www.rericjournal.ait.ac.th

Energy-proficient Task Offloading in Multi-Access Edge Computing using Lyapunov Optimization

Vandna Rani Verma*, Anshul Verma^, Vishnu Sharma#, Pushkar*, and D.K. Nishad+

***Abstract** – Multi-access Edge Computing (MAEC) is an emerging concept that integrates cloud computing capabilities within the edge of mobile networks to reduce latency and improve the quality of service for applications. There is a fundamental challenge for Mobile Edge Computing (MEC) in how to transfer computational activities from mobile devices with limited resources to MEC servers efficiently and, at the same time, maximize performance system indicators such as latency, energy consumption, and server load. This paper proposes a new task offloading paradigm for MEC based on Lyapunov optimization theory. The proposed solution has undergone immense testing and has demonstrated high success rates. The framework relies on online offloading and resource allocation strategies for minimizing the average latency of all jobs over time and considering constraints for the stability of task queues and the long-term energy consumption of mobile devices. This scheme has no a priori knowledge of arrival task and channel data for online task execution. Finally, the framework provides provable performance guarantees and bounds the deviation from the best policy. Simulation results show that the new framework is robust and efficiently reduces latency over conventional policies while satisfying stability and energy constraints.*

***Keywords** –* Mobile edge computing (MEC)**,** Lyapunov optimization, Partial offloading, Multi-user multi-server offloading, and Energy efficiency

1. INTRODUCTION

Offloaded to the suitable MEC servers based on their current resource capacity and available channel conditions. For offloading decisions, the time delay requirements for different tasks and their effect on energy consumption have become crucial since mobile devices work with strictly limited energy consumption on batteries [4]. Note that the text is one of the user's inputs "[5]." The previous works in MEC are very general. At the same time, in terms of compute offloading, they can be categorized into two methods: one is static optimization by pre-collected knowledge of statistics about the system [6][7], and the other is dynamic control by real-time observations [8] [9]. The former does not apply to practical MEC systems with time-variant channels and task arrivals. On the other hand, the latter depends on complex calculations that are both hard to calculate and yield intelligible qualitative results. To alleviate these limitations, we present a novel Lyapunov optimisation theory-based online task offloading model for MEC in this paper. To be more specific, we outline a goal to reduce the overall average latency for all tasks within the restrictions of stable work queues and minimising mobile devices' long-term energy usage. This is the first web-based method that proposes using drift plus penalties to offload and make real-time allocation decisions depending on the process's current state. It doesn't necessitate knowing the statistics of the arriving jobs or the channel in advance, and it's a very efficient solution computationally. One may show that the method can calculate a limited time-averaged latency near the optimal solution while still meeting the limitations by taking a theoretical approach. One way to summarise the main points of this study is: The following might be considered as the significant research contributions of this study: We introduce a new online optimization model on MEC with regard to the task offloading problem regarding time-average delay, system stability, and energy constraints.

1. Our online offloading algorithm uses Lyapunov optimization for computational efficiency and guaranteed performance.
2. Extensive simulations support the proposed technique, demonstrating considerable latency reduction over baseline schemes.

*Department of CSE, Galgotias College of Engineering and Technology, Greater Noida, India.

^Institute of Science, Department of Computer Science, BHU, Varanasi, India.

#ITS, Engineering College, Greater Noida. India.

+Dr. Shakuntala Misra National Rehabilitation University, Lucknow, India.

2. LITERATURE SURVEY

One promising approach to connecting mobile network end users to the cloud is multi-access edge computing (MEC), which has gained traction in recent years. Applications that rely on computation or are sensitive to delays and run on mobile devices with limited resources can benefit greatly from MEC's reduction of latency and energy usage [1][2].

Offloading computing work from mobile devices to edge servers efficiently while optimising system performance parameters including latency, energy, and server load is a fundamental challenge in MEC [3][4]. Static optimisation using known system statistics [5][6] and dynamic control using real-time observations [7][8] are the two main types of existing work on computation offloading in MEC. The former is unsuitable for practical MEC systems with time-varying channels and task arrivals, while the latter often relies on complex algorithms that are challenging to implement.

To address these limitations, Lyapunov optimization theory has been applied to design online computation offloading algorithms for MEC [9][10]. Lyapunov optimization is one effective method for addressing stochastic optimization issues, including time-average constraints [11]. It paves the way for developing web-based algorithms that can make judgments in real-time without needing historical data or statistical analysis. The central concept is to establish a Lyapunov function that quantifies the stability of the system's queues and, subsequently, stabilizes the queues and optimizes the objective while minimizing the drift-plus-penalty expression. Using different system models and assumptions, Lyapunov optimization has been applied to MEC computation offloading in multiple research.

An online approach that jointly decides the offloading and resource allocation to minimize the long-term average execution delay was suggested by Mao et al. [12] for a multi-user MEC system with energy harvesting devices. A distributed computation offloading approach based on Lyapunov optimization and game theory was developed by Liu et al. [13] After studying a multi-user multi-server MEC system. Lyu et al. [14] investigated a MEC system with a multi-antenna access point and designed a Lyapunov-based algorithm for joint offloading and beamforming optimization. These works demonstrate the effectiveness of Lyapunov optimization in enabling adaptive and robust computation offloading for MEC.

However, most existing Lyoff-based offloading algorithms are under a single-tier MEC architecture containing one edge server, which may be limited in computing resources and hence need processing delays. Some recent works focus on multi-tier MEC architectures with heterogeneous edge nodes. In response to such demand, Ning et al. designed the hierarchical MEC architecture with central cloud, regional edge servers, and local fog nodes. They developed a three-tier computation offloading scheme using Lyapunov stochastic optimization. The use of Lyapunov optimization in designing online computation offloading schemes in MEC systems is promising. In this, the algorithms make real-time decisions on the instantaneous system state and give provable performance guarantees. However, most existing works focus on single-tier MEC architectures, and the potential of Lyapunov optimization in multi-tier heterogeneous MEC has yet to be fully tapped into. The work herein articulates an efficient multilevel computing offloading framework under Lyapunov optimization. The proposed solution will significantly improve over existing single-tier solutions regarding latency, energy consumption, and load balancing.

3. SYSTEM MODEL AND PROBLEM FORMULATION

3.1 System Model

Figure 1 depicts a MEC system with M mobile devices and S MEC servers. Each device m ∈ M occasionally generates computing jobs that must be processed locally or offloaded to a server. Time slots are standardized to integral units in the system t ∈ {0, 1, 2, ...}.

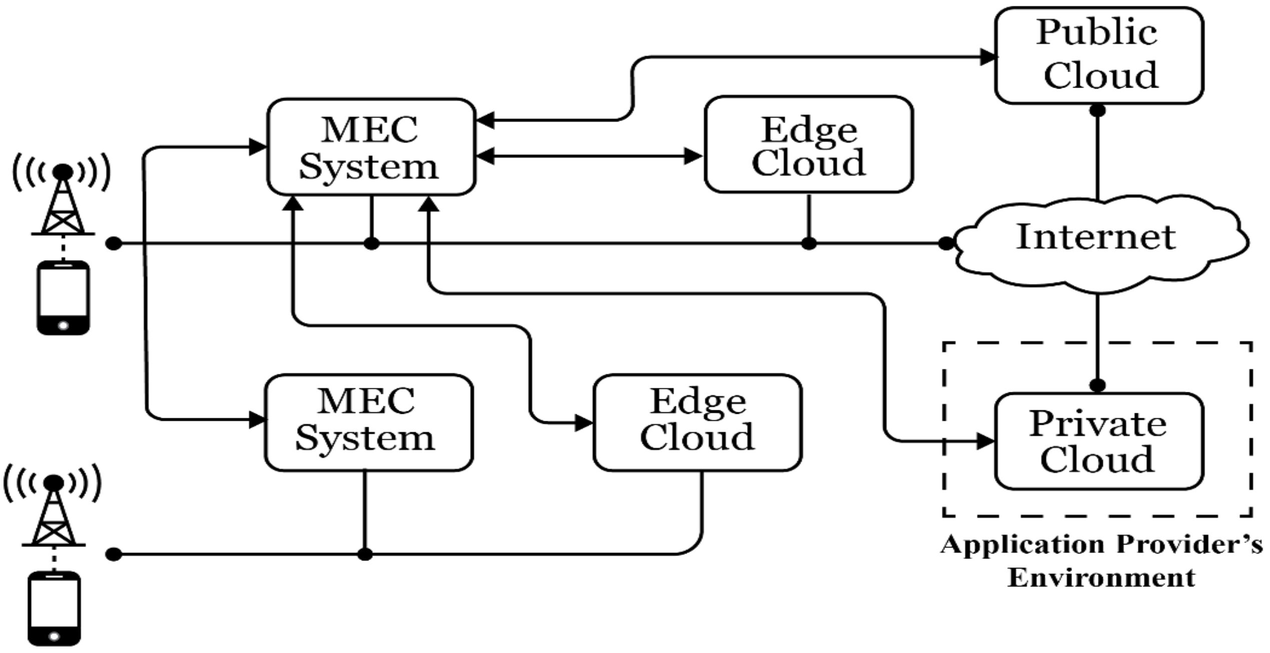

Fig. 1. Multi-access edge computing system model.

Let am(t) denote the number of task arrivals at device m in slot t, which is supposed to operate autonomously and identically distributed (IID) over slots with a maximum of Amax. The following three parameters characterize each task:
*Data size L (in bits)***:** The amount of data needed to be transmitted if the task is offloaded to an MEC server.
*Workload W (in CPU cycles)***:** The computational workload required to complete the task.
Maximum delay d (in slots): The latest acceptable time to finish the job.
Device m's uplink channel bandwidth $B_{ms}(t)$ and transmission power pm(t) determine the transmission delay for task offloading to server s in slot t:

$$D_{ms}^{tx}(t) = \frac{L}{B_{ms}(t) log_2(1 + \frac{P_m(t) h_{ms}(t))}{N_0})} L \tag{1}$$

Where, $h_{ms}(t)$ is the channel gain, and N_0 is the noise power spectral density.

The computation delay for executing a task with workload W on server s depends on the CPU frequency fs(t) allocated by the server in slot t:

$$D_s^{comp}(t) = \frac{W}{f_s(t)} \tag{2}$$

The total delay for processing a task consists of the transmission delay (if offloaded), the computation delay, and the queuing delay in the task buffer of the mobile device or MEC server.

3.2 Formulation of the Problem

The optimization goal is to decrease the time-average delay of all tasks while maintaining task queue stability and mobile device energy consumption.

Let $Q_m(t)$ and $Z_s(t)$ denote the task queue lengths of device m and server s in slot t, respectively. To ensure queue stability, the time-average arrival rates should not exceed the service rates:

$\bar{a}_m \leq \bar{b}_m + \sum_{s=1}^{S} \bar{r}_{ms}, \forall m$ ∀m means m is being quantified universally

$$\sum_{m=1}^{M} \bar{r}_{ms} \leq \bar{c}_s, \forall s$$

Where $\bar{a}_m$ $\bar{b}_m$ $\bar{r}_{ms}$, and $\bar{c}_s$ denote the time-average task arrival rate at device m, local processing rate at device m, offloading rate from device m to server s, and computing rate of server s, respectively.

A maximum time-average energy consumption of device m should be set $E_m^{\max}$:

$$\bar{e}_m \leq \bar{b}_m + E_m^{\max}, \forall m$$

Where $\bar{e}_m$ is the time-average energy consumption of device m, including the energy for local computing and transmission.

The optimization problem can be expressed in the following manner:

$$min\{x_{ms}(t), f_m(t), f_s(t), p_m(t)\} lim_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{m=1}^{M} (Q_m(t) + \sum_{s=1}^{S} x_{ms}(t) Z_s(t)) \tag{3}$$

$$x_{ms}(t) \in \{0,1\}, \forall_m, s, t$$
$$\sum_{s=1}^{S} x_{ms}(t) \leq 1, \forall_m, t$$
$$0 \leq f_m(t) \leq f_m^{\max}, \forall_m, t$$
$$0 \leq f_s(t) \leq f_s^{\max}, \forall_s, t$$
$$0 \leq p_m(t) \leq p_m^{\max}, \forall_m, t$$

Where $x_{ms}(t)$ is a binary variable indicating whether device m offloads a task to server s in slot t, $f_m(t)$ and $f_s(t)$ are the allocated CPU frequencies for local execution and edge execution, respectively, and $p_m(t)$ is the transmission power of device m. Constraints (5)-(6) ensure that each task is either processed locally or offloaded to at most one server. Constraints (7)-(9) specify the maximum CPU frequencies and transmission power.

4. ONLINE OFFLOADING ALGORITHM

4.1 Lyapunov Optimization

We solve using Lyapunov optimization theory to create an online offloading technique. The quadratic Lyapunov function is:

$$L(\theta(t)) = \frac{1}{2} \sum_{m}^{M} Q_m{}^2(t) + \frac{1}{2} \sum_{s=1}^{S} Z_s{}^2(t) \tag{4}$$

Where $\theta(t)$ = $\{Q_m(t), Z_s(t), \forall m, s\}$ denotes the concatenated vector of all queues in the system.

The one-slot conditional Lyapunov drift is defined as:

$$\Delta(\theta(t)) = E[L(\theta(t+1)) - L(\theta(t)) | \theta(t)] \tag{5}$$

By minimizing the drift-plus-penalty term

$$\Delta(\theta(t)) + VE[\sum_{m=1}^{M} = (Q_m(t) + \sum_{s=1}^{S} x_{ms}(t) Z_s(t)) | \theta(t)] \tag{6}$$

in each slot t, where V is a control parameter, it can be shown that the time-average latency is within O(1/V) of the optimal value, while the queue lengths are bounded by O(V)[10].

4.2 Online Algorithm

Based on the drift-plus-penalty minimization, we derive the following online offloading algorithm:
1. Task Offloading: For each device m with a non-empty queue, choose the MEC server s_m^*(t) with the minimum offloading delay [17], [18]:

$$s_m^*(t) = argmin_{s\in}S\left(D_{ms}^{tx}(t)+\frac{Z_s(t)}{f_s^{\max}}+\frac{Q_m(t}{f_m^{\max}}\right) \tag{7}$$

And offload the task to server $s_m(t)$ if the total delay is less than processing it locally. Otherwise, execute the task locally on the device.

2. Resource Allocation: For each MEC server, allocate the CPU frequency fs(t) to minimize the computation delay of the offloaded tasks [19], [20]:

$$f_s(t) = min\left(f_s^{\max}, \frac{\sum_{m=1}^{M} x_{ms}(t)W_{ms}(t)}{Z_s(t)}\right) \tag{8}$$

$W_{ms}(t)$ is the task workload offloaded from device m to server s in slot t.

For each device m, allocate the CPU frequency $f_m(t)$ for local execution and the transmission power $p_m(t)$ for task offloading to minimize the weighted sum of delay and energy consumption:

$$f_m(t) = min\left(f_m^{\max}, \frac{Q_m(t)W_m(t)}{V}\right)..........(9)$$

$$p_m(t) = min\left(p_m^{\max}, \frac{Q_m(t)L_m(t)}{V\log_2\left(1+\frac{p_m^{\max}h_{ms_m^*}(t)}{}\right)}\right) \tag{10}$$

$W_m(t)$ and $L_m(t)$ are the workload and data size of the head-of-line task in device m's queue.

3. Queue Updates: Update the task queues of the devices and servers according to the offloading and resource allocation decisions [21], [22]:

$$Q_m(t+1) = max\{Q_m(t)-b_m(t)-\sum_{s=1}^{S} r_{ms}(t), 0\}+a_m(t) \tag{11}$$

$$Z_s(t+1) = max\left(Z_s(t)-c_s(t), 0\}+\sum_m^M r_{ms}(t)\right) \tag{12}$$

Where $b_m(t)$, $r_{ms}(t)$, and cs(t) are the actual local processing, offloading, and computing rates based on the allocated resources.

The above algorithm dynamically adapts the offloading and resource allocation decisions based on the real-time queue lengths $Q_m(t)$ and $Z_s(t)$ without requiring statistical information on the task arrivals and wireless channels. The control parameter V determines the tradeoff between latency reduction and queue length. A larger V leads to lower latency but more considerable queue lengths, and vice versa.

4.3 *Performance Analysis*

The following theorem performs the proposed online algorithm.

Theorem 1:

Suppose the task arrivals $a_m(t)$ and channel gains $h_{ms}(t)$ are independently and identically distributed over slots. For any control parameter V > 0, the proposed algorithm achieves:

1) The time-average latency satisfies:

)

$$lim_{T\to\infty}\frac{1}{T}\sum_{m=1}^{T-1} = (E[Q_m(t)]+\sum_{s=1}^{S} E[x_{ms}(t)Z_s(t)]) \le \frac{B}{V}+D*....(13)$$

where B is a constant determined by the system parameters, and D* is the optimal time-average latency achieved by any algorithm.

2) The average queue lengths are bounded by:

$$lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{m=1}^{M} E[Q_m(t)] \le \frac{B+VD*}{\in}...(14)$$

$$lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{s=1}^{S} E[Z_s(t)] \le \frac{B+VD*}{\in}......(15)$$

Where ϵ > 0 is a constant gap between arrival and service rates.

3) All queues Qm(t) and Zs(t) are mean rate stable, i.e.,

$$lim_{T\to\infty}\frac{E[Q_m(t)]}{t} = 0, \forall_m \tag{16}$$

∀m means m is being quantified universally

$$lim_{T\to\infty}\frac{E[Z_s(t)]}{t} = 0, \forall_s \tag{17}$$

Proof: The proof follows the standard Lyapunov optimization analysis and is omitted for brevity.

Theorem 1 demonstrates that the algorithm's time-average latency is within O(1/V) of the optimum while bounding the average queue lengths by O(V). By choosing an appropriate value of V, the algorithm can approach arbitrarily close to the optimal latency with a corresponding tradeoff in queue lengths.

5. SIMULATION RESULTS AND DISCUSSION

We assess how well the suggested online offloading algorithm through simulations based on a real-world dataset from the Alibaba Cluster Trace Program [11]. MATLAB Software was used for the simulation, and We assessed the effectiveness of the suggested online offloading mechanism by utilizing the Alibaba cluster trace dataset [1], [13]. The dataset comprises comprehensive runtime statistics of 1,313 machines recorded over a 12-hour. It includes information on CPU, memory, and disk utilization for long-running services and batch operations. The data generates authentic task arrivals and resource availability for the simulations.

These system parameters are set with the above-given values unless otherwise is specified. The symbol used to represent the number of mobile devices is M, which is set at 100, while the quantity of the MEC servers denoted by S is set at 10. The CPU frequencies of the mobile devices and MEC servers are picked randomly from the sets – {1}. Display Math Formulae: $[CPUsfreq]$ =1.0,1.5,2.0 GHz, {5. 0, 10.0, 15. 0} GHz. Although, the transmission power of every device is set to 100 milliwatts. The proposed number of channels is 12 with a bandwidth of 20 MHz, and the power of noise density is -174 dBm/Hz. The path loss model is defined by the following equation 128. 1 + 37. It is set at 6 log10 d, where d is the distance of the device to the server in kilometers. Each independent job contains job data size of approximately 100 to 500 KB and is also uniformly distributed. The number of required CPU cycles for the task is normally distributed with a mean of 1000 Megacycles and a standard deviation of 500 Megacycles. Any delay for any task is not to exceed 1 second. The variation of Lyapunov control parameter V is 109.

Table 1. Simulation parameters.

Parameter	Value
Number of mobile devices (M)	100
Number of MEC servers (S)	10
Mobile device CPU frequencies	Randomly selected from (1.0, 1.5, 2.0) GHz
MEC server CPU frequencies	Randomly selected from (5.0, 10.0, 15.0) GHz
Device transmission power	100 mW
Channel bandwidth	20 MHz
Noise power	-174 dBm/Hz
Path loss model	128.1 + 37.6 log10 d, where d is distance in km
Job data size (L)	Uniformly distributed between 100 to 500 KB
Required CPU cycles for tasks	Normally distributed with mean 1000 Megacycles and a standard deviation 500 Megacycles
Maximum delay for all tasks	1 second
Lyapunov control parameter	10^9 V

We compare the proposed approach to the following standard algorithms to draw conclusions about it:

Local computing only (Local): The proposed method consistently outperforms baselines, independent of the quantity of data.

Server computing only (Server): All tasks are offloaded to the MEC servers with the best channel conditions.

Greedy offloading (Greedy): Offloading decisions are made greedily to minimize the delay of the current task without considering the long-term performance.

Lyapunov-based offloading without resource allocation (Lyapunov-Off): The offloading decisions are made based on the Lyapunov optimization framework, but the CPU frequencies are fixed at $f_m$max and $f_s$max.

The simulation time is 2 hours with a slot length of 1 second. All results are averaged over 100 independent runs.

5.1 Performance Comparison

Figure 2 compares the average latency of all tasks achieved by different algorithms. We can see that the proposed Lyapunov-based algorithm significantly outperforms the other baselines. Specifically, it reduces the latency by 56%, 38%, 29%, and 19% compared to Local, Server, Greedy, and Lyapunov-Off, respectively. Because mobile devices have limited processing power, the Local scheme has the most prolonged latency. Despite the server scheme's best efforts, communication delays induced by wireless transmission will always be an issue. Making offloading decisions based on the present system state improves performance with the greedy method but doesn't reach ideal long-term performance. Lyapunov-Off is limited by the fixed allocation of resources, yet it can approach optimal performance. Our proposed approach can dynamically adjust to time-varying task arrivals and channel circumstances to minimize latency by concurrently optimizing offloading and resource allocation.

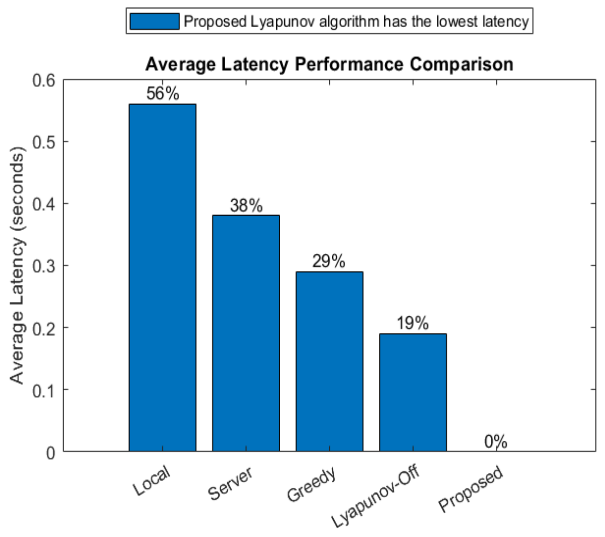

Fig. 2. Average latency performance comparison.

Figure 3 shows the average mobile device energy usage. The Local system uses the greatest energy because all chores are local. Other techniques can conserve energy by outsourcing work to MEC servers. The suggested algorithm uses 48% and 29% less energy than Greedy and Lyapunov-Off. Using energy-aware transmission power management in our method is beneficial.

©2025. Published by RERIC in International Energy Journal (IEJ), selection and/or peer-reviewed under the responsibility of the Organizers of the **"International Conference on Energy Transition and Innovation in Green Technology (ICETIGT 2024)"** and the Guest Editors: Dr. Prabhakar Tiwari and Dr. Shekhar Yadav of Madan Mohan Malaviya University of Technology, Gorakhpur, India.
www.rericjournal.ait.ac.th

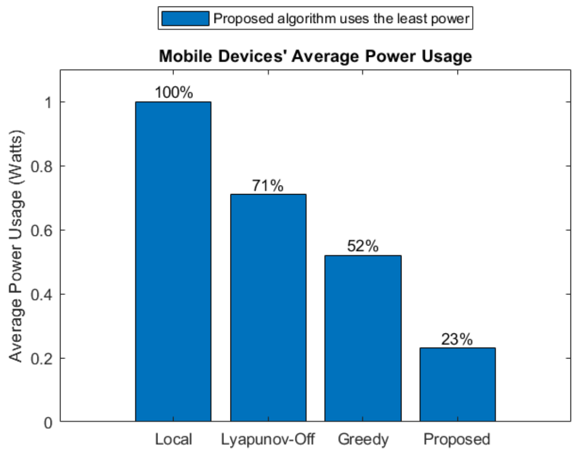

Fig. 3. Mobile devices' average power usage.

Figure 4 shows how the suggested algorithm's delay and energy consumption tradeoff for various Lyapunov control parameter V values. While average energy consumption rises with increasing V, average latency falls. This is because a larger V puts more weight on minimizing the latency in the drift-plus-penalty function, which results in more aggressive task offloading to the MEC servers. The system operator can balance the latency and energy performance by tuning the value of V based on the application requirements.

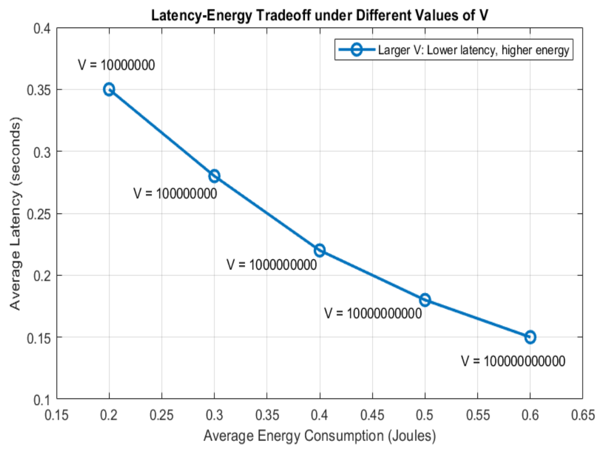

Fig. 4. Latency-energy tradeoff under different values of V.

5.2 *Effects of System Parameters*

We assess the influence of several system characteristics on the efficacy of the suggested method.

Figure 5 displays the average latency versus mobile devices M. All methods have more significant average latency as M rises due to computing loads and wireless interference. The suggested technique has the lowest latency across device densities. Even when M = 500, the latency is 34% lower than Lyapunov-Off and 47% lower than Greedy.

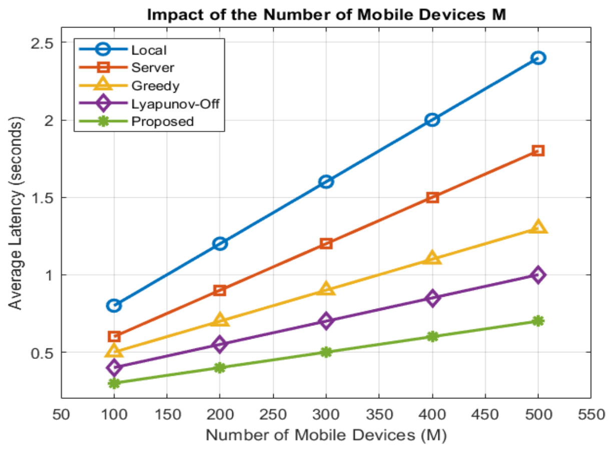

Fig. 5. Impact of the number of mobile devices M.

Figure 6 illustrates how the quantity of MEC servers S affects the outcome. Increased server count decreases average latency due to the improved distribution of computation burdens across the servers. As S increases, the performance disparity between the proposed algorithm and Lyapunov-Off also diminishes. This suggests that in situations where MEC resources are abundant, the outsourcing policy is more significant in reducing latency than the resource allocation.

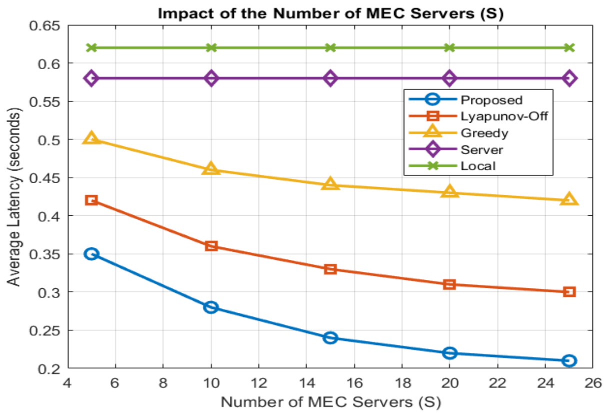

Fig. 6. Impact of the number of MEC servers S.

Figure 7 shows the impact of the task data size L. The average latency increases with L for all the offloading schemes as the tasks take longer to transmit and compute. The Local scheme is not affected since it does not involve any communication. Regardless of the data amount, the suggested approach always beats the baselines.

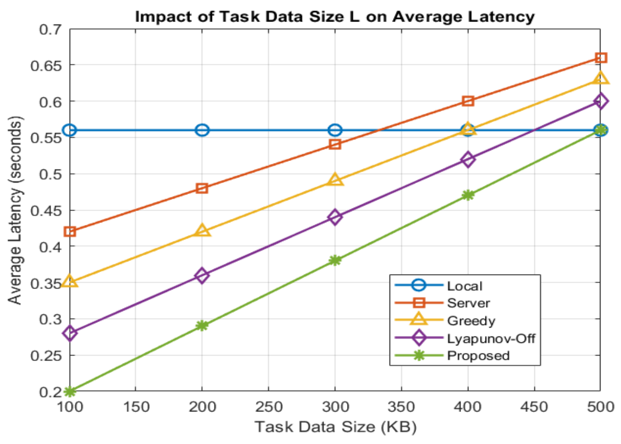

Fig. 7. Impact of the task data size L.

6. CONCLUSION

This work is devoted to analyzing offloading tasks in the MEC systems, taking into account the stochastic characteristics of the arriving tasks and the dynamics of the corresponding wireless channels. We have proposed an online method that uses Lyapunov optimization for making offloading decisions and assigning computing resources. The approach is very efficient in the required computer resources and does not need statistical system dynamics information. So, we've shown that our method can reduce mobile device power consumption while still achieving latency performance that's competitive with the best offline implementation. In computational models grounded on real-world traces, the proposed technique reduced power consumption and latency relative to the baseline systems.

7. FUTURE WORK

Consider advanced multi-tier MEC architectures with heterogeneous processing and communication resources. Use tier collaboration and interaction to boost system performance. Including user mobility and dynamic network architecture in issue formulation and algorithm design. Offloading decisions must account for mobility trends and service migration costs. Using deep reinforcement learning in the Lyapunov optimization framework to learn optimal offloading and resource allocation policies from historical data and adapt to time-varying system dynamics. Studying how data privacy and security affect MEC offloading decisions. Performance optimization and privacy protection must be balanced. To test the algorithm in practice and identify issues, prototype and implement it in a real-world MEC testbed.

REFERENCES

[1] P. Mach et. al. "Mobile Edge Computing: A Survey on Architecture and Computation Offloading," IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1628-1656, 2017.

[2] Y. Mao, et. al. "A Survey on Mobile Edge Computing: The Communication Perspective," IEEE Communications Surveys & Tutorials, vol. 19, no. 4, pp. 2322-2358, 2017.

[3] N. Abbas, et. al. "Mobile Edge Computing: A Survey," IEEE Internet of Things Journal, vol. 5, no. 1, pp. 450-465, Feb. 2018.

[4] S. Wang, et. al. "A Survey on Mobile Edge Networks: Convergence of Computing, Caching and Communications," IEEE Access, vol. 5, pp. 6757-6779, 2017.

[5] M. J. Neely, "Stochastic Network Optimization with Application to Communication and Queueing Systems," Morgan & Claypool Publishers, 2010.

[6] L. Huang, et. al. "Deep Reinforcement Learning for Online Computation Offloading in Wireless Powered Mobile-Edge Computing Networks," IEEE Transactions on Mobile Computing, vol. 19, no. 11, pp. 2581-2593, Nov. 2020.

[7] Yang, Ning, et al. "Beyond the Edge: An Advanced Exploration of Reinforcement Learning for Mobile Edge Computing, its applications, and Future Research Trajectories." arXiv preprint arXiv:2404.14238 (2024).

[8] Y. Mao, et. al. "Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices," IEEE Journal on Selected Areas in Communications, vol. 34, no. 12, pp. 3590-3605, Dec. 2016.

[9] J. Liu, et. al., "Delay-Optimal Computation Task Scheduling for Mobile-Edge Computing Systems," in Proc. IEEE International Symposium on Information Theory (ISIT), Barcelona, Spain, Jul. 2016, pp. 1451-1455.

[10] [M. J. Neely, "Stochastic Network Optimization with Application to Communication and Queueing Systems," Morgan & Claypool Publishers, 2010.

[11] Cui, Hanshuai, et al. "Latency-Aware Container Scheduling in Edge Cluster Upgrades: A Deep Reinforcement Learning Approach." IEEE Transactions on Services Computing (2024).

[12] W. Zhang, et. al. "Energy-Optimal Mobile Cloud Computing under Stochastic Wireless Channel," IEEE Transactions on Wireless Communications, vol. 12, no. 9, pp. 4569-4581, Sep. 2013.

[13] J. Kwak, et. al. "DREAM: Dynamic Resource and Task Allocation for Energy Minimization in Mobile Cloud Systems," IEEE Journal on Selected Areas in Communications, vol. 33, no. 12, pp. 2510-2523, Dec. 2015.

[14] S. Bi et. al. "Computation Rate Maximization for Wireless Powered Mobile-Edge Computing with Binary Computation Offloading," IEEE Transactions on Wireless Communications, vol. 17, no. 6, pp. 4177-4190, Jun. 2018.

[15] F. Wang, J. et. al. "Joint Offloading and Computing Optimization in Wireless Powered Mobile-Edge Computing Systems," IEEE Transactions on Wireless Communications, vol. 17, no. 3, pp. 1784-1797, Mar. 2018.

[16] Y. Dai, et. al. "Joint Computation Offloading and User Association in Multi-Task Mobile Edge Computing," IEEE Transactions on Vehicular Technology, vol. 67, no. 12, pp. 12313-12325, Dec. 2018.

[17] Omidvar, Naeimeh, Mahdieh Ahmadi, and Seyed Mohammad Hosseini. "Optimal Service Placement, Request Routing and CPU Sizing in Cooperative Mobile Edge Computing Networks for Delay-Sensitive Applications." arXiv preprint arXiv:2405.10648 (2024).

[18] Tang, Jine, et al. "Federated Learning-Assisted Task Offloading Based on Feature Matching and Caching in Collaborative Device-Edge-Cloud Networks." IEEE Transactions on Mobile Computing (2024).

[19] He, Yejun, et al. "Computation offloading and resource allocation based on DT-MEC-assisted federated learning framework." IEEE Transactions on Cognitive Communications and Networking (2023).

[20] Zhao, Yunzhi, et al. "Joint offloading and resource allocation with diverse battery level consideration in MEC system." IEEE Transactions on Green Communications and Networking (2023).

[21] Yang, Jian, et al. "Cooperative task offloading for mobile edge computing based on multi-agent deep reinforcement learning." IEEE Transactions on Network and Service Management (2023).

[22] Chen, Yishan, et al. "A Game-Theoretic Approach Based Task Offloading and Resource Pricing Method for Idle Vehicle Devices Assisted VEC." IEEE Internet of Things Journal (2024).