth of IoT devices and delay-critical applications has caused an unpredicted surge in data generation at the network edge. These applications like industrial automation, smart transportation, smart healthcare, augmented reality, etc. demand faster response time and hard QoS guarantees which are likely to be impossible in cloud-assisted applications due to the physical gaps in datacenter location. These traditional cloud-centric paradigms often fail to meet these requirements due to traffic overhead, prolonged transmission delays, and limited adaptability to variable network conditions [1].

Mobile Edge Computing (MEC) [2] has become a promising solution by putting computing and storage resources closer to the people who use them. MEC greatly lowers service latency and lightens the load on the core network by letting tasks run on edge servers that are next to base stations (BSs). But the fast growth of different IoT services and the changing needs of users have made it harder to manage limited edge resources efficiently. In MEC-enabled networks, intelligent task offloading and service caching are especially important for making sure that the system is always responsive and can run for a long time [3].

By connecting centralized cloud servers with distributed edge nodes, cloud-assisted MEC [4] makes the system even more flexible. This hierarchical framework allows latency-sensitive tasks to be

processed at the edge, while the compute-intensive or cache-miss tasks to be offloaded to the Cloud. However, effectively coordinating the Cloud and edge nodes under the dynamic traffic and resource conditions remains a challenge, motivating the need for adaptive, learning-based optimization framework [5].

Challenges in Task Offloading and Service Caching in MEC

Efficient task offloading in MEC determines the offloading location, such as a task should be computed on the device itself, offloaded to a nearby edge server, migrated to a nearby BS, or sent it to the cloud. Critically of tasks, computing power requirements, good network conditions, and disparate service requirements make task offloading challenging. The existing static and rule-based approaches are not suitable for these real-world situations [6]. Since edge nodes lack sufficient storage space to cache all the computing services, service caching becomes even more complicated. It becomes very challenging to determine which services to cache, when to update the cached services, when to replace the cached service to optimize the storage, and what should be done when cache miss occurs. If caching strategies aren't working well, services may go down often at the edge, the system may rely more on cloud offloading, and performance may suffer [7].

Furthermore, task offloading and service caching operate at different time scales. Task offloading decisions must be made quickly to respond to real-time task arrivals and network changes. In contrast, service caching and replacement are performed less frequently i.e. on longer timescales because of storage limits and the costs involved in deploying and updating services at the edge. Coordinating these decisions across various timescales while considering fluctuating traffic intensities continues to be a primary challenge in cloud-assisted MEC systems [8].

Related work

Caching and offloading strategies have become a predominant solution to reduce the latency overhead improving the overall performance for the rapid growth of resource-intensive and latency-critical applications. This paper explores a range of optimization methods, from heuristics to metaheuristics and machine learning algorithms to advanced deep learning algorithms, to leverage dynamic tasks and heterogeneous computing resources. This review synthesizes findings from 12 research works (2024-2025) highlighting QoS metrics, simulator used, problem addressed, optimization method used and research gaps identified.

Fu *et al.* (2025) [9] formulated an optimization problem in terms of costs for IaaS big data Cloud. They proposed deterministic and randomized algorithms to address the issues. In this work, authors have optimized cost function across task reuse frequencies. Experiments were simulated on Alibaba cloud. Xie *et al.* (2025) [10] devised a D3QN-based algorithm for dynamic service caching and computation offloading for MEC. The authors have formulated the problem using Markov decision process. In this, a single agent RL framework is used with the future work pointing to use the deep streaming RL methods for improved scalability and reliability. The authors have improved the overall service latency, energy efficiency and overall reward. Chen *et al*. (2025) [11] utilized DDPG and DDQN for joint service caching and task offloading in MEC with the improvement of task delay and energy efficiency leveraging hierarchical decision making. However, this study lacks real-world validation and comparison with advanced related baselines. Lu *et al.* (2025) [12] introduced a multilevel framework with dynamic programming, tabu search, Q-learning, and MDP for dynamic and QoS-aware service caching for MEC network. The authors efficiently improved the

total cost and system delay. However, it lacks real-world validation on large-scale setup with security and privacy concerns. Zhai *et al.* (2025) [13] devised a framework using block coordinate descent, progressive rounding and the entropy weight method for task offloading and multi-cache placement for multi-access edge computing network. This framework effectively meliorated service latency, cache hit ratio and energy efficiency, however, through a stable network prone to dynamic conditions. Zhao *et al.* (2025) [14] leveraged DDPG and MDP as a joint framework for task offloading and service caching for dependency-aware scheduling. Total reward, delay, and success rate are optimized, however, it failed to cater to the needs for dynamic caching and dynamic workload conditions with no server load prediction. Li *et al.* (2025) [15] proposed a framework for service caching for resource-constrained devices using centralized training with decentralized execution (CTDE) algorithm. This model is validated through total latency, local hit ration, and computational complexity under a python-based simulation. Zeng *et al.* (2025) [16] introduced an online framework to make a trade-off for resource utilization, total load, and energy usage using Lyapunov optimization. They have used Gurobi optimization for efficient decision-making for this online framework which integrates cachebality-driven interactive method. However, it suffers from unpredictable surge and fails to balance energy use and offloading to Cloud. Zhao *et al.* (2024) [17] presented a UAV-based MEC network for joint caching, offloading and service placement. They have optimized cache hit ratio, service latency, energy consumption and thus overall QoE performance. However, this model does not investigate unrealistic assumptions about user homogeneity and delay-metric. Xu *et al.* (2024) [18] proposed a hybrid soft actor-critic approach for task offloading and service caching in MEC network. They have used a standard simulator. This simulation primarily focuses on reducing total task delay and computational cost; however, it does not consider privacy-preserving task offloading for broader deployment. Wu *et al.* (2024) [19] used cross entropy-based optimization method for secure offloading in a hierarchical framework. This method is used to tackle delay-sensitive issues. This model improved and optimized task delay with maximization of secrecy offloading rates. However, it is limited by assumptions of perfect CSI (Channel State Information), binary-only decision spaces, and single time block scenarios. Ren *et al.* (2024) [20] devised a caching strategy which combines a learning-driven hybrid method of MAB and GAN for traffic prediction and minimizing average delay. The proposed method shows an improvement in optimizing efficient traffic prediction, minimized average delay and regret bound. Authors have used a real-world data trace using ILP and LP relaxation for optimized caching. However, this model fails to learn about traffic knowledge and the complexities of these learning models incur high.

Across the reviewed works, reinforcement learning (especially DRL and PPO) and convex optimization techniques dominate the optimization landscape. While simulation environments and custom-built testbeds are common, real-world implementation and validation remain rare. Emerging areas such as multi-agent systems, meta-learning, digital twins, and LLM integration are promising directions for future research. Existing research on MEC optimization [9-21] mainly treats task offloading and service caching as separate issues or presumes implausible system conditions. Several studies concentrate exclusively on task offloading, presuming either infinite or fixed service availability at edge servers, thereby neglecting the practical limitations imposed by constrained edge storage. On the other hand, a lot of service caching methods use static popularity models or offline optimization, which don't work well when service demand and traffic patterns change quickly. Recent learning-based solutions,

especially those using deep reinforcement learning (DRL), have shown promise in dealing with changing environments. Most DRL-based methods, on the other hand, use centralized learning architectures, which don't work well in large-scale MEC deployments because they can't scale and have too much communication overhead. Also, end-to-end DRL solutions that optimize both offloading and caching at the same time often have problems with convergence stability because rewards are sparse and delayed, especially when caching decisions have a long-term effect on system performance. Also, current frameworks often don't consider how cloud assistance can help with edge resource limits, if all tasks need to be done in the edge layer. This edge-only assumption makes the system less stable when there are a lot of tasks to do or when there are cache misses, and it limits how useful these kinds of solutions can be in real cloud-edge-IoT ecosystems. Table 1 summarizes the related work.

Table 1. Summary Table

Ref.	**Objective**	**Methodology**	**Key Limitations**	**Improvements by Our Proposed Work**
[9]	Cost optimization for cloud task reuse	Linear programming, competitive analysis	No edge support; ignores latency and caching	Edge-aware, latency- and energy-driven optimization
[10]	Joint offloading and caching	Centralized D3QN	Scalability issues; static edge configs	Distributed DDRL with adaptive traffic awareness
[11]	Cooperative offloading and caching	HRL (DDPG + DDQN)	No explicit service replacement; weak timescale separation	Explicit multi-timescale hierarchy with replacement
[12]	QoS-aware caching	DP, tabu search, Q-learning	High complexity; limited scalability	Lightweight, decentralized learning framework
[13]	Offloading and multi-cache placement	BCD, entropy weighting	Assumes stable network; high optimization cost	Traffic-aware abstraction for dynamic environments
[14]	Dependency-aware offloading	DDPG + knapsack	Static caching; limited delay modeling	Dynamic caching and cache-aware rewards
[15]	Service caching optimization	Transformer-based CTDE	High complexity; no energy awareness	Energy–latency tradeoff with lightweight learning
[16]	Load balancing and caching	Lyapunov + Gibbs sampling	Sensitive to demand uncertainty	ICLA-based traffic-aware adaptation
[17]	UAV-assisted MEC optimization	Gibbs sampling, matching games	Homogeneous user assumptions	Supports heterogeneous IoT services
[18]	Delay and cost minimization	Hybrid SAC	Assumes perfect system knowledge	Operates under partial observability
[19]	Security-aware offloading	Convex optimization (SCA)	Perfect CSI assumption; limited adaptability	Scalable, adaptive decision-making
[20]	Delay-aware caching	MAB + GAN prediction	High model complexity	Lightweight traffic abstraction

Key Contributions

The main contributions of this work can be summarized as follows:

- A thorough literature review (2024–2025) is performed on task offloading, service caching, and joint optimization in cloud-assisted MEC, systematically classifying existing studies by methodology, learning paradigm, and system assumptions.
- The research gaps have been identified including weak integration between task offloading and service caching, a lack of explicit service replacement mechanism, limited coordination across multiple time scales, and a heavy reliance on centralized or computationally heavy optimization methods.
- A structured comparison table brings together different methods and makes it clear that the proposed framework fills in the gaps in scalability, adaptability, and practical deployability that were found in earlier research.

Conclusions and Future Works

- Latency is a pivotal component to be reduced in Cloud-assisted mobile edge computing. The rapid growth of IoT and latency-sensitive applications demand a faster response time for execution. Therefore, caching strategies appears to be a prominent solution for minimizing latency which stores services in edge devices or base stations for a quick fetch.
- There are several exiting works which consider metaheuristics to ML and DL techniques to address these issues. However, none of the papers considers time-aware and traffic-aware service caching which is suitable for unpredicted workloads and dynamic network conditions.
- This study performs a thorough literature review considering 12 journal articles from 2024 to 2025 and presented a systematic review analyzing future trends, research gaps, performance metrics used, simulations carried out and methodology used to address the latency overhead issue in MEC.
- This leverages to design a time and traffic-aware task offloading based on learning, service caching and replacement based on optimization, and dynamic decision making by edge nodes.
- Furthermore, a full performance test by running a lot of simulations will be carried out. We will look at things like how long it takes for a request to be processed, how much power it uses, how well it uses its cache, and how well it can grow.
- Future research will also investigate how to make the framework work with user mobility, service migration, security and privacy issues, and real-world workloads or experimental testbeds.

References

1. Li, N., Zhai, L., Ma, Z., Zhu, X., Li, Y.: Lyapunov-guided Deep Reinforcement Learning for service caching and task offloading in Mobile Edge Computing. Comput. Netw. (2024).
2. Liu, J., Li, C., Luo, Y.: Efficient resource allocation for IoT applications in mobile edge computing via dynamic request scheduling optimization. Expert Syst. Appl. **255**, 124716 (2024).
3. Somesula, M.K., Mothku, S.K., Annadanam, S.C.: Cooperative service placement and request routing in mobile edge networks for latency-sensitive applications. IEEE Syst. J. **17**(3), 4050– 4061 (2023).
4. Somesula, M.K., Mothku, S.K., Kotte, A.: Deep reinforcement learning mechanism for deadline-aware cache placement in device-to-device mobile edge networks. Wireless Netw. **29**(2), 569–588 (2023).
5. Xie, M., Ye, J., Zhang, G., Ni, X.: Deep reinforcement Learning-based computation offloading and distributed edge service caching for mobile edge computing. Comput. Netw. **250**, 110564 (2024).
6. Rajareddy, G.N.V., Mishra, K., Majhi, S.K., Sahoo, K.S., Bilal, M.: M-SOS: Mobility-aware secured offloading and scheduling in dew-enabled vehicular fog of things. IEEE Trans. Intell. Transp. Syst. **26**(4), 4851–4864 (2025).
7. Zhao, X., Wu, Y., Zhao, T., Wang, F., Li, M.: Federated deep reinforcement learning for task offloading and resource allocation in mobile edge computing-assisted vehicular networks. J. Netw. Comput. Appl. **229**, 103941 (2024).
8. Chhabra, G.S., Satti, S.K., Rajareddy, G.N.V. *et al.* Time-and-Traffic-aware collaborative task offloading with service caching-replacement in cloud-assisted mobile edge computing. *Cluster Comput* **28**, 900 (2025). https://doi.org/10.1007/s10586-025-05629-x
9. Fu, X., Pan, L., & Liu, S. (2025). Caching or re-computing: Online cost optimization for running big data tasks in IaaS clouds. *Journal of Network and Computer Applications*, *235*, 104080. https://doi.org/10.1016/j.jnca.2024.104080
10. Xie, B., Xie, J., Cui, H., He, Y., & Guizani, M. (2025). Dynamic Service Caching Aided Computation Offloading Optimization Algorithm for Mobile Edge Networks. *IEEE Internet of Things Journal*. 10.1109/JIOT.2025.3555978
11. Chen, T., Ai, J., Xiong, X., & Hu, G. (2025). Cooperative Service Caching and Task Offloading in Mobile Edge Computing: A Novel Hierarchical Reinforcement Learning Approach. *Electronics*, *14*(2),380.https://doi.org/10.3390/electronics14020380
12. Lu, S., Jin, X., Wu, J., Zhou, S., Yang, J., Yan, R., ... & Cai, Z. (2025). Enhanced Multi-Stage Optimization of Dynamic QoS-Aware Service Caching and Updating in Mobile Edge Computing. *IEEE Transactions on Network and Service Management*. 10.1109/TNSM.2025.3570427
13. Zhai, L., Zhao, P., Xue, K., Li, Y., & Cheng, C. (2025). Task offloading and multi-cache placement in multi-access mobile edge computing. *Computer Networks*, *258*, 111030. https://doi.org/10.1016/j.comnet.2024.111030
14. Zhao, L., Zhao, Z., Hawbani, A., Liu, Z., Tan, Z., & Yu, K. (2025). Dynamic caching dependency-aware task offloading in mobile edge computing. *IEEE Transactions on Computers*. 10.1109/TC.2025.3533091
15. Li, Y., Zhang, Z., & Chao, H. C. (2025). Service caching with multi-agent reinforcement learning

in cloud-edge collaboration computing. *Peer-to-Peer Networking and Applications*, *18*(2), 93. 5) https://doi.org/10.1007/s12083-025-01915-y

16. Zeng, J., Zhou, X., & Li, K. (2024). Resource-Efficient Joint Service Caching and Workload Scheduling in Ultra-Dense MEC Networks: An Online Approach. *IEEE Transactions on Network and Service Management*. 10.1109/TNSM.2024.3511537
17. Zhao, Y., Liu, C., Hu, X., He, J., Peng, M., Ng, D. W. K., & Quek, T. Q. (2024). Joint content caching, service placement and task offloading in UAV-enabled mobile edge computing networks. *IEEE Journal on Selected Areas in Communications*. 10.1109/JSAC.2024.3460049
18. Xu, Y., Peng, Z., Song, N., Qiu, Y., Zhang, C., & Zhang, Y. (2024). Joint optimization of service caching and task offloading for customer application in MEC: A hybrid SAC scheme. *IEEE Transactions on Consumer Electronics*. 10.1109/TCE.2024.3443168
19. Wu, M., Li, K., Qian, L., Wu, Y., & Lee, I. (2024). Secure computation offloading and service caching in mobile edge computing networks. *IEEE Communications Letters*, *28*(2), 432-436. 10.1109/LCOMM.2023.3347218
20. Ren, W., Xu, Z., Liang, W., Dai, H., Rana, O. F., Zhou, P., ... & Wu, G. (2024). Learning-driven service caching in MEC networks with bursty data traffic and uncertain delays. *Computer Networks*, *250*, 110575. https://doi.org/10.1016/j.comnet.2024.110575