IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 26 April 2023, accepted 24 May 2023, date of publication 30 May 2023, date of current version 21 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3281348

RESEARCH ARTICLE

Resource Provisioning Using Meta-Heuristic Methods for IoT Microservices With Mobility Management

SHINU M. RAJAGOPAL[1], M. SUPRIYA[1], AND RAJKUMAR BUYYA[2], (Fellow, IEEE)
[1]Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Bengaluru 560035, India
[2]CLOUDS Laboratory, School of Computing and Information Systems, The University of Melbourne, Melbourne, VIC 3010, Australia

Corresponding author: Shinu M. Rajagopal (mr_shinu@blr.amrita.edu)

ABSTRACT The fog and edge computing paradigm provide a distributed architecture of nodes with processing capability for smart healthcare systems driven by Internet of Thing (IoT) applications. It also provides a method to reduce big data transmissions that cause latency and enhance the system's efficiency. Resource provisioning and scheduling in edge and fog systems is a significant problem due to heterogeneity and dispersion of edge/fog/cloud resources. The goal of scheduling is to map tasks to appropriate resources, which belong to NP-hard problems, and it takes much time to find an optimal solution. Meta-heuristic methods achieve near-optimal solutions within a reasonable time. Current edge/fog resource allocation research does not sufficiently address resource allocation problems in mobility-aware microservice-based IoT applications. This paper proposes a meta-heuristic-based micro-service resource provisioning model with mobility management for smart healthcare systems. The proposed approach has been tested on an experimental set-up with a simulation of a critical real-time smart healthcare application with and without considering the mobility of the devices. It applies meta-heuristic methods such as modified genetic and flower pollination algorithms for resource management. The proposed method outperforms the existing solutions in energy consumption, network usage, cost, execution time, and latency by 17%, 20%, 22%, 17%, and 63%, respectively.

INDEX TERMS Edge computing, fog computing, Internet of Things, meta-heuristic, microservices, mobility, smart healthcare, time critical applications.

I. INTRODUCTION

The vast amounts of data generated by Internet of Thing (IoT) devices require latency-sensitive processing, which is impossible when applications are placed in distant cloud data centers [1]. Hence cloud computing seems infeasible for delay-sensitive medical real-time IoT applications [2]. Fog or edge computing, a cloud extension, can address these needs for smart IoT systems to provide better resource management. The purpose of the fog or edge computing model is to process tasks using local computing resources in fog or edge devices that have storage, computation, and communication capabilities to enhance mobility, confidentiality, privacy, reduced latency, and bandwidth [3], [4]. It allows a seamless connection between smart medical devices and computational resources to have more control over data privacy and security [5]. It also supports critical latency-sensitive IoT applications that require faster responses with reduced energy and bandwidth consumption [6]. In some emergencies, making a decision using the appropriate resources in a fraction of a minute directly impacts the patient's life [7]. To avoid issues with over-provisioning or under-provisioning while satisfying QoS requirements, it is critical to dynamically provide the right amount of available fog or edge resources to manage the workload of IoT services [8]. Because of its dynamic, unpredictable nature, resource management is one of the most challenging problems to tackle in such environments [9]. To create a comprehensive solution for proactive resource

The associate editor coordinating the review of this manuscript and approving it for publication was Cong Pu.

management, the transition between multiple resource management strategies based on various network circumstances is presented [10].

As a user relocates from one place to another, the proximity to a fog or edge service may change; hence, user mobility restricts such benefits in practice [11]. IoT device mobility can impact fog computing systems when they repeatedly change access points. The mobility of end IoT devices causes migration of the requested application services from one computing node to another to maintain the desired QoS. The deployment of local, small-scale data processing and storage at the network's edge using edge computing makes computations closer to the source data, thus ensuring the QoS requirements. In addition to meeting the demands of latency and bandwidth on the network, it offers intelligent services at the edge to fulfill the vital needs of IoT applications in real-time [12], [13].

IoT applications use various technologies to connect, manage, and operate IoT smart devices. Microservices, a service-oriented architecture, have attracted much interest nowadays. It is an emerging technology based on the microservices concept to enable services with the smallest granularities that perfectly complement the distributed nature of IoT devices [14]. Each microservice is responsible for a single sub-task or service, requiring fewer compute resources and lowering communication overhead. Based on the resource availability and workload of fog nodes, microservices can scale up and down dynamically due to loosely coupled modules. Compared to an existing monolithic design, integrating distributed microservices into the application process provide advantages such as independent deployment, scalability, and fault isolation [15].

Resource management is a mechanism that manages the system's finite physical and virtual resources for the execution of user tasks. Resource provisioning involves the process of allocating and provisioning resources, such as computing, storage, network bandwidth, and others, to support the demands of applications or services. Resource scheduling, on the other hand, deals with the process of determining when and how to allocate resources to tasks or jobs based on their requirements and priorities. Resource provisioning methods when combined with resource scheduling policies, allocate resources to applications efficiently in distributed scenarios, especially in edge/fog computing environments. To maximize the utilization of these resources and improve the efficiency of applications, efficient scheduling, and resource allocation are required. To avoid over-provisioning and under-provisioning, dynamic resource provisioning, an effective method of preparing resources based on changes in the workload of IoT applications, is required [16]. Due to the vast solution space, scheduling in fog/edge computing is classified as an NP-hard problem, which means it takes a long time to discover an optimal solution. No algorithms can handle these issues in polynomial time and yield optimal results. Finding a sub-optimal solution in a short period is preferred in such scenarios. To address such issues, meta-heuristic-based strategies have been experimented to generate near-optimal solutions in a reasonable amount of time [17]. Meta-heuristic scheduling algorithms flower pollination algorithm (FPA) and genetic algorithm (GA) can emulate the best characteristics are seen in nature and assist the scheduling algorithm in performing better. The modified GA and the modified FPA are meta-heuristic methods used in the proposed scheduling method.

The genetic algorithm is one of the most effective population-based algorithms in terms of ease of use and effectiveness for different problem contexts. Each chromosome on GA consists of a set of genes representing a possible solution to a problem. The initial population, chosen at random, is the starting point of the algorithm. A fitness function determines whether a chromosome is suitable for its environment. Single-point crossover and mutation methods are used to create a new population. This procedure is repeated when there are enough children. To select the best solution, heuristic algorithms use the objective function, but the fitness function is implemented by the genetic algorithm for choosing the optimal solution [18]. The flower pollination algorithm, which simulates the pollination process of flowering, has been proposed in the literature for multi-objective optimization applications. Global and local pollination are the two most important processes in this algorithm. A flower and its pollen gametes provide a stable approach to the optimization problem. The benefits of FPA include a more straightforward flower analogy and lightweight computing depending on just one control factor [19]. This paper proposes using the modified genetic algorithm and flower pollination algorithm to effectively provision fog and edge resources.

The significant contributions of our work are as follows:

- Proposing a resource provisioning solution using IoT microservices with mobility management for healthcare applications.
- Utilizing modified meta-heuristic scheduling techniques for efficient resource provisioning in fog and edge devices.
- Simulating a set of experiments to validate the effectiveness of our proposed solution under real workloads in terms of energy consumption, network use, cost, execution time, and latency.

The rest of the paper is organized as follows: Sections II and III present the motivation, background, and related work. Sections IV and V discuss the proposed method and the experimental setup. Results are presented in Section VI. The concluding comments are set out in Section VII.

II. BACKGROUND AND MOTIVATION

Vast amounts of sensor data from IoT endpoints are expected to be transferred to the cloud in a conventional health monitoring system, requiring significant network resources. Delay is crucial in real-time contexts like health monitoring systems, and it gets complex if the technology is used on a big scale. The paradigm for resolving the issues mentioned above is fog/edge computing. Fog/edge computing reduces latency

by bringing resources close to the network's edge [20]. Microservice architecture is used to keep up with the IoT applications' demand for fast development and deployment. Due to the microservices' fine-grained modularity and being independently deployable and scalable, they show tremendous promise for utilizing both cloud and fog/edge resources to satisfy various QoS demands of IoT application services [21]. Latency brought on by the mobility of a sensor device can significantly impact edge-based IoT applications for remote monitoring, including erroneous analysis and poor service quality. In order to reduce the latency while maintaining a connection, high-quality of service and energy-efficient mobility support techniques are necessary [15]. Resource provisioning and scheduling is a significant problem due to heterogeneity, mobility, and dispersion of edge/fog/cloud resources. Scheduling aims to match tasks with the right resources, which are included within the scope of NP-hard issues, and it takes a long time to discover the optimal solution.

Meta-heuristic-based techniques have been proven to achieve near-optimal solutions within a reasonable time for such problems [22]. Finding global optimum solutions to several complicated multi-modal design problems in engineering and industry seems to be very difficult. In such scenarios, conventional optimization techniques perform inadequately because they may become locked in local optima. The utilization of meta-heuristic algorithms derived from nature is hence proposed. Due to their ability to avoid stagnation in local optima and high convergence speed in the right direction of the near-optimal solution, meta-heuristic optimization algorithms have greatly impacted many fields in recent decades. These algorithms tackle many optimization problems, especially problems in the engineering domain. Meta-heuristic algorithms currently occupy the leading technique as optimization models for solving various optimization problems in a reasonable amount of time due to their significant results. By making a few assumptions about the optimization problems, meta-heuristics offer a set of solutions that can be applied to various issues. These algorithms use less processing to find feasible results.

According to the source of their inspiration, meta-heuristic algorithms have been divided into four groups: human-based, swarm-based, physics-based, and evolutionary algorithms. The first group of algorithms, referred to as ''evolution-based algorithms,'' imitate biological evolution by using reproduction, mutation, recombination, and selection to create new offspring that are more powerful than their parents. The majority of population evolutionary algorithms, such as genetic algorithms, evolution strategy, genetic programming, biogeography-based optimizer, and probability-based incremental learning, have been extensively used for different optimization issues. The swarm-based or social behavior-based algorithms include the harris hawks algorithm, particle swarm optimization, cuckoo search, whale optimization algorithm, slime mold algorithm, marine predators algorithm, grey Wolf optimizer, ant colony optimization, bat algorithm, and flower pollination algorithm. Virtualizing fog computing includes an essential concept for optimal task scheduling.

The best way to solve task scheduling issues in fog computing is to use meta-heuristic algorithms that can quickly handle a large search area and find the best solution. Hence we have selected meta-heuristic approaches for the proposed method. We have also chosen a genetic algorithm for the proposed approach since genetic algorithms use several sets in a search space where a search space is a collection of all possible solutions to the problem. It requires one objective function to calculate an individual's fitness and can work in parallel. Genetic algorithms operate on potential solutions' representations, known as chromosomes, rather than the actual solutions. Genetic Algorithms are not guaranteed to produce global optimal solutions as well but genetic operators like crossover and mutation increase the likelihood of producing global optimal solutions. Genetic algorithms are stochastic and probabilistic in nature. With the right parameter setting, due to their large solution space, genetic algorithms are highly effective at handling multi-modal problems [23].

Because of the following characteristics, we have chosen the flower pollination method as the other meta-heuristic approach. Swarm intelligence (SI) optimization algorithms, which are modeled after numerous forms of biological behavior found in nature, have the advantages of being easy to use, performing well in optimization, and having strong robustness. The flower pollination algorithm (FPA) is a meta-heuristic inspired by flowering plants for artificial intelligence. Flower pollination is the process of transferring pollen from one flower to another. Animals, such as birds, bats, insects, and so forth, are the principal actors in such transfers. Flowers and insects will form a flower-pollinator alliance. These blooms can attract birds which are part of the pollination, and these insects are the primary pollinators of the flowers. A flower and its pollen gametes provide a reliable answer to the optimization problem. With only one control parameter, FPA gives a simplified flower analogy with lightweight computing and provides a balanced intensification and diversity of solutions by implementing the Lévy flight and switch condition, which may be used to switch between local and global search. The pollinator transports pollen over greater distances to high-fitting flowers in case of global pollination; however, in other circumstances, local pollination is carried out inside a small area of an exclusive bloom. Switch probability is a possibility for global pollination. Local pollination can be used to replace phased elimination. The flower optimization algorithm (FPA), which was developed to address global optimization based on simulating the pollination process of flowers, has successfully addressed several optimization problems. FPA is distinguished by its formulation's simplicity, adaptability, and great computational performance efficiency. According to numerous studies, it can also outperform other well-known meta-heuristic optimization methods. As a result, FPA has

been incorporated into several optimization studies and successfully used to solve numerous optimization issues in a variety of scientific domains.

To summarize, current research on edge/fog resource allocation does not sufficiently address resource allocation issues in mobility-aware microservice-based IoT applications. The rise of edge/fog computing has opened up new possibilities for healthcare sector, leading us to select a crucial medical application as the focus of our study. Edge/fog computing allows real-time processing of data generated by medical devices and wearable sensors, thus enabling remote patient monitoring, faster diagnosis, and more personalized treatment. Efficient resource provisioning is crucial for healthcare applications in edge/fog computing because of the reasons such as low-latency requirements, limited network bandwidth, and resource constraints. Therefore, we conducted a review of existing literature on resource allocation methods employed in healthcare applications, which led us to consider utilizing metaheuristic techniques for resource provisioning. We chose GA and FPA because similar applications of this category of heuristics, such as resource management on cloud infrastructure, have produced promising results.

III. RELATED WORK

Reserving data processing, network, and storage resources for use by IoT applications is called provisioning. A fog or an edge node must effectively provide resources to allocate IoT requests. In the literature, resource provisioning has been studied for years and has gained much attention recently [24]. This section reviews resource management concerns in fog and edge computing.

Shakarami et al. propose an overview of resource provisioning methods in fog computing environments and discuss the open challenges in this area. Machine learning-based, heuristic/meta-heuristic-based, framework-based, game theoretic-based, and model-based are the five primary classifications presented [7]. Masoumeh et al. provide a resource provisioning technique that uses a Bayesian learning-based autonomic computing model for decision-making and control loop planning. This work has been carried out using time series prediction models [8]. The work proposed in [25] also uses Bayesian learning along with linear regression and autonomic computing to efficiently allocate the cloud resources. Dinesh et al. suggest an improved resource provisioning method based on the JAYA (a sanskrit word meaning victory) approach for placing virtual machines in a data center which aims to reduce energy consumption by effectively organizing the migrated VMs [26]. Literature also presents many heuristic-based and evolutionary-based techniques for task scheduling. Heuristic algorithms are faster than evolutionary algorithms but unsuitable for finding an optimal solution in NP-complete situations. Recently, meta-heuristics have also been employed to generate optimal solutions [27].

Mishra et al. use meta-heuristic service allocation algorithms for a heterogeneous fog computing system that processes heterogeneous jobs, formulate the linear programming problem for time and energy optimization and uses particle swarm optimization (PSO), binary PSO, and bat algorithm [22]. Hosseinioun et al. propose a strategy based on the dynamic voltage and frequency scaling (DVFS) technique that is energy aware and saves it using hybrid invasive weed optimization [28]. Ashkan et al. recommend FOGPLAN, a QoS-aware dynamic fog service provisioning framework, by defining it as an optimization problem and evaluating it using a simulation based on real-world traffic traces [29]. Naranjo et al. propose a penalty-aware bin packing heuristic algorithm for resource management hosted by each fog node, allowing resource consolidation and admission control by scaling up or scaling down computation frequencies [30]. To reduce the average peak age of information, Fang et al. designed the associated time slot allocation problems and, using an exact linear search strategy, found the best solutions to the resulting non-convex problems [33].

FCM-FPA, a new fuzzy clustering with flower pollination method as a resource provisioning model for fog computing proposed in the literature, includes resource normalization and fuzzy clustering and has been evaluated using the Iris and Wine datasets [19]. Abdel et al. introduce an energy-aware meta-heuristic approach for task scheduling based on Harris hawk optimization to enhance QoS, which also assesses energy consumption, cost, makespan, flowtime, and carbon dioxide emission [34]. To more effectively address numerous research difficulties, such as resource placement and scheduling, mobility, communication and edge control, many nature-inspired meta-heuristic (NIMH) methods have been applied in edge computing. Fuzzy logic, edge network systems, and various research issues are all included in the survey conducted by Adhikari et al., which divides the current NIMH into three categories based on the nature of their work [35]. To reduce Service Level Agreement (SLA) violations caused by the limitations of edge computing resources and to handle the computational complexity of edge computing problems, Adyson et al. propose a random and heuristic approach to initialize the population for multi-objective genetic algorithm. The solution thus developed is found to be close to optimal and is employed to examine the placement and load distribution of IoT applications. It performs better than existing benchmark algorithms in response to deadline violation, cost, and service accessibility [36]. Fang et al. provide a heuristic particle swarm optimization (PSO) approach built on a Lyapunov framework to balance system queue backlog and energy efficiency for trajectory scheduling and allocation of computational resources for Internet of Underwater Things [37]. Table 1 presents the summary of related works in meta-heuristic methods.

Among many applications, the healthcare sector deserves to be prioritized in terms of service quality compared to other domains. Critical functions such as simultaneous reporting and monitoring, tracking and alerts, and remote medical aid are all possible with IoT-based apps. The center for connected health policy conducted a study that observed that remote

TABLE 1. Summary of related works in heuristic/meta-heuristic methods.

Paper	Purpose	Technique	Energy consumption	Execution time	QOS/QOE	Cost	Delay	Network use
[22]	Service allocation	PSO,BPSO,BAT	✔	✔				
[28]	Task scheduling	IWO-CA	✔					
[29]	Dynamic resource provisioning	Greedy algorithm			✔	✔		
[30]	Dynamic resource management	Bin packing based	✔				✔	
[19]	Resource provisioning	Optimal FPA			✔			
Proposed	**Resource provisioning**	**GA, FPA**	✔	✔		✔	✔	✔

TABLE 2. Summary of related works in mobility implementations of fog computing.

Paper	Tool	Realtime or Simulation	Mobility	Migration	Clustering
[11]	MobFogSim	Simulation	✔	✔	
[31]	YAFS	Simulation	✔		
[32]	FogNetSim++	Simulation	✔		
Proposed	**iFogSim2**	**Simulation**	✔	✔	✔

health monitoring systems lower the re-admission rates of heart failure patients by 50 percent [38], [39]. The following paragraph discusses fog and edge technology in the medical sector.

Nashatt et al. propose an E-health and wellness monitor application to encourage a healthier lifestyle. This work gathers the user behaviors, analyzes them, and later predicts certain events with personalized recommendations [40]. However, the latency in data processing affects critical emergencies. Adesh et al. suggest a real-life cloud-based smart medical system using a communication networking where-in a doctor treats his patients via internet. This proposed application uses mobile, wireless body area networks, and so on, intending to be extended to fog technology. The proposal claims that the suggested framework is more effective in computation and communication expenditure than the existing protocols in smart healthcare [41]. Tuli et al. propose a lightweight healthcare fog service that manages cardiac patients' IoT data. The FogBus framework allows efficient edge/fog/cloud integration for reliable and fast results [3]. A comprehensive overview of fog-based technologies in healthcare IoT systems has been conducted by Ammar et al. [42]. To identify the challenges and requirements of edge devices for diverse use cases, Morghan et al. explore current and developing edge computing architectures and approaches for health care applications [43]. To evenly distribute the load amongst fog nodes when the health monitoring system is installed on a big scale, Asghar et al. offer a new load balancing scheme (LBS). It presents the comparison of the parameters, network use, and latency for different placements namely cloud-only implementation, load Balancing scheme, and fog node placement [44].

The mobility of IoT end devices can impact the performance of fog-based provisioning. The provisioning of resources considering mobility for critical real-time applications is a growing field with several unsolved issues. The following articles examine the implementation of end node mobility in fog and edge computing environments.

Puliafito et al. implement the MobFogSim, an add-on to the iFogSim simulator that can account for user mobility, wireless connectivity, and the virtual machine/container migration process [11]. Isaac et al. propose yet another fog simulator (YAFS) for fog computing environments that model network failures and thereby allow the evaluation of service placement solutions in failure cases through run time creation/deletion of cloudlets and network links, as well as functions invoked at run time to implement actual events [31]. FogNetSim++ is a framework for constructing network simulators that extend OMNeT++14 to mimic all aspects of energy consumption, pricing, mobility, and handoff mechanisms [32].

Jayasena et al. use a whale optimization meta-heuristic algorithm for optimal task scheduling in a smart healthcare application model and found that it outperforms PSO, shortest job first and round robin in terms of energy usage and cost [45]. Qiu et al. analyze the minimization optimization in fog computing-based Internet of Medical Things, which is considered a non-convex and non-linear problem [46]. The work considers the quality of service, power limit, and wireless constraint as optimization parameters. Abdel et al. propose a fog-based IoT platform for real-time diabetic patient tracking, using a hybrid strategy based on type-2 neutrosophic with the help of VIKOR method [47]. Hasse et al. present an e-health system that collects general and physiological health indicators from older people using My signals HW V2 technology using a fog computing mobile application for monitoring health [48]. The recommended strategies have the advantage of handover latency. However, they are unsuitable for IoT fog systems since the message notifications, and distributed storage cannot be refreshed when moving [49]. Table 2 presents the summary of related works in mobility implementations of fog computing.

This paper's primary focus is building IoT microservices for the healthcare sector. The goal of microservice architecture is to break down the system into small, self-contained components linked to shared services with more significant service decoupling compared to service-oriented and monolithic architectures [50]. Because of its essential characteristics, such as small granularity and low coupling, microservices architecture is recommended as a new design method that is easier to update and deploy fog IoT applications. Due to its excellent performance and applicability for IoT applications, microservice deployments are more

relevant nowadays [51]. Each microservice is responsible for a single sub-task or service, requiring fewer compute resources and lowering communication overhead. Zhao et al. offer an architecture based on the microservice container fog system to execute delay-sensitive and cost-effective mobility applications in which the costs are calculated as the sum of the computation and communication expenses [52]. A novel approach to learn the microservice applications using predictive autoscaling deployed on containerized fog computing infrastructure has been proposed by Abdullah et al., which seems to have less number of rejected requests and SLA violations compared to existing systems [53]. Samodha et al. investigate the factors that distinguish microservices-based application scheduling in fog computing from other application models. In addition, the work analyzes the integration of microservices for IoT applications using application modeling, placement composition, and performance evaluation [54].

We aim to develop a framework based on meta-heuristic approach for mobile-aware IoT microservices that could be implemented on edge/fog computing scenarios for the medical IoT applications. Our proposed approach involves modifying and integrating the mobility module within iFogSim2. Along with latency addressed in the existing works, other parameters like energy consumption, network use, cost, and execution time are also to be considered while developing a healthcare IoT system. The focus of this paper is to develop a resource provisioning solution using IoT microservices with mobility management by deploying meta-heuristic methods for healthcare applications and comparing the results against the existing implementations.

The main contributions of this paper are as follows:

- Resource provisioning of IoT microservice applications considering the mobility of nodes.
- Experimentation on the meta-heuristic microservice framework for efficient resource management.

IV. PROPOSED METHOD

A. APPLICATION MODEL

Healthcare systems are facing enormous challenges due to the pandemic and chronic diseases. The proposed architecture is based on expectations extracted from the actual application of the healthcare sector. If successfully applied, fog computing can reduce the latency experienced in the quality of service (QoS) and minimize the bandwidth usage in any healthcare application and later can be extended to other time-critical applications. The main difference between fog-based and cloud-based systems is the computing and storage capability of the fog devices between the patient and the cloud data center. With fog technology, the underutilization of intermediate devices can also be addressed. Virtualized processing cores, storage, and memory are considered resources at fog nodes. If a request meets the resource requirements such as CPU, memory, and bandwidth, the request can be processed by the current fog or edge device; else the request can be transmitted to the neighboring device. The proposed fog-based architecture includes the concept of fog node virtual machine partitioning to run medical IoT device data efficiently. Multiple processes running on a single fog/edge node might cause congestion, preventing tasks from running smoothly. To solve this problem, fog/edge nodes can establish virtual machines responsible for delivering computational resources to the tasks. Virtual machines are primarily separate modules, each doing a specific task. Medical IoT applications are increasingly being implemented using modular architectures, which use microservices architecture and allow time-sensitive tasks inside fog/edge and latency-tolerant jobs inside the cloud. Hence we have selected microservice architecture for designing and modeling critical real-time medical applications. The application model is discussed in detail in the following paragraphs.

For an integrated fog healthcare application, a multitier architecture paradigm has been chosen for the proposed approach. IoT devices such as sensors and actuators constitute tier 0. Electrocardiogram (ECG) signals are sensed by sensors connected to the patient and transmitted to the fog nodes via smartphones. All sensors have the same sensing frequency. In the proposed system, the tuple transmit rates of the sensors are set using transmitDistribution, which is an attribute in the Sensor class. DeterministicDistribution (EEG_TRANSMISSION_TIME) defines the transmitDistribution of the sensor, and EEG_TRANSMISSION_TIME is selected as 5ms in the proposed system. Actuators are used to carry out corresponding actions based on the outcomes of applications. The fog nodes (proxy servers, gateways) constitute the architecture's tier-1 and tier-2. In the proposed fog-based smart healthcare system, the fog layer act as a supportive intermediary layer for processing and analyzing real-time critical healthcare data close to end-users. The fog nodes process the data from the sensor IoT device, and the patient's health condition is relayed to the patient's smartphone. The fog nodes are located near the network's edge, which is closer to the IoT device, hence the patients receive a quick response in a real-time application. The data center at the cloud layer is the upper layer of the proposed system, which stores the permanent healthcare data of the patient. The cloud, which acts as tier 3, supplies additional processing and storage resources if fog devices cannot handle the incoming request requirements. Data stored in the cloud can be retrieved anytime by the users connected with the application. Our proposed health monitoring system's multitier architecture is depicted in Figure 1.

B. PROBLEM FORMULATION

A systematic strategy of allocating available resources to clients is known as resource allocation. Every fog device has a data center with similar capabilities as a cloud but with fewer hosts, computing elements, and storage. Hence, the fog computing model has limited resources to serve the incoming user requests with stringent latency requirements. The fog network's client applications can run as cloudlets in virtual machines (VM). The cloudlet tasks are executed in the

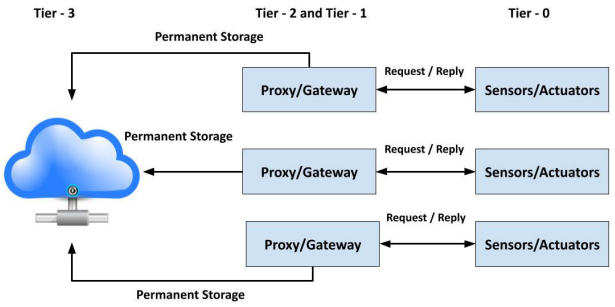


FIGURE 1. Multitier architecture.

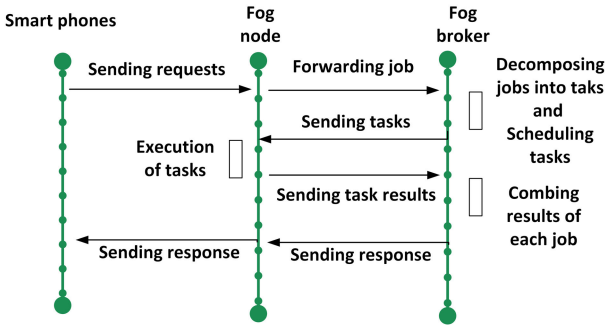


FIGURE 2. The operation of task scheduling.

corresponding virtual machine. By creating multiple virtualized copies of the hardware, the VMs provide resources to the applications. In such distributed environments, task scheduling involves allocating resources to the tasks efficiently. A task scheduler is in charge of mapping an application's task to the available resources so that specific requirements can be met. Since each VM is cost-effective, task scheduling is a challenging issue. To address the challenges faced, the literature finds the use of meta-heuristic approaches, namely the genetic algorithm and the flower pollination algorithm, for resource allocation. As a resource provisioning model, this study proposes a mobility-aware IoT healthcare application using modified meta-heuristic approaches for scheduling. This proposed work integrates mobility, clustering, and microservice techniques for healthcare applications and analyzes energy consumption, network usage, cost, execution time, and latency parameters. IFogSim2 simulator has been used to evaluate the proposed system [15]. The following subsections go through the proposed method in considerable detail.

1) SYSTEM MODEL

The proposed system model assumes the presence of a task scheduler placed at the edge/fog nodes, that is in-charge of the scheduling of the tasks which are forwarded for execution once they are submitted. As a result, a resource provisioning technique is to be used in the nodes that help in finding an efficient match between tasks and resources.

The main aim of this research is to allocate resources efficiently by considering users' mobility with minimal processing time. The applications considered in this work are time-sensitive and hence processing user requests only using a cloud is not a preferable option due to the incurring latency issues. The participation of fog/edge devices is a need to complete the tasks in a timely manner. The proposed model should address the limitations of such fog/edge devices like highly distributed nature and minimal resource availability. It should also consider the hybrid fog-cloud environment with the deadline and dynamic behavior of the users. Prior research have not considered resource allocation in the fog by considering the location change of users. As mentioned earlier, this paper experiments the mobility of the devices and analyzes the results for varying conditions. A detailed description of the mobility of the users is described in Section IV-B5 and results in Section VI-B1.

IoT applications handle the processing of user requests and respond to consumers' need on a regular basis. The flow of the task scheduling module for the proposed approach is described below:

A mobile user first sends a request, which is processed by the fog node to whom it is linked. The fog broker receives job request. Each job is broken down into a series of tasks to be handled in the distributed system, and the resource requirement for each of them is then calculated. The fog broker manages the data about tasks and nodes and uses a scheduling algorithm to determine the suitable assignment of resources to the task. Tasks are then dispatched to the corresponding fog nodes. Each node is responsible for completing all the tasks allocated to it before sending the completed work back to the fog broker. The connected fog node then transmits the response to the mobile user. The sequence diagram depicting the above discussions is presented in Figure 2.

An edge/fog computing system consists of a fully interconnected set of m servers. Each server consists of n VMs. Virtualization allows the creation of multiple virtual machines on the same host or server. We assume that all the resources are homogeneous with respect to computing capacity and capability. Each host may be assigned to perform different or the same services. The workload submitted to the system is the tasks that are later submitted to the scheduler. The scheduler allocates the tasks to the VMs on different computing hosts. Tasks are independent scheduling entities that cannot be preempted and generally represent the user's compute or service request.

Efficient resource provisioning for edge /fog systems can be represented as an optimization problem to minimize the execution time, as represented in the following paragraph. Since time-critical applications have a greater impact on this parameter, we optimize the execution time in our model.

Requests from user applications are broken down into small, independent tasks when they are transferred to the

fog layer to be processed through the cloud-fog computing infrastructure. The following characteristics belong to each task: the number of instructions, memory required, and size of input and output files. A set of n independent tasks are delivered to the system at each time, assuming that T_k represents the k^{th} task, as follows:

$$T = \{T_1, T_2, T_3, \ldots, T_n\} \tag{1}$$

The cloud-fog infrastructure comprises cloud nodes and fog nodes, which are processors with characteristics such as CPU rate, CPU usage cost, bandwidth usage cost, and memory usage cost. The cost of using cloud nodes is higher than that of fog nodes, despite the fact that they are often more powerful. The system's set of m processors is made up of fog nodes, as follows.

$$N = \{N_1, N_2, N_3, \ldots, N_m\} \tag{2}$$

where N_i presents the i^{th} processing node. The processor N_i is given each job T_k, which is denoted by T_k^i.

A set of one or more tasks may be assigned to one processor for computing:

$$N_i Tasks = \{T_x^i, T_y^i, \ldots, T_z^i\} \tag{3}$$

The execution time (EXT) required by node N_i to finish a set of $N_i Tasks$ assigned to it, as follows:

$$EXT(N_i) = \sum_{T_k^i \in N_i Tasks} EXT(T_k^i) \tag{4}$$

$$EXT(T_k^i) = \frac{length(T_k^i)}{CPU_i} \tag{5}$$

where length(T_k^i) denote the number of instructions in the task T_k and the node N_i's CPU rate is represented by CPU_i and is depending on factors such as clock rate, core count, instruction level parallelism, etc. Total execution time is the total time taken by the system to complete all the tasks, defined from the time when the request is received until the last task, or the time when the last machine completes. Total execution time is determined by the formula:

$$TXT = \sum_m [EXT(N_i)] \tag{6}$$

The execution time of a task includes the time taken to complete the job while utilizing system services. The duration of execution varies for different tasks depending on the level of processing and input-output activities involved. Enhancing execution speed increases system effectiveness; as a result, it becomes a crucial factor. Efficient resource provisioning for edge /fog systems can be represented as an optimization problem to minimize the execution time, as represented in the following paragraph. Since time-critical applications have a greater impact on this parameter, we optimize the execution time in our model.

2) EVALUATION PARAMETERS

The objective function used in our work intends to minimize the execution time. Additional metrics considered are latency, energy consumption, network use, and cost. This section explains these performance measures which are used to evaluate the modified meta-heuristic approaches (explained in the next sub-section) for task scheduling in mobility-aware microservice-based IoT medical applications proposed in this study.

a: LATENCY

The application's control loop is Client Microservice → Preprocessing Microservice → Decision Making Microservice → Client Microservice which is described in Section IV-B7. The relatively lower latency of this control loop suggests improved allocation and coordination of computer resources. Microservices in the fog hierarchy are upgraded by edgeward placement, resulting in a single instance of each microservices from an edge to a cloud path. The limited resource capacity of fog nodes necessitates a strategy where microservice instances are placed in higher fog layers. However, this approach leads to an increase in average latency.

$$TL = \sum_m CAL \tag{7}$$

Total latency (TL) represented in Equation 7 directly depends on the allocation of VMs in fog devices in which the tasks are distributed for execution. Total latency is the summation of the current average delay (CAL) experienced by every VM inside the host for m servers, where CAL is calculated as follows.

$$CAL = CC - ET \tag{8}$$

In the above Equation, CC is the simulator clock and ET is the execution time of the tuple.

b: ENERGY CONSUMPTION

Energy consumption is a crucial factor in computing systems, encompassing various components such as data communication infrastructure, backup power supplies, cooling systems, fire control, and security technologies. The operational costs of infrastructure are influenced by the power supply, making it essential to implement strategies for reducing these costs. The edgeward method involves deploying the majority of microservices on cloud virtual machines, which leads to increased energy consumption of cloud resources. The extent of energy usage depends on the number of active microservices in each tier. Reducing energy consumption significantly affects the efficiency, affordability, availability, dependability, and environmental sustainability of devices.

The energy consumption of a server or host can be represented by separating it into two main components: the fixed energy consumption when the server is in an idle state, and the variable energy consumption when the server is actively

processing requests. Energy consumption depends on the server's number of VMs and the allocated MIPS allocated for each VM.

E_0 is the current energy consumption, the fixed energy for the server in an idle state. The variable energy for server utilization while processing the requests is EN_{i_k}. E is the total energy consumption which can be calculated by

$$E = EN_{ik} + E_0 \tag{9}$$

$$EN_{i_k} = c_1 * EXT(T_k^i) \tag{10}$$

where EN_{i_k} is the energy consumption by the task T_k running on the virtual machine or node i, E_0 is the power required to operate a data center which is the fixed energy for the server in an idle state, where c_1 denote the CPU usage fee per time unit in node N_i

c: NETWORK USAGE

A crucial metric for comparing various approaches is the overall volume of data delivered across a network. Particularly on large networks, high data transfer techniques may result in network congestion, service interruptions, or an increase in the control loop's average delay for the applications. Compared to cloud operations, latency can be significantly minimized by offloading a portion of the workload to the network edge. It is important to maintain efficient communication between the edge and the cloud. Data pre-processing at edge and fog devices can greatly reduce the size of data transmission. However, it is crucial to conserve bandwidth due to the large number of endpoints connected to the network and the requirement for multiple database servers to support them. Network usage depends on the latency experienced by the network and the tuple size of the data for $'n'$ VMs in the host as listed in Equation 11.

$$NU = \sum_{n} (l * TNS) \tag{11}$$

where l denote the latency experienced by the network and TNS denote the tuple network size. Total network use depends on the number of VMs in fog devices in which the tasks are distributed for execution.

d: COST

Costs include infrastructure, network hardware, processing, network communications and storage costs. The investment made by service providers in fog computing also includes the placement of communication and processing workloads in the fog device. One of the main issues with fog computing is cost saving.

Processing cost is defined as:

$$cost(T_k^i) = c_1 * EXT(T_k^i) + c_2 * M(T_k^i) + c_3 * Bw(T_k^i) \tag{12}$$

where c_1 denote the CPU usage fee per time unit in node N_i, and EXT(T_k^i) is given in Equation 5. c_2 denote the memory usage fee per data unit in node N_i and M(T_k^i) represent the memory needed by task T_k. Task T_k processed in node N_i needs an amount of bandwidth Bw(T_k^i), which is the sum of input and output file size. c_3 is the bandwidth usage fee per data unit. The following formula is used to determine how much each task in the cloud-fog system will cost in total:

$$Totalcost = \sum_{T_k^i \in N_i Tasks} cost(T_k^i) \tag{13}$$

Our proposed strategy aims to deploy meta-heuristic resource-provision methods and evaluate the above-mentioned parameters.

Table 3 contains the list of acronyms used in this section.

TABLE 3. Notations used in evaluation metrics equations.

Symbol	Meaning	Symbol	Meaning
T	Tasks	E_0	Power required for the server in an idle state
N	Nodes	NU	Network use
m	The Number of servers	l	Latency experienced by the network
n	Number of VMs inside the host	TNS	Tuple network size
$EXT(N_i)$	The execution time required by node N_i	$cost(T_k^i)$	Cost for processing task T_k^i
TXT	Total execution time	$M(T_k^i)$	Memory needed by task T_k^i
TL	Total latency	$Bw(T_k^i)$	Bandwidth needed by task T_k^i
CAL	Current average latency	NU	Network use
CC	Simulator clock		
ET	Execution time of the tuple		
E	Total energy consumption		
EN_{i_k}	Energy consumption by the task T_k		

3) MULTI-OBJECTIVE OPTIMIZATION

This paper proposes the solution to the resource provisioning problem in two approaches. The first approach is addressing the resource provisioning problem as a multi-objective optimization problem with the objective to minimize the evaluation parameters considered in this work. This approach uses the weighted sum method to aggregate the results. The second approach is to solve the problem using modified meta-heuristic methods. This subsection explains the formulation of the multi-objective optimization problem and the next subsection explains the meta-heuristic approach in detail.

The parameters considered for optimization are energy consumption, network use, execution time, cost and latency represented by the weight vectors w_1, w_2, w_3, w_4, w_5 respectively. They hold the values between 0 and 1 and sum to 1. These weights are subjective and define the contribution of each parameter in the solution space. The multi criteria decision making methods like analytical hierarchical process could be used to derive the weight vector. Any point of a convex pareto front can be obtained by altering the weights. The model is solved for each combination of weighted coefficients, and the objective function values returned are saved in the pareto set. The objective functions considered in this work are presented in Equations 14, 15, 16, 17, and 18. The multi-objective optimization problem is given in Equation 19.

$$obj_1 = min(TXT) \tag{14}$$

$$obj_2 = min(TL) \tag{15}$$

$$obj_3 = min(E) \tag{16}$$

$$obj_4 = min(NU) \tag{17}$$

$$obj_5 = min(Totalcost) \tag{18}$$

TABLE 4. Weight selection for weighted sum method.

w_1	w_2	w_3	w_4	w_5	objective function value
0.165	0.143	0.143	0.221	0.329	2.21E+06
0.18	0.12	0.14	0.21	0.32	2.40E+06
0.18	0.06	0.06	0.1	0.58	2.40E+06
0.18	0.05	0.08	0.08	0.5	2.40E+06
0.2	0.2	0.2	0.2	0.2	2.67E+06

$$min(w_1 * obj_1 + w_2 * obj_2 + w_3 * obj_3 + w_4 * obj_4 + w_5 * obj_5) \quad (19)$$

where

$$w_1 + w_2 + w_3 + w_4 + w_5 = 1 \quad (20)$$

The constraints for this problem include ensuring that the total CPU requirements of all the jobs assigned to a particular host do not surpass the host's capacity. Additionally, each Virtual Machine should be allocated to only one host. The start time for the simulation should be greater than zero, and the tuple network size must not exceed the network size limit.

Analytical Hierarchical Process (AHP) is a structured technique that is used to solve multi-criteria decision making problems in many areas including e-commerce, transportation, portfolio selection, supplier selection etc. It helps in organizing and analyzing complex decisions, based on mathematics and psychology, where alternatives are ranked using the pairwise comparison of multiple criteria [55]. The weighted sum for each alternative is computed by multiplying each objective value with its corresponding weight and summing up the result, ending in a singular value representing the performance of each alternative. These alternatives are then ranked in descending order based on their weighted sum values, creating a list from best to worst. Beginning with the top alternative, non-dominated solutions are identified by comparing their weighted sum value to those of the alternatives below it. If an alternative has a better or equal weighted sum value for at least one objective and a better-weighted sum value for at least one other objective, it is considered non-dominated. This process is repeated for all alternatives, and the preferred solution is selected from the non-dominated set. However, this weight assignment is subjective and can vary from problem to problem. The use of AHP in this proposed study leads to the set of weight vectors listed in the Table 4 for the parameters considered.

The results comparing this with the existing and the proposed methods are presented in Fig. 13.

4) PROPOSED META-HEURISTIC METHODS

This section explains the solution to the resource provisioning problem using the meta-heuristic methods. As described in section IV-B, this work uses the modified version of genetic algorithm and the flower pollination algorithm which is described below.

a: FITNESS FUNCTION

Execution time is the the most important factors influencing energy consumption, network use, cost, and latency. Thus, an objective function is employed for evaluating the candidate solutions as same as Equation 5, which can be represented as follows:

$$Fitness = \sum_{i=1}^{m} \frac{length(T_k^i)}{CPU_i} \quad (21)$$

where length(T_k^i) denote the number of instructions in the task T_k and the node N_i's CPU rate is represented by CPU_i

Algorithm 1 Modified Genetic Algorithm

```
Input: An application's set of resources available and unmapped tasks.
Output: Output mapping
Construct a list of the available resources.
Create population
for Each chromosome do
    Determine the optimum resources (best fit) for every activity based on the time it will take to complete it.
    Go to the next resource in the list.
    if The counter of index = last resource then
        Go to the first resource on the list.
    end if
end for
Evaluate all chromosomes using the fitness function
while Termination condition not reached do
    Random selection and crossover
    Mutation
    select best chromosomes
end while
Save the best solution
Map the tasks on resources
```

b: MODIFIED GENETIC ALGORITHM

By taking into consideration the fog/cloud system features, a modified meta-heuristic method based on GA is proposed in this approach.

- Initial population: In a standard genetic algorithm, it is generated at random. In fog-critical time applications, creating a good and goal-oriented initial population that leads to finding the response promptly is preferable. Hence to build the initial population, the tasks/cloudlets are sorted by execution times and their processing capacity in MIPS. The chromosome's first row of genes is occupied with sorted cloudlets. The best-fit technique is to select a suitable VM with the shortest operating time for each task from the virtual resource list.
  The proposed pseudo-code is presented as Algorithm 1. Figure 3 presents the flowchart for the same.
- Crossover: A random gene selection is used in this case. Two parents are chosen randomly, and their genes are

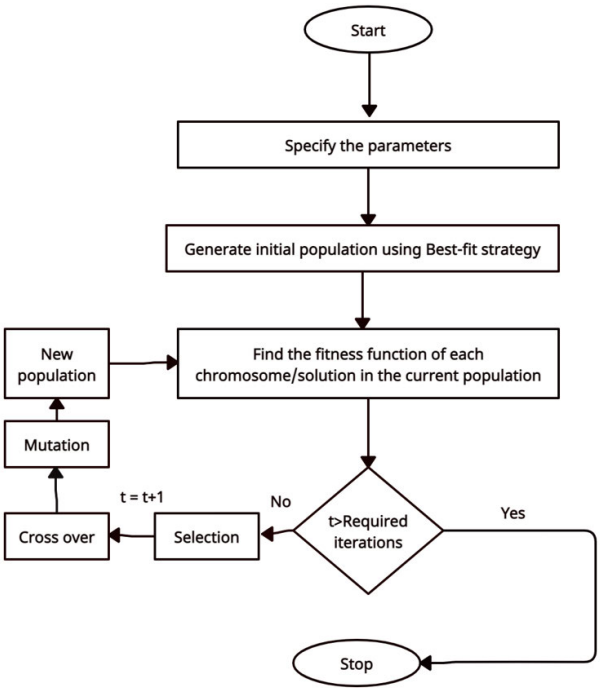


FIGURE 3. Modified genetic algorithm flowchart.

chosen at random. Then, two other solutions are developed by altering resource regions of selected genes.

- Mutation: A chromosome and one of its genes along with a resource from a virtual list is chosen at random. If the execution time is faster than the last candidate, the selected resource will be replaced with the selected gene. The mutation leads to the speedy discovery of a good solution.
- Evaluation and selection solutions: It uses a fitness function with efficient parameters to recognize the value of a solution. The fitness function is applied to all the solutions, and their values are determined in GA. Then the solution with the best value is identified as the maximum or minimum for the fittest solution using parameter placement guidelines. In the proposed approach, solutions with minimum execution time can be considered the best value after fitness function estimation. The number of virtual resources in the list is denoted as $'n'$. Each solution's fitness value is calculated using Equation 21. The chromosome with minimum fitness value is considered as the best solution among others. The target is to minimize the fitness function. The method tries to find a solution by using crossover and mutation operations to reduce the fitness value as much as feasible. Single-point crossover has been applied in the proposed approach due to its advantages, which include enhanced genetic diversity, expedited convergence, effective exploration of the search space, and retention of favorable genetic material. By utilizing single-point crossover, the diversity of solutions can be increased, and the search for optimal solutions can be accelerated. The population selection strategy opted is the elite selection, which is a simple method to implement. It only involves selecting a fixed number of the most best individuals from the population. Elite selection helps quicken the convergence, enhance the genetic diversity, and keep the good genetic material by preserving the best individuals in the population. It prefers some of the best chromosomes for the next iteration, considered elites. The time complexity of GA for the resource allocation problem considered in this work depends on the number of generations (g), the population size (n), and the complexity of the fitness function (m) represented as O(g * n * m). The memory complexity of the GA implementation depends on the representation of the chromosome, the population size, and the size of the problem. The memory required to store the population is proportional to the number of chromosomes (n) and the length of each chromosome (l) denoted as O(n * l).

c: MODIFIED FLOWER POLLINATION ALGORITHM

The flower pollination algorithm (FPA) is a meta-heuristic inspired by flowering plants for artificial intelligence. Flower pollination is the process of transferring pollen from one flower to another. Animals, such as birds, bats, insects, and so forth, are the principal actors in such transfers. Flowers and insects will form a flower-pollinator alliance. These blooms can attract birds which are part of the pollination, and these insects are the primary pollinators of the flowers. A flower and its pollen gametes provide a reliable answer to the optimization problem. With only one control parameter, FPA gives a simplified flower analogy with lightweight computing and provides a balanced intensification and diversity of solutions by implementing the Lévy flight and switch condition, which may be used to switch between local and global search. The pollinator transports pollen over greater distances to high-fitting flowers in case of global pollination; however, in other circumstances, local pollination is carried out inside a small area of an exclusive bloom. Switch probability is a possibility for global pollination. Local pollination can be used to replace phased elimination. The steps of modified FPA in the proposed method are described below:

- Step 1: Initialize the population with the help of the best fit strategy to get the response rapidly. Hence to build the initial population, the tasks/cloudlets are sorted by execution times, and their processing capacity in MIPS. Cloudlets are allocated to VMs in the virtual resource list using the best-fit strategy.
- Step 2: Evaluate performance for each solution in the initial population using the fitness function in Equation 21. The fitness function is applied to all the solutions, and their values are computed in FPA, after which the solution with the best value is selected as the maximum or minimum for the fittest solution using parameter placement guidelines. In the proposed approach, solutions with minimum execution time can be considered

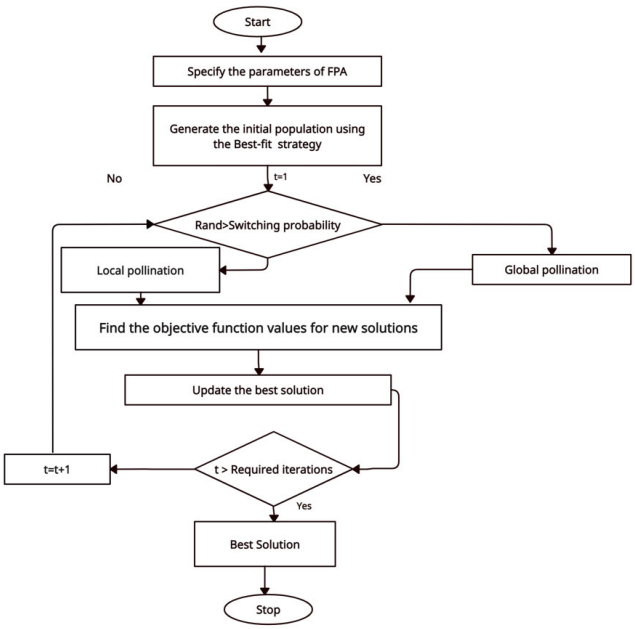


FIGURE 4. Modified flower pollination algorithm flowchart.

the best value after fitness function calculation. Each solution's fitness value is calculated using Equation 21.

- Step 3: Find the best solution among all. The best solution among the others is the one with the lowest fitness value. The goal is to minimize the fitness function as much as possible.
- Step 4: Define switch probability. In the proposed approach, the switch probability is considered as 0.8.
- Step 5: Check to stop criteria. In the proposed approach, the algorithm stops once it reaches the required number of iterations.
- Step 6: Flower pollination main loop start. According to switch probability, perform local pollination or global pollination.
- Step 7: Update new solution and compare with old solutions.
- Step 8: Display the best solution among all.

Figure 4 shows the flowchart for the same. The proposed algorithm's pseudo-code is included in Algorithm 2. The switching probability p, the scaling parameter, and the population size n are the parameters in FPA. Empirical findings and numerical simulations imply that a small population is adequate, regardless of whether the real-world population sizes are large. The time complexity of FPA for this implementation depends on the number of iterations (t), the population size (n), and the complexity of the objective function or the number of decision variables (m) which is given as O(n * m * t). The memory complexity of FPA for this work relies on the population size (n) and the number of decision variables (m) denoted as O(n * m).

The proposed approach described in Algorithm 3 is the implementation of investigating resource provisioning based on meta-heuristic methods for microservice-based IoT medical applications in a fog computing environment utilizing mobility components of iFogSim2.

Algorithm 2 Modified Flower Pollination Algorithm

```
Input: An application's set of resources available and unmapped tasks.
Output: Output mapping
Construct a list of the available resources.
Create population
for Each solution do
    Determine the optimum resources (best fit) for every activity based on the time it will take to complete it.
    Go to the next resource in the list.
    if The counter of index = last resource then
        Go to the first resource on the list.
    end if
end for
Find the fitness of each solution in the population using the fitness function
Find the best solution
while Termination condition not reached do
    if (rand) is less than switching probability (.8) then
        performs global pollination
        else
        performs local pollination
    end if
    Find fitness of the new solution
    if new solution better than existing solutions then
        swap with the new solution
    end if
end while
Save the best solution
Map the tasks on resources
```

5) MOBILITY

The mobility of fog nodes or users raise an issue for the fog computing platform by maintaining resources close to users at all times [56]. IoT device mobility can impact the performance of fog applications because of the rapid movement of the devices from one access point to another. This causes IoT device application service migration from one computing node to another to maintain the committed QoS. When opting for mobility, one must consider identifying an intermediary node to which the origin fog node can upload the relevant application service, so that the final fog node can download. Location data for IoT devices, the device's mobility trajectory or mobility direction and speed, are the factors we must consider for service migration. If we consider end node mobility, network metrics such as delay, energy use, and other variables also fluctuate dramatically. Considering all these characteristics, we have extended the mobility component of iFogSim2 to customize with the proposed application. A comprehensive description of these classes is provided in Figure 5.

Algorithm 3 Healthcare Application

```
Input: Sensor data
Output: Response to an actuator
Create new FogBroker
Create new Application
Create new dataparser object and location handler object
Adding microservices (Client module, datapreprocessing, decision making, storage) to the application
Using edges to connect application modules in the application model
Defining the relationships of the input and output microservice modules
Defining the microservice application loop as sensor-client module-data preprocessing-decision making-client-actuator
Add application to fogbroker
Create fog devices in the physical topology
Create Cloud datacenter
for each i to locator.getLevelWiseResources do
    Create proxy fog device
    Apply modified genetic algorithm for task scheduling and Apply modified flower pollination algorithm for task scheduling
end for
for each i=0 to locator.getLevelWiseResources do
    Create gateway fogdevice
end for
Initialize microservice mapping
Create MicroservicesMobilityClusteringController and submit the controller to the application
Start Simulation
Stop Simulation
End
```

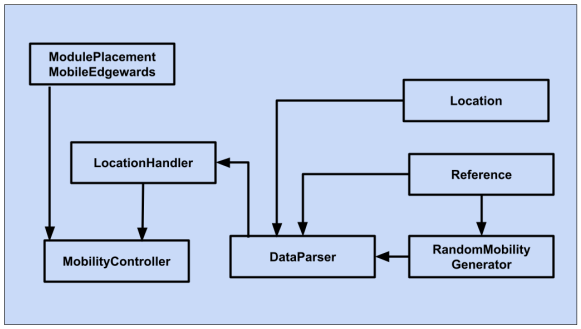


FIGURE 5. IFogSim2 mobility.

The modified mobility components can expand data from external sources using the DataParser class as an interface by reading data from .csv files. The multitier logical hierarchy of edge/fog nodes described in the previous subsection has been employed to deploy end devices, edge, fog, and cloud. The DataParser class can independently assimilate the location data of various mobile users and IoT devices, allowing application services to be handled according to each user's unique mobility pattern without harming other users. The DataParser class parses the location data of edge, fog, and end nodes and generates separate location object for each coordinate. A Location object may also contain the relevant entities' block-wise information. Additionally, DataParser can use these Location objects to refer to the LocationHandler class to sequence all mobile entities' movement events. The MobilityController class wraps the object references of various mobility-specific classes to collectively address mobility-related problems.

The proposed use case dynamically initiates the necessary sequential or parallel events on various FogDevice and AppModule referenced objects. The MobilityController refers to an object of the MobilityPlacementMobileEdgewards class for the initial placement of AppModules. Including a feature that tracks the fog node-wise deployment of microservices or application modules for each end mobile device in the simulation environment, this class duplicates the ModulePlacementEdgewards of the simulator. When the simulation begins, MobilityController moves in to carry out activities like module launch and resource management following initial placement. It has many steps, including initialization, new parent selection, migration of intra-cluster and inter-cluster modules, and update. The mobility management operation's necessary setup is carried out in the initialization phase. The new parent selection step establishes the mobile entity's new upper-tier contact's minimum distance due to the shifting of locations. During the migration of intra-cluster modules and inter-cluster phase, the migration of modules between clusters happen. In the final update stage, the simulation environment must be updated based on intra-cluster or inter-cluster module migration. The above operations are described in Algorithm 4.

The consideration of mobility component in the application causes the migration of applications modules from one fog device to another which affects the scheduling of tasks to resources and hence the evaluation parameters.

6) CLUSTERING

Resource augmentation can greatly help applications that require a finite amount of fog resources, particularly in terms of storage and computing power. To meet their QoS requirements, some tasks performed by IoT nodes may be delegated to adjacent nodes [57]. Therefore, a clustering method that permits resource augmentation among fog resources is essential. The iFogSim2 clustering component enables distributed dynamic coordination and collaboration between several nodes. In accordance with the particular clustering policies, every node can probe and register its cluster members.

7) MICROSERVICES

To effectively exploit the fog computing paradigm, application development has transitioned from monolithic design to microservice architecture. The application model for the suggested system consists of a number of microservices.

Algorithm 4 Customized Mobility Management Logic

Input: Dataparser object, reference object, location handler object, location of mobile entity m at time t, parent of mobile entity, level of the mobile entities
Output: Migration of application modules to different fog nodes due to the mobility of mobile entities
ρ' = null
δ = getmaximum distance denoted in the reference class
η = operation tier
ρ = upper tier contact due to the location change
for Every fog node of the upper layer of mobile entity **do**
    L_{f_u} =location of fog node
    δ' = distance between m and fog node
    Fog node residing at a minimum distance from mobile entity marked as new upper tier contact or parent node ρ
    if The current and new upper tier contact of mobile entity m differs **then**
        Application modules of mobile entity deployed in ρ' push to ρ, which is a new parent in the same cluster
        else
        Application modules of mobile entity deployed in ρ' push to ρ, which is a new parent in the different cluster
    end if
end for

A methodology called microservices allow us to create an application out of a number of small services, each of which operates in its own process and uses simple protocols to communicate. Microservices enable the creation of systems made up of a number of small, independent components that may each manage their own data. The development of large-scale IoT applications is made possible by advantages including heterogeneity, robustness, scalability, ease of deployment, organizational alignment, and composability. Microservice deployments are more important in the modern world because of their high performance and suitability for IoT applications. Figure 6 shows the application model of the proposed system, which consists of a group of microservices. Each microservice is represented by a vertex, while the edges depict the data connections between them. In this design, there are three microservices, which are listed below.

- Client microservice: It is the front end of the fog computing-based healthcare system. The client microservice is deployed on the users smartphone and receives the sensor ECG signals associated with the patient. Later it transmits sensor data to the preprocessing microservice which is stored on the fog device.
- Preprocessing microservice: The preprocessing microservice performs data validation and cleaning to reduce noise from ECG sensor data before transmitting it for further processing.
- Decision-making microservice: To alert the client microservice about the patient's health, the microservice must determine whether an emergency situation exists from the data in real-time. This decision-making is performed in this microservice.

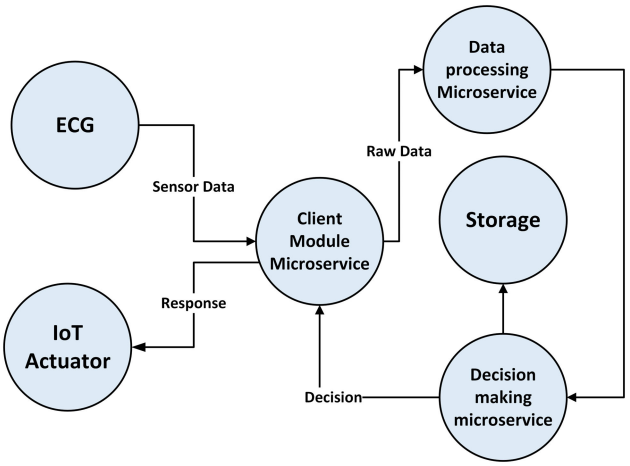


FIGURE 6. Data flow diagram with microservice modules.

To monitor user health, these microservices communicate with one another. Depending on the placement policy, time-critical microservices for preprocessing and decision-making can be deployed in fog or in the cloud. The storage module receives the data that will be residing in the cloud permanently. This study suggests a useful resource provisioning mechanism that takes advantage of a multi-level hierarchical fog architecture, where multiple levels of application placement requests are processed at the fog nodes. It uses a decentralized placement method to disperse microservices across the fog environment to model a critical IoT medical application.

V. EXPERIMENTAL SETUP

The market is very competitive with simulators for simulating cloud, fog, and edge devices. One such discrete event network simulator is NS3, which enables us to create various virtual nodes and install devices, internet stacks, programs, etc., on our nodes with the aid of various classes. For modeling and simulating edge/fog/cloud computing infrastructures and services, we have chosen iFogSim2, an extension of Cloudsim, since this framework can be used to develop and deploy experiments for edge/fog/cloud devices that handle compute, memory, I/O, and VM allocation, as well as VM power models, among other things. The iFogSim2 Simulator, an extension of the iFogSim simulator, holds the properties of service migration, distributed cluster building across fog nodes, and microservice orchestration. This simulator helps validate the proposed approach's performance in the fog computing environment. The components of the iFogSim2, such as mobility, clustering, and microservices, are loosely coupled and can be utilized for simulation in such scenarios. iFogSim2 incorporates real datasets for assessing the performance of different service management strategies in fog computing settings, unlike most existing solutions. It includes

node clustering, mobility management, and microservice orchestration methodologies that can be used as benchmarks for comparing performance [15].

All iFogSim core classes, such as FogDevice, Actuator, sensor, and AppModule, have object references in the Controller class, and it can access the Tuple class through an Application object. The Clustering element of iFogSim2 allows distributed dynamic coordination and collaboration among multi-fog nodes. To add the modified mobility component, which is customized for the proposed use case to the iFogSim2 simulation, it includes classes such as DataParser and MobilityController. The functions of these classes are described below:

- The DataParser class allows the separation and assimilation of location data from many IoT end devices so that application services may be handled based on their unique mobility patterns.
- MobilityController class helps to dynamically start the required sequential or parallel actions on separate FogDevice and AppModule referenced objects for mobility management.

During simulations, the proposed model considers two different types of mobility patterns namely 'RANDOM MOBILITY' and 'DIRECTIONAL MOBILITY'. In 'DIRECTIONAL MOBILITY' model, the time period between any two of these motions is made to be equal to ensure that the user/IoT device maintains a constant speed. The 'DIRECTIONAL MOBILITY' model is being used, which has a significant number of consecutive coordinates lying at the same distances across the Melbourne central business district (CBD) for a user/IoT device. Based on those coordinates, events are constructed to simulate the movement of the associated end IoT device. There are numerous random mobility patterns to represent users 'RANDOM MOBILITY' model behaviors. According to numerous mobility criteria, user speed, direction, stopping time at each location, and duration within each edge/fog node's communication range can be produced by the RandomMobilityGenerator class and can be expanded for multiple random mobility models. The diagrammatic representation of the connection between these modules are presented in Figure 5.

This section explains the simulation environment used to evaluate the suggested approach's performance. The sensors detect ECG and regularly send data to the fog nodes through a smartphone. Data is processed and analyzed on the fog nodes to determine whether the patient's health status is normal or critical. The results are subsequently sent to the patient's smartphone and to the cloud for storage. The fog nodes' connection to the cloud server is established through a proxy server. The client module is embedded in IoT devices to get the sensor data. The processing module is embedded in fog nodes to process and analyze the incoming data to assess the patient's health status. The fog node then communicates the results to the associated IoT device, which displays them. It must define values for numerous parameters such as CPU length, bandwidth, RAM, and so on, in iFogSim2 when

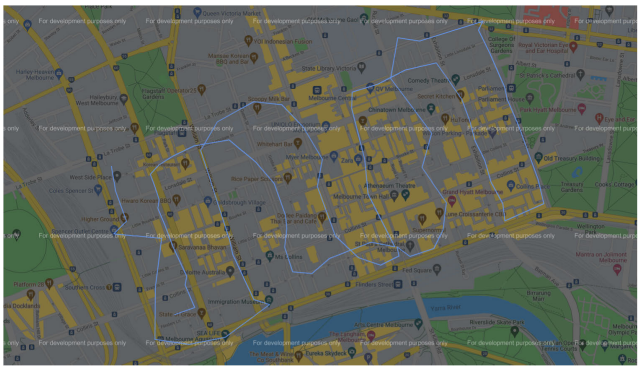


FIGURE 7. User mobility.

FIGURE 8. Resource location.

generating fog devices. The settings used for device configuration in iFogSim2 [58] are listed in Table 5. The typical unit of measurement for latency is milliseconds, which indicates the amount of time it takes for a tuple to travel from the sensor to the mobile device, from the mobile device to the Fog, from the Fog to the Proxy, from the Proxy to the Cloud, and between clusters of Fog nodes. Table 6 displays the values assigned to these parameters in the configuration settings of iFogSim2.

Fog devices are the computational devices in iFogSim2. Computational gadgets, on the other hand, come in various levels. On Level 3, the parent node is a cloud server. The fog nodes are connected to the cloud server via a proxy server at Level 2. Fog nodes are located closer to the user at Level 1, giving more frequent computational and storage capacities. Sensors and actuators are used in Level 0 IoT devices. The MicroserviceFogDevice, Actuator, and Sensor classes of iFogSim2 simulate the physical topology.

These scenarios were simulated on an Intel Core i7 CPU running at 1.80 GHz and 4GB of RAM. The fractional selectivity of the input-output relationship inside a module is set to 1.0.

VI. RESULTS AND DISCUSSION

A. DATA SET

This work uses EUA dataset [59], which provides the position information of the fog nodes deployed across Central

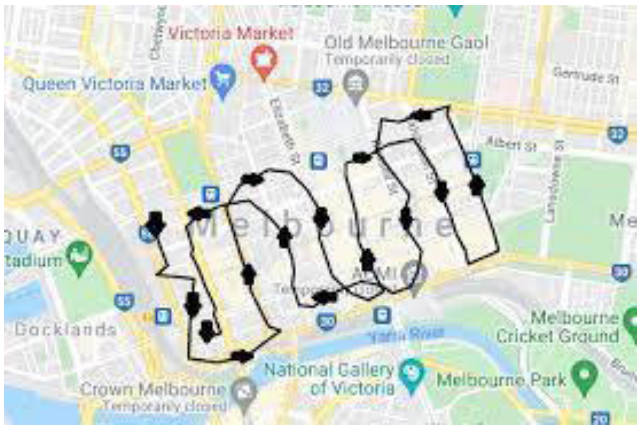


FIGURE 9. Directional mobility of the user.

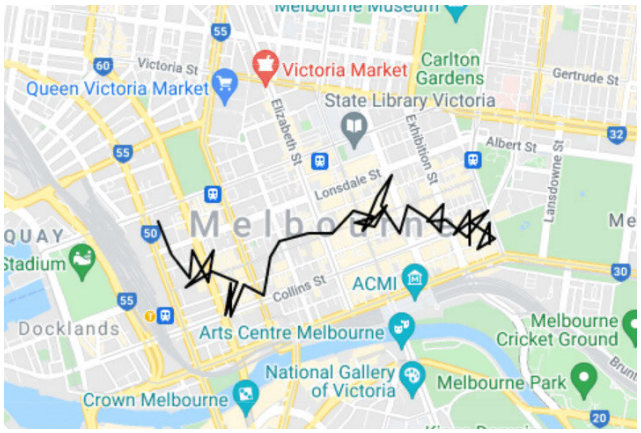


FIGURE 10. Random mobility of the user.

TABLE 5. Configuration parameters [58].

Parameter	Cloud	Fog	Smartphone
CPU length(MIPS)	44800	2800	2800
RAM (MB)	40000	4000	4000
Uplink BW(MB)	100	10000	10000
Downlink BW(MB)	10000	10000	10000
Busy power(J)	16*103	107.339	87.53
Idle power(J)	16*83.25	83.433	82.44

TABLE 6. Simulation parameters [58].

Latency (ms)	Value
Sensor to mobile	5
mobile to Fog	20
Fog to Proxy	20
Proxy to Cloud	30
Fog node clusters	2

Business District zones of major Australian cities such as Melbourne and Sydney. The dataset is segmented into various regions and is divided into several blocks, among which a specific node is picked as the proxy server at random to ensure impartiality. Within a block, all nodes except the proxy server serve as the gateway for IoT devices. This repository includes a collection of EUA datasets gathered from real-world data sources. The datasets have been made available to the public in assisting edge computing research. The data corresponds to the Australian region and helps in better simulation of a real-time environment. Figures 7 and 8 show the characteristics

TABLE 7. Results of Modified Genetic algorithm based resource provisioning.

Genetic algorithm	Edge	Fog	Cloud
Cloud energy(J)	2728189	2664000	2717900
Router energy (J)	174607	174718	172110
Cost ($)	67162	81529	92414
Network use (B)	8309	9961	13907
Latency (ms)	23.4	23	247
Execution time (ms)	747	1417	1660

TABLE 8. Results of Modified Flower pollination algorithm based resource provisioning.

Flower pollination algorithm	Edge	Fog	Cloud
Cloud energy (J)	2728556	2732075	2734776
Router energy (J)	174608	174718	172719
Cost($)	58354	68384	79714
Network use(B)	8709	10045	13773
Latency(ms)	23.3	25.6	249
Execution time(ms)	781	1337	1381

of user mobility and resource location in the dataset, and Figures 9 and 10 show directional movement, and random movement of the user [59].

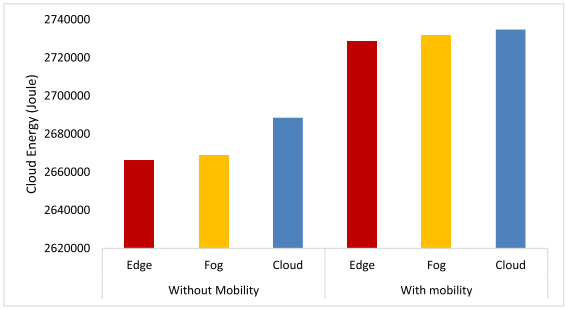
B. OBSERVATIONS

The proposed work compares the system's performance with two approaches, namely, cloud-only and edge-fog with cloud, changing the deployment of meta-heuristic algorithms for resource management. The section also presents the results with and without mobility considerations of IoT devices. Parameters, namely cloud energy, router energy, cost, network use, latency, and execution time, are analyzed with and without mobility as presented in Figure 11. The proposed system's performance is evaluated using the modified genetic algorithm for resource provisioning, considering the identified parameters, and are presented in Table 7 by utilizing the evaluation metrics discussed in section IV-B. The proposed system's performance is also evaluated using the modified flower pollination algorithm, considering the same parameters, and is presented in Table 8. The fitness of solutions for modified FPA and modified GA are depicted in Figure 12. The comparison of fog implementation, edge implementation, and cloud-only implementations for the identified parameters using modified GA, modified FPA, the multi-objective optimization method and the existing (First Come First Serve) FCFS method in the simulator are presented in Figure 13.

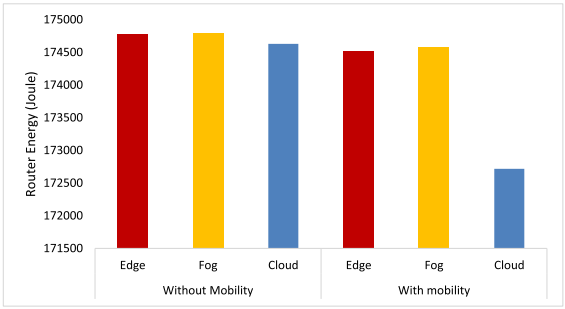
The results presented in Figure 14 and Figure 15 presents the results of modified FPA and modified GA for resource provisioning while taking into account directional and random mobility models for user movements with microservice placement and clustering approach and compares the proposed models for different movement patterns of a patient.

1) OBSERVATIONS FROM THE INTEGRATION OF MOBILITY AND MICROSERVICE FEATURES IN EDGE-FOG-CLOUD COMPUTING ENVIRONMENTS

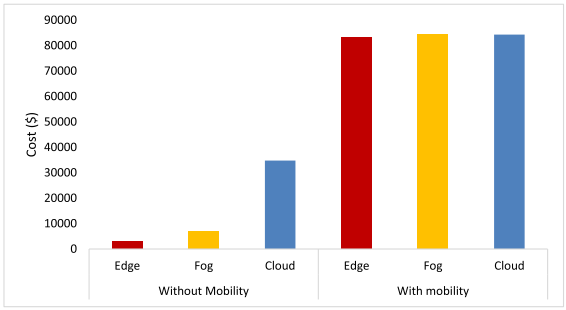
- In comparison to simulations lacking mobility components, simulations with mobility systems are practical.



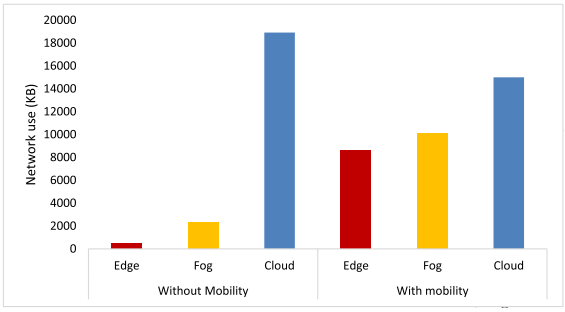
(a) Cloud Energy



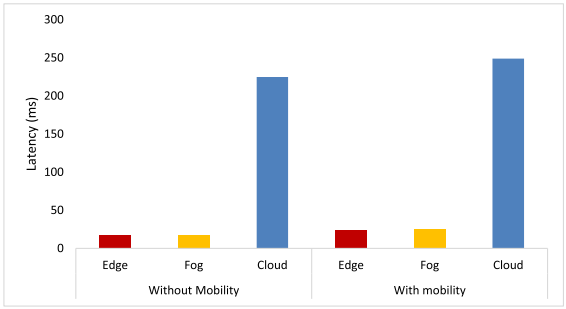
(b) Router Energy



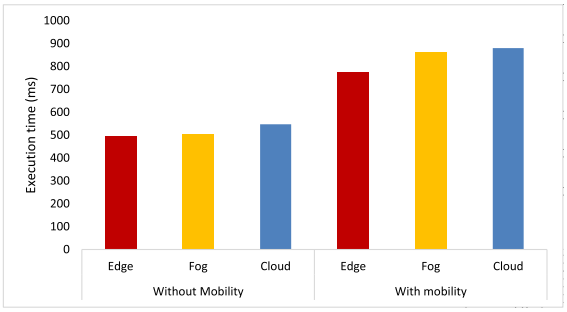
(c) Cost



(d) Network use



(e) Latency



(f) Execution time

FIGURE 11. Result graphs with mobility and microservice integration.

TABLE 9. Sensitivity analysis for modified GA.

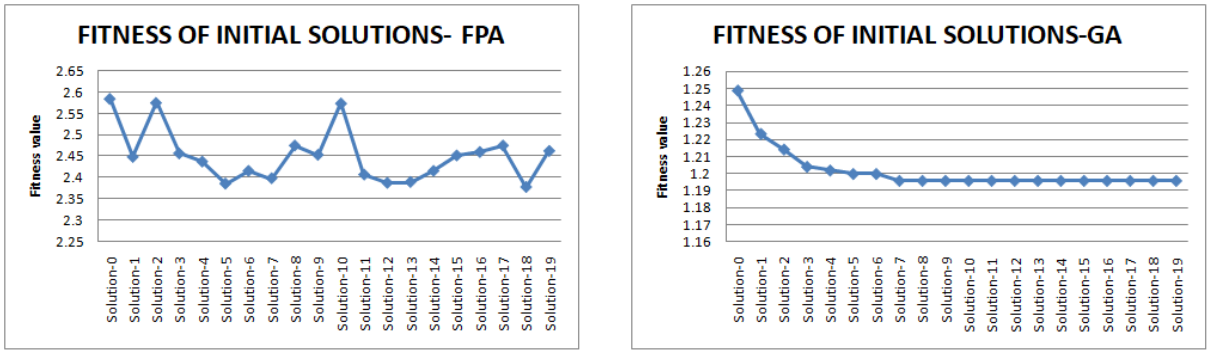
(a) One percentage of increase in input

Genetic algorithm	Edge	Fog	Cloud
Cloud energy(J)	2720126	2721507	2729185
Router energy (J)	173851	173934	172348
Cost ($)	67162	81529	92414
Network use (B)	8309	8668	13586
Latency (ms)	24.3	25.8	249
Execution time (ms)	808	995	1001

(b) One percentage of decrease in input

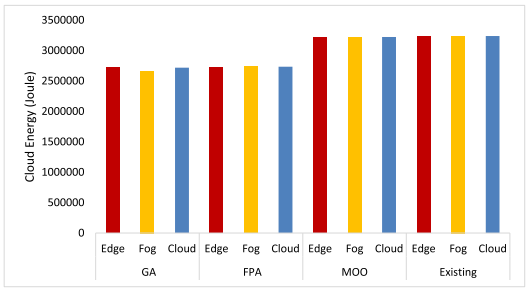
Genetic algorithm	Edge	Fog	Cloud
Cloud energy(J)	2705161	2712235	2720227
Router energy (J)	173845	174028	172583
Cost ($)	58353	68384	79714
Network use (B)	9857	11029	17340
Latency (ms)	21.3	23.8	249
Execution time (ms)	736	825	954

- Cloud energy, cost, network usage, execution time, and latency are all reduced in the fog computing model since processing happens at the lower fog level, which is very close to the end device both in mobility and without mobility implementations. This brings practical proof of performing the computation at the fog level.
- Router energy consumption is less for cloud-only implementations if we deploy mobility and microservice

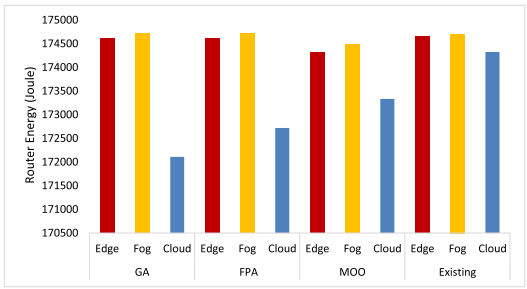


(a) Modified FPA (b) Modified GA

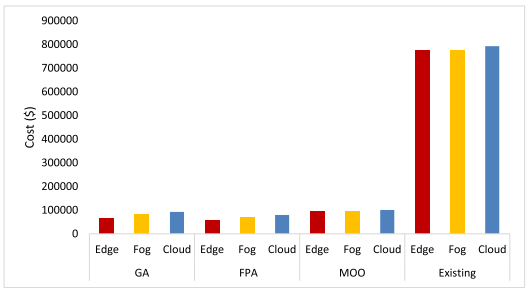
FIGURE 12. **Fitness values.**



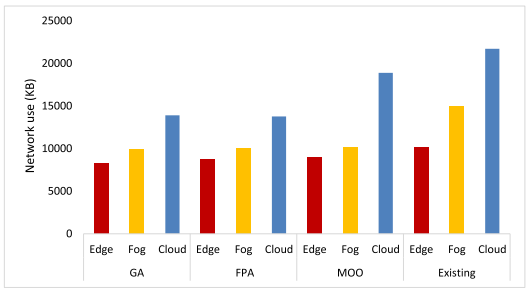
(a) Cloud Energy



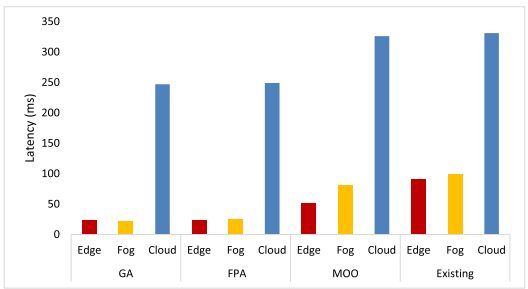
(b) Router Energy



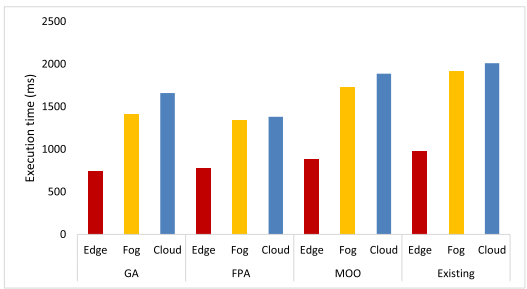
(c) Cost



(d) Network use



(e) Latency

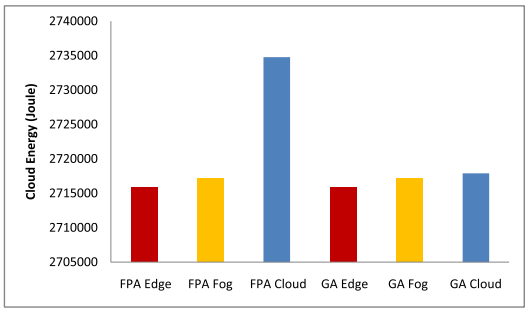


(f) Execution time

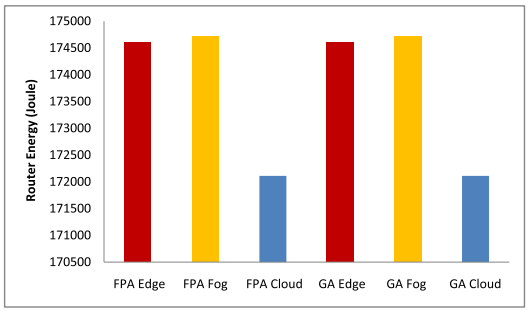
FIGURE 13. **Result graphs comparing modified GA, modified FPA, multi-objective optimization method and existing FCFS method.**

concepts since microservice modules are loosely coupled, which consume less energy than monolithic architecture.

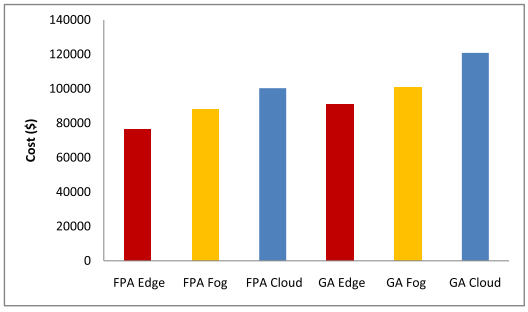
- If we include mobility in our application, the network use, execution time, and delay for cloud and fog scenarios is slightly longer in milliseconds since mobility



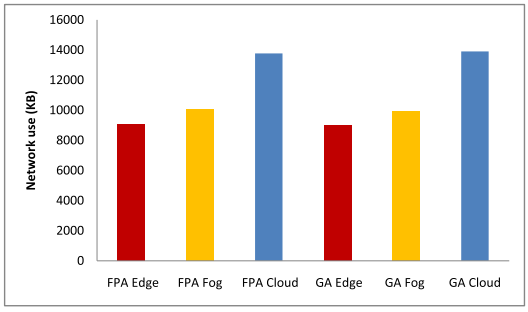
(a) Cloud Energy



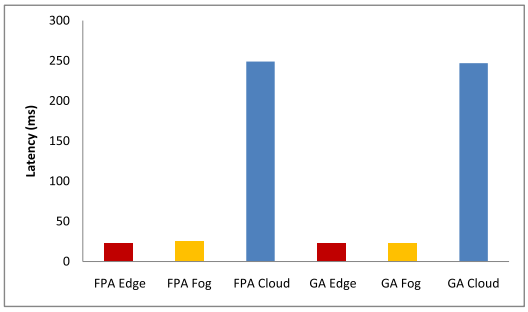
(b) Router Energy



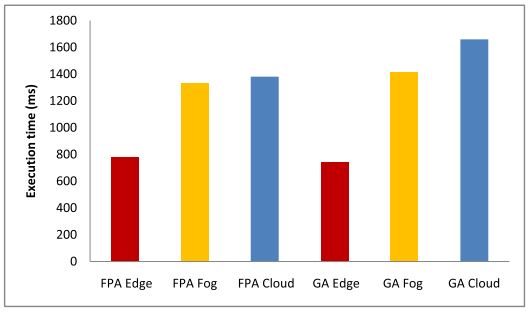
(c) Cost



(d) Network use



(e) Latency



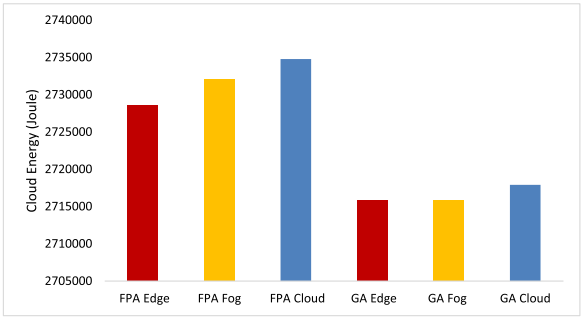
(f) Execution time

FIGURE 14. **Results of modified FPA and modified GA in fog and edge computing using directional mobility user model.**

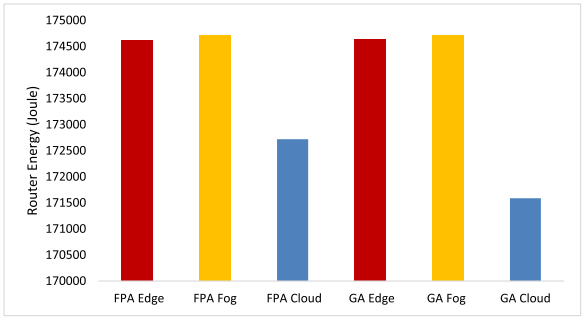
requires service migration which needs more network requirements and causes an increase in execution time and delay.

2) OBSERVATIONS FROM THE INTEGRATION OF META-HEURISTIC METHODS IN EDGE-FOG-CLOUD COMPUTING ENVIRONMENTS

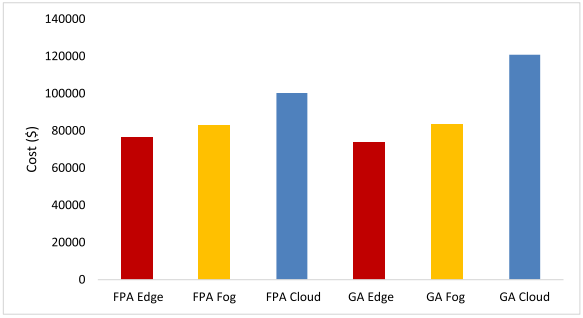
- Metaheurtic methods provide efficient provisioning compared to existing systems. The reason being, meta-heuristic methods converge to optimal or sub-optimal solutions at a faster rate when compared to multi-objective optimization approaches.
- The energy consumption of routers are same for GA and FPA since more computations are happening in the fog layer.
- When compared to GA, FPA has less cost of execution since FPA converges faster than GA. GA and FPA have almost the same network use and latency.
- In cloud scenarios, FPA takes less time to execute since FPA converges fast, but GA and FPA take about the same amount of time to execute in edge/fog scenarios.
- While considering the parameters, cloud energy, cost, network use, execution time, and latency, genetic algorithm, and FPA outperforms the existing resource provisioning methods since meta-heuristic methods converge to optimal or sub-optimal solutions fast. However, router energy is slightly higher than traditional resource provisioning methods since more computations are happening in the fog layer.
- In comparison, cloud energy consumption decreases by 15%, network use by 7%, cost by 29%, execution time



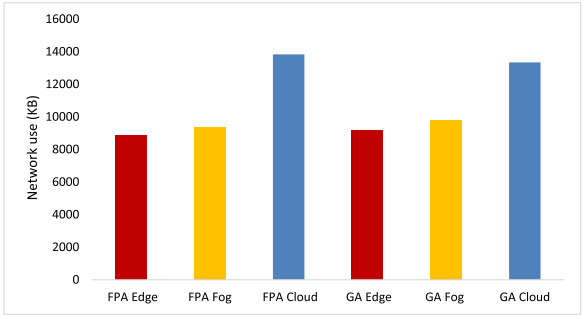
(a) Cloud Energy



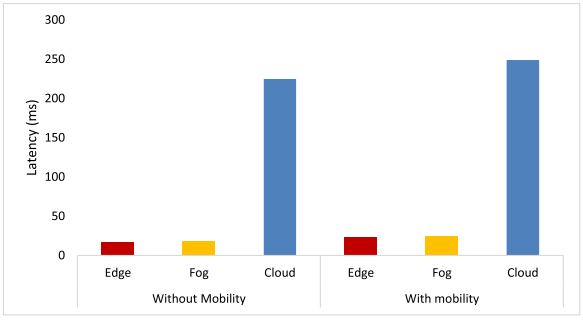
(b) Router Energy



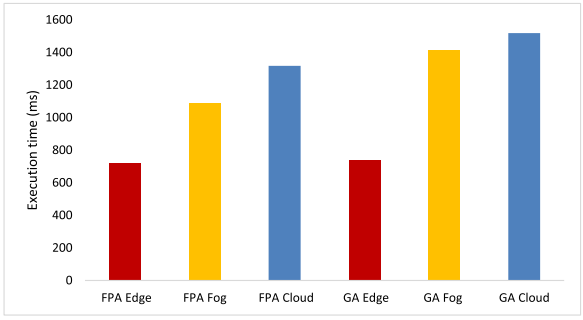
(c) Cost



(d) Network use



(e) Latency



(f) Execution time

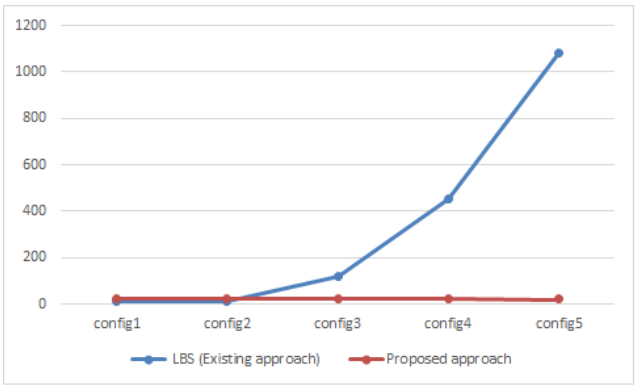
FIGURE 15. **Results of modified FPA and modified GA in fog and edge computing using random mobility user model.**

by 16% and latency by 55% when using meta-heuristic based resource provisioning approach for edge computing. While cloud energy consumption decreases by 18%, network use by 33%, cost by 16%, execution time by 18% and latency by 72% when using meta-heuristic based resource provisioning approach for fog computing.

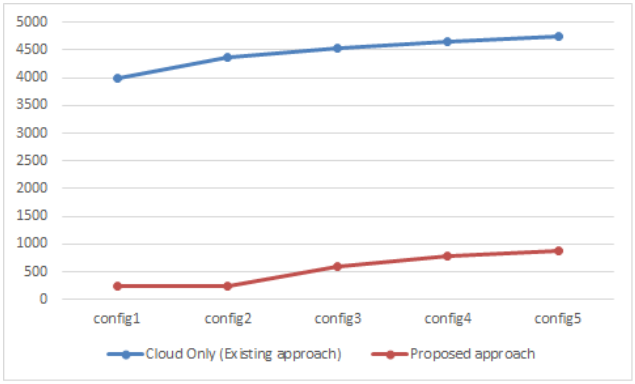
- The router energy consumption is 0.1% more for edge/fog computing since more computations are happening in the fog layer for the meta-heuristic methods considered in this work.

3) OBSERVATIONS OF DIRECTIONAL AND RANDOM MOBILITY USER MODELS IN EDGE-FOG-CLOUD COMPUTING ENVIRONMENTS

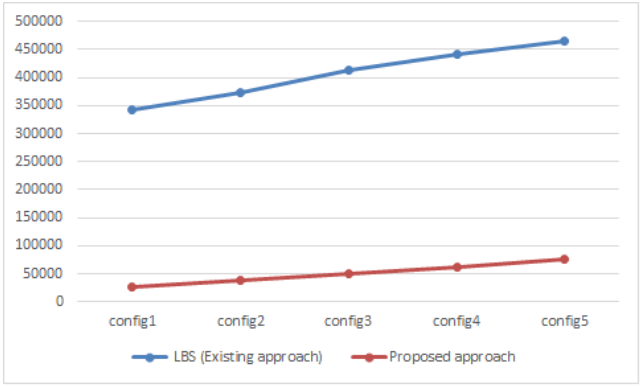
- Cloud energy is lesser for edge computing than fog computing since computation occurs only in edge devices.
- Router energy is the same for edge computing compared to fog computing because computations are happening in the router for edge and fog computing in a similar manner.
- Since computations are happening in the edge layer, edge computing has less cost of execution, network



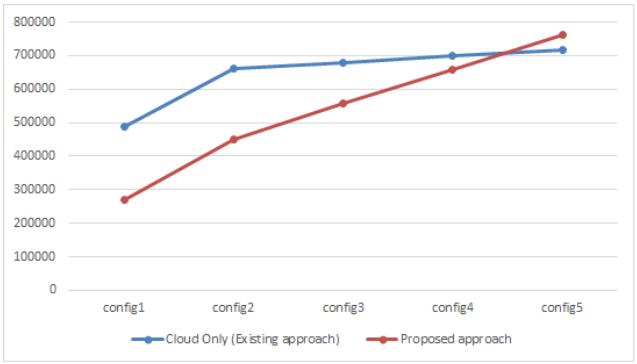
(a) Latency fog



(b) Latency cloud

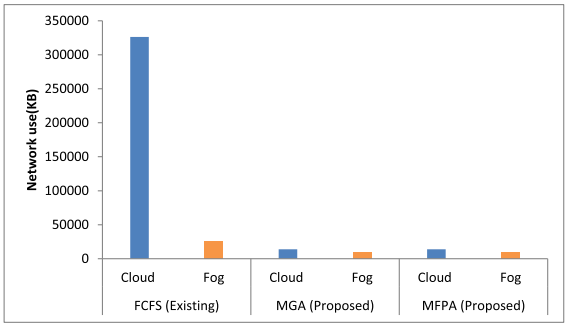


(c) Network use fog

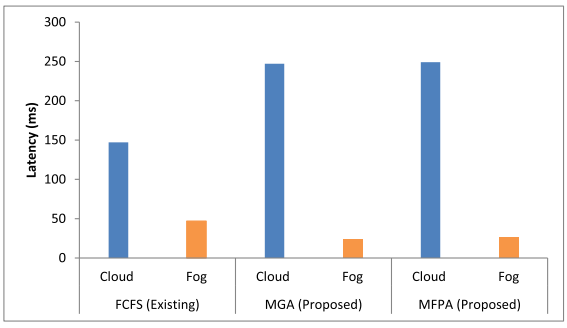


(d) Network use cloud

FIGURE 16. Comparative analysis [44].



(a) Network use comparison with the existing method



(b) Latency comparison with the existing method

FIGURE 17. Comparative analysis [60].

use, latency, and execution time than fog computing for directional and random user mobility models.

In conclusion, edge computing is superior to fog computing, and the resource provisioning efficiency of the meta-heuristic techniques is high. However, in real-time scenarios, there may be around 30 hops between the IoT device and the destination server making fog computing and cloud computing very distinct in real-world deployments. In such scenarios, it is evident that the deployment of the microservices in the edge/fog layers would be beneficial. However, since we only simulate a few hops, there will not be a significant difference between edge computing and fog computing in the simulator.

4) COMPARATIVE ANALYSIS

In order to have an effective conclusion, the proposed approach has been compared for its network use and the latency parameters against the existing results presented in

TABLE 10. Sensitivity analysis for modified FPA.

(a) One percentage of increase in input

FPA	Edge	Fog	Cloud
Cloud energy(J)	2715807	2717151	2723475
Router energy (J)	174061	173649	172052
Cost ($)	73448	75352	83116
Network use (B)	8308	8803	13539
Latency (ms)	24	25.6	248
Execution time (ms)	764	806	965

(b) One percentage of decrease in input

FPA	Edge	Fog	Cloud
Cloud energy(J)	2704068	2706462	2717702
Router energy (J)	173819	174005	172664
Cost ($)	56806	60199	76135
Network use (B)	10181	10948	16789
Latency (ms)	21.4	25	249
Execution time (ms)	788	790	840

TABLE 11. Topologies for simulation.

Configurations	End IoT devices	Total IoT devices
config 1	4	24
config 2	6	36
config 3	8	48
config 4	10	60
config 5	12	72

the literatures [44] and [60] which are presented in Figures 16 and 17. The comparison of the parameters for the different architectures named config 1, 2, 3, 4, and 5, the description of which is listed in Table 11, is depicted in Figure 16. These results also prove the conclusion statement in the previous sub-section.

5) SENSITIVITY ANALYSIS

The proposed model has been tested for its sensitivity to minor variations in the input feed. The results thus obtained conclude that the small changes in the input cause little to no change in the output measurements. This proves that the model is less sensitive to minimal changes in the input and is robust. It has been seen that a slight change in the input causes little to no change in the output parameters. Tables 9 and 10 show the findings for the same.

VII. CONCLUSION AND FUTURE WORK

Due to the heterogeneous and dynamic nature of critical medical IoT applications in fog scenarios, efficient resource management becomes a crucial problem. This paper investigates the trade-off between mobility and non-mobility of IoT devices using the modified meta-heuristic-based fog resource provisioning approach in microservice-based IoT medical applications. In addition, we also examine the performance of the proposed algorithms using the identified parameters and prove that the modified algorithms outperform the existing ones, which is very much needed for a time-critical medical application. The simulation's assumptions include the availability of resources in the edge and fog computing infrastructure and reliable network connectivity to handle application traffic. However, it may not take into account regulatory and legal requirements for patient data collection and usage, as well as the behavior of real-world users and stakeholders, which can affect the application's adoption and effectiveness and the proposed system considers only ECG sensor values for deployment, but it can be extended to any other sensor by changing the attributes of the sensor class.

As part of future work, we plan to pursue various research directions in this domain, put the proposed strategy into practice, and address the proposed system's scalability. We will investigate, enhance, and apply more hybrid meta-heuristic algorithms to solve scheduling problems that take into account task dependency and priority. We aim to tackle and overcome the scalability limitations present in the proposed approach, providing valuable information for real-world deployments. We plan to extend our technique to other critical real-time IoT applications which can be deployed in real-time edge/fog/cloud environments. Future directions for the simulation include considering the Edge AI and Blockchain technologies on the application's deployment and performance in edge and fog computing environments.

REFERENCES

[1] J. Santos, T. Wauters, B. Volckaert, and F. D. Turck, "Towards end-to-end resource provisioning in fog computing over low power wide area networks," *J. Netw. Comput. Appl.*, vol. 175, Feb. 2021, Art. no. 102915.
[2] V. K. Quy, N. V. Hau, D. V. Anh, and L. A. Ngoc, "Smart healthcare IoT applications based on fog computing: Architecture, applications and challenges," *Complex Intell. Syst.*, vol. 8, no. 5, pp. 3805–3815, 2021.
[3] S. Tuli, N. Basumatary, S. S. Gill, M. Kahani, R. C. Arya, G. S. Wander, and R. Buyya, "HealthFog: An ensemble deep learning based smart healthcare system for automatic diagnosis of heart diseases in integrated IoT and fog computing environments," *Future Gener. Comput. Syst.*, vol. 104, pp. 187–200, Mar. 2020.
[4] V. Sankar, M. N. Devi, and M. Jayakumar., "Data augmented hardware trojan detection using label spreading algorithm based transductive learning for edge computing-assisted IoT devices," *IEEE Access*, vol. 10, pp. 102789–102803, 2022.
[5] I. Martinez, A. S. Hafid, and A. Jarray, "Design, resource management, and evaluation of fog computing systems: A survey," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2494–2516, Feb. 2021.
[6] M. Laroui, B. Nour, H. Moungla, M. A. Cherif, H. Afifi, and M. Guizani, "Edge and fog computing for IoT: A survey on current research activities future directions," *Comput. Commun.*, vol. 180, pp. 210–231, Dec. 2021.
[7] A. Shakarami, H. Shakarami, M. Ghobaei-Arani, E. Nikougoftar, and M. Faraji-Mehmandar, "Resource provisioning in edge/fog computing: A comprehensive and systematic review," *J. Syst. Archit.*, vol. 122, Jan. 2022, Art. no. 102362.
[8] M. Etemadi, M. Ghobaei-Arani, and A. Shahidinejad, "Resource provisioning for IoT services in the fog computing environment: An autonomic approach," *Comput. Commun.*, vol. 161, pp. 109–131, Sep. 2020.
[9] M. Ghobaei-Arani, A. Souri, and A. A. Rahmanian, "Resource management approaches in fog computing: A comprehensive review," *J. Grid Comput.*, vol. 18, no. 1, pp. 1–42, Mar. 2020.
[10] Y. Wang, Q. Cui, and K. Chen, "Machine learning enables predictive resource recommendation for minimal latency mobile networking," in *Proc. IEEE 32nd Annu. Int. Symp. Pers., Indoor Mobile Radio Commun. (PIMRC)*, Sep. 2021, pp. 1363–1369.
[11] C. Puliafito, D. M. Gonçalves, M. M. Lopes, L. L. Martins, E. Madeira, E. Mingozzi, O. Rana, and L. F. Bittencourt, "MobFogSim: Simulation of mobility and migration for fog computing," *Simul. Model. Pract. Theory*, vol. 101, May 2020, Art. no. 102062.

[12] M. Shinu and M. Supriya, "Performance comparison of VM allocation and selection policies in an integrated fog-cloud environment," in *Proc. Int. Conf. Ubiquitous Commun. Netw. Comput.* Cham, Switzerland: Springer, 2021, pp. 169–184.

[13] H. Pydi and G. N. Iyer, "Analytical review and study on load balancing in edge computing platform," in *Proc. 4th Int. Conf. Comput. Methodolog. Commun. (ICCMC)*, Mar. 2020, pp. 180–187.

[14] B. E. Khalyly, A. Belangour, M. Banane, and A. Erraissi, "A comparative study of microservices-based IoT platforms," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 8, pp. 389–398, 2020.

[15] R. Mahmud, S. Pallewatta, M. Goudarzi, and R. Buyya, "IFogSim2: An extended iFogSim simulator for mobility, clustering, and microservice management in edge and fog computing environments," *J. Syst. Softw.*, vol. 190, Aug. 2022, Art. no. 111351.

[16] M. Faraji Mehmandar, S. Jabbehdari, and H. H. S. Javadi, "A dynamic fog service provisioning approach for IoT applications," *Int. J. Commun. Syst.*, vol. 33, no. 14, Sep. 2020, Art. no. e4541.

[17] M. Shakil, A. F. Y. Mohammed, R. Arul, A. K. Bashir, and J. K. Choi, "A novel dynamic framework to detect DDoS in SDN using metaheuristic clustering," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 3, Mar. 2022, Art. no. e3622.

[18] W. Rankothge, F. Le, A. Russo, and J. Lobo, "Optimizing resource allocation for virtualized network functions in a cloud center using genetic algorithms," *IEEE Trans. Netw. Service Manage.*, vol. 14, no. 2, pp. 343–356, Jun. 2017.

[19] V. Porkodi, A. R. Singh, A. R. W. Sait, K. Shankar, E. Yang, C. Seo, and G. P. Joshi, "Resource provisioning for cyber–physical–social system in cloud-fog-edge computing using optimal flower pollination algorithm," *IEEE Access*, vol. 8, pp. 105311–105319, 2020.

[20] B. Farahani, M. Barzegari, F. S. Aliee, and K. A. Shaik, "Towards collaborative intelligent IoT eHealth: From device to fog, and cloud," *Microprocessors Microsyst.*, vol. 72, Feb. 2020, Art. no. 102938.

[21] F. Al-Doghman, N. Moustafa, I. Khalil, N. Sohrabi, Z. Tari, and A. Y. Zomaya, "AI-enabled secure microservices in edge computing: Opportunities and challenges," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1485–1504, Mar. 2023.

[22] S. K. Mishra, D. Puthal, J. J. P. C. Rodrigues, B. Sahoo, and E. Dutkiewicz, "Sustainable service allocation using a metaheuristic technique in a fog server for industrial applications," *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4497–4506, Oct. 2018.

[23] A. Bajaj and O. P. Sangwan, "A systematic literature review of test case prioritization using genetic algorithms," *IEEE Access*, vol. 7, pp. 126355–126375, 2019.

[24] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Application management in fog computing environments: A taxonomy, review and future directions," *ACM Comput. Surv.*, vol. 53, no. 4, pp. 1–43, Jul. 2021.

[25] R. Panwar and M. Supriya, "Dynamic resource provisioning for service-based cloud applications: A Bayesian learning approach," *J. Parallel Distrib. Comput.*, vol. 168, pp. 90–107, Oct. 2022.

[26] D. R. Vemula, M. K. Morampudi, S. Maurya, A. Abdul, M. M. Hussain, and I. Kavati, "Enhanced resource provisioning and migrating virtual machines in heterogeneous cloud data center," *J. Ambient Intell. Humanized Comput.*, vol. 13, pp. 1–12, Jul. 2022.

[27] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah, "Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing," *IEEE Access*, vol. 11, pp. 20635–20646, 2023.

[28] P. Hosseinioun, M. Kheirabadi, S. R. K. Tabbakh, and R. Ghaemi, "A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm," *J. Parallel Distrib. Comput.*, vol. 143, pp. 88–96, Sep. 2020.

[29] A. Yousefpour, A. Patil, G. Ishigaki, I. Kim, Wang, Xi, H. C. Cankaya, Q. Zhang, W. Xie, and J. P. Jue, "FOGPLAN: A lightweight QoS-aware dynamic fog service provisioning framework," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 5080–5096, Jun. 2019.

[30] P. G. V. Naranjo, E. Baccarelli, and M. Scarpiniti, "Design and energy-efficient resource management of virtualized networked fog architectures for the real-time support of IoT applications," *J. Supercomput.*, vol. 74, no. 6, pp. 2470–2507, Jun. 2018.

[31] I. Lera, C. Guerrero, and C. Juiz, "YAFS: A simulator for IoT scenarios in fog computing," *IEEE Access*, vol. 7, pp. 91745–91758, 2019.

[32] T. Qayyum, A. W. Malik, M. A. K. Khattak, O. Khalid, and S. U. Khan, "FogNetSim++: A toolkit for modeling and simulation of distributed fog environment," *IEEE Access*, vol. 6, pp. 63570–63583, 2018.

[33] Z. Fang, J. Wang, Y. Ren, Z. Han, H. V. Poor, and L. Hanzo, "Age of information in energy harvesting aided massive multiple access networks," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 5, pp. 1441–1456, May 2022.

[34] M. Abdel-Basset, D. El-Shahat, M. Elhoseny, and H. Song, "Energy-aware metaheuristic algorithm for industrial-Internet-of-Things task scheduling problems in fog computing applications," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12638–12649, Aug. 2021.

[35] M. Adhikari, S. N. Srirama, and T. Amgoth, "A comprehensive survey on nature-inspired algorithms and their applications in edge computing: Challenges and future directions," *Softw., Pract. Exper.*, vol. 52, no. 4, pp. 1004–1034, Apr. 2022.

[36] A. M. Maia, Y. Ghamri-Doudane, D. Vieira, and M. F. de Castro, "An improved multi-objective genetic algorithm with heuristic initialization for service placement and load distribution in edge computing," *Comput. Netw.*, vol. 194, Jul. 2021, Art. no. 108146.

[37] Z. Fang, J. Wang, J. Du, X. Hou, Y. Ren, and Z. Han, "Stochastic optimization-aided energy-efficient information collection in Internet of underwater things networks," *IEEE Internet Things J.*, vol. 9, no. 3, pp. 1775–1789, Feb. 2022.

[38] D. H. Brahmbhatt and M. R. Cowie, "Remote management of heart failure: An overview of telemonitoring technologies," *Cardiac Failure Rev.*, vol. 5, no. 2, pp. 86–92, May 2019.

[39] M. Prabhu and A. Hanumanthaiah, "Edge computing-enabled healthcare framework to provide telehealth services," in *Proc. Int. Conf. Wireless Commun. Signal Process. Netw. (WiSPNET)*, Mar. 2022, pp. 349–353.

[40] H. Nashaat, E. Ahmed, and R. Rizk, "IoT application placement algorithm based on multi-dimensional QoE prioritization model in fog computing environment," *IEEE Access*, vol. 8, pp. 111253–111264, 2020.

[41] A. Kumari, V. Kumar, M. Y. Abbasi, S. Kumari, P. Chaudhary, and C. Chen, "CSEF: Cloud-based secure and efficient framework for smart medical system using ECC," *IEEE Access*, vol. 8, pp. 107838–107852, 2020.

[42] A. A. Mutlag, M. K. A. Ghani, N. Arunkumar, M. A. Mohammed, and O. Mohd, "Enabling technologies for fog computing in healthcare IoT systems," *Future Gener. Comput. Syst.*, vol. 90, pp. 62–78, Jan. 2019.

[43] M. Hartmann, U. S. Hashmi, and A. Imran, "Edge computing in smart health care systems: Review, challenges, and research directions," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 3, Mar. 2022, Art. no. e3710.

[44] A. Asghar, A. Abbas, H. A. Khattak, and S. U. Khan, "Fog based architecture and load balancing methodology for health monitoring systems," *IEEE Access*, vol. 9, pp. 96189–96200, 2021.

[45] K. P. N. Jayasena and B. S. Thisarasinghe, "Optimized task scheduling on fog computing environment using meta heuristic algorithms," in *Proc. IEEE Int. Conf. Smart Cloud (SmartCloud)*, Dec. 2019, pp. 53–58.

[46] Y. Qiu, H. Zhang, and K. Long, "Computation offloading and wireless resource management for healthcare monitoring in fog-computing-based Internet of Medical Things," *IEEE Internet Things J.*, vol. 8, no. 21, pp. 15875–15883, Nov. 2021.

[47] M. Abdel-Basset, G. Manogaran, A. Gamal, and V. Chang, "A novel intelligent medical decision support model based on soft computing and IoT," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4160–4170, May 2020.

[48] H. Ben Hassen, W. Dghais, and B. Hamdi, "An E-health system for monitoring elderly health based on Internet of Things and fog computing," *Health Inf. Sci. Syst.*, vol. 7, no. 1, pp. 1–9, Dec. 2019.

[49] T. Nguyen Gia, A. M. Rahmani, T. Westerlund, P. Liljeberg, and H. Tenhunen, "Fog computing approach for mobility support in Internet-of-Things systems," *IEEE Access*, vol. 6, pp. 36064–36082, 2018.

[50] A. Benayache, A. Bilami, S. Barkat, P. Lorenz, and H. Taleb, "MsM: A microservice middleware for smart WSN-based IoT application," *J. Netw. Comput. Appl.*, vol. 144, pp. 138–154, Oct. 2019.

[51] D. Yu, Y. Jin, Y. Zhang, and X. Zheng, "A survey on security issues in services communication of microservices-enabled fog applications," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 22, Nov. 2019, Art. no. e4436.

[52] X. Zhao and C. Huang, "Microservice based computational offloading framework and cost efficient task scheduling algorithm in heterogeneous fog cloud network," *IEEE Access*, vol. 8, pp. 56680–56694, 2020.

[53] M. Abdullah, W. Iqbal, A. Mahmood, F. Bukhari, and A. Erradi, "Predictive autoscaling of microservices hosted in fog microdata center," *IEEE Syst. J.*, vol. 15, no. 1, pp. 1275–1286, Mar. 2021.

[54] S. Pallewatta, V. Kostakos, and R. Buyya, "Placement of microservices-based IoT applications in fog computing: A taxonomy and future directions," 2022, *arXiv:2207.05399*.

[55] M. Supriya, K. Sangeeta, and G. K. Patra, "Trustworthy cloud service provider selection using multi criteria decision making methods," *Eng. Lett.*, vol. 24, no. 1, pp. 1–10, 2016. [Online]. Available: www.scopus.com
[56] A. Mseddi, W. Jaafar, H. Elbiaze, and W. Ajib, "Joint container placement and task provisioning in dynamic fog computing," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10028–10040, Dec. 2019.
[57] L. Yang, H. Yao, J. Wang, C. Jiang, A. Benslimane, and Y. Liu, "Multi-UAV-Enabled load-balance mobile-edge computing for IoT networks," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 6898–6908, Aug. 2020.
[58] H. Gupta, A. Vahid Dastjerdi, S. K. Ghosh, and R. Buyya, "IFogSim: A toolkit for modeling and simulation of resource management techniques in the Internet of Things, edge and fog computing environments," *Softw., Pract. Exper.*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017.
[59] P. Lai, Q. He, M. Abdelrazek, F. Chen, J. Hosking, J. Grundy, and Y. Yang, "Optimal edge user allocation in edge computing with variable sized vector bin packing," in *Proc. Int. Conf. Service-Oriented Comput.* Cham, Switzerland: Springer, 2018, pp. 230–245.
[60] R. Paesani, G. Paolone, P. D. Felice, D. Iachetti, and M. Marinelli, "iFogSim simulations on IoT computational alternatives," *Eng. Proc.*, vol. 31, no. 1, p. 44, 2022.

SHINU M. RAJAGOPAL received the B.Tech. degree in electronics and communications from Calicut University, India, and the M.E. degree in computer science and engineering from Anna University, India, in 2008. She is currently pursuing the Ph.D. degree with Amrita Vishwa Vidyapeetham, Bengaluru, India. She has more than ten years of teaching experience. Her research interests include distributed computing, cloud computing, fog computing, and edge computing.

M. SUPRIYA received the Ph.D. degree in CSE from Amrita Vishwa Vidyapeetham, Bengaluru. She is currently an Associate Professor with the Department of Computer Science and Engineering, Amrita School of Computing, Bengaluru, India. She has more than 22 years of teaching experience, among which 12 years contribute to research. She has more than 40 Scopus-indexed publications in reputed journals and conferences. She is an active ACM professional member and leads various ACM club activities at the college. Her research interests include parallel/distributed computing, cloud computing, and fog computing. She also works on integrating cloud computing with the Internet of Things, which has emerged as a new challenging domain in research.

RAJKUMAR BUYYA (Fellow, IEEE) is currently a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, The University of Melbourne, Australia. He is also the Founder and the CEO of Manjrasoft, a spin-off company of the university, commercializing cloud computing technologies. He was an Honorary/Visiting Professor for several elite universities, including Imperial College London, U.K., and University of Birmingham, U.K. He has written over 850 publications, including seven textbooks. He has been recognized as a Web of Science Highly Cited Researcher for five consecutive years, since 2016, and a Scopus Researcher of the Year 2017 with Excellence in Innovative Research Award by Elsevier and the "Best of the World," in computing systems field, by The Australian 2019 Research Review.