Running Virtual Services for the Intelligent Edge: A Review

Kevin Afachao[1]
[1]*Department of Electrical, Electronic and Computer Engineering*
University of Pretoria
Pretoria, 0028, South Africa
u22851217@tuks.co.za

Adnan M. Abu-Mahfouz[1,2]
[2]*Council for Scientific and Industrial Research (CSIR)*
Pretoria, 0083, South Africa
a.abumahfouz@ieee.org

***Abstract*— Emerging technologies like Artificial Intelligence, the Internet of Things, Virtualization and Edge Computing are driving research towards making the Intelligent Edge conducive for varied application growth. Virtualization is adopted because it provides an adaptive development of services. Artificial Intelligence provides promising data manipulation opportunities while Edge Computing grants lower latency and higher data privacy to users. This article highlights significant architecture for virtual service deployment in the Intelligent Edge. The first basic concepts are presented: Virtualization, Network Function Virtualization, Artificial Intelligence and the Intelligent Edge. Then various virtual services modifications are discussed such as the Round Robin Algorithm, the Minimum-Minimum algorithm, the Multiheuristic Algorithm and the Particle Swarm Optimization Algorithm. The algorithms are contrasted on performance indicators such as makespan, standard deviation, degree of imbalance and processing speed. Results vary from algorithm to algorithm, indicating room for improvements. In addition the paper presents future opportunities for virtual service deployment in Intelligent Edge Computing.**

Keywords—Internet of Things, edge computing, artificial intelligence, scheduling algorithms, virtualization

I. INTRODUCTION

Big technology companies such as Google, Microsoft and Amazon are making strides in the Internet of Things (IoT) by offering commercially viable applications that run Artificial Intelligent (AI) services at the edge of the network, namely Microsoft Azure IoT Edge, Amazon's Greengrass, and Google Cloud IoT Edge [1]. Although these are also referred to as Edge applications, they rely heavily on Cloud processing and cannot handle real-time applications such as live video streaming, Augmented Reality (AR) tasks and real-time object detection scenarios due to runtime delays. A recent framework that combines AI and Edge computing called the Intelligent Edge (IE) solves this problem and offers additional privacy and security enhancements for the network. The goal of the Intelligent Edge is a framework that combines data processing, data analytics, data storage, deep learning inference, deep learning training, and privacy and security systems all at the network edge [2]. With recent advancements in hardware computing technology, and increasing production of user data accompanied by vast service deployment scenarios, Intelligent Edge adapts services to users, whiles providing low latency, low communication cost and localized data storage.

The number and complexity of application scenarios for IoT devices are increasing. However, these tasks can be divided into sub-tasks using virtualization technology to reduce their complexity. Despite this, the application requirements such as flexibility of hardware, low latency, high data throughput, and high quality of service (QoS) are still not sufficiently satisfied by Intelligent Edge computing [3]. Some of the challenges faced include how to adequately manage resources at the edge for training and inference of deep learning models while meeting the minimum requirements for a good user experience. IoT devices at the edge of the network are application-specific and constrained due to a lack of resources. Additionally, they have a smaller battery lifespan compared to cloud data servers. Therefore, the Intelligent Edge computing architecture needs to embrace collaboration with technologies such as Virtualization, Software Defined Networking, and Network Function Virtualization to boost service provision [4].

The motivation for this literature survey is to present a novel scope of virtualized services in the Intelligent Edge, its advances, and opportunities. Simulations are further conducted to compare existing techniques implemented to meet the inadequacies of service deployment.

The paper is outlined as follows. In the next section, the background is presented, we continue by discussing the various studies of virtualized services in the Intelligent Edge, baseline algorithms and opportunities for research. Finally, the paper ends with a conclusion.

II. BACKGROUND

Several surveys have discussed different aspects of virtual services in different computing environments. Service placement and migration surveys consider where to place the virtual services during runtime and how to deploy such services [5],[6]. Resource allocation and system consolidation surveys consider schemes to optimize overall system performance for virtual service continuity [7]. Surveys on security, explore the methods and policies to control and mitigate cyber-attacks at the network edge [8]–[10]. Literature reviews on content caching and computation offloading present works on how devices can save content for reuse as well as save device battery lifespan through efficient computation[11]–[14]. Task scheduling involves assigning resources to applications in a queue and deploying them based on a priority scheme. Surveys in this area are presented in [22]. However, this study presents a survey that concerns running virtual services for the Intelligent Edge, with emphasis on reduction in execution delay.

The early forms of edge computing, such as distributed clouds [15], aimed to improve the user experience by propagating cloud computing resources across a network. However, the distributed cloud architecture relies solely on the cellular network, which introduces operational costs and unstable user service continuity. This is because cloud content providers must compromise with network service providers. In the distributed cloud data center, the migration of Virtual Machines (VM) faces the challenge of execution latency, considering that VM services introduce delay during deployment.

Urgaonkar et al [16] presented the challenge of determining where a user's content and services are migrated when they move to a new area. They emphasized the need for a policy to choose a node for service migration, but this poses a challenge as user request patterns, the number of users, and the number of applications are unknown and difficult to model. They also noted that using only the Markov Decision Process (MDP) to model the system is impractical since it cannot account for user mobility and service request arrival values. To overcome this challenge, they proposed a control algorithm called Lyapunov optimization, which decouples the MDP and solves it using non-stochastic shortest path optimization. The Lyapunov optimization algorithm effectively reduces the queue backlog, even though it incurs additional migration costs. Taleb et al [17] introduced Follow Me Cloud (FMC), which was implemented using two deployment schemes, Location-Identifier Service Protocol (LISP) and Software Defined Networking (SDN). They used MDP to model service migration, but also considered 2D service mobility. The updated version of FMC performed better than previous versions. In addition to addressing the challenges of user mobility and service migration, there is also a need to consider the security of the Intelligent Edge. As more devices are connected to the network, the attack surface increases, and there is a greater risk of cyber threats. This has led to the development of edge security mechanisms that can protect the devices and data at the edge. Some of these mechanisms include firewalls, intrusion detection systems, and data encryption. However, there is still a need for more research in this area to develop more effective security mechanisms that can address the unique challenges of the Intelligent Edge.

In [18], the authors designed a scheduling scheme, KnapSOS for virtual services in Fog Computing to meet this challenge of execution latency. Fog nodes employ VMs to extend network resources by creating virtual Cloudlets for the network edge. Against schemes that exclude virtual services, results show improvements in energy consumption, network utilization, execution costs and node lifetime expectancy. In [19], researchers designed a dynamic graph partitioning theory to ensure load balancing among distributed virtual Fog computing systems. By introducing lightweight virtualization, [20] implements container virtualization in collaboration with task scheduling and load reallocation mechanisms to realize a reduction in task execution delays in smart manufacturing IoT application scenarios. In a recent study, [21] formulate decision policy algorithms for task scheduling in virtual edge systems using DRL, with the goal of profit maximization. The algorithms outperform baselines in terms of service delay, computation cost, energy consumption and task execution.

The distributed and heterogeneous nature of IoT puts infrastructure under pressure to deliver quality services to users. Scheduling schemes are employed to ensure the maximization of profit, the satisfaction of users, efficient resource allocation and reduction of execution latency at the network edge [7]. There are many task scheduling and workload distribution methods for Edge Computing architectures [6], however, there are not many papers dedicated to task scheduling for virtual services in Intelligent Edge computing.

III. SCHEDULING VIRTUAL SERVICES

Virtual services address different problems and challenges for the Intelligent Edge through various algorithms and models. Applications are made up of unique sub-tasks that are responsible for features in the application. These tasks all together make the application function optimally and can be executed as virtual services. Multiple VMs are created and allocated to each task for resource maximization.

A. Key Parameters

1) Execution time (ET_i*):* The expected time it takes for a VM to execute a task in milliseconds.

2) Completion time (CT_{ij}*):* The time in milliseconds for the VM to execute a task until the next task arrives. It usually comprises the execution time and the ready time.

3) *Make Span:* The time interval between execution initiations for a process in a VM until its termination.

4) Ready time (r_j*):* The time in milliseconds that the VM resource remains idle until the next task is assigned.

5) Time quantum: The allocated time in milliseconds for which the scheduler must process a task.

B. Algorithms

1) Minimum-Minimum (Min-Min) Algorithm: The main goal of the Min-Min Algorithm is to reduce the makespan for processes run in the VM [22]. Intelligent Edge applications vary in the computational demand required to meet each application which implies that with multiple users' service requests, edge servers get congested trying to meet all the requests. VMs extend the capacity of physical devices implementing the services, however executing the requests on a First-Come-First-Serve (FCFS) does not make efficient utilization of resources because heavy computation tasks can become a bottleneck to the entire system. The Min-Min algorithm ensures that computation is utilized efficiently by running timesaving virtual services first before other services. Although the Min-Min algorithm ensures computation efficiency, it is not context-aware and executes tasks irrespective of context priority. Context awareness is a characteristic of Intelligent Edge Computing because the Intelligent Edge considers the priority of task context before executing them. For instance, emergency services irrespective of computational time should get run first before other services.

2) *Multiheuristic Min-Min Algorithm:* Multiheuristic Min-Min, a variation of the Min-Min algorithm, considers the case where the initial task–machine pair is incapable of being run by the Virtual Machine due to incompatibility [20]. Hence, the alternative pair chosen should have a similar minimum value as the preferred choice. Incompatibility occurs when a virtual service cannot be deployed in a container due to the configuration of that container. This is not captured by the Min-Min algorithm. Multiheuristic Min-Min Algorithm suffers from the defects of the Min-Min algorithm in that it does not consider application context in

service execution making it less ideal for Intelligent Edge Computing.

3) *Round Robin Algorithm:* The round-robin scheduling algorithm is designed to allocate a minimum amount of time to each task known as the time quantum, to reduce VM idle time [21]. The VM idle time is the time the VM waits until another task is received. In practise the Round Robin algorithm is most efficient for modular virtual services run at the edge, however monolithic virtual services with computational requirements greater than the time quantum face waiting queue delays. For instance at service initiation the scheduler creates a waiting queue for tasks. At the arrival of the first task, the VM executes this task within the quantum given. Any task that arrives while the VM is busy is put in the waiting queue until the task is completed and the next task is received. For a task that remains incomplete, it is added to the waiting queue once again as the latest arrival awaiting processing. This leads to unforeseen service delays experienced by users especially when multiple services are initiated.

4) Particle Swarm Optimization (PSO): PSO algorithm mimics animal behaviour in swarms, example ants and migratory birds, indicating the unit's performance to fit into the group [22]. The algorithm optimises the performance of several tasks by employing randomness and probabilities expressed as velocity to allow the swarm to learn the optimal performance. Each unit in the swarm moves with a stochastic velocity and constantly updates its position from the previous best position to the current best position. Concurrently, the unit aims to reach the global best position, which indicates the unit's overall best performance in the swarm. In Intelligent Edge applications, the PSO algorithm continuously modifies the system resources given the system parameters until a desirable state is attained. The downside to this algorithm lies with the search space. In large-scale implementations the swarm prolongs the modification in attempt to reach the best performance. However only after several iterations, does the system attain a global best position which is desirable in the long term.

IV. Simulations

A. Simulation workflow

The simulation models a real-time environment by first creating figures of devices to be employed, then creating logical components of the system and finally, the management policies are realized. The CloudSim simulator is used in this study. An adjunctive model called ReCloud is used to compare the various algorithms in the Edge-Cloud simulation. The simulations are carried out by following many procedures: initializing ReCloud, creating servers for the cloud, creating cloudlets, creating brokers, creating VMs, creating tasks, declaring algorithms for scheduling, simulating the environment, and monitoring results. The ReCloud library allows comparisons between scheduling algorithms initiated in the broker. Although an extension of CloudSim, ReCloud is not preconfigured to provide components like sensors and actuators.

B. Environment setup

In this review, the implementation involves three server nodes, one as a cloud server and the other two, as gateway servers, three VMs are created, three brokers and three hosts. The tasks are designed to be distributed evenly across the VMs with sizes of 1MB and 1000 MIPS. The setup is run for three iterations and the average is taken. In this simulation the cloudlets and virtual machines are configured as follows, the virtual machine is set with 9726 MIPS, 512 MB RAM, 1000 MB/s bandwidth and 10000 MB image size with a single processing element. The host servers are set with 177730 MIPS, 16000 MB of RAM, 15000 MB/s bandwidth, and 4000000 MB of storage with six processing elements. Table I shows the simulation environment setup.

TABLE I. Simulation parameter settings for the experiment

PARAMETER	**VALUES**
Number of tasks	200
Number of VMs	3
Number of hosts	3
MIPS of VM	9726
MIPS of host	177730
ram of VM (MB)	512
ram of host (MB)	16000
bandwidth of VM (MB/s)	1000
bandwidth of host (MB/s)	15000
VM Image Size (MBs)	10000
Storage on Host (MBs)	4000000

C. Results

The experiment is run over 200 tasks in three rounds and the average values of the algorithms are illustrated in figures. The bar graph in Fig. 1 indicates the average performance of the algorithms in terms of makespan. Fig. 2 shows the standard deviation of each of the algorithms. Standard deviation measures the spread of the values from the average makespan recorded. This measure indicates how varied the makespan recorded for each task varies from the expected value. The lower the deviation the closer the values are to the standard value, the higher the deviation, the more distant the values are from the average value. The Round-Robin algorithm records the lowest standard deviation, with the Multiheuristic algorithm following and the PSO algorithm coming in third whereas the Minimum-Minimum algorithm records the highest values for standard deviation. Round Robin algorithm is handicapped when the quantum allocated to the processor is large. With a large quantum, the recorded makespan and service delay increases rapidly. Thus with an increasing number of tasks, the round-robin algorithm quickly descends to a First-Come-First-Serve (FCFS) algorithm. In the simulation, the Minimum-Minimum algorithm works with the execution times of the tasks and locates the pair with the minimum task-to-VM execution ratio, however, the delays experienced in the operating system implementing this algorithm refer to several configurations on the VM, that make certain tasks incompatible with particular virtual machines.

Fig. 3 shows the degree of imbalance among the VMs. The degree of imbalance is simply a measure of the maximum completion time in addition to the minimum completion time, divided by the average completion time across VMs. The Minimum-minimum algorithm records the highest degree of imbalance because in executing the shortest jobs first, the resources across the VMs are not used efficiently. For instance, VMs responsible for handling heavy computing tasks would remain idle whiles the VMs which handle light computations with shorter execution times remain busy. Multiheuristic Min-Min corrects this by considering an additional condition to ensure load rebalancing. The PSO algorithm follows after the Minimum-Minimum algorithm, whiles the Round Robin algorithm records the least values. In Fig. 4 the processing speeds are compared among the mentioned algorithms. The PSO algorithm records the highest processing speed due to the search space expansion when dealing with a huge number of tasks, this means many more computations are performed than necessary leading to an increase in the processing time. The Minimum-Minimum algorithm, Multiheuristic Algorithm and Round Robin record similar values. The round-robin algorithm which uses a standard quantum of 100ms per task makes efficient use of the processing unit since computation length is minimized as well as idle time. The PSO algorithm was introduced to optimize service delays specifically. The processing speed of tasks for the PSO algorithm is large and hence contributes to the recorded delay, however, with an increasing number of tasks, the system is optimized to produce desirable results. Thus, PSO records high performance for large-scale systems than for small-scale systems. Moreover, PSO utilises basic mathematical operators compared to other algorithms which make use of matrix-level computations, this makes the processing relatively better.

There have been many variations of PSO, Round Robin, and Minimum-Minimum algorithms because these algorithms serve as the basis for many job scheduling algorithms. Ideally, these algorithms vary in how they assign tasks to the scheduler, how they utilize the task queues and how they manage resources to monitor task completion and redeployment for Intelligent Edge Computing applications.

V. FUTURE DIRECTIONS

A. *Emerging Technologies*

Cloud service architecture often relies on service deployment schemes such as Microservices, which enable the deployment of multiple small services to achieve a common goal. Microservices offer language-agnostic and independent building blocks that are well-suited for the Intelligent Edge environment, which comprises distributed software frameworks. Big tech companies like Netflix and Amazon are already testing microservices in their applications [25]. One of the benefits of microservices is easy deployment and reconfiguration of services, made possible by lightweight virtualization technology like Docker. Docker Swarm, Kubernetes, and Mesos are examples of container orchestration technologies that enable complex and dynamic orchestration and maintenance of microservices [26]. Emerging technologies like Software Defined Networking (SDN) [6], [22] and Information-Centric Networking (ICN) [27] are also being considered for deploying virtual services at the Intelligent Edge. SDN offers scalability, availability, resilience, and interoperability, making it an ideal technology for deploying any type of service on machines. ICN, on the other hand, provides a robust and efficient service architecture that ensures minimal delays during service updates or migration, and it is constructed with IoT and Intelligent Edge Computing as the focus, making it more compatible than traditional IP networks.

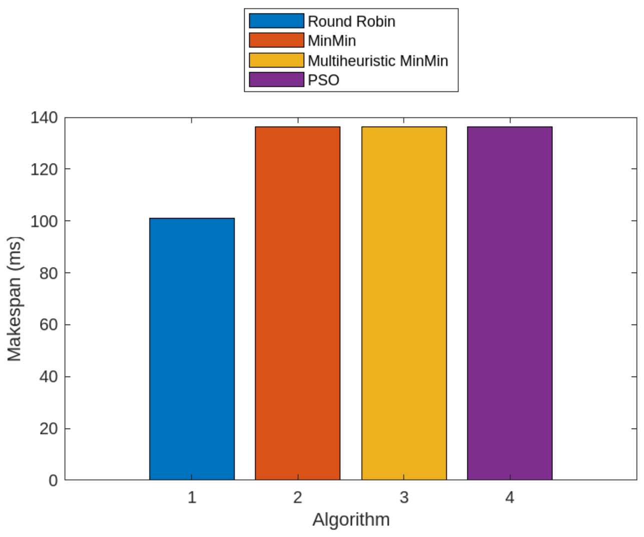

Fig 1. A bar graph of the average makespan recorded from the algorithms.

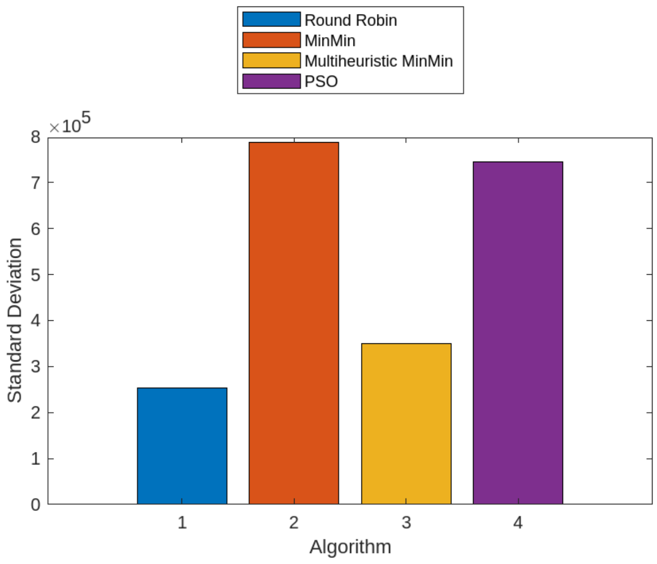

Fig.2 A bar graph of the standard deviation recorded from the algorithms.

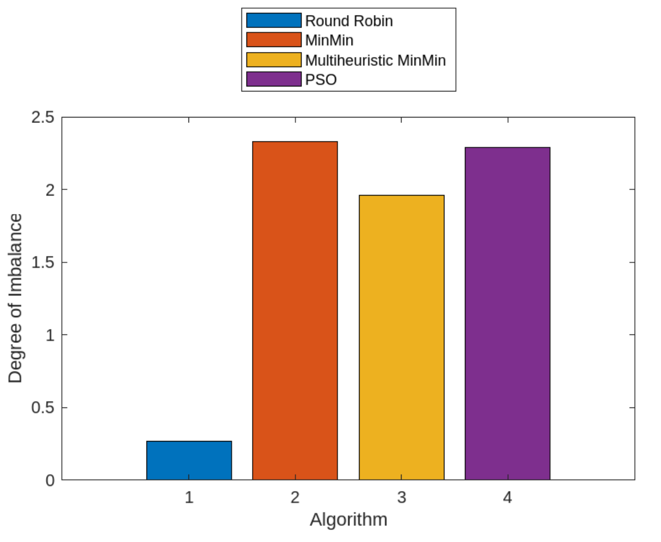

Fig.3 A bar graph of the degree of imbalance recorded from the algorithms.

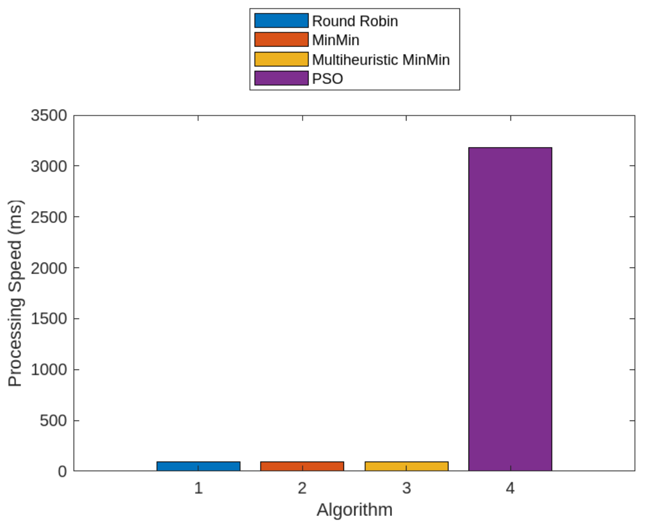

Fig.4 A bar graph of the processing speeds recorded from the algorithms.

Another emerging technology that has been gaining attention in the context of Intelligent Edge Computing is serverless computing. Serverless computing allows developers to build and run applications without having to manage the underlying infrastructure. This approach can significantly reduce costs and simplify application development, deployment, and scaling. Platforms such as AWS Lambda, Azure Functions, and Google Cloud Functions have been widely adopted and are increasingly being used for Intelligent Edge Computing applications. With serverless computing, developers can focus on writing code and designing applications, while leaving the underlying infrastructure and scaling to the cloud provider.

B. *Frameworks for Optimizing Intelligent Edge Computing*

In addition to the technologies mentioned earlier, recent developments in the field of intelligent edge computing include Arm's ArmNN and Arm Computing Library, which provide efficient execution of neural network models on Arm-based processors, and Qualcomm's Snapdragon Neural Processing Engine, which is designed to optimize neural network performance on mobile devices. The AI-enabled technologies placed a demand for computation resources which inhibits the performance of edge computing devices since the hardware available is limited in its computational power. The need has given rise to libraries and hardware providing increased processing speeds and lightweight sizes ideal for the network edge. Currently, CPU chip designs are taking a step further by providing edge nodes with extra computing power through the incorporation of neuromorphic chips, TPUs, GPU and FPGAs. Hardware matched with light frameworks like TensorFlow Lite, Caffe2, and Arm Computing Library give virtual services an expansive domain to provide novel ideas like Intelligent Edge As a Service (EIaaS), Resource as a Service(RaaS) and also Security as a Service(SaaS).

C. *Varying Application Scenarios*

Applications at the edge require different levels of priority based on their demands. While some smart applications need a real-time response and run at the edge, others require a best-effort response. For instance, online gaming demands high throughput, but even slight delays can lead to service degradation [1]. Applications that need a real-time response, such as virtual and augmented reality, terminate when they encounter latency delays. As such, future applications will require a balance between real-time response and best effort. The Intelligent Edge plays a crucial role in enabling applications such as Telepresence, Ubiquitous Computing, and the Internet of Everything. Telepresence [6], which is an extension of mixed reality, allows users to be present in another location and respond to stimuli from that location. Another promising future application scenario is Gadget-Free computing [26], where users can interact with services through the environment without carrying mobile devices. The Intelligent Edge ensures that content and services are available to users even as they move from one location to another. Furthermore, advancements in technologies such as 5G, AI, and IoT are facilitating the development of new applications at the Intelligent Edge, opening up new possibilities for the future.

VI. CONCLUSIONS

With the increasing popularity of Edge computing and Artificial Intelligence (AI), there is now greater potential for complex AI applications to be implemented in the volatile environment of the Intelligent Edge. To support this, various technologies, including virtualization, are being introduced to enable the efficient deployment of virtual services at the Edge. This literature review has identified four different approaches to virtual service deployment and evaluated their performance using the CloudSim simulator. The round-robin algorithm was found to have the best average makespan, assuming a minimal quantum, and comparison was made based on factors such as standard deviation, degree of imbalance, and processing speeds. Further research opportunities were also highlighted, particularly in developing algorithms that can minimize service delay and execute tasks in the shortest possible time.

REFERENCES

[1] Z. Zhou, X. Chen, E. Li, L. Zeng, K. Luo, and J. Zhang, “Edge Intelligence: Paving the Last Mile of Artificial Intelligence With Edge Computing,” *Proceedings of the IEEE*, 2019, doi: 10.1109/JPROC.2019.2918951.

[2] J. Chen and X. Ran, "Deep Learning With Edge Computing: A Review," *Proceedings of the IEEE*, 2019, doi: 10.1109/JPROC.2019.2921977.

[3] X. Wang, Y. Han, V. C. M. Leung, D. Niyato, X. Yan, and X. Chen, "Convergence of Edge Computing and Deep Learning: A Comprehensive Survey," *IEEE Communications Surveys and Tutorials*, vol. 22, no. 2. Institute of Electrical and Electronics Engineers Inc., pp. 869–904, Apr. 01, 2020. doi: 10.1109/COMST.2020.2970550.

[4] Y. Y. Shih, H. P. Lin, A. C. Pang, C. C. Chuang, and C. T. Chou, "An NFV-Based Service Framework for IoT Applications in Edge Computing Environments," in *IEEE Transactions on Network and Service Management*, Dec. 2019, vol. 16, no. 4, pp. 1419–1434. doi: 10.1109/TNSM.2019.2948764.

[5] S. Wang, R. Urgaonkar, T. He, M. Zafer, K. Chan, and K. K. Leung, "Mobility-induced service migration in mobile micro-clouds," in *Proceedings - IEEE Military Communications Conference MILCOM*, Nov. 2014, pp. 835–840. doi: 10.1109/MILCOM.2014.145.

[6] H. Tabatabaee Malazi *et al.*, "Dynamic Service Placement in Multi-Access Edge Computing: A Systematic Literature Review," *IEEE Access*, vol. 10, pp. 32639–32688, 2022, doi: 10.1109/ACCESS.2022.3160738.

[7] L. Ni, J. Zhang, C. Jiang, C. Yan, and K. Yu, "Resource Allocation Strategy in Fog Computing Based on Priced Timed Petri Nets," *IEEE Internet Things J*, vol. 4, no. 5, pp. 1216–1228, Oct. 2017, doi: 10.1109/JIOT.2017.2709814.

[8] Y. Xiao, Y. Jia, C. Liu, X. Cheng, J. Yu, and W. Lv, "Edge Computing Security: State of the Art and Challenges," *Proceedings of the IEEE*, 2019, doi: 10.1109/JPROC.2019.2918437.

[9] S. Lal, T. Taleb, and A. Dutta, "NFV: Security Threats and Best Practices," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 211–217, May 2017, doi: 10.1109/MCOM.2017.1600899.

[10] J. Pan and Z. Yang, "Cybersecurity challenges and opportunities in the new 'edge computing + iot' world," in *SDN-NFVSec 2018 - Proceedings of the 2018 ACM International Workshop on Security in Software Defined Networks and Network Function Virtualization, Co-located with CODASPY 2018*, Mar. 2018, vol. 2018-January, pp. 29–32. doi: 10.1145/3180465.3180470.

[11] J. Yao, T. Han, and N. Ansari, "On Mobile Edge Caching," *IEEE Communications Surveys and Tutorials*, vol. 21, no. 3, pp. 2525–2553, Jul. 2019, doi: 10.1109/COMST.2019.2908280.

[12] L. C. Mutalemwa and S. Shin, "A classification of the enabling techniques for low latency and reliable communications in 5G and beyond: Ai-enabled edge caching," *IEEE Access*, vol. 8, pp. 205502–205533, 2020, doi: 10.1109/ACCESS.2020.3037357.

[13] H. Zhu, Y. Cao, W. Wang, T. Jiang, and S. Jin, "Deep Reinforcement Learning for Mobile Edge Caching: Review, New Features, and Open Issues," *IEEE Netw*, vol. 32, no. 6, pp. 50–57, Nov. 2018, doi: 10.1109/MNET.2018.1800109.

[14] R. S. Alonso, J. Prieto, F. de la Prieta, S. Rodriguez-Gonzalez, and J. M. Corchado, "A Review on Deep Reinforcement Learning for the management of SDN and NFV in Edge-IoT," in *2021 IEEE Globecom Workshops, GC Wkshps 2021 - Proceedings*, 2021. doi: 10.1109/GCWkshps52748.2021.9682179.

[15] T. Taleb and A. Ksentini, "An analytical model for follow me cloud," in *GLOBECOM - IEEE Global Telecommunications Conference*, 2013, pp. 1291–1296. doi: 10.1109/GLOCOM.2013.6831252.

[16] R. Urgaonkar, S. Wang, T. He, M. Zafer, K. Chan, and K. K. Leung, "Dynamic service migration and workload scheduling in edge-clouds," *Performance Evaluation*, vol. 91, pp. 205–228, Sep. 2015, doi: 10.1016/j.peva.2015.06.013.

[17] T. Taleb, A. Ksentini, and P. A. Frangoudis, "Follow-me cloud: When cloud services follow mobile users," *IEEE Transactions on Cloud Computing*, vol. 7, no. 2, pp. 369–382, Apr. 2019, doi: 10.1109/TCC.2016.2525987.

[18] D. Rahbari and M. Nickray, "Scheduling of Fog Networks with Optimized Knapsack by Symbiotic Organisms Search." in *FRUCT*, 2018, pp.278-283

[19] N. Song, C. Gong, X. An, and Q. Zhan, "Fog computing dynamic load balancing mechanism based on graph repartitioning," *China Communications*, vol. 13, no. 3, pp. 156–164, Mar. 2016, doi: 10.1109/CC.2016.7445510.

[20] L. Yin, J. Luo, and H. Luo, "Tasks Scheduling and Resource Allocation in Fog Computing Based on Containers for Smart Manufacturing," *IEEE Trans Industr Inform*, vol. 14, no. 10, pp. 4712–4721, Oct. 2018, doi: 10.1109/TII.2018.2851241.

[21] P.Gazori, D. Rahbari, and M. Nickray, "Saving time and cost on the scheduling of fog-based IoT applications using deep reinforcement learning approach" *Future Generation Computer Systems*, vol. 110, pp.1098-1115, Sep 2020.

[22] I. D. Filip, F. Pop, C. Serbanescu, and C. Choi, "Microservices scheduling model over heterogeneous cloud-edge environments as support for IoT applications," *IEEE Internet Things J*, vol. 5, no. 4, pp. 2672–2681, Jan. 2018, doi: 10.1109/JIOT.2018.2792940.

[23] J. Yao, J. Guo, and L. N. Bhuyan, "Ordered Round-Robin: An efficient sequence preserving packet scheduler," *IEEE Transactions on Computers*, vol. 57, no. 12, pp. 1690–1703, 2008, doi: 10.1109/TC.2008.88.

[24] H. S. Al-Olimat, M. Alam, R. Green, and J. K. Lee, "Cloudlet scheduling with particle swarm optimization," in *Proceedings - 2015 5th International Conference on Communication Systems and Network Technologies, CSNT 2015*, Sep. 2015, pp. 991–995. doi: 10.1109/CSNT.2015.252.

[25] Z. Chang, S. Liu, X. Xiong, Z. Cai, and G. Tu, "A Survey of Recent Advances in Edge-Computing-Powered Artificial Intelligence of Things," *IEEE Internet of Things Journal*, vol. 8, no. 18. Institute of Electrical and Electronics Engineers Inc., pp. 13849–13875, Sep. 15, 2021. doi: 10.1109/JIOT.2021.3088875.

[26] E. Harjula *et al.*, "Decentralized Iot edge nanoservice architecture for future gadget-free computing," *IEEE Access*, vol. 7, pp. 119856–119872, 2019, doi: 10.1109/ACCESS.2019.2936714.

[27] T. le Duc, R. G. Leiva, P. Casari, and P. O. Östberg, "Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey," *ACM Comput Surv*, vol. 52, no. 5, Sep. 2019, doi: 10.1145/3341145.

[28] S. Bansal and D. Kumar, "IoT Ecosystem: A Survey on Devices, Gateways, Operating Systems, Middleware and Communication," *Int J Wirel Inf Netw*, vol. 27, no. 3, pp. 340–364, Sep. 2020, doi: 10.1007/s10776-020-00483-7.

[29] M. Zolotukhin, T. Hamalainen, T. Kokkonen, and J. Siltanen, "Increasing web service availability by detecting application-layer DDoS attacks in encrypted traffic," in *2016 23rd International Conference on Telecommunications, ICT 2016*, Jun. 2016. doi: 10.1109/ICT.2016.7500408.

[30] N. M. Yungaicela-Naula, C. Vargas-Rosales, and J. A. Perez-Diaz, "SDN-based architecture for transport and application layer DDoS attack detection by using machine and deep learning," *IEEE Access*, vol. 9, pp. 108495–108512, 2021, doi: 10.1109/ACCESS.2021.3101650.