A New Task Scheduling Framework for Internet of Things based on Agile VNFs On-demand Service Model and Deep Reinforcement Learning Method

Li YANG
Department of Electronic Information Engineering, Leshan Vocational and Technical College,
Leshan 614099, Sichuan, China

***Abstract*—Recent innovations in the Internet of Things (IoT) have given rise to IoT applications that require quick response times and low latency. Fog computing has proven to be an effective platform for handling IoT applications. It is a significant challenge to deploy fog computing resources effectively because of the heterogeneity of IoT tasks and their delay sensitivity. To take advantage of idle resources in IoT devices, this paper presents an edge computing concept that offloads edge tasks to nearby IoT devices. The IoT-assisted edge computing should meet two conditions, edge services should exploit the computing resources of IoT devices effectively and edge tasks offloaded to IoT devices do not interfere with local IoT tasks. Two main phases are included in the proposed method: virtualization of edge nodes, and task scheduling based on deep reinforcement learning. The first phase offers a layered edge framework. In the second phase, we applied deep reinforcement learning (DRL) to schedule tasks taking into account the diversity of tasks and the heterogeneity of available resources. According to simulation results, our proposed task scheduling method achieves higher levels of task satisfaction and success than existing methods.**

***Keywords*—Internet of things; task scheduling; edge computing; resource allocation**

I. INTRODUCTION

The recent rapid development of artificial intelligence [1, 2], machine learning [3], optical networks [4, 5], smart grids [6], cloud computing [7, 8], 5G connectivity [9], Blockchain, and Internet of Things (IoT) [10, 11] is leading to an exponential growth in data usage across a wide range of engineering and commerce disciplines. Over the last decade, the IoT has been recognized for its remarkable potential in computer science. It brings out an environment in which many intelligent objects with limited resources can interact with each other through different technologies [12]. The IoT tries to link the physical and virtual worlds by equipping physical devices with processing, networking, detection, and identification functions. The IoT objects are employed in different applications, such as vehicle networks, energy management, traffic control, medical treatment, and healthcare, aiming to gather information about the physical world [13, 14]. In this regard, to obtain valuable information and fulfil the assigned tasks, a massive amount of data is produced that brings challenging problems for efficient information processing, especially for the specific scenarios requiring real-time data handling. Considering edge computing capability in real-time processing, the computing IoT tasks can be offloaded to edge devices for implementation [15, 16].

Moreover, since the IoT services and applications are increasing daily, a practical approach to serving the growing needs in different application domains becomes vital [17, 18]. To address the mentioned problems, achieve high resource utilization, reduce communication costs, and improve the lifetime of IoT networks, task scheduling methods can have significant impacts [19, 20]. These methods aim to schedule the existing tasks in a suitable sequence to accomplish tasks under problem-specific constraints, such as communication costs among IoT objects, resource utilization, and the operational lifetime of sensor nodes [21].

Nowadays, Network Function Virtualization (NFV) is known as a superior technology in telecommunication networks. It refers to centralizing and virtualizing network functions that can be run in data centres on standard and commercial off-the-shelf (COTS) hardware instead of distributed and proprietary hardware [22]. Using NFV, network device purchases and related maintenance costs can be reduced effectively. As a logical consequence of NFV, Virtual Network Functions (VNFs) take advantage of middleware to virtualize network functions. The deployment of VNFs on commodity hardware reduces the need for dedicated hardware devices to perform individual network functions. Our main objective is to improve edge computing operational efficiency by implementing the agile VNF on-demand model and the deep reinforcement learning-based task scheduling method. The proposed method includes two main phases, virtualization of edge nodes and task scheduling based on the deep reinforcement learning method. In the first step, a layered edge framework is presented. In the second step, we employed deep reinforcement learning (DRL) to solve task scheduling problems considering the tasks' diversity and the heterogeneity of available resources. This article makes the following major contribution.

- We analyze the optimization of time scheduling and the assignment of virtual machines in edge computing using model-free DRL-based task scheduling. VM availability, task characteristics, and queue dynamics are considered in the formulation of the problem as an MDP problem.

- In this case, the action is represented as a pair of VMs and tasks, whose dimension may be extremely large. In the MDP formulation, a new mechanism is designed to decouple the scheduling time step from the real-time step. With this mechanism, the action space remains linear with the product of the number of VMs and the queue size, and multiple tasks can be scheduled simultaneously.

The rest of the paper in organized as follows. Related works are reviewed in Section II. The proposed task scheduling method is described in Section III. Section IV reports the simulation results. Finally, the conclusion is presented in Section V.

II. Related Work

Al-Habob, Dobre [23] considers offloading multiple tasks concurrently to several mobile edge computing servers. Offloading latency and failure probability are minimized by scheduling interdependent subtasks for servers. Models based on conflict graphs and genetic algorithms are used to solve scheduling problems. Experimental findings demonstrate that these algorithms approach optimal solutions found by exhaustive searches. In addition, even though parallel offloading employs orthogonal channels, sequential offloading is more likely to fail than parallel offloading. In contrast, parallel offloading provides lower latency. Latency gaps between parallel and sequential schemes decrease as the dependency between sub-tasks increases.

Chen, Guo [24] examine the business processes of edge cloud task scheduling. The authors propose an algorithm to optimize resource-constrained task scheduling using the construction of profit matrices, classification, and clustering preprocessing. Tasks with similar characteristics are grouped and categorized using clustering preprocessing. On the basis of the constructed profit matrix, a resource-constrained task scheduling strategy is derived. In the second step, edge cloud components such as virtual machines, user requests, tasks, and resources are constructed using Petri nets. As a final step, the suggested mechanism is evaluated through several experiments. According to simulations, the algorithm maximizes profit while managing tasks efficiently, reliably, and with a high load balance.

A mathematical formulation of the fog node task scheduling problem is presented by Azizi, Shojafar [25] to reduce fog node energy consumption while maintaining IoT QoS requirements. The goal of their model was to reduce deadline violation time. The authors proposed two partially greedy algorithms, semi-greedy with priority awareness and multi-start semi-greedy with priority awareness, to map IoT tasks to fog nodes efficiently. The proposed approaches were evaluated in terms of system lifespan, latency, energy, and the IoT task completion rate. In comparison with existing algorithms, tests indicate that the introduced algorithms improve task deadline compliance and reduce the total time spent violating deadlines.

Kanbar and Faraj [26] developed the Region Aware Dynamic Scheduling (RADISH) model, which consists of five consecutive processes. To reduce latency in scheduling, they implemented a bi-class neural network based on task nature to classify incoming tasks based on their nature, taking into account login credentials, emails, passwords, types of services, and quality of service parameters. In the second process, the improved moth flame optimization is used to schedule classified tasks considering workload, deadline, priority, and energy. Third, load balancing is accomplished by clustering potential fields. With the aim of balancing server load and improving efficiency, three repository systems have been implemented. The final step involves introducing the Hopcroft-Karp algorithm that takes into account the VM state and minimizes allocation times and enhances the quality of service.

Hybrid Flamingo Search with a Genetic Algorithm (HFSGA) is implemented by Hussain and Begh [27] to optimize task scheduling for cost minimization. HFSGA and other well-known optimization algorithms are compared on seven essential benchmark optimization functions. Furthermore, Friedman Rank Tests are conducted to ascertain the results' significance. Implementing the model produces better results regarding task completion percentages, makespan, and costs. This work shows better results than existing algorithms such as round-robin, genetic, PSO, and ACO.

In order to secure the allocation of tasks on cloud and fog nodes, Najafizadeh, Salajegheh [28] proposed a multi-objective simulated annealing algorithm. A compromise solution is found by applying the goal-programming approach. In addition, a new goal called client-driven access level and schedule is created in regard to distributing tasks among fog and cloud nodes. The proposed algorithm was found to be 50% more efficient regarding deadlines, 88% more efficient regarding control levels, and 10% more efficient regarding service delays in comparison to moth-flame optimization, tabu search, and PSO algorithms.

Task scheduling in edge computing requires the consideration of two special problems: time scheduling and resource allocation. The task execution order is determined by time scheduling, while resource allocation is responsible for allocating tasks to suitable virtual machines (VMs) for execution. In the field of edge computing, a number of scheduling issues have been explored [29-33]. However, the majority of existing works focus on resource allocation, while time scheduling has received little attention. Tan, Han [34] proposed a general model to minimize task response times when tasks are offloaded to edge servers. Zhang, Du [35] proposed a scheduling algorithm based on Lyapunov optimization in order to minimize the communication delay and the computing delay. Chen, Thomas [36] developed a dual-scheduling framework to accommodate the unstable capacity of servers and task arrival rates in heterogeneous vehicular edge computing. In [37], a mixed integer nonlinear programming (MINLP) algorithm was employed for data-parallel offloading and scheduling of computationally-intensive data-parallel tasks in order to minimize the average completion time. According to [38], tasks with the lowest delay are scheduled first using the shortest-job-first (SJF) scheduling method.

Alameddine, Sharafeddine [39] explored the use of device-to-device collaboration for task offloading by taking into consideration the mobility of humans in order to optimize the task assignment and power management. The issue of energy-efficient task scheduling for IoT edge computing has been addressed in [20] by a heuristic algorithm. In these methods, ideal mathematical models are used and optimization is achieved through mixed-integer nonlinear programming (MINLP) or heuristic algorithms. In spite of the fact that these model-oriented algorithms can produce excellent results, they are not well suited to dynamic environments in which task arrival rates and popularity are unknown in advance. It is also important to note that the model-based task scheduling algorithms are largely concerned with optimizing one-step instead of pursuing long-term objectives. In these algorithms, the availability of resources is assumed to be fixed during the scheduling period.

III. Proposed Task Scheduling Method

In our proposed mechanism, each gateway node serves as a connection point for several IoT nodes connected to the gateway node based on their shortest distance. Our main aim is to propose an edge computing service model based on gateways to maximize IoT resource utilization, accelerate the processing of users' service requests, and enhance edge computing efficiency. The proposed model involves user requests passing through the edge gateway, which determines whether to process tasks. To decrease data processing time, the controller forwards service requests to the cloud if the edge gateway is unable to process them. The edge computing service model contains three main components, lightweight VNF configuration, scheduler, and resource estimation. According to the proposed model, when a request for a service is received, it is checked whether or not an adequate amount of computing resources is available to fulfil such demand. In this regard, one of the following events may occur. Depending on the outcome of the task scheduling algorithm, the edge gateway processes the requests through the scheduler and queues them in the system. On the other hand, requests are transferred directly to the cloud if the edge gateway lacks the resources.

Improving edge gateway operational efficiency is the main aim of task scheduling methods. Since the service requests are different from each other, a task scheduling algorithm determines how to meet the demands of each request. A scheduling method reduces time spent on tackling high-demand tasks as a primary objective. Our main objective is implementing on-demand models for agile VNFs and deep reinforcement learning-based task scheduling methods to enhance edge computing efficiency. The proposed method includes two main phases, virtualization of edge nodes and task scheduling based on the deep reinforcement learning method.

In order to simplify the task scheduling process, we will only focus on computational resources. In order to make scheduling decisions, the scheduler monitors the status information of incoming tasks and virtual machines (VMs), including the task sizes, the expected completion time, the computing speed (in million instructions per second (MIPS)), and the waiting time. The scheduler determines when to schedule (e.g., the scheduling order and the start time for each task) and where to schedule (e.g., which VM is assigned to each task) based on the observations. In order to schedule the tasks, they are divided into two sets: a waiting set and a backlog queue. A task in the waiting set occupies a waiting slot that can be observed fully, whereas the scheduler can only observe the number of tasks in the backlog queue. In each scheduling time step, the scheduler selects at most one task from the waiting slot for scheduling. In this study, we examine the scheduling of tasks in edge computing when only one edge server is deployed. It is the objective to maximize the long-term task satisfaction of all tasks, which is achieved by:

$$\boldsymbol{max} \sum_{\boldsymbol{t=1}}^{\boldsymbol{T}} \sum_{\boldsymbol{i \in J, j \in V}} \boldsymbol{g_{i,j}} \quad (1)$$

where $g_{i,j}$ is the task satisfaction of the task i scheduled to VM j.

A. First Phase: Virtualization of Edge Nodes

This step proposes a virtualized edge framework to support Cloud-to-Things applications at all layers. The virtualization process involves three stages: virtualizing objects, network functions, and services. The implementation of object virtualization allows physical sensors to gain IP capability without compromising their unique functionalities. In order to represent heterogeneous physical entities, we need to create a unified software Virtual Object (VO) on edge nodes. An abstract VO is capable of interacting over the Internet with various hosts.

Additionally, it can act as a close neighbour to physical objects through wireless or wired connections available at the edge. VOs provide semantic descriptions of actual objects. Although physical objects have heterogeneous functions, they generally have limited memory and components such as communication modules, sensing modules, and power supply units. These components can be expanded and installed on edge devices for real physical objects as virtual software instances.

As illustrated in Fig. 1(a), two critical streams of object virtualization are taken into account to illustrate VO-hosting solutions: hardware-level and OS-level virtualization. VOs are compatible with established operating systems. Edge players, such as providers, developers, and end users, can develop dedicated hosting platforms with advanced features, such as memory, hardware interface, and CPU, in order to achieve better performance. Fig. 1(b) provides more details about the sensor virtualization framework. Both virtual sensor instructions and physical sensor data are stored in the "Sync Flag". Version numbers are used to synchronize actual and virtual objects. "Energy Manager" indicates the battery life of physical sensors and turns them on. Compared to physical sensors, virtual sensors are composed of V-communication, V-processing, and V-sensing components. In the left column, "Actuator Flag" contains the instructions given to the physical sensor, and "Sensor Flag" holds the sensor's collected data. V-processing is the equivalent of physical sensor processing. A predefined local or network storage device can be used as an external memory.

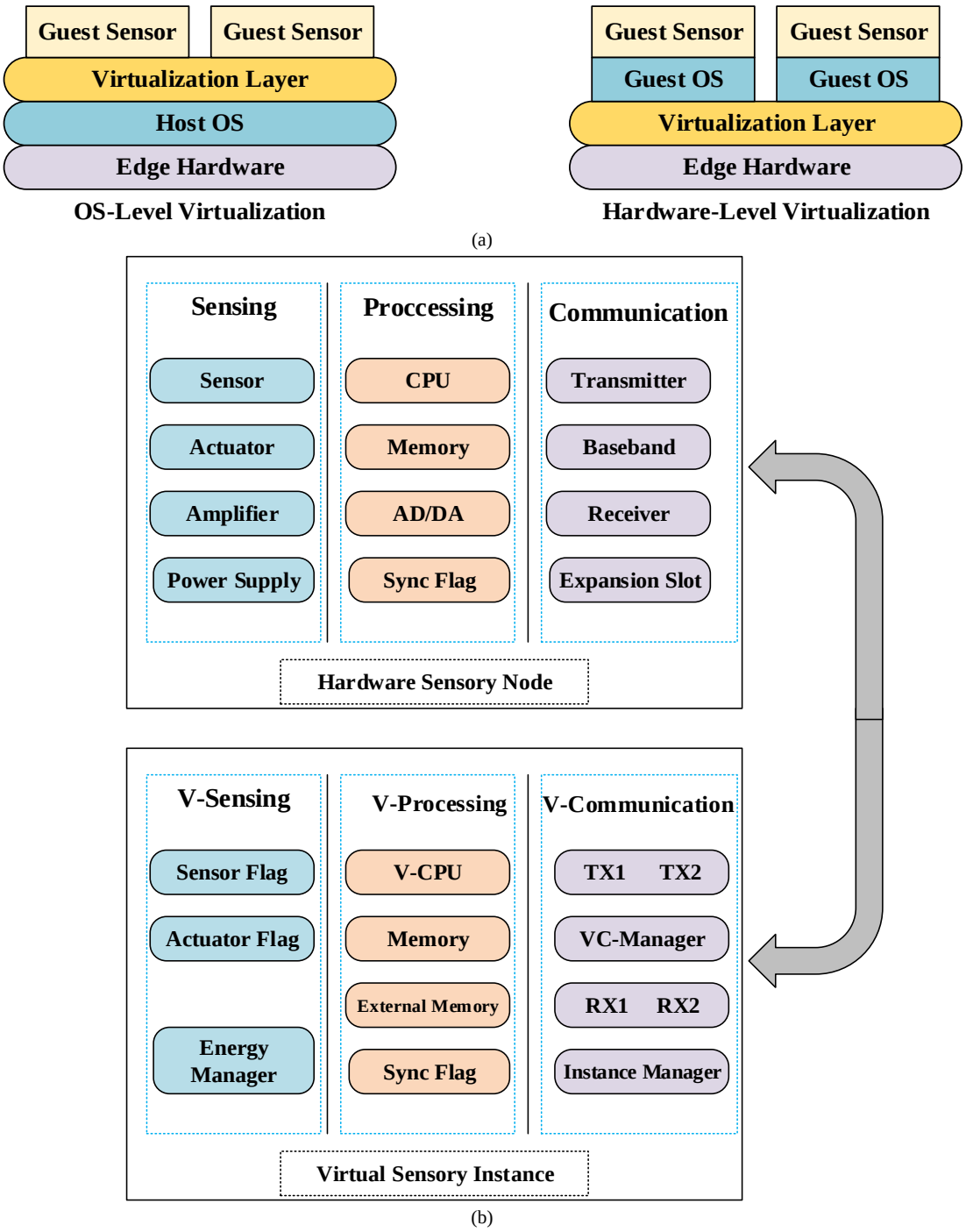

Fig. 1. Object virtualization scheme.

To manage virtual and physical nodes as well as VNFs in the proposed method, an Object Virtualization Manager (OVM) is required. As depicted in Fig. 2, the OVM manages and orchestrates physical and virtual entities. Initially, the OVM deploys and programs the corresponding VOs, monitors and coordinates their operation, discovers and registers them, and creates and terminates them. In this regard, the OVM stores the configuration files of VOs editable by remote or local users. In order to perform automatic self-configuration for service deployment, the OVM downloads the configuration profiles of registered VOs. A VO will be migrated to a more resourceful container if its container is overloaded. VNFs, as software instances, contain several portions of VMs that run network functions on standard hardware. In order to decompose the VNF into reusable components, which can be designed as executable microservices and optimized, upgraded, and configured independently, network functions typically involve several approaches, including verification, computing, media access control, coding, and signaling. These components combine to form VNFs in micro containers.

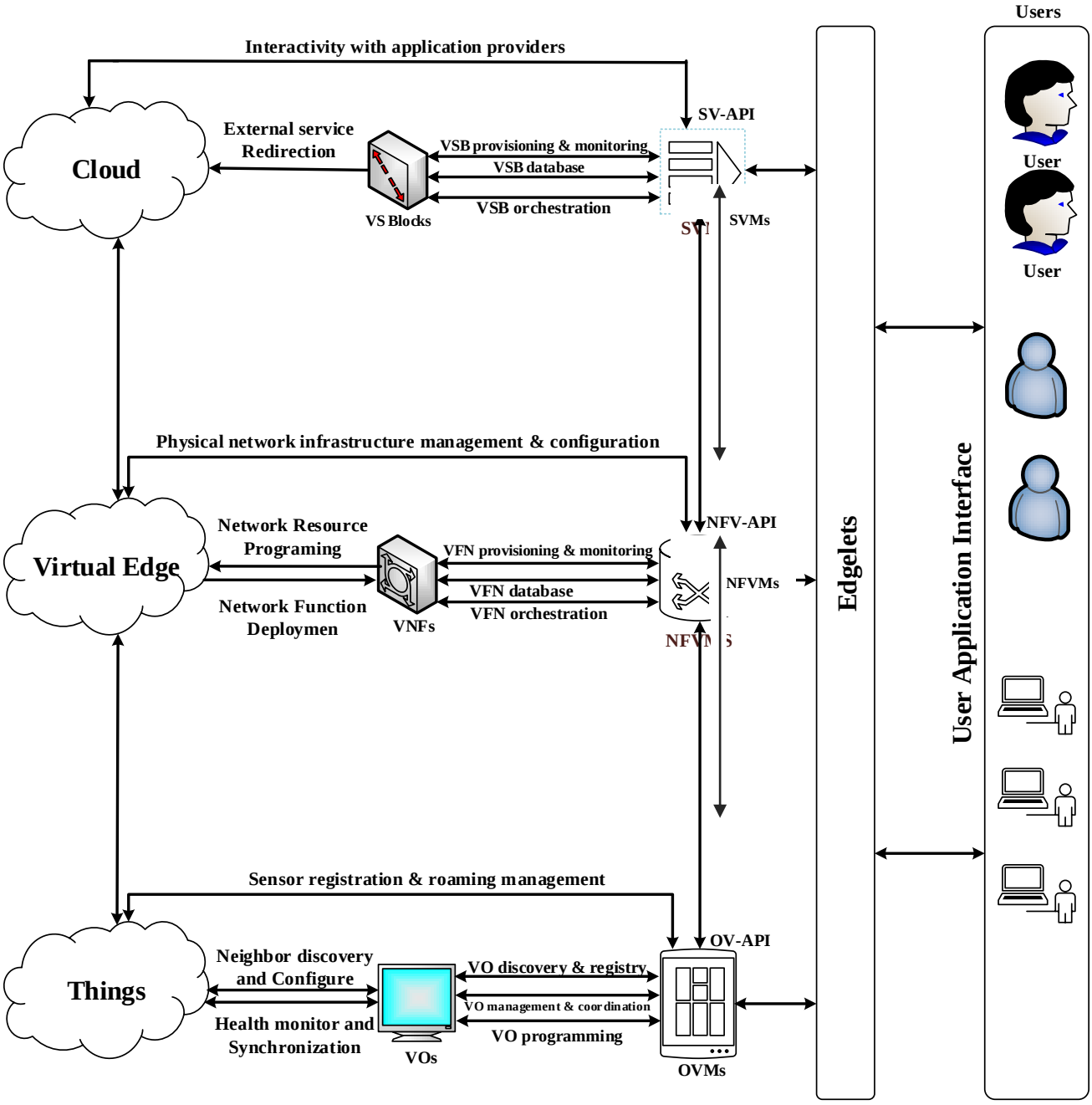

Fig. 2. Virtualization of edge computing framework for IoT environment.

B. *Second Phase: Task Scheduling based on Deep Reinforcement Learning*

Scheduling tasks in edge computing involves two main concerns, allocation of resources and schedule. Resource allocation involves assigning tasks to the appropriate virtual machines, and time scheduling determines the order of task execution. To maximize the quality of experience, our task scheduling strategy considers the expected delay requirement for heterogeneous virtual machine resources. The level of task satisfaction acts as a reward in the deep reinforcement learning algorithm. Network edge servers handle computationally-intensive tasks owing to the difficulty of performing them on local devices. The edge servers are configured with various virtual machines that differ in terms of computational capacity and execution time. The scheduler is responsible for monitoring the status of virtual machines and incoming tasks. The waiting time, the speed of computing, the expected completion time, and task size are significant factors in scheduling decision-making. Observations confirm that the scheduler decides the scheduling order and the start time of each task, determining suitable virtual machines. Fig. 3 shows the general layout of the proposed task-scheduling framework.

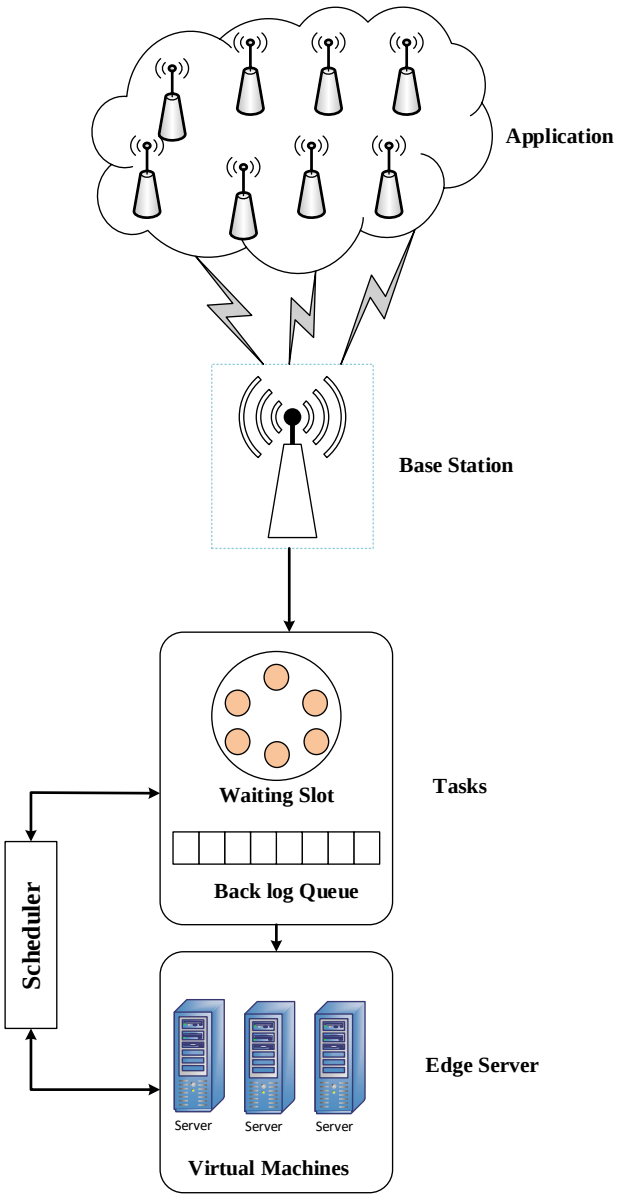

Fig. 3. Proposed task scheduling framework.

IV. Experimental Results

A numerical evaluation of the proposed task scheduling mechanism is presented in this section. Pytorch running on Python 3 produced all simulation results. A comparison was made between the proposed mechanism and two baselines. We implemented the task scheduling policy using a four-layer DNN structure. Each hidden layer encompasses 64 neurons with a rectified linear unit (ReLU) as an activation function. The output layer consists of (M+1)×O neurons, where M refers to the number of VMs and O denotes the maximum tasks in the waiting slot. In training, we set the discount factor to 0.99, which indicates that the current decision is affected by future steps. The learning rate is set to 10−4, and gradient descent is performed using the Adam optimizer. The paper focuses on scheduling tasks in the edge system by considering only computational resources. VM resources and task characteristics are the only environment parameters considered in calculating residual computation delay. IoT devices generate tasks and send them to the base station for transmission. The edge server receives these tasks periodically. Latency is expected to range from 5 to 10 seconds, with transmission delays ranging from 1 to 5 seconds. Task sizes range from 500 to 4000 MI. Virtual machines have a processing capacity ranging from 1000 to 2000 MIPS. Waiting slots are set to O=5 and backlog queues to |b|=5.

The effects of task popularity skewness, task arrival rate, and virtual machine count on task satisfaction and success ratio were studied. Fig. 4 and Fig. 5 illustrate the excremental outcomes. As illustrated in Fig. 4, the number of virtual machines and the task arrival rate are associated with the cumulative degree of task satisfaction. With the popularity skewness set at 0.3, the task arrival rate ranges from 3 to 7, and the number of virtual machines increases from 3 to 5. According to Fig. 5, the cumulative task satisfaction degree

decreases as the task arrival rate increases. A higher arrival rate means that more tasks are waiting for scheduling in the edge system simultaneously, which lengthens their waiting time. As the number of VMs increases, the average task satisfaction degree also increases. Tasks can be scheduled across multiple virtual machines, resulting in shorter waiting times. Fig. 5 shows the skewness of task popularity versus task satisfaction degree. Task popularity skewness ranges from 0.1 to 0.9 with three virtual machines. A higher cumulative task satisfaction is associated with greater skewness in task popularity. Each type of task has a different popularity skewness. As task popularity skewness increases, smaller tasks become more popular while larger tasks become less popular, thus reducing the overall waiting time.

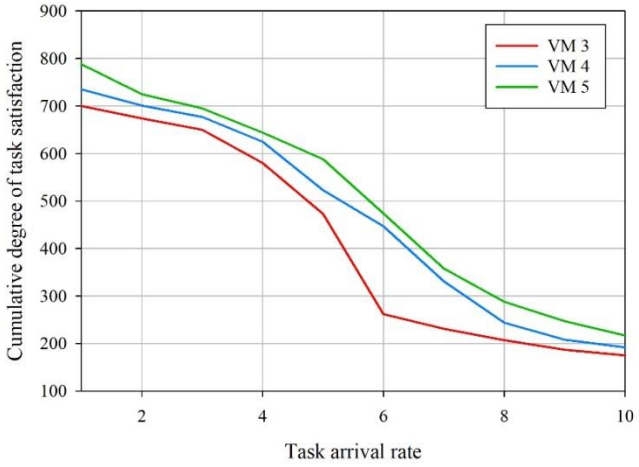

Fig. 4. Task satisfaction degree vs. task arrival rate.

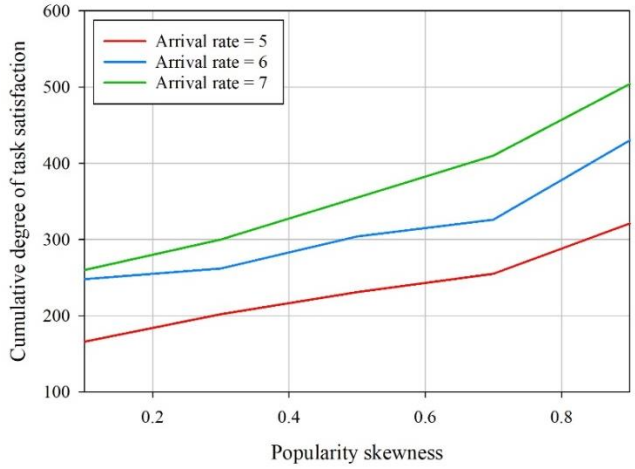

Fig. 5. Task satisfaction degree vs. popularity skewness.

Shortest Job First (SJF) [38] and First Come First Service (FCFS) [37] algorithms are chosen as benchmarks for evaluating the performance of the proposed task scheduling mechanism. SJF and FCFS assign the scheduled task to virtual machines with the highest instant reward. This leads to the scheduling of tasks in a greedy manner. These benchmarks can therefore be categorized into greedy-FCFS and greedy-SFJ. A comparison was made between our proposed method and these benchmarks in terms of task success rate and task satisfaction rate. Total task satisfaction is affected by the average task satisfaction degree, which makes it possible to evaluate the algorithm's overall quality. The task is considered complete when the response time is less than the expected delay. According to Eq. (1), the task success ratio can be obtained by dividing the number of satisfied tasks by the number of tasks in total.

$$\varepsilon_s = \frac{N_s}{\sum_{j \epsilon J} N_T} \quad (1)$$

Fig. 6 and Fig. 7 show how performance is affected by varying task arrival rates. In all algorithms, the average task success ratio and level of task satisfaction decrease as task arrival rates rise. Our method is significantly more efficient than the greedy-FCFS and greedy-SJF scheduling algorithms. In particular, the suggested method can increase average task satisfaction degrees by around 50% and 25% over greedy-FCFS and greedy-SJF. As FCFS schedules earlier-arriving tasks first, subsequent tasks may be delayed when the earlier-arriving tasks demand a lot of CPU power. For long tasks, greedy SFJ prioritizes shorter tasks over longer ones. The expected delay demand is not taken into account by greedy-FCFS or greedy-SJF algorithms.

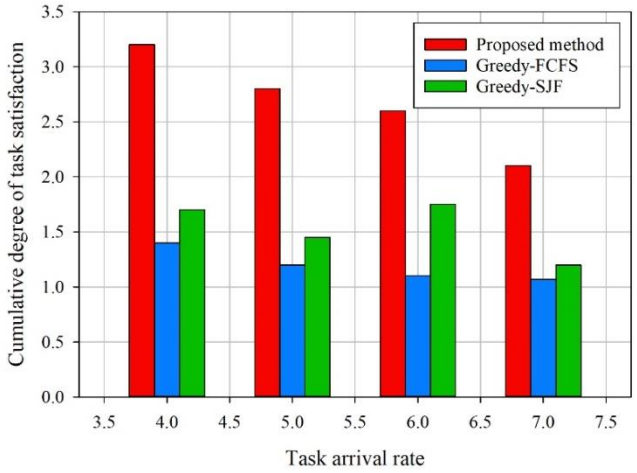

Fig. 6. Task satisfaction degree vs. task arrival rate.

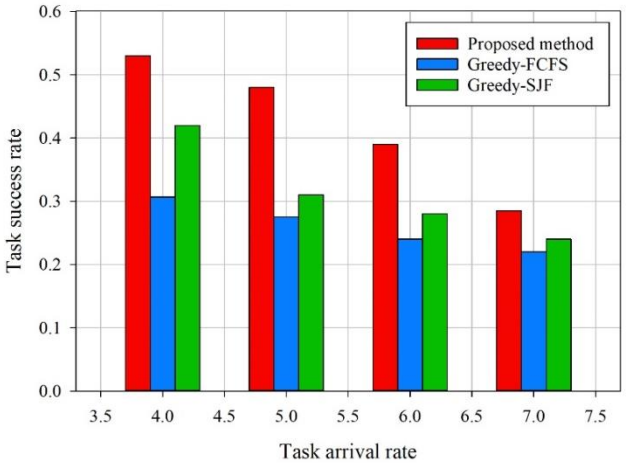

Fig. 7. Task success rate vs. task arrival rate.

Fig. 8 and Fig. 9 compare our task scheduling mechanism with the baselines in terms of task popularity. A higher skewness factor results in a higher degree of task satisfaction and a higher success ratio. The proposed algorithm can significantly improve task satisfaction, as shown in Fig. 8. The gap widens with increasing popularity factor values compared to greedy-SJF and greedy-FCFS algorithms. The greedy-SJF algorithm suffers from performance degradation due to a larger proportion of small tasks being assigned as popularity increases. The higher skewness factor allows for more accurate predictions of task lengths, leading to more efficient scheduling. This leads to better results for popularity-based algorithms compared to greedy-SJF and greedy-FCFS algorithms. The improved task satisfaction and success ratio observed in Fig. 8 are a direct result of this improved scheduling efficiency.

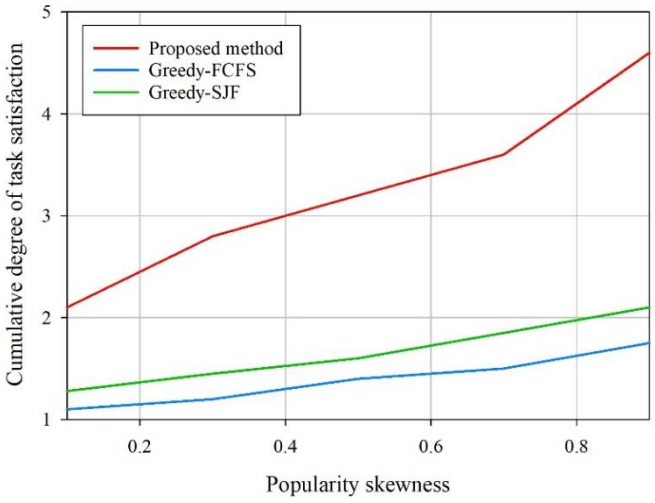

Fig. 8. Task satisfaction degree vs. populariry skewness.

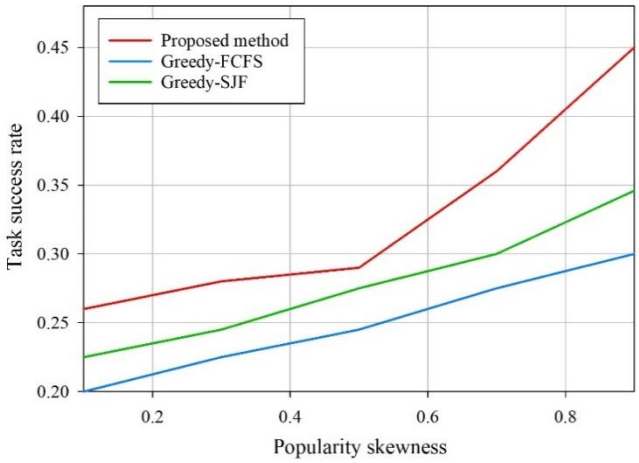

Fig. 9. Task success rate vs. popularity skewness.

V. CONCLUSION

With the rapid growth of IoT applications, edge and IoT devices have expanded their services in recent years. Fog computing offers a latency sensitivity advantage over cloud computing for IoT-enabled smart applications. Task scheduling effectively reduces application computation time and latency while improving quality of service. This paper's proposed task scheduling framework comprises two main phases: virtualization of edge nodes and task scheduling based on deep reinforcement learning. The first phase offers a layered edge framework. In the second phase, we applied DRL to schedule tasks taking into account the diversity of tasks and the heterogeneity of available resources. Simulations indicate that our proposed task scheduling method leads to greater levels of task satisfaction and success than existing approaches.

REFERENCES

[1] Vahedifard, F., et al., Artificial intelligence for radiomics; diagnostic biomarkers for neuro-oncology. World Journal of Advanced Research and Reviews, 2022. 14(3): p. 304-310.

[2] Saeidi, S.A., et al. A novel neuromorphic processors realization of spiking deep reinforcement learning for portfolio management. in 2022 Design, Automation & Test in Europe Conference & Exhibition (DATE). 2022. IEEE.

[3] Akhavan, J. and S. Manoochehri. Sensory data fusion using machine learning methods for in-situ defect registration in additive manufacturing: a review. in 2022 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS). 2022. IEEE.

[4] Khosravi, F., et al. Implementation of an Elastic Reconfigurable Optical Add/Drop Multiplexer based on Subcarriers for Application in Optical Multichannel Networks. in 2022 International Conference on Electronics, Information, and Communication (ICEIC). 2022. IEEE.

[5] Khosravi, F., et al., Improving the performance of three level code division multiplexing using the optimization of signal level spacing. Optik, 2014. 125(18): p. 5037-5040.

[6] Haghshenas, S.H., M.A. Hasnat, and M. Naeini, A Temporal Graph Neural Network for Cyber Attack Detection and Localization in Smart Grids. arXiv preprint arXiv:2212.03390, 2022.

[7] Taami, T., S. Krug, and M. O'Nils. Experimental characterization of latency in distributed iot systems with cloud fog offloading. in 2019 15th IEEE International Workshop on Factory Communication Systems (WFCS). 2019. IEEE.

[8] Pourghebleh, B., et al., The importance of nature-inspired meta-heuristic algorithms for solving virtual machine consolidation problem in cloud environments. Cluster Computing, 2021: p. 1-24.

[9] He, P., et al., Towards green smart cities using Internet of Things and optimization algorithms: A systematic and bibliometric review. Sustainable Computing: Informatics and Systems, 2022. 36: p. 100822.

[10] Pourghebleh, B., et al., A roadmap towards energy ‐ efficient data fusion methods in the Internet of Things. Concurrency and Computation: Practice and Experience, 2022: p. e6959.

[11] Kumar, A., et al., Smart power consumption management and alert system using IoT on big data. Sustainable Energy Technologies and Assessments, 2022: p. 102555.

[12] Stoyanova, M., et al., A survey on the internet of things (IoT) forensics: challenges, approaches, and open issues. IEEE Communications Surveys & Tutorials, 2020. 22(2): p. 1191-1221.

[13] Khan, W.Z., et al., Industrial internet of things: Recent advances, enabling technologies and open challenges. Computers & Electrical Engineering, 2020. 81: p. 106522.

[14] Mohseni, M., F. Amirghafouri, and B. Pourghebleh, CEDAR: A cluster-based energy-aware data aggregation routing protocol in the internet of things using capuchin search algorithm and fuzzy logic. Peer-to-Peer Networking and Applications, 2022: p. 1-21.

[15] Cui, Y.-y., et al., A novel offloading scheduling method for mobile application in mobile edge computing. Wireless Networks, 2022. 28(6): p. 2345-2363.

[16] Alqarni, M.M., A. Cherif, and E. Alkayal, A Survey of Computational Offloading in Cloud/Edge-based Architectures: Strategies, Optimization Models and Challenges. KSII Transactions on Internet and Information Systems (TIIS), 2021. 15(3): p. 952-973.

[17] Zhang, D., et al., Task offloading method of edge computing in internet of vehicles based on deep reinforcement learning. Cluster Computing, 2022. 25(2): p. 1175-1187.

[18] Kamalov, F., et al., Internet of Medical Things Privacy and Security: Challenges, Solutions, and Future Trends from a New Perspective. Sustainability, 2023. 15(4): p. 3317.

[19] Hasan, M.Z. and H. Al ‐ Rizzo, Task scheduling in Internet of Things cloud environment using a robust particle swarm optimization. Concurrency and Computation: Practice and Experience, 2020. 32(2): p. e5442.

[20] Abdel-Basset, M., et al., Energy-aware marine predators algorithm for task scheduling in IoT-based fog computing applications. IEEE Transactions on Industrial Informatics, 2020.

[21] Shadroo, S., A.M. Rahmani, and A. Rezaee, The two-phase scheduling based on deep learning in the Internet of Things. Computer Networks, 2021. 185: p. 107684.

[22] Guizani, N. and A. Ghafoor, A network function virtualization system for detecting malware in large IoT based networks. IEEE Journal on Selected Areas in Communications, 2020. 38(6): p. 1218-1228.

[23] Al-Habob, A.A., et al., Task scheduling for mobile edge computing using genetic algorithm and conflict graphs. IEEE Transactions on Vehicular Technology, 2020. 69(8): p. 8805-8819.

[24] Chen, L., et al., Resource constrained profit optimization method for task scheduling in edge cloud. IEEE Access, 2020. 8: p. 118638-118652.

[25] Azizi, S., et al., Deadline-aware and energy-efficient IoT task scheduling in fog computing systems: A semi-greedy approach. Journal of network and computer applications, 2022. 201: p. 103333.

[26] Kanbar, A.B. and K.H.A. Faraj, Region aware dynamic task scheduling and resource virtualization for load balancing in IoT-fog multi-cloud environment. Future Generation Computer Systems, 2022.

[27] Hussain, S.M. and G.R. Begh, Hybrid heuristic algorithm for cost-efficient QoS aware task scheduling in fog–cloud environment. Journal of Computational Science, 2022. 64: p. 101828.

[28] Najafizadeh, A., et al., Multi-objective Task Scheduling in cloud-fog computing using goal programming approach. Cluster Computing, 2022. 25(1): p. 141-165.

[29] Li, Z., et al., Credit-based payments for fast computing resource trading in edge-assisted Internet of Things. IEEE Internet of Things Journal, 2019. 6(4): p. 6606-6617.

[30] Wang, P., et al., Joint task assignment, transmission, and computing resource allocation in multilayer mobile edge computing systems. IEEE Internet of Things Journal, 2018. 6(2): p. 2872-2884.

[31] Li, S., et al., Joint admission control and resource allocation in edge computing for internet of things. IEEE Network, 2018. 32(1): p. 72-79.

[32] Zhang, X., et al., Resource allocation for a UAV-enabled mobile-edge computing system: Computation efficiency maximization. IEEE Access, 2019. 7: p. 113345-113354.

[33] Ataie, I., et al. D 2 FO: Distributed Dynamic Offloading Mechanism for Time-Sensitive Tasks in Fog-Cloud IoT-based Systems. in 2022 IEEE International Performance, Computing, and Communications Conference (IPCCC). 2022. IEEE.

[34] Tan, H., et al. Online job dispatching and scheduling in edge-clouds. in IEEE INFOCOM 2017-IEEE Conference on Computer Communications. 2017. IEEE.

[35] Zhang, Y., et al., Resource scheduling for delay minimization in multi-server cellular edge computing systems. IEEE Access, 2019. 7: p. 86265-86273.

[36] Chen, X., et al., A hybrid task scheduling scheme for heterogeneous vehicular edge systems. IEEE Access, 2019. 7: p. 117088-117099.

[37] Chiang, Y.-H., T. Zhang, and Y. Ji, Joint cotask-aware offloading and scheduling in mobile edge computing systems. IEEE Access, 2019. 7: p. 105008-105018.

[38] Li, C., et al., Collaborative cache allocation and task scheduling for data-intensive applications in edge computing environment. Future Generation Computer Systems, 2019. 95: p. 249-264.

[39] Alameddine, H.A., et al., Dynamic task offloading and scheduling for low-latency IoT services in multi-access edge computing. IEEE Journal on Selected Areas in Communications, 2019. 37(3): p. 668-682.