International Journal of Applied Mathematics

Volume 38 No. 3 2025, 436–446

ISSN: 1311-1728 (printed version); ISSN: 1314-8060 (on-line version)

BIPHASIC EFFICIENCY MODELING FOR STABLE OFFLOADING IN RESOURCE CONSTRAINED EDGE NETWORKS USING PUREEDGESIM

Amit Malik[1], Amita Rani[2]

[1]Department of Computer Science and Engineering, SRM University,
Delhi-NCR, Sonepat, Haryana, India.
Email: amit.m@srmuniversity.ac.in, ORCID: 0009-0008-0407-7183

[2]Department of Computer Science and Engineering, DCRUST,
Murthal, Sonepat, Haryana, India.
Email: amitamalik.cse@dcrustm.org, ORCID: 0000-0002-7385-4045

Abstract: Current edge computing research strongly favors complex deep learning for managing resources. However, these data-heavy models often clash with the physical needs of edge devices, such as low latency, low energy, and total predictability. This study examines the trade-offs between unpredictable learning methods and the proposed deterministic approach, Biphasic Efficiency Model (BEM). Performance evaluation in PureEdgeSim shows that BEM maintains a stable completion rate between 38% and 44%, while traditional methods collapse to nearly 7% under high workload.

Keywords: Edge Computing, Task Offloading, Biphasic Efficiency Model, Deterministic, Deep Learning.

Received: 29 March, 2025

1. INTRODUCTION

In the foundational era of sensor networks and satellite communications, systems were largely designed to be tolerant of delays. This approach is frequently referred to as the store-and-forward model. In these environments, if a network link failed or a server became overwhelmed, the data was simply placed in a queue to wait until resources became available. The speed of the immediate response was considered less critical than the eventual reliability of the delivery [1].

However, the emergence of the modern computing edge appears to have changed this priority. Contemporary applications such as autonomous driven vehicles, industrial robotics, and emergency safety systems operate on a strict principle of delay intolerance [2]. For a drone attempting to stabilize its flight path against sudden wind resistance, a computational decision that is delayed by even a fraction of a second late is often equivalent to a system failure. The primary objective of edge computing is to bring processing power closer to the physical source of data to eliminate these transmission delays. Still, there exists a potential conflict at the core of this design. These high-speed applications are increasingly being deployed on hardware that is physically constrained. Edge nodes frequently operate with limited battery reserves, modest processing capabilities, and intermittent connectivity. Consequently, the industry is attempting to run delay-intolerant software on infrastructure that behaves similarly to a delay-tolerant network [3].

This contradiction presents a significant challenge for resource orchestration. The system must decide in real time where to execute a task to ensure it meets strict timing deadlines. To address this, the research community has exceedingly turned toward Artificial Intelligence. The prevailing hypothesis suggests that because the edge environment is volatile and unpredictable, the software managing it must possess a high degree of adaptability [4]. This has resulted in a steep rise in studies proposing Deep Learning and Reinforcement Learning models to manage edge resources. The underlying logic is that a sophisticated agent can learn the hidden patterns of network traffic and predict the optimal path for data transmission.

While this approach holds theoretical promise, it may not be practically feasible in some scenarios. By embedding complex learning models directly into the edge, there is a risk of reintroducing the very latency that the system aims to eliminate. These models are often computationally intensive [5]. They require significant time to train, and they consume energy to generate decisions. In an environment where real-time responses are required, an algorithm that consumes excessive time to calculate an optimal move becomes a liability. It acts as a bottleneck. This effectively introduces delay into the control loop in the pursuit of intelligence.

In contrast to these complex methods, many industrial implementations prefer to avoid such overhead. These systems often rely on simple and deterministic rules. Common strategies include Round Robin, which assigns tasks to servers in a strictly sequential order. Other systems employ Greedy algorithms that simply select the node with the highest amount of available storage at that specific moment [6]. These methods are often favored practically because they are computationally fast and their behavior is predictable. However, these simplified methods appear to suffer from a different form of limitation. They operate on the assumption that computer performance scales in a linear fashion. The assumption is that a server running at near-maximum capacity is still fully functional and simply requires marginally more time to complete its workload.

This assumption of linearity appears to be mathematically imprecise when applied to high-load scenarios. Computing systems do not behave like simple containers that can be filled to capacity without consequence. Evidence suggests they behave more like complex traffic systems. As a server approaches high utilization, it begins to experience internal friction. Tasks begin to compete for access to memory, storage locks, and processor cycles. This state is referred to as contention. When contention occurs, performance does not degrade gradually [7]. It tends to degrade rapidly. A server operating near its physical limit may spend more time managing the internal queue than actually processing data. Simple heuristic rules often fail to detect this tipping point. They continue to direct tasks to a node that appears available on paper but is physically on the verge of saturation. This triggers a backlog and forces the system back into a delay-tolerant state where tasks must wait in long queues to be processed.

This paper explores a third methodological path and proposes that the solution may not lie in the addition of more artificial intelligence, or in the reliance on static linear rules. The solution may lie in a more accurate model of which is able to capture the relationship between system workload and its efficiency. This paper introduces it as a Biphasic Efficiency Model (BEM), with the understanding that initially the distribution of workload increases the efficiency of the system but then overload results into performance degradation. By mathematically modeling this transition, it is possible to derive a control strategy that retains the speed of a simple rule while maintaining the stability required for preventing performance dip [8].

The remainder of this paper is organized as follows. Section 2 briefly reviews the related work in edge orchestration and identifies the limitations of current linear and stochastic approaches. Section 3 details the mathematical formulation of the Biphasic Efficiency Model and describes the mechanics of conflict in computing systems. Section 4 presents the experimental setup using the PureEdgeSim simulator, where the proposed deterministic strategy is compared against standard Round Robin and Greedy algorithms. Section 5 discusses the performance results, highlighting the trade-offs between speed and stability. Finally, Section 6 concludes the study and outlines future research directions, specifically focusing on the development of more adaptive and robust biphasic modeling techniques.

2. RELATED WORK

The evolution of resource management at the edge has transitioned through several distinct paradigms, moving from rigid heuristics to complex adaptive models. The trend observed in initial deployments favored simplicity for the sake of execution speed, as showcased by [1], [3] and [9]. These early efforts prioritized low-latency decision-making by utilizing static rules that did not account for the fluctuating nature of network traffic or node availability.

As the complexity of edge environments grew, research shifted toward more sophisticated solvers. In [10] and [11], the ML/DL approaches are explained to handle the inherent unpredictability of mobile edge networks. These studies demonstrate that deep reinforcement learning can effectively navigate multi-dimensional search spaces to find optimal offloading paths. However, as noted in recent critiques, these models often introduce significant computational jitter and high energy overhead, which can be counterproductive in real-time, power-constrained scenarios. Parallel to the rise of AI-driven methods, deterministic strategies have remained a staple in industrial applications due to their predictability [12]. The linear method used in provides a clear example of how threshold-based balancing is applied to distribute tasks. While such methods are mathematically transparent and fast, they frequently rely on the assumption that performance scales evenly with load. This oversight leads to a failure in detecting the "knee of the curve," where internal system contention begins to degrade throughput. Consequently, there is a visible gap in the literature for a model that combines the speed of deterministic rules with a non-linear understanding of system physics.

3. BIPHASIC EFFICIENCY MODEL (BEM)

The discussion in this section begins with the critical observation that traditional resource management strategies in edge computing often consider system workload as a linear quantity. In most standard models, it is assumed that performance decreases steadily and predictably as more tasks are added to a processing node [13]. However, empirical evidence and a deep analysis of high-density computing environments suggest a more complex, biphasic relationship that cannot be captured by simple linear slopes [14]. In the first phase of this relationship, which can be described as constructive stimulation, a computing node often operates at a suboptimal efficiency level when under-utilized. This is largely because the fixed overhead of the system, including power leakage, basic operating system maintenance, and active cooling, remains high regardless of the workload. As the distribution of tasks increases, the system reaches a peak state where these static costs are effectively balanced against a higher rate of successful task completion, and hardware pipelines are kept sufficiently full to justify the energy expenditure.

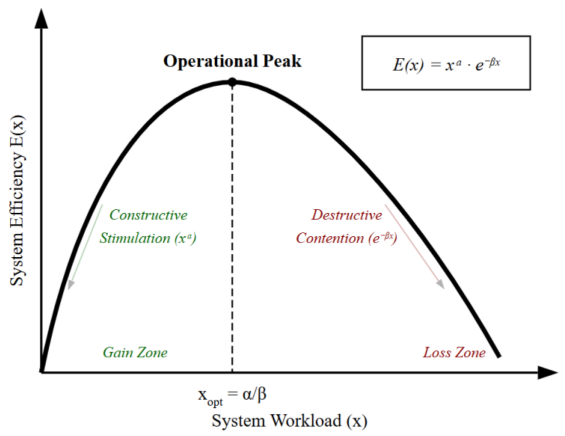

Figure 1: The Biphasic Efficiency Model (BEM) curve illustrating the non-monotonic relationship between workload and efficiency.

As depicted in the figure 1, the initial rise in efficiency eventually reaches a tipping point where a second, more dangerous phase, known as destructive contention, begins to dominate the system dynamics. Once this physical threshold is crossed, the addition of more tasks no longer results in higher throughput or better resource utilization. Instead, the hardware begins to suffer from internal conflicts that are inherent to the physics of computing architectures. These conflicts include memory bus saturation, cache thrashing, and excessive context switching, where the processor spends more time managing the administrative overhead of multitasking than it does executing actual application logic.

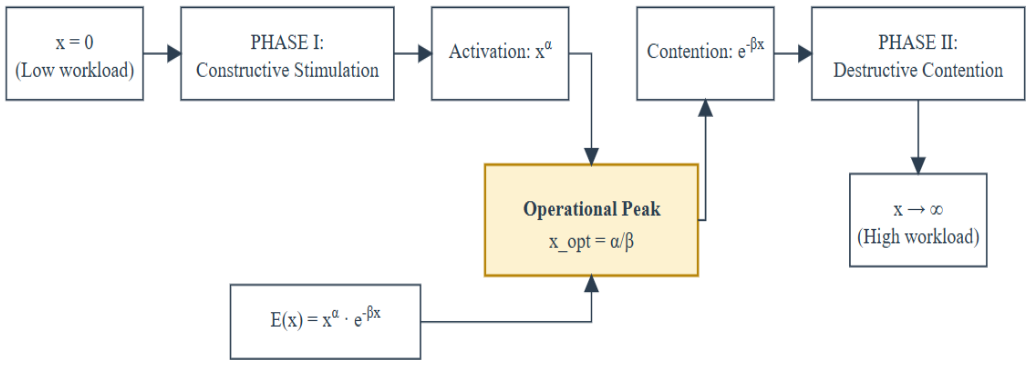

Figure 2: Flowchart mapping the logical progression from initial low-workload states to the identification of the operational peak.

This transition from beneficial resource use to harmful saturation creates an inverted-U trajectory that defines the actual performance boundaries of the hardware. Recognizing the "knee" or the apex of this curve is essential for edge orchestration, as it represents the precise point where a system achieves its highest potential without risking a sudden collapse into a state of heavy queuing and delay.

To capture this behavior mathematically and provide a deterministic alternative to heavy learning models, the state of the system can be formalized through a non-linear efficiency function that treats workload as a form of system pressure. This model proposes that the total efficiency of an edge node is the product of two competing forces that act in opposition to one another and models it as Equation 1:

$$E(x) = x^{\alpha} \cdot e^{-\beta x} \qquad \ldots (1)$$

Figure 2 illustrates that the first force is an activation term, which models the initial rise in performance as hardware resources are engaged and fixed overhead is overcome. This is represented by a power function, x^{α}, where x is the normalized workload currently assigned to the node, ranging from zero to one $(0 \leq x \leq 1)$. The exponent α represents the hardware's ability to scale with increasing load, reflecting factors such as pipeline depth and multi-threading efficiency. The second force is a contention term, which models the exponential increase in internal system conflict as the node approaches its physical limits. This is represented by an exponential decay function, $e^{-\beta x}$, where the coefficient β accounts for the specific sensitivity of the system to congestion and resource locking.

When these two terms are multiplied, they form the complete Biphasic Efficiency Model (BEM), formalized as Equation 1. This continuous and predictable curve allows the orchestrator to identify the optimal operational threshold through basic calculus rather than trial-and-error training. It is possible to pinpoint the exact workload level where the marginal gain from resource activation is perfectly balanced by the marginal cost of contention by calculating it as per Equation 2,

$$x_{opt} = \frac{\alpha}{\beta} \qquad \ldots (2)$$

by equating $\frac{dE}{dx} = 0$. In a real-world orchestration scenario, a controller based on this deterministic model does not simply search for the node with the lowest current load. Instead, it evaluates the current state of all available candidate nodes and selects the one where the addition of a new task will move the system's state closest to that identified peak. This mechanism allows the orchestrator to act with the execution speed of a simple reflex while maintaining a sophisticated, context-aware understanding of the boundaries that prevent performance degradation and system collapse. By keeping nodes in this "hot" but safe operational zone, the system maintains its delay-intolerant nature even under heavy demand.

4. SIMULATION SETUP

To evaluate the performance of the proposed Biphasic Efficiency Model (BEM) against traditional orchestration strategies, the study utilizes PureEdgeSim [15], a specialized toolkit designed for simulating large-scale edge computing environments.

Table 1: Key simulation parameters for biphasic efficiency model evaluation

Parameter		Value
Simulator	:	PureEdgeSim
Simulation area	:	2000 m x 2000 m
Network type	:	wlan
Simulation durations	:	1h, 2h, 4h
Number of edge devices	:	400, 800
Number of edge datacenters	:	9
Datacenter range	:	400 m
Datacenter cores	:	8–12
Datacenter MIPS	:	30,000 – 50,000
Datacenter RAM	:	16 GB – 32 GB
Edge device types	:	4 (sensor, wearable, camera, fixed compute)

Task generation rate	**:**	26 tasks/sec per device
Offloading scenario	**:**	EDGE_ONLY (no cloud fallback)

As given by Table 1, the simulation environment is configured within a $2000m \times 2000m$ area, representing a dense urban deployment. The infrastructure consists of nine edge datacenters, each equipped with 8 to 12 CPU cores, processing speeds ranging from 30,000 to 50,000 MIPS, and RAM between 16 GB and 32 GB. These datacenters communicate via a WLAN network and serve a dynamic population of edge devices, ranging from 400 to 800 units. To reflect real-world heterogeneity, four distinct device types are modeled: sensors, wearable, cameras, and fixed computing units. Each device generates tasks at a high frequency of 26 tasks per second, creating a high-pressure workload environment. The simulation is conducted under an EDGE_ONLY offloading scenario, intentionally removing cloud fallback to strictly test the resilience and stability of the edge nodes under stress. Simulations are executed for durations of 1, 2, and 4 hours to observe long-term system behavior. The BEM strategy is compared against the Round Robin and Trade-off (load-balancing) algorithms, both of which are natively available in PureEdgeSim. To implement the BEM framework, custom modifications were made to the Simulation Manager and Orchestrator modules. These changes allow the simulator to move beyond linear task distribution by calculating the non-linear efficiency of candidate nodes in real-time, enabling the orchestrator to target the optimal operational peak rather than simply seeking the lowest available load.

5. RESULT AND ANALYSIS

The performance of the proposed BEM Model is evaluated against the inbuilt Trade-off and Round Robin algorithms of PureEdgeSim Simulator. The evaluation is done across varying number of device (400 and 800) and simulation intervals (1, 2 and 4 Hour). The empirical results, summarized in Table 2, demonstrate a significant performance gap between the proposed deterministic model and standard linear heuristics.

Table 2: Comparative performance analysis of orchestration algorithms across different simulation durations and device loads

No. Of Devices	Simulation Time (Hour)	Algorithm	Total Task	Tasks Completed (%)	Latency(s)
400	1	BEM	16,58,679	41.65	0.079
		TRADE_OFF	16,58,001	13.01	0.047
		ROUND_ROBIN	16,57,940	13.89	0.043
	2	BEM	32,00,231	43.78	0.062
		TRADE_OFF	32,99,677	09.74	0.032
		ROUND_ROBIN	31,91,492	09.88	0.034
	4	BEM	64,23,751	40.38	0.067

		TRADE_OFF	64,67,552	07.12	0.036
		ROUND_ROBIN	46,54,259	07.91	0.040
800	1	BEM	32,17,784	38.21	0.697
		TRADE_OFF	32,15,347	12.45	0.061
		ROUND_ROBIN	32,23,982	12.74	0.061
	2	BEM	64,76,318	37.80	0.834
		TRADE_OFF	64,37,324	09.10	0.049
		ROUND_ROBIN	64,34,669	10.30	0.057
	4	BEM	1,27,87,287	39.01	1.033
		TRADE_OFF	1,27,38,267	06.99	0.049
		ROUND_ROBIN	1,27,51,482	07.28	0.052

The most notable observation is the substantial increase in the task completion percentage achieved by BEM. While the Trade-off and Round Robin strategies struggle to maintain efficiency as the simulation duration increases, BEM consistently completes approximately 38% to 43% of the total generated tasks. In contrast, the baseline algorithms experience a sharp decline in success rates. For instance, at 4-hour duration with 800 devices, the Trade-off and Round Robin algorithms drop to a completion rate of roughly 7%, while BEM maintains a robust 39.01%. This is a validation of the core principle that by identifying the Operational Peak x_{opt}, BEM prevents nodes from entering the "Loss Zone" of destructive contention, whereas standard methods continuously push nodes into congestion.

The data in Table 2 also reveals an expected trade-off regarding system latency. BEM exhibits higher average latency which ranges from 0.062 seconds in low-density scenarios to 1.033 seconds under high-stress conditions (800 devices, 4 hours). While the baseline algorithms report lower average latency (approx. 0.05 seconds), these figures are misleading as they only reflect the small fraction of tasks (~7%) that were successfully processed.

The higher latency in BEM is a direct consequence of its ability to keep more tasks "alive" in the system. By refusing to blindly offload tasks to saturated nodes, BEM ensures that a much larger volume of work is eventually completed, rather than being discarded due to system thrashing. As the number of devices doubles from 400 to 800, BEM successfully manages over 12.7 million tasks in the 4-hour window, proving its stability in high-pressure, resource-constrained edge environments.

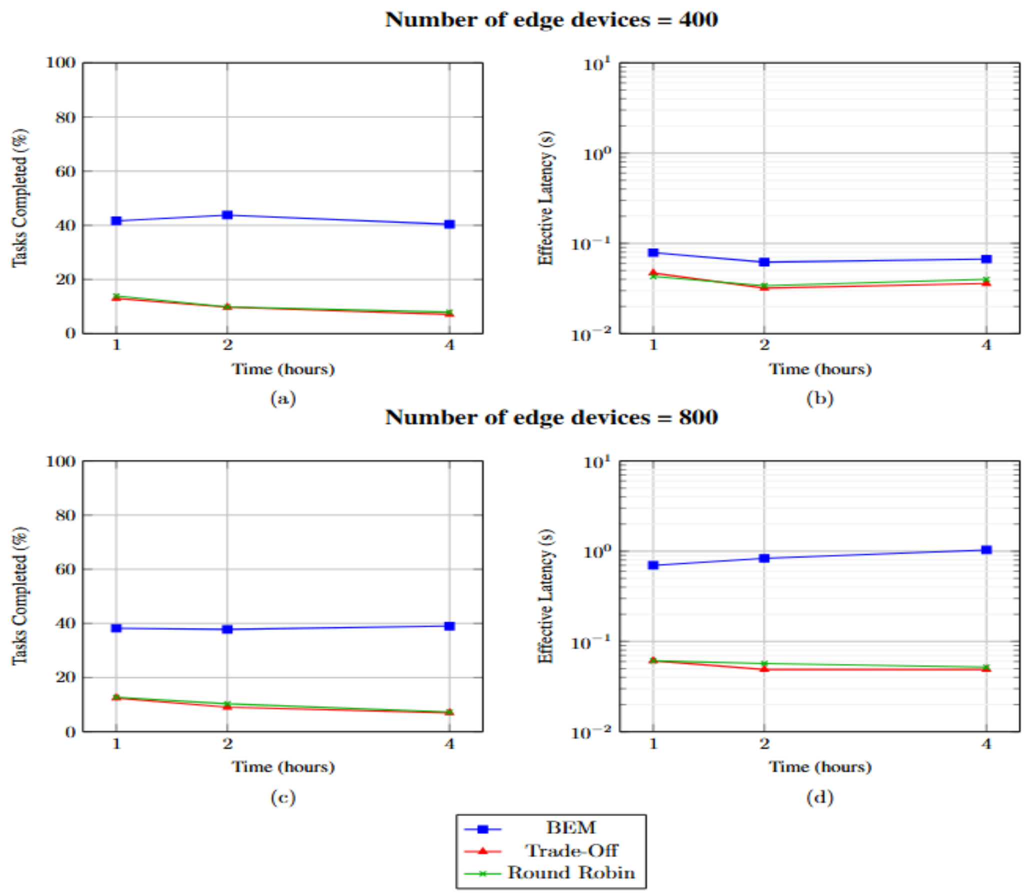

Figure 3: Performance comparison for BEM, Trade-Off, and Round Robin orchestrators. Subfigures (a, b) show results for 400 devices and (c, d) for 800 devices, evaluating Task Completion Rate (a, b) and Effective Latency (c, d).

The performance trends established in the comparative analysis are further validated by the graphical data plotted in Figure 3 above, which illustrates the critical advantage of the Biphasic Efficiency Model (BEM) over traditional linear strategies. The task completion plots (Figures 3a and 3c) reveal a significant "performance collapse" in both the Trade-Off and Round Robin strategies as simulation time progresses. While BEM maintains a relatively stable completion rate between 38% and 44% regardless of device density, the baseline heuristics show a continuous downward trend. As the workload increases, these traditional methods drop to completion rates as low as approximately 7%, visually representing the transition where nodes cross from the "Gain Zone" of constructive stimulation into the "Loss Zone" of destructive contention. The latency analysis (Figures 3b and 3d) clarifies the trade-offs required for system stability in resource-constrained networks. Although BEM exhibits higher average latency but the seemingly lower latency reported by baseline algorithms is statistically biased. This is because Trade-Off and Round Robin only process a small fraction of tasks.

6. CONCLUSION

The results of this study demonstrate that the Biphasic Efficiency Model (BEM) provides an efficient deterministic alternative to traditional linear offloading methods. By shifting the orchestration paradigm from simple load balancing to a context-aware analysis of the "Operational Peak," the model successfully prevents edge nodes from degrading into states of destructive behaviour. Empirical evaluations using PureEdgeSim confirm that BEM maintains significantly higher task completion rates which ranges between 38% and 44% even as system workload increases, whereas standard Round Robin and Trade-off strategies suffer from a rapid performance decay to approximately 7%. While BEM introduces higher latency, but it is a result of managing a much larger number of tasks, it ensures system-wide stability and prevents the total throughput degradation observed in baseline methods. Despite these advantages, there is a clear path for improvement to bridge the gap between this deterministic model and the contemporary state-of-the-art (SOTA) solutions. Future work may focus on evolving the BEM into an adaptive model where the method is able to dynamically get tuned in real-time through lightweight feedback loops. Such improvements would allow the model to compete more effectively with SOTA algorithms. Additionally, future research may explore the integration of energy-aware constraints to further optimize the trade-off between task completion and power consumption in resource-constrained edge environments.

7. REFERENCES

[1] S. R. Das, K. Sinha, N. Mukherjee, and B. P. Sinha, "Delay and Disruption Tolerant Networks: A Brief Survey," 2021. doi: 10.1007/978-981-15-5971-6_32.

[2] E. P. van Horssen, J. A. A. van Hooijdonk, D. Antunes, and W. M. Heemels, "Event- and Deadline-Driven Control of a Self-Localizing Robot With Vision-Induced Delays," *IEEE Transactions on Industrial Electronics*, Feb. 2020, doi: 10.1109/TIE.2019.2899553.

[3] R. Yu and G. Xue, "Principles and Practices for Application-Network Co-Design in Edge Computing," *IEEE Network*, Jan. 2022, doi: 10.1109/mnet.128.2200430.

[4] M. B. F. Sanjetha, Y. Kanagaraj, V. Herath, and S. Lokuliyana, "Deep Learning for Edge Computing Applications: A Comprehensive Survey," *Asian journal of computer science and technology*, Dec. 2022, doi: 10.51983/ajcst-2022.11.2.3456.

[5] "Challenges and opportunities in edge computing architecture using machine learning approaches," 2022. doi: 10.1016/b978-0-12-824054-0.00002-2.

[6] C. Comte and C. Comte, "Dynamic Load Balancing with Tokens," May 2018. doi: 10.23919/IFIPNETWORKING.2018.8697018.

[7] D. Dice and A. Kogan, "Avoiding Scalability Collapse by Restricting Concurrency," 2019.

[8] T. Kontogiannis and S. Malakis, “A system dynamics approach to the efficiency thoroughness tradeoff,” *Safety Science*, Oct. 2019, doi: 10.1016/J.SSCI.2019.06.011.

[9] C.-H. Hong and B. Varghese, “Resource Management in Fog/Edge Computing: A Survey on Architectures, Infrastructure, and Algorithms,” *ACM Computing Surveys*, Sep. 2019, doi: 10.1145/3326066.

[10] X. Liu, S. Jiang, and Y. Wu, “A Novel Deep Reinforcement Learning Approach for Task Offloading in MEC Systems,” *Applied Sciences*, Nov. 2022.

[11] I. Akturk and U. R. Karpuzcu, “Trading Computation for Communication: A Taxonomy of Data Recomputation Techniques,” *IEEE Transactions on Emerging Topics in Computing*, Jan. 2021, doi: 10.1109/TETC.2018.2883286.

[12] M. H. Najafi and D. J. Lilja, “High Quality Down-Sampling for Deterministic Approaches to Stochastic Computing,” *IEEE Transactions on Emerging Topics in Computing*, vol. 9, no. 1, pp. 7–14, Jan. 2021, doi: 10.1109/TETC.2017.2789243.

[13] R. Shi, Y. Gan, and Y. Wang, “Evaluating Scalability Bottlenecks by Workload Extrapolation,” *Modeling, Analysis, and Simulation On Computer and Telecommunication Systems*, pp. 333–347, Sep. 2018, doi: 10.1109/MASCOTS.2018.00039.

[14] M. Jin and J. Lavaei, “Stability-Certified Reinforcement Learning: A Control-Theoretic Perspective,” *IEEE Access*, vol. 8, pp. 229086–229100, Dec. 2020.

[15] C. Mechalikh, H. Taktak, and F. Moussa, “PureEdgeSim: A simulation framework for performance evaluation of cloud, edge and mist computing environments,” *Computer Science and Information Systems*, vol. 18, no. 1, pp. 43–66, Jan. 2021, doi: 10.2298/CSIS200301042M.