IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 20 March 2024, accepted 11 April 2024, date of publication 16 April 2024, date of current version 23 April 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3388837

RESEARCH ARTICLE

Energy Efficient Real-Time Tasks Scheduling on High-Performance Edge-Computing Systems Using Genetic Algorithm

HAMEED HUSSAIN[1], MUHAMMAD ZAKARYA[2,3], (Senior Member, IEEE), AHMAD ALI[4], AYAZ ALI KHAN[5], MOHAMMAD REZA CHALAK QAZANI[2], MAHMOOD AL-BAHRI[2], AND MUHAMMAD HALEEM[6]

[1]Department of Computer Science, University of Buner, Buner 19290, Pakistan
[2]Faculty of Computing and Information Technology, Sohar University, Sohar 311, Oman
[3]Department of Computer Science, Abdul Wali Khan University, Mardan 23200, Pakistan
[4]College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China
[5]Department of Computer Science, University of Lakki Marwat, Taja Zai 28420, Pakistan
[6]Department of Computer Science, Kardan University, Kabul 1007, Afghanistan

Corresponding author: Muhammad Haleem (m.haleem@kardan.edu.af)

ABSTRACT With an increase in the number of processing cores or systems, the high-performance edge-computing system's power consumption along with its computational speed will increase, essentially. However, this comes at the expense of high-energy utilization. One notable solution to reduce the energy consumption of these systems is to execute these systems at the slowest feasible speed so that the job's deadline times are met. Unfortunately, this method is at the expense of more response time and performance loss. To resolve this issue, in this paper, we propose a scheduling approach that associates the genetic algorithm (GA) with the first feasible speed (FiFeS) technique i.e. GA-FiFeS algorithm. This does not jeopardize real-time tasks' deadlines. The GA-FiFeS algorithm proposes an energy-efficient schedule while still ensuring high response times. The results of the proposed approach, using plausible assumptions and experimental parameters, are compared with currently in-practice approaches, i.e. FiFeS and LeFeS (least feasible speed) approaches. Using numerical simulations and plausible assumptions, our investigation suggests that the proposed GA-FiFeS technique outperforms the FiFeS technique in terms of energy consumption (∼18.56%) and response times (∼2.78%). Furthermore, the GA-FiFeS has comparable outcomes with the LeFeS method while taking the expected time of execution as an assessment feature for analysis.

INDEX TERMS Genetic algorithm, edge-computing, multi-core, real-time systems, feasibility analysis, HPC.

I. INTRODUCTION

The power computation of various resources can be increased by using either high-performance computing (HPC) systems or multi-core technology. Distributed [1] and non-distributed [2] systems are the two main categorizations of the HPC systems, and the idea of distribution means the arrangement of physical processors on diverse system boards. From the notion of distributed HPC systems [1], the concepts of grids, cloud computing systems, and clusters are derived. Whereas, in the class of non-distributed HPC systems, multi-core systems are dominated [3]. To form a single system image in cluster computing systems, multiple storage and processing resources are interconnected through high-speed networks [1], [4]. For availability, load balancing, and performance improvement, the integration of software, hardware, and network resources is the primary goal of cluster computing systems [1], [4], [5], [6]. As a medium of resource sharing, the grid-computing concept depends on the internet [1]. For spread availability through the medium, the powerful computing resources are connected [7]. Grid computing encompasses user privileges, geographically

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

distributed resources, and belonging to different administrative domains as described in [1], [8], and [9]. Moreover, grids are loosely coupled, and more heterogeneous, in contrast to the traditional cluster computing systems [10], [11].

The basic motivation behind grid computing is complex problem-solving and powerful resource sharing. The authors in [12] and [13] have provided a comprehensive survey on grid computing systems. Cloud computing is another dimension of distributed computing systems that facilitates the users by providing three basic services: (1) software-as-a-service (SaaS), (2) platform-as-a-service (PaaS), and (3) infrastructure-as-a-service (IaaS). These services are provided through the Internet using the virtualization concept under a pay-as-you-use policy. The detailed comparison among cluster, grid, and cloud computing systems can be found in [1] and [14]. Cloud and edge computing are the two diverse architecture paradigms [15], [16], [17]. Edge computing is normally used for the data having time constraints as an important issue, while, in the cloud, the time sensitivity is not that important. In far-flung regions having low or no connectivity to a centralized location, edge-computing is a good choice as compared to the cloud, besides delay [15], [18].

Another technique for increasing the computation power of the resources is using multi-core technology [2], [19], [20]. The multi-core front drastically enhances existing technology, however, this technology faces some crucial challenges like thermal dissipation and lack of mature scheduling for executing computational-intensive applications [21]. Usually, cores run symmetrically by using the same power and timer occurrences stages [22], [23], wherever cores run asymmetrically if the situation exists and in asymmetric circumstances it is very precarious to maintain and keep the symmetrical performance [19], [24], [25]. Therefore, to tackle this issue, two kinds i.e. hardware and software solutions are planned [21], [26]. If all cores operate symmetrically, the software solution is the intelligent scheduling of applications [21]. While solution through hardware gives circuitry of dynamic voltages to every individual core [21]; however, the eventual outcome is power leakage and thermal throttling [21], [24]. The resolution of the said problem through software is less expensive and efficient, but it has given relatively less weight in the state-of-the-art literature from a scheduling perspective [21]. To fill the aforementioned gap, authors have used a rate-monotonic (RM) scheduling technique for power-efficient scheduling of real-time applications by executing all cores on the same frequency [21].

Applying diverse techniques, the processing power consumption of the computing systems can be reduced. As we know $P = v^2.f$ implies that power is straightforwardly corresponding to frequency or speed and voltage from the voltage, frequency, and power relationship. Thus, power can be managed either by changing the working voltage or speed progressively. This powerful change of voltage and speed parameters can be done by using the dynamic voltage scaling (DVS) method. From the above relation, it can be observed that the DVS technique can help in minimizing power consumption. However, this method increases response time [21]. Hence, response time acquired through DVS is challenging to get a handle on, especially continuous climate where the application needs to be processed within the deadline. So, minimizing power by adjusting system speed is a good option for real-time application [27].

In previous works [19], [21], authors have examined the speed factor in the prior research work presented in [23]; and termed it as the first feasible speed (FiFeS) method. Then, the authors addressed the problem of executing a system at the lowest possible speed and named this method as the least feasible speed (LeFeS) method. In this research, to improve the energy consumption and response time of real-time tasks, we are presenting a methodology by applying a notable heritable genetic algorithm (GA) in addition to the FiFeS approach and called it as GA-FiFeS approach.

Furthermore, influenced by the concept of Darwinian's theory of the ''survival of the fittest'' [28], [29], GA [30] is an optimization algorithm and the meaning is that the GA recalls the fittest genes in each repetition. By using any selection method, GA optimization progression, primarily fitness values of the offsprings are intended, and then some offsprings are designated, and the designated offsprings are then passed over to mutation and cross-over phases [30]. The fitness values are recalculated for the new offspring. The optimization process takes place if the new offspring's fitness values are good as compared to their old fitness values. The key contribution of this research work is that we propose a novel algorithm which is known as GA-FiFeS that outperforms FiFeS in terms of energy and LeFeS in view of execution time, respectively. The major contributions of this work are as follows:

- we propose a scheduling approach i.e. GA-FiFeS that associates the genetic algorithm (GA) with the first feasible speed (FiFeS) technique;
- the proposed approach offers an energy-efficient schedule for real-time tasks while still ensuring tasks' deadlines i.e. response times;
- GA-FiFeS does not jeopardize real-time tasks' deadlines i.e. tasks finished their execution within their deadlines; and
- empirical evaluation using plausible assumptions and comparison with state-of-the-art techniques.

The rest of the research work is organized as follows. An overview of the related work is presented in section II. In section III, we present the system model which is related to the system speed, power consumption, and system model. In section IV, we discuss the proposed work. After that, section V elaborates on the critical analysis of the empirical results formed by the proposed work. Furthermore, experimental setup and evaluation metrics are also described in this section. Finally, in section VI we conclude the work and discuss some directions for future research and investigation.

II. RELATED WORK

Designing energy-efficient scheduling strategies for real-time periodic tasks on heterogeneous platforms poses a formidable challenge, compounded by the computational complexity inherent in the problem of task scheduling. Consequently, there exists a considerable amount and quantity of real-time energy-aware scheduling approaches that are, in fact, applicable and have been studied in the context of such heterogeneous computational environments.

The authors in [31] have explored the evolving landscape of processing platforms in battery-supported real-time systems, which increasingly incorporate specialized multi-cores to address the demands of modern applications. It underscores the pressing need for energy-efficient schedulers in such environments. SEAMERS is a low-overhead heuristic strategy designed for dynamic voltage and frequency scaling (DVFS)-based energy-aware scheduling of real-time periodic tasks on heterogeneous multi-core platforms [31]. SEAMERS operates through four phases: (i) deadline partitioning; (ii) core clustering; (iii) task allocation; and (iv) energy-aware scheduling, which collectively aim to optimize task scheduling while considering both energy consumption and real-time constraints such as deadline and response time.

The authors in [32] have introduced HEALERS, a novel low-overhead heuristic strategy tailored for dynamic voltage and frequency scaling (DVFS) and dynamic power management (DPM) enabled energy-aware scheduling of periodic tasks on a heterogeneous multi-core system. The HEALERS strategy begins by employing deadline partitioning to delineate distinct time slices. At each time-slice boundary, a three-phase operation is executed to determine the schedule for the subsequent time-slice. First of all, the technique calculates the execution demand pieces of every activity on the platform across different processor cores. In the second phase, it then creates a schedule for every job on one or more processor cores, making sure that the combined execution requirements of all of the tasks are satisfied. Lastly, in order to reduce energy usage within the time slice while maintaining the execution needs of the planned activities, HEALERS applies DVFS and DPM mechanisms to all processor cores.

Reference [33] discusses the challenge of energy-efficient task scheduling on edge devices with limited power, proposing a multi-objective energy-efficient task scheduling technique (METSM). This technique establishes a mathematical model considering make-span and total energy consumption as optimization objectives, and it introduces a problem-specific algorithm called the iterated greedy-based multi-objective optimizer (IMO) to address task scheduling and resource allocation. The increasing adoption of Cloud services by Internet of Things (IoT) devices has raised concerns about latency, leading to the emergence of fog computing. Fog computing has an objective to minimize latency (enhance response time) by bringing processing capabilities closer to the end-users with the help of installing extra resources in the proximity of users and at the edge of the Cloud infrastructure. However, reducing latency or improving response times without negatively affecting the total energy consumption remains a challenge. In [34], a novel genetic-based algorithm called GAMMR is proposed to address the task scheduling problem in a fog-cloud-based environment, achieving an optimal balance between total consumed energy and total response time, outperforming standard genetic algorithms in simulations on various datasets.

Current real-time systems offer increased computational power to handle CPU-intensive applications, but this leads to higher energy consumption and heat dissipation [21]. To address these issues, system speed is dynamically adjusted to meet application deadlines while reducing overall energy consumption. Although multi-core technology presents opportunities for energy-efficient scheduling, it remains relatively unexplored. This paper proposes techniques to optimize core speed for individual tasks and balance core utilization, ensuring task deadlines are met, and overall system energy consumption is minimized, with experimental results demonstrating the effectiveness of the approach over existing methods.

Reference [23] introduces a framework for analyzing and designing embedded systems to minimize energy consumption while meeting timing requirements. Realistic assumptions are made, including discrete operating modes for the processor with associated speed and power consumption, and consideration of energy overhead and transition delay during mode switches. The method enables computation of the optimal sequence of voltage/speed changes to approximate the minimum continuous speed, ensuring the feasibility of real-time tasks without deadline violations and is applicable under both fixed and dynamic priority assignments.

The authors created a resource management framework for cloud computing systems for the first time in [35]. This framework clarifies the resource management process in relation to cloud job scheduling. Next, we examine how the physical resources of a Virtual Machine (VM) affect the consistency of cloud service execution. Next, we created an algorithm called priority-based fair scheduling (PBFS) to schedule jobs in a way that ensures they have access to the necessary resources at the best possible times. Three important parameters—CPU time, arrival time, and work length—have been taken into consideration when developing the algorithm. We have developed a backfilling method called Earliest Gap Shortest Job First (EG-SJF) that emphasizes filling up schedule gaps in a certain sequence for the best scheduling of cloud workloads. A novel method is proposed in [36] to enhance the current Priority Rules (PR) for cloud schedulers by creating a new dynamic scheduling algorithm and adjusting the gaps in the cloud work schedule. First, in order to schedule jobs so that they may access the necessary resources at the best times, the Priority-Based Fair Scheduling (PBFS) algorithm has been devised. Next, a backfilling technique known as Shortest Gap - Priority-Based Fair Scheduling (SG-PBFS) is created in an effort to control the spaces in the cloud tasks' schedule.

The Johnson Sequencing algorithm is used by the authors of [37] to provide a unique approach to work scheduling in cloud computing across three servers. The Johnson Sequencing approach, which was first created for job scheduling in a manufacturing setting, has shown efficaciousness in addressing scheduling difficulties. In this case, we modify this approach to deal with task scheduling amongst three servers in a cloud computing setup. The algorithm's main goal is to reduce the makespan, which is a measure of the overall amount of time needed to do all activities. The authors provide a three-step procedure to implement the Johnson Sequencing method for cloud computing work scheduling. Initially, we create a precedence graph by examining the connections between the various jobs. The precedence graph is then assigned to servers, converting it into a two-machine Johnson Sequencing issue. Lastly, in order to minimize the makespan, we use the Dynamic Heuristic Johnson Sequencing approach to figure out the optimal sequence of jobs on each server. The relevant work to our proposed system is summarized in Table 1. We think the data in this table will enable readers to rapidly pinpoint areas in need of more study, analysis, and development.

III. SYSTEM MODEL AND BACKGROUND

Before explaining the systematic process and background in detail, the schematic symbols used in the research work are given in Table 2.

A. MATHEMATICAL TASK MODEL

A periodic task model is used where a task τ_i is characterized by the following three parameters.

- Period P_i
- Relative deadline D_i
- Worst-case execution time C_i

A task τ_i needs C_i units of central processor shares by D_i. The set $\mathrm{T} = \tau_1, \tau_2, \ldots \tau_n$ comprises all n our tasks or activities. The resources' usage of a single task and total core or handling system use is given by (1) and (2).

$$U_i = \frac{C_i}{P_i} \tag{1}$$

$$U(\delta_i) = \sum_{i=1}^{k} \frac{C_i}{P_i} \tag{2}$$

The core or system number i is represented by δ_i.

The rate-monotonic scheduler assigns high priorities to tasks having smaller periods, i.e., priority $\tau_i \propto \frac{1}{P_i}$.

The first feasibility test for RM scheduling is called LL-bound (Liu and Layland) which states that inherent deadline task proposed by Liu and Layland in [18]. Wherever $d = p$ is possible on core i if and only if (3) is satisfied:

$$n\left(2^{\frac{1}{n}} - 1\right) \geq U(\Delta_i) \tag{3}$$

The hyperbolic bound (HB) test [26] shows that a task is operatable on core i if (4) is satisfied:

$$2 \geq \prod_{h=1}^{n} (U_i(\Delta_i) + 1) \tag{4}$$

When all cores function at low speed, the symmetric performance of the multi-core system is possible [21]. The multi-core systems in edge-computing result in power leakage when they operate at high speed. To avoid the above-mentioned problem and conserve energy, the cores need to operate at the same minimum possible frequency. To formulate our problem, we assume that a Γ is schedulable on a universal core. As indicated by authors in [16], the total responsibility of τ_i any time occurrence t is the total execution time demand of τ_i and the total workload of higher priority tasks than τ_i. Mathematically, it is given by (5):

$$L_i(t) = C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{P_j} \right\rceil \cdot C_j \tag{5}$$

From (5), it can be observed that a task τ_i is possible on a general core δ_i at any time occurrence t if and only if the following (6) is satisfied:

$$t \geq \min_{t \in S_i} L_i(t) \tag{6}$$

In (6), S_i represents scheduling points and t represents a scheduling point and set it in appropriate way such that the criteria in (7) is met:

$$S_i = \{l \cdot P_i \mid j = 1, \ldots, i; l = 1, \ldots, \lfloor P_i/P_j \rfloor\} \tag{7}$$

After signifying the feasibility of a task, the task set Γ suits possible when criteria in (8) is met:

$$1 \geq \max_{i=1,\ldots,k} \left\{ \frac{\min_{t \in S_i} L_i(t)}{t} \right\} \tag{8}$$

The authors in [21] incorporated the speed component in the task's schedulability analysis, which expresses that a task is operatable with speed f_i when it is schedulable at any point of $t \in S_i$. Formally, it can be mathematically written as given in (9):

$$f_i \geq \min_{t \in S_i} \left(\frac{C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{P_j} \right\rceil C_j}{t} \right) \tag{9}$$

In the author's previous work [21], the lowest speed for a task τ_i processing was obtained by testing all values of $t \in S_i$. The lowest possible speed termed as the LeFeS can be obtained by (10).

$$f_i \geq \min \left(\max_{t \in S_i} \frac{C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{P_j} \right\rceil C_j}{t} \right) \tag{10}$$

The mathematical formulations and obtained results reveal that speed gained through $FiFeS \geq LiFeS$ and therefore, power consumption (energy use) spent by LeFeS is in a

TABLE 1. Summary of the related work, closest to GA-LeFeS, with respect to various evaluation criteria.

Parameters	Related Work [31]	[32]	[33]	[34]	[23]	[35]	[36]	[37]	[21]	[19]	GA-FiFeS
Energy	✓	✓	✓	✓					✓	✓	✓
Deadlines	✓		✓	✓				✓	✓		✓
DVFS	✓	✓							✓		✓
Homogeneous	✓								✓		
Heterogeneous		✓			✓	✓				✓	✓
Edge			✓								✓
Priority						✓	✓				✓
Back-filling							✓				
GA			✓								✓

TABLE 2. List of notations and symbols.

Notations	Description
Γ	Tasks set
τ_i	Tasks i
C_i	Worst-case execution time of task i
P_i	Period of task i
D_i	Relative deadline of task i
n	Total number of tasks in task set
U_i	Cumulative utilization of task i
$U(\delta)$	Cumulative core utilization
P	Power
V	Voltage
f	Frequency
T	Time instant
L_i	Cumulative workload of task i
S_i	Scheduling points set
$LeFeS$	Least Feasible Speed
$FiFeS$	First Feasible Speed

smaller amount than or equivalent to the power consumed by FiFeS i.e., $FiFeS_{power} \geq LeFeS_{power}$. In the present research, the FiFeS paradigm is further extended by hybridizing with a genetic algorithm in order to further improve the system speed, which in turn improves the system power. This hybridized technique is known as the GA-FiFeS technique [31], [32].

IV. THE PROPOSED MODEL

In Fig. 1, we elaborate on the working process of the proposed method. The whole process is divided into two sub-processes; the first feasible speed calculation and the optimization using genetic algorithm. The process of the first feasible speed calculation is denoted by the red box. Initially, for a generic task, τ_i using (7) the scheduling points set is calculated. Then task workload on scheduling points is determined by (5). Using (6) the execution possibility of a task τ_i on a core is checked. The system speed by using this t is calculated by (9), if the task τ_i is possible at a point t, which excludes further scheduling points from feasibility analysis amid this speed called FiFeS. If the task τ_i is not viable at the point t, then the next t is tested. Thus, this interaction goes on until the first possible point is reached by reusing a similar cycle. It becomes the initial population for the Genetic algorithm if FiFeS for every task in the task set $\Gamma = \tau_1, \tau_2, \ldots, \tau_n$ is calculated.

The process of GA is shown in Fig. 1 outside the red box. In GA, further processing the values of phenotype must be converted into genotype because the FiFeS values behave in phenotype. As such, for further steps, the FiFeS values are necessary to transform into binary form because its values are in decimal form. In the initial population, all the offsprings' fitness values are calculated. But in (11) the fitness function in our case is given.

$$\min_{f_i} \text{ Such that}$$
$$\min \left(\max_{t \in S_i} \frac{C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{P_j} \right\rceil C_j}{t} \right)$$
$$\leq f_i \quad \text{and } f_i \geq f_{i-1} \tag{11}$$

This condition f_i is the FiFeS obtained by (10) are satisfied because by $\min(f_i)$ we mean minimizing the FiFeS. The next step is the selection of offspring, when the fitness values are calculated then the offspring's selection has many ways such as: (i) tournament selection, (ii) roulette wheel selection, and (iii) random selection [30]; because each of the selections has its pros and cons. Due to the likelihood of the concept of GA [29] "survival of the fittest", the author of [29] uses a tournament selection method. For the cross-over phase, selecting two offsprings from the original population is achieved and then the tournament is applied to randomly selecting four offsprings in the selection phase. As our problem is a minimization problem, that is why we select the offspring having minimum speed in the Tournament Selection Process (TSP). The whole TSP mechanism is pictorially presented in Fig. 2.

A. KEY DRIVERS OF THE GENETIC ALGORITHM

The key drivers of the GA algorithm are mutation and cross-over. The Mutation process happens in a single offspring, while for the cross-over process, at least two offsprings are mandatory.

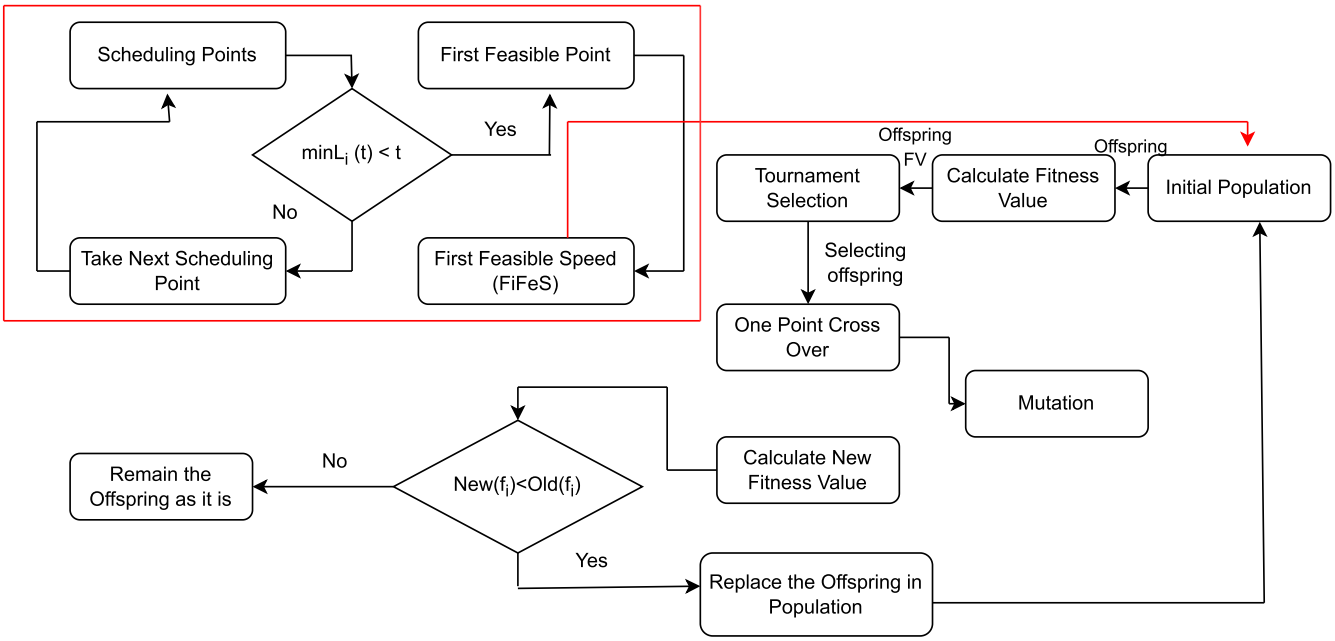

FIGURE 1. The workflow of the proposed model.

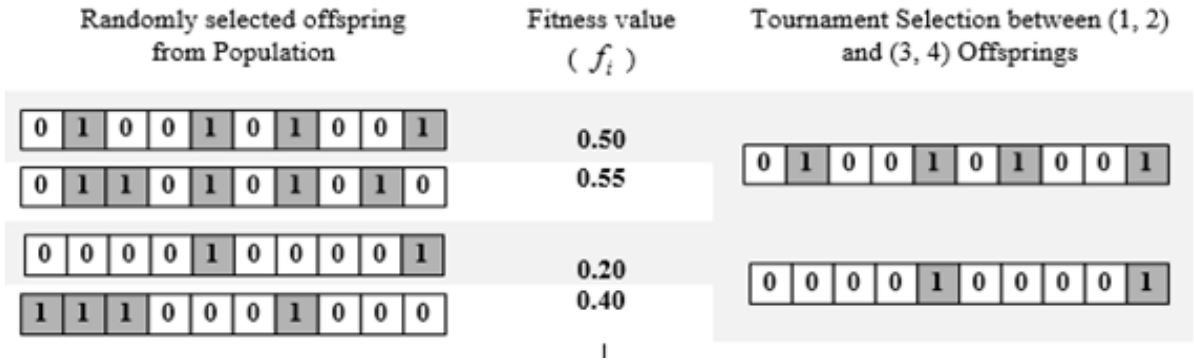

FIGURE 2. The Tournament selection process.

1) THE CROSS-OVER PROCESS

The cross-over is the process of reproduction where parents meet to produce children/offspring. The offspring get some characteristics of their own but also inherit some features from their parents. Between two offsprings, both at single or multiple points, the cross-over occurred. These points may vary or be fixed, but the authors used a single-point cross-over while this point is not fixed. The following Fig. 3 (a) and Fig. 3 (b) show the cross-over process.

2) THE MUTATION PROCESS

Traits are transmitted from parents to offspring through cross-over, and mutation is liable for the offspring's specific qualities. A minor change in the structure of children's genes is called Mutation. After crossover, the mutation occurs in a single offspring and also can be single-bit or bitwise. But usually, the probability of mutation is retained very low, which is why mutation does not occur in offspring occasionally. In the objective function, there are high hazards of convergence, if the probability is high of the mutation. To trap out from local minima, the mutation is utilized. But in our case, the probability of mutation is 0.1 because the authors used single-point mutation. First authors check the possibility that mutation will occur at the point or not, authors arbitrarily select a mutation point and the bit altered consequently, if it fulfills the mutation probability. In any case, leaves the offspring, all things considered, on the off chance that it is absurd. Fig. 4 shows the process of mutation.

In the single-point-mutation process, we select randomly a point and checked only once the probability of mutation occurrence. In bitwise-mutation, the chance is tested at each piece to transform the piece or hold it, all things considered. Using (11) the fitness values are calculated for the fresh offspring next to the mutation and cross-over processes. If the offspring's old fitness value is greater than the new fitness value of an offspring, then all old offsprings are replaced with the new offsprings as shown in (12).

In the single-point-mutation process, we select randomly a point and check only once the probability of mutation

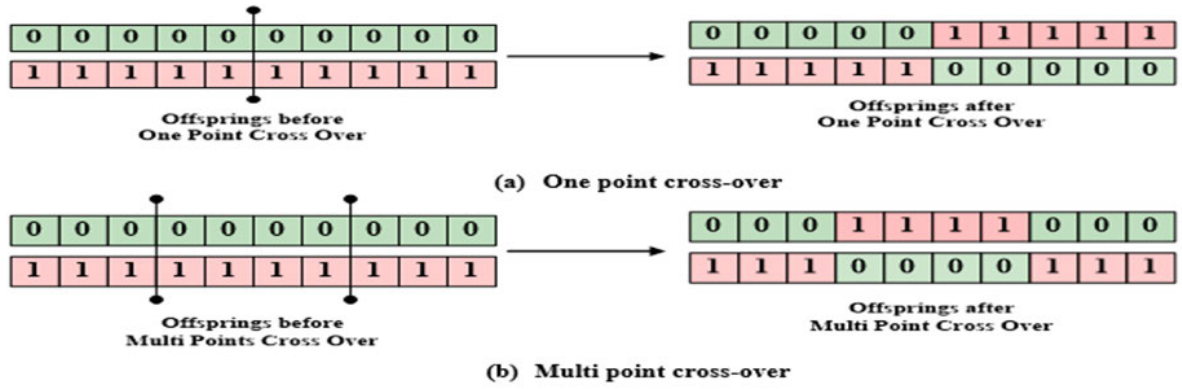

FIGURE 3. The process of cross-over.

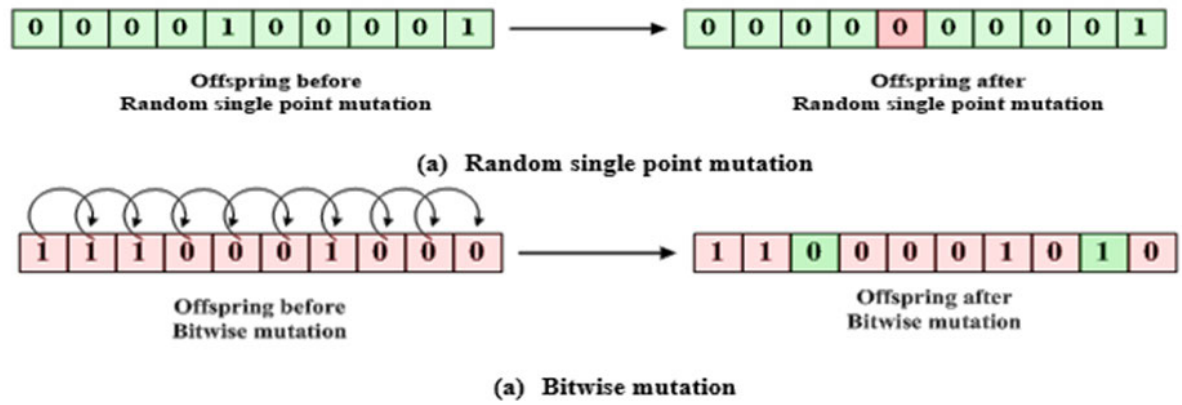

FIGURE 4. The mutation process.

occurrence. In the bitwise-mutation, the chance is tested at each piece to transform the piece or hold it, all things are considered. Using (11) the fitness values are calculated for the fresh offspring next to the mutation and cross-over processes. If the offspring's old fitness value is greater than the new fitness value of an offspring, then all old offsprings are replaced with the new offsprings as shown in (12).

$$\text{Old}(f_i) > \text{New}(f_i) \tag{12}$$

For further iterations, in the population the values are retained if the offspring whose new fitness value is not less than its old fitness.

By summing up the above conversation, GA-FiFeS works on the acquired f_i through FiFeS by utilizing the primary drivers of GA, with the end goal that the new speed is not more than that got through LeFeS. Thus, we can undoubtedly get f_i that $FiFeS \geq GA - FiFeS \geq LeFeS$. Various steps in the proposed process are described in Alg. 1.

B. FEASIBILITY CHECKING THROUGH GA-FiFeS APPROACH

A task τ_i execution is possible using the FiFeS technique if and only if, By rearranging (9) and (10), we get (13):

$$t \geq \min_{t \in S_i} \left(\frac{C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{P_j} \right\rceil C_j}{f_i} \right) \tag{13}$$

And using LeFeS if, and only if, a task τ_i is feasible if (14) holds:

$$t \geq \min \left(\max_{t \in S_i} \frac{C_i + \sum_{j=1}^{i-1} \left\lceil \frac{t}{P_j} \right\rceil C_j}{f_i} \right) \tag{14}$$

Like $FiFeS \geq GA - FiFeS \geq LeFeS$, as we determine that the f_i acquired through LeFeS, FiFeS and GA-FiFeS and the required execution time increases if the f_i value decreases. If a task execution is possible at FiFeS, it may also be possible at LeFeS as presented in [21]. We can determine that $LeFeS_{time} \geq GA - FiFeS_{time} \geq FiFeS_{time}$ from the above discussion. Therefore, it might be possible to execute a task at GA-FiFeS, if a task is viable at LeFeS. Fig. 5 elaborates on the flow of the GA-FiFeS technique.

C. COMPUTATIONAL COMPLEXITY

As discussed in [2], the time complexity of FiFeS is $mlog(n)$ where m shows the amount of time taken in calculating the number of scheduling points and $log(n)$ is the amount of time taken to find the first feasible point i.e. to calculate the speed at that particular point. The time complexity of LeFeS is $O(mn)$, where m shows the amount of time taken in calculating the number of scheduling points, and n is the amount of time taken in calculating the speed at every feasible point, and finally selecting minimum speed among the calculated speeds. The time complexity of GA-FiFeS is

Algorithm 1 The GA-FiFeS Algorithm

Input: GA-FiFeS values
Output: Set of $min(f_i)$ values
Calculate individual f_i of all the offsprings in the initial population (FiFeS) ;
for *epoch:1 to total number of epochs* **do**
  Step 1: Tournament selection ;
  Select randomly Off_i where i_1^4 and arrange a tournament between Off (1, 2) and (3, 4). ;
  Select those Off as parent whose $f_i >$ opponent i.e. Handle ties arbitrary ;
  Step 2: Cross over ;
  Select p where $1 \leq p \leq size(Off)$;
  Cross over the two parents Off at p. Step 3: Mutation ;
  Select p where $1 \leq p \leq size(Off)$;
  if $prob_{mut}$ *lies in the range of* $threshold_{mut}$ **then**
    mutate the bit at p ;
  else
    retain the bit as it is ;
  end
  Step 4: Calculate f_i of the two new Off ;
  if $New(f_i) < Off(f_i)$ *i.e. Handle ties arbitrary* **then**
    Replace old Off with new Off ;
  else
    Retain the old Off as it was (before crossover and mutation) ;
  end
  epoch = epoch +1 ;
end
return

$O(mlog(n) + k)$, where $mlog(n)$ is the amount of time taken by the FiFeS and k is the amount of time taken by the genetic algorithm portion of the GA-FiFeS algorithm. The amount of k depends on the number of iterations (epochs/population) in the genetic algorithm. If the value of k is high enough to the amount of n then the GA-FiFeS will take more time than LeFeS and if the value of k is small then GA-FiFeS will take less time than the LeFeS technique. However, existing counterparts for the FiFeS beat our both proposed approaches if time is the comparison parameter [38], [39].

As we know that $f = 1/t$ which is of concern i.e. we are interested in lowering the speed (frequency) of the CPU and, hence, power consumption (energy); therefore, the relationship of GA and LeFeS methods may become like $f = LeFeS \in O(g(FiFeS))$. This means that if the speed of the CPU is taken as a comparison parameter then the FiFeS approach is the upper bound for the LeFeS method i.e. $0 \leq f(LeFeS) \leq C(g(FiFeS))$ and if time is the comparison parameter then all these terms occurs in a reverse order. However, if speed is the comparison parameter then the above relationship can also be written as $f(FiFeS) \in \omega(g(LeFeS))$. This means that the LeFeS method is the lower bound for the FiFeS technique i.e. $0 \leq C(g(LeFeS)) \leq f(FiFeS)$.

V. RESULTS AND ANALYSIS

A. EXPERIMENTAL SETUP

For experimental results and analysis, the three techniques FiFeS, GA-FiFeS, and LeFeS were compared using numerical simulations in the MATLAB software. With a step size of 1, random task sets with sizes between [5, 50] were created in order to evaluate and compare the three methods. The average values of 300 runs for each of the task sets 5 through 50 are evaluated and visually plotted in the following sections. The task durations were chosen at random from a range of [100, 10,000] that was evenly distributed. In order to derive the relevant task execution needs C_i for τ_1, uniformly distributed random values were selected from the range $[1, P_i]$. The tasks were prioritized in accordance with the Rate Monotonic (RM) scheduling guidelines. In other words, the job priority increases with a shorter task period and vice versa. We first maintain the system utilization at 0.69 or $ln(2)$, which is rather low, in order to have a viable RM schedulable task set. Moreover, the CPU runs at various speeds, i.e. frequencies, where a lower speed has a lower energy consumption and vice versa. We assume that a processor has 10 major operational levels as detailed in Table 3.

B. EVALUATION METRICS

We use three evaluation metrics to compare the results of the proposed algorithm with the closest rivals: (i) speed; (ii) energy consumption; and (iii) execution time. Note that speed is related to the CPU frequency (measured in Hz) that denotes the CPU speed for a particular task set. We assume that the CPU operates at multiple speeds i.e. dynamic voltage and frequency scaling (DVFS). Energy consumption is measured in Watts hours (Wh) and denotes the amount of

TABLE 3. Operational levels and the respective speed ranges.

Level(i)	f_i	δf_i (respective subranges/minor-levels for speed f_i)
0	0.1	0.01, 0.02, ..., 0.09, 0.10
1	0.2	0.11, 0.12, ..., 0.19, 0.20
2	0.3	0.21, 0.22, ..., 0.29, 0.30
3	0.4	0.31, 0.32, ..., 0.39, 0.40
4	0.5	0.41, 0.42, ..., 0.49, 0.50
5	0.6	0.51, 0.52, ..., 0.59, 0.60
6	0.7	0.61, 0.62, ..., 0.69, 0.70
7	0.8	0.71, 0.72, ..., 0.79, 0.80
8	0.9	0.81, 0.82, ..., 0.89, 0.90
9	1.0	0.91, 0.92, ..., 0.99, 1.00

TABLE 4. Experimental results in terms of task set sizes, required speeds, normalized energy consumption, and required execution times for various techniques.

Task Set Size	5	10	20	30	40	50
Required Speed						
FiFeS	0.904982619	0.972157773	0.992753623	0.997436993	0.999608611	1
LeFeS	0.511280551	0.553269655	0.579440521	0.594227956	0.605409814	0.628962088
GA-FiFeS	0.5625	0.59375	0.609375	0.62109375	0.627929688	0.78515625
Normalized Energy Consumption						
FiFeS	0.840278158	1	1	1	1	1
LeFeS	0.296865217	0.375918616	0.406418616	0.438941335	0.468941335	0.495941335
GA-FiFeS	0.375976563	0.41015625	0.5	0.50390625	0.517578125	0.765625
Execution Time Required						
FiFeS	1	1	1.001602564	1.003902439	1.017283951	1.104993598
LeFeS	1.589920949	1.634133965	1.663017815	1.701449964	1.760596091	1.95587334
GA-FiFeS	1.273631841	1.406593407	1.6	1.620253165	1.662337662	1.777777778

energy consumed by all CPUs at the end of each experiment. Note that energy consumption is directly proportional to the speed of the CPU. Finally, execution time is measured in seconds (s) and denotes the total CPU time for all tasks in a particular task set.

C. RESULTS DISCUSSION

The empirical results and evaluation of FiFeS, LeFeS, and GA-FiFeS are considered in this section. The obtained results for various techniques, in terms of task set sizes, required speeds, normalized energy consumption, and required execution times are shown in Table. 4.

Under the FiFeS, LeFeS, and GA-FiFeS procedures, we plot the speeds essential for task sets viability shown in Fig. 5. which show, obviously, that FiFeS is fewer high-speed runs than GA-FiFeS, and that is why FiFeS performs better than GA-FiFeS. Fig. 5 (a) is based on 150 and Fig. 5 (b) is on 400 epochs for GA-FiFeS. The LeFeS is the optimal solution so with an increase in the number of epochs (new populations) the GA-FiFeS will behave like LeFeS while as the number of epochs decreases then the GA-FiFeS approach behaves like the FiFeS technique.

Following are some of the inferences, justifications, and mathematical perspectives that are discussed based on our empirical evaluation, the results obtained, and their analysis:

- When speed is used as a testing attribute, then GA-FiFeS achieves better than FiFeS. From a Justification and mathematical perspective, for speed calculation, FiFeS uses (9), and the result of this is the input for the GA-FiFeS. We also apply mutation and cross-over operations due to the optimization properties of GA-FiFeS that improve the outcomes obtained by FiFeS due to the objective or fitness function of the GA-FiFeS which is min(f_i). At the point when the speed is thought about, we noted that the GA-FiFeS method outperforms the FiFeS technique.
- If speed is the testing criterion, the supremacy of LeFeS over FiFeS and GA-FiFeS is shown in Fig. 5. From the obtained results, anyone can easily conclude that the LeFeS algorithm runs with less feasible speed than GA-FiFeS and FiFeS. We observed from justification and mathematical perspective; that when speed is under consideration, LeFeS is efficient [21]. Hence, as clear from GA-FiFeS constraints, we set a restriction that the speed got through GA-FiFeS should not be exactly the speed acquired by the LeFeS technique. This is evident from the GA-FiFeS constraints given by (15):

$$\min\left(\max_{t\in S_i}\frac{C_i+\sum_{j=1}^{i-1}\left\lceil\frac{t}{P_j}\right\rceil C_j}{t}\right)\leq f_i \qquad (15)$$

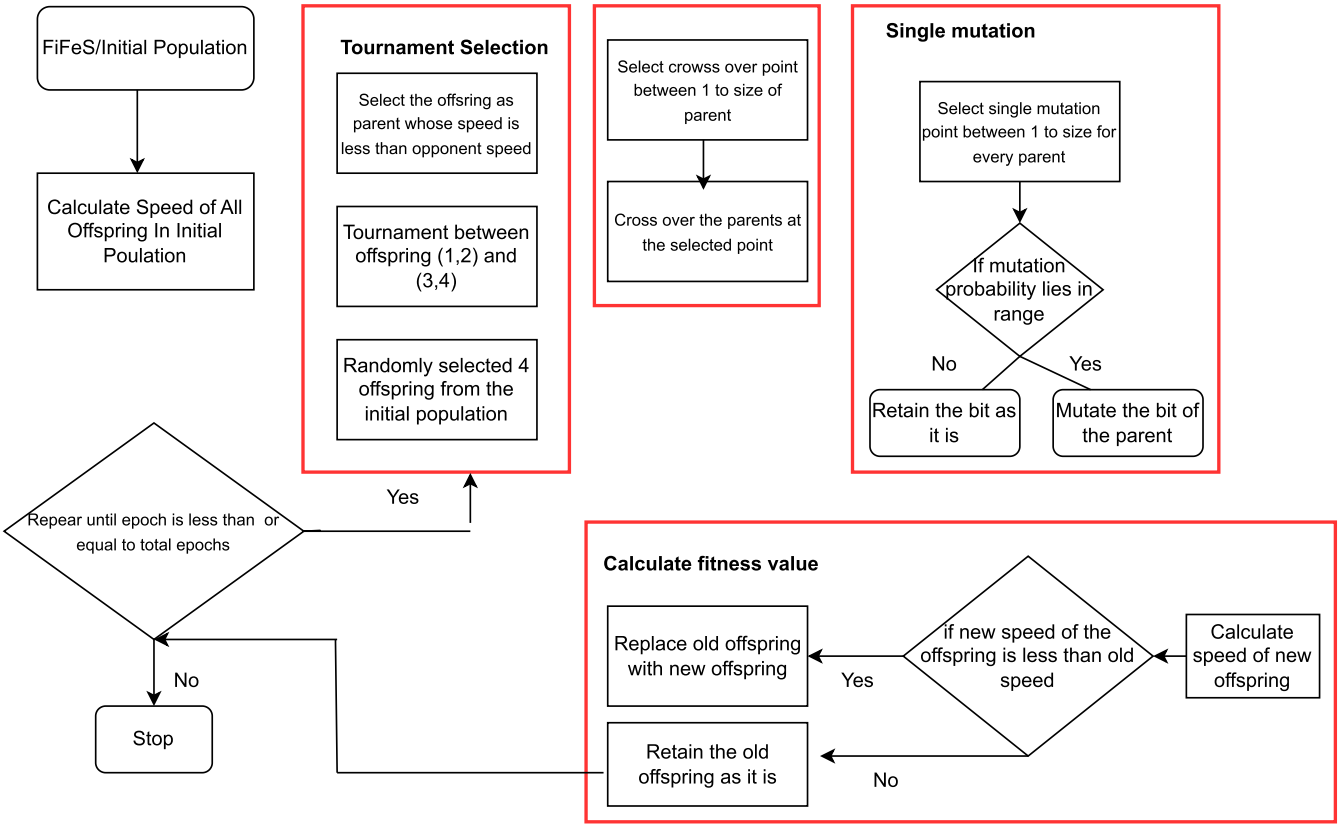

FIGURE 5. Flow chart of the Genetic Algorithm based First Feasiable Speed (GA-FiFeS) Technique.

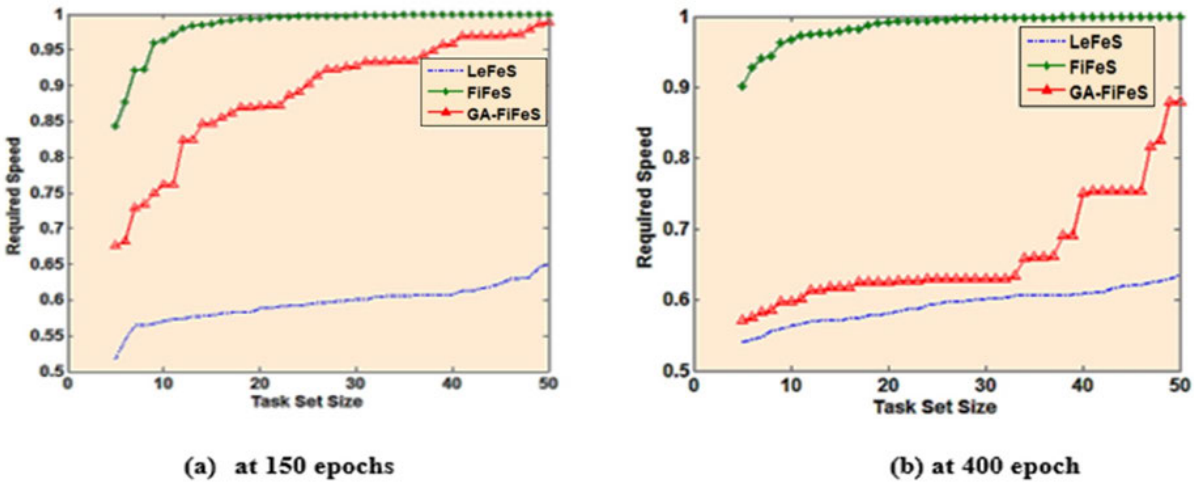

FIGURE 6. The required speeds for LeFeS, FiFeS, and GA-FiFeS approaches [lower graphs denote minimum CPU speed for tasks execution] – The GA-FiFes method performs better than FiFes and for higher epoch GA-FiFes is comparable to the LeFes approach.

The energy usage of GA-FiFeS, LeFeS, and FiFeS algorithms are shown in Fig. 6, respectively. Fig. 6 (a) used 150 and Fig. 6 (b) used 400 epochs for the GA-FiFeS technique.

- The GA-FiFeS does not have as much power (energy) as compared to the FiFeS method. The mathematical viewpoint and justification is, that it is evident from the first inference that the speed got through GA-FiFeS is not exactly as of FiFeS. As we know that $P = V^2 \times f$ and $E = P \times T$ and the power (energy) values for GA-FiFeS will outperform FiFeS values, as we put the speed values acquired by FiFeS and GA-FiFeS in the previous relations.
- The LeFeS uses less energy or power than GA-FiFeS and FiFeS, the mathematical perspective and justification are, as clear from the justification of previous

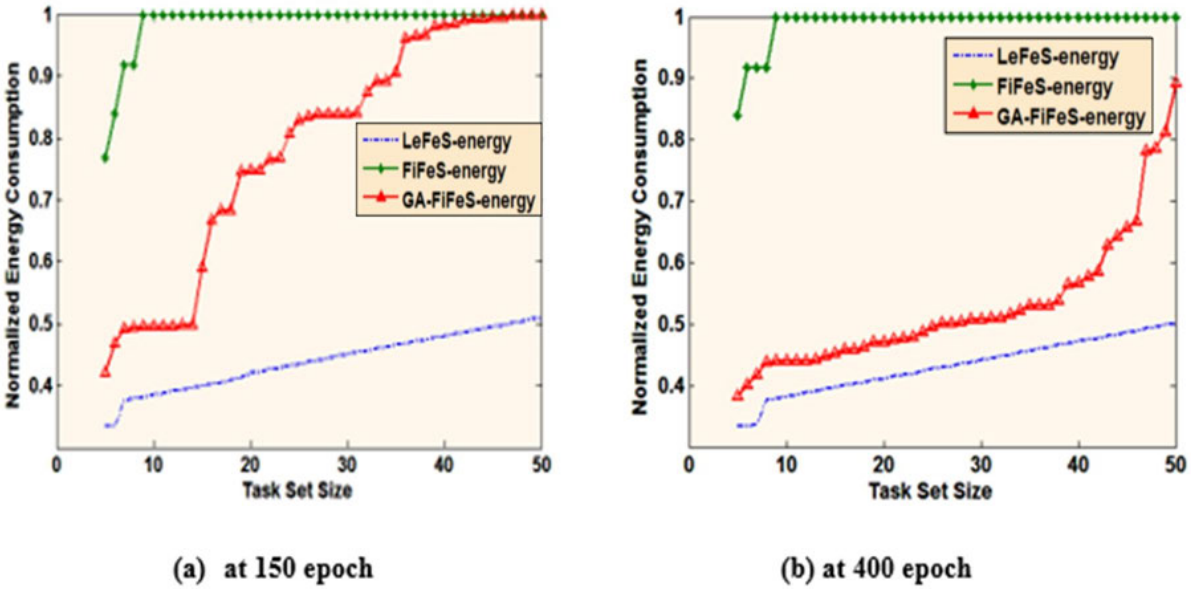

FIGURE 7. The Normalized energy usage of LeFeS, FiFeS, and GA-FiFeS approaches [lower graphs denote minimum energy consumption for tasks execution] – The GA-FiFes method performs better than the FiFes approach and for higher epoch GA-FiFes is comparable to the LeFes approach.

implications 2 and 3. The essential execution times of GA-FiFeS, LeFeS, and FiFeS approaches are shown in Fig. 7 and it is clear that when desirable execution-time is the analysis factor, GA-FiFeS defeats LeFeS. The results based on 150 epochs and 400 epochs are shown in Fig. 7 (a) and Fig. 7 (b) for GA-FiFeS, respectively.

- In the required execution time, the proposed GA-FiFeS defeats the LeFeS approach. In the justification, for the detection of the least feasible speed, the LeFeS method checks all scheduling points of every task. Therefore, it takes extra execution time. Whereas, GA-FiFeS takes the first feasible point, for each task, and subsequently uses GA for optimization. The GA-FiFeS takes a negligible portion of performance time. Thus, GA-FiFeS overthrows LeFeS in the viewpoint of expected execution time. The execution times are shown in Fig. 8.
- In required execution (response) time, FiFeS defeats both the GA-FiFeS and LeFeS approaches. In its justification, as clear from previous inferences, in terms of execution time GA-FiFeS defeats the LeFeS approach. The result of FiFeS turns into input for GA-FiFeS. Consequently, in the required execution time, the FiFeS defeats LeFeS and GA-FiFeS approaches.
- If GA-FiFeS is used, a task execution is possible using GA-FiFeS if a task finishes within its assigned deadline. The mathematical perspective and justification is, as inferred from inference 5, the necessary execution time of GA-FiFeS is not as much as reported for the LeFeS technique i.e., $LeFeS_{time} \geq GA - FiFeS_{time}$. Furthermore, the outcomes in [19] show that a task is feasible at FiFeS, it might likewise be possible at the LeFeS. From this, we can determine that $LeFeS_{time} \geq GA - FiFeS_{time} \geq FiFeS_{time}$.
- Survival of the fittest proof through GA is as follows. For the Justification, the concept of Darwin's philosophy and hence of GA is "Survival of the fittest" [29]. It is clear from the results that when the amount of epochs is less, then the GA-FiFeS works like FiFeS as shown in Fig. 5, Fig. 6, and Fig. 7. But the fittest values are attained when the number of epochs increases, then the GA-FiFeS acts like the LeFeS method as the number of iterations increases.
- The last inference is, through the GA-FiFeS, the obtained f_i values are non-decreasing. For this justification, the second constraint of the fitness function is referred to where it is clear, that $f_i \geq f_(i-1)$. Therefore, through the GA-FiFeS technique, the f_i values are non-decreasing.

D. CRITICAL ANALYSIS AND SCALABILITY

With the size increase of the task set, as evident from the experiments, the proposed solution has shown no significant performance improvements over the FiFeS algorithm. There are two possible reasons for this situation: (i) existing tradeoff between task parameters; and (ii) resource utilization [9].

In respect to (i), the characteristics of the tasks themselves might be influencing the performance comparison due to the fact that certain types of tasks may favor one algorithm over another, depending on factors like task dependencies, resource requirements, or deadlines [28]. In respect of (ii), a solution might perform similarly in terms of raw execution time but be more resource-efficient. Therefore, further assessment is needed whether your proposed solution effectively utilizes available resources (e.g., CPU, memory) compared to the FiFeS technique [19].

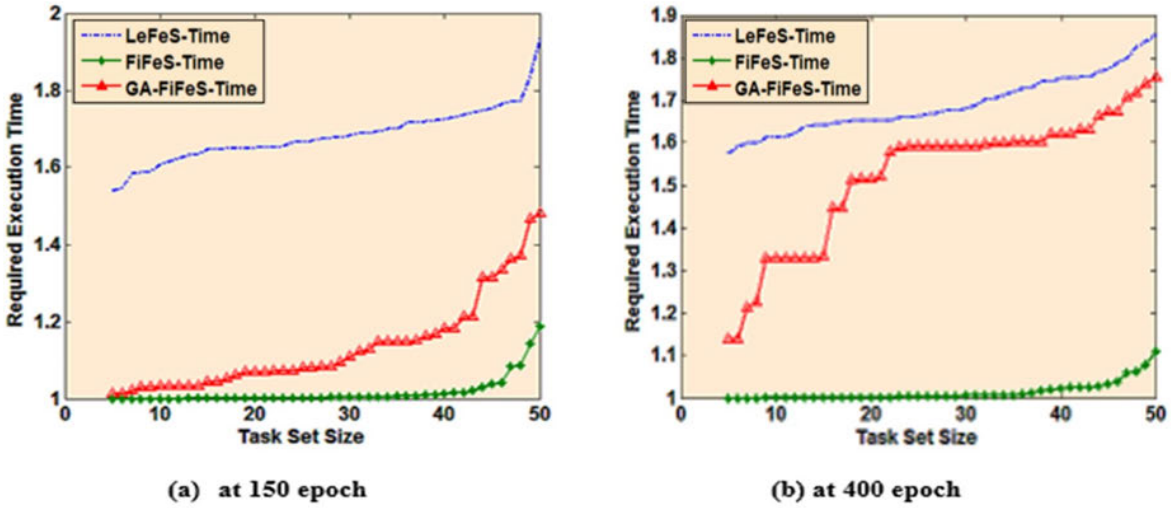

FIGURE 8. Execution times for FiFeS, GA-FiFeS, and LeFeS approaches [lower graphs denote minimum execution times for tasks] – The GA-FiFes method performs better than the FiFes approach and for higher epoch GA-FiFes is comparable to LeFes approach.

Moreover, the complexity of the GA algorithm can also affect the overall performance of the proposed GA-FiFeS method. In case both algorithms i.e. GA-FiFeS and FiFeS have similar time complexities, then it is expected that their performance would be comparable, regardless of the size of the task set.

The scalability of a task scheduling algorithm for real-world large-scale scenarios is essential for ensuring that it can effectively handle increasing workloads, task volumes, and system demands [40], [41]. This should be noted that GA-FiFeS incorporates load-balancing mechanisms, which can lead to balanced resource utilization in large-scale scenarios. Ensuring efficient load balancing becomes increasingly important as the number of tasks and resources grows to prevent the overloading of individual resources and maximize overall system throughput. Moreover, the proposed algorithm can easily be modified with fault-tolerance approaches.

We studied the scalability of the proposed method under various workload conditions, task set sizes, task resource utilization, number of processors or cores, and heterogeneity of resources. We observed an existing tradeoff among various performance evaluation metrics. Moreover, the FiFeS is known for its simplicity and efficiency, often with a time complexity that is linear or close to linear with respect to the number of tasks and resources [42], [43]. We believe, that the GA method can make the complexity of GA-FiFeS a bit high but still, it is very close to linear. Therefore, linear scaling makes it well-suited for handling large-scale task scheduling scenarios efficiently.

VI. CONCLUSION AND FUTURE WORK

In this research, we focused on energy and performance-efficient scheduling of real-time tasks on high-performance edge computing systems while ensuring that they meet their deadlines. By using the concepts of GA, we enhanced the power and speed viewpoints of the classical FiFeS approach, and this improved variation of the FiFeS algorithm is named as GA-FiFeS technique. We observed that the GA-FiFeS method is more effective in power and speed consumption than the FiFeS approach which is proved through several plausible assumptions and experimental results. The GA-FiFeS algorithm has improved results as compared to the LeFeS method while taking the expected time of execution as an assessment feature for analysis.

For future directions, we suggest that the analysis of the performance of a specific task time of execution C_i through alteration in speed f_i will give reasonable results. Furthermore, for a distinct task, the correlation between C_i and f_i is $C_i \propto \frac{1}{f_i}$ and the value of C_i decreases consequently if there is an increase in f_i. In addition, this is for a singular task, but if there are multiple dependent and independent tasks then what will be the impact? Another alternative for future direction is to balance the load among the processing units or cores in edge computing if there are dependent tasks and what will be the impact on individual execution unit speed when tasks are independent. we will also investigate the scalability of our approach to large-scale cloud computing servers while accounting for the heterogeneity of resources and workloads.

ACKNOWLEDGMENT

This research has received support from the Department of Computer Science, Abdul Wali Khan University, Mardan (AWKUM), Pakistan; the University of Buner, Khyber Pakhtun Khwa, Pakistan; and Kardan University, Kabul, Afghanistan. The study was carried out within the framework of the Ph.D. program.

REFERENCES

[1] H. Hussain et al., "A survey on resource allocation in high performance distributed computing systems," *Parallel Comput.*, vol. 39, no. 11, pp. 709–736, 2013.

[2] H. Hussain, M. Shoaib, M. B. Qureshi, and S. Shah, "Load balancing through task shifting and task splitting strategies in multi-core environment," in *Proc. 8th Int. Conf. Digit. Inf. Manage. (ICDIM)*, Sep. 2013, pp. 385–390.

[3] A. Burmyakov, E. Bini, and C.-G. Lee, "Towards a tractable exact test for global multiprocessor fixed priority scheduling," *IEEE Trans. Comput.*, vol. 71, no. 11, pp. 2955–2967, Nov. 2022.

[4] W. Shu, K. Cai, and N. N. Xiong, "Research on strong agile response task scheduling optimization enhancement with optimal resource usage in green cloud computing," *Future Gener. Comput. Syst.*, vol. 124, pp. 12–20, Nov. 2021.

[5] G. L. Valentini, W. Lassonde, S. U. Khan, N. Min-Allah, S. A. Madani, J. Li, L. Zhang, L. Wang, N. Ghani, J. Kolodziej, H. Li, A. Y. Zomaya, C.-Z. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, and P. Bouvry, "An overview of energy efficiency techniques in cluster computing systems," *Cluster Comput.*, vol. 16, no. 1, pp. 3–15, Mar. 2013.

[6] S. Bharany, S. Badotra, S. Sharma, S. Rani, M. Alazab, R. H. Jhaveri, and T. R. Gadekallu, "Energy efficient fault tolerance techniques in green cloud computing: A systematic survey and taxonomy," *Sustain. Energy Technol. Assessments*, vol. 53, Oct. 2022, Art. no. 102613.

[7] F. Pinel, J. E. Pecero, S. U. Khan, and P. Bouvry, "Energy-efficient scheduling on milliclusters with performance constraints," in *Proc. IEEE/ACM Int. Conf. Green Comput. Commun.*, Aug. 2011, pp. 44–49.

[8] L. Wang, S. U. Khan, D. Chen, J. Kołodziej, R. Ranjan, C.-Z. Xu, and A. Zomaya, "Energy-aware parallel task scheduling in a cluster," *Future Gener. Comput. Syst.*, vol. 29, no. 7, pp. 1661–1670, Sep. 2013.

[9] A. Chhabra, G. Singh, and K. S. Kahlon, "Performance-aware energy-efficient parallel job scheduling in HPC grid using nature-inspired hybrid meta-heuristics," *J. Ambient Intell. Humanized Comput.*, vol. 12, no. 2, pp. 1801–1835, Feb. 2021.

[10] S. Ullah Khan, "A goal programming approach for the joint optimization of energy consumption and response time in computational grids," in *Proc. IEEE 28th Int. Perform. Comput. Commun. Conf.*, Dec. 2009, pp. 410–417.

[11] D. Chen, L. Wang, X. Wu, J. Chen, S. U. Khan, J. Kołodziej, M. Tian, F. Huang, and W. Liu, "Hybrid modelling and simulation of huge crowd over a hierarchical grid architecture," *Future Gener. Comput. Syst.*, vol. 29, no. 5, pp. 1309–1317, Jul. 2013.

[12] C. Wen, J. Yang, L. Gan, and Y. Pan, "Big data driven Internet of Things for credit evaluation and early warning in finance," *Future Gener. Comput. Syst.*, vol. 124, pp. 295–307, Nov. 2021.

[13] D. Kanakadhurga and N. Prabaharan, "Demand side management in microgrid: A critical review of key issues and recent trends," *Renew. Sustain. Energy Rev.*, vol. 156, Mar. 2022, Art. no. 111915.

[14] C.-G. Wu, W. Li, L. Wang, and A. Y. Zomaya, "An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing," *Future Gener. Comput. Syst.*, vol. 117, pp. 498–509, Apr. 2021.

[15] O. M. AlMendah and S. M. Alzahrani, "Cloud and edge computing security challenges, demands, known threats, and vulnerabilities," *Academic J. Res. Sci. Publishing*, vol. 2, no. 21, pp. 156–175, 2021.

[16] J. P. Lehoczky, "Fixed priority scheduling of periodic task sets with arbitrary deadlines," in *Proc. 11th Real-Time Syst. Symp.*, Dec. 1990, pp. 201–209.

[17] M. B. Qureshi, M. M. Dehnavi, N. Min-Allah, M. S. Qureshi, H. Hussain, I. Rentifis, N. Tziritas, T. Loukopoulos, S. U. Khan, C.-Z. Xu, and A. Y. Zomaya, "Survey on grid resource allocation mechanisms," *J. Grid Comput.*, vol. 12, no. 2, pp. 399–441, Jun. 2014.

[18] C. L. Liu and J. W. Layland, "Scheduling algorithms for multiprogramming in a hard-real-time environment," *J. ACM*, vol. 20, no. 1, pp. 46–61, Jan. 1973.

[19] H. Hussain, "Power efficient resource allocation in high performance computing systems," Ph.D. dissertation, Dept. Comput. Sci., COMSATS Inst. Inf. Technol., Islamabad, Pakistan, 2016.

[20] Y. Chang, C. Gu, and F. Luo, "Energy efficient virtual machine consolidation in cloud datacenters," in *Proc. 4th Int. Conf. Syst. Informat. (ICSAI)*, Weihai, China, Nov. 2017, pp. 61–72.

[21] N. Min-Allah, H. Hussain, S. U. Khan, and A. Y. Zomaya, "Power efficient rate monotonic scheduling for multi-core systems," *J. Parallel Distrib. Comput.*, vol. 72, no. 1, pp. 48–57, Jan. 2012.

[22] X. Huang, K. Li, and R. Li, "A energy efficient scheduling base on dynamic voltage and frequency scaling for multi-core embedded real-time system," in *Algorithms and Architectures for Parallel Processing: 9th International Conference, ICA3PP 2009, Taipei, Taiwan, June 8–11, 2009. Proceedings 9*. Berlin, Germany: Springer, 2009, doi: 10.1007/978-3-642-03095-6.

[23] E. Bini, G. Buttazzo, and G. Lipari, "Minimizing CPU energy in real-time systems with discrete speed management," *ACM Trans. Embedded Comput. Syst.*, vol. 8, no. 4, pp. 1–23, Jul. 2009.

[24] D. Ramegowda and M. Lin, "Energy efficient mixed task handling on real-time embedded systems using FreeRTOS," *J. Syst. Archit.*, vol. 131, Oct. 2022, Art. no. 102708.

[25] B. Khodabandeloo, A. Khonsari, P. Behnam, A. Majidi, and M. H. Hajiesmaili, "Stereo: Assignment and scheduling in MPSoC under process variation by combining stochastic and decomposition approaches," *IEEE Trans. Comput.*, vol. 71, no. 11, pp. 2940–2954, Nov. 2022.

[26] J. Huang, R. Li, J. An, H. Zeng, and W. Chang, "A DVFS-weakly dependent energy-efficient scheduling approach for deadline-constrained parallel applications on heterogeneous systems," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 12, pp. 2481–2494, Dec. 2021.

[27] S. Jadon and R. S. Yadav, "Deadline-constrained tasks' scheduling in multi-core systems using harmonic-aware load balancing," *Arabian J. Sci. Eng.*, vol. 46, no. 4, pp. 3099–3113, Apr. 2021.

[28] K. S. Garud, S. Jayaraj, and M. Lee, "A review on modeling of solar photovoltaic systems using artificial neural networks, fuzzy logic, genetic algorithm and hybrid models," *Int. J. Energy Res.*, vol. 45, no. 1, pp. 6–35, Jan. 2021.

[29] R. Smith, "Darwin and his critics. The reception of Darwin's theory of evolution by the scientific community," *Brit. J. Hist. Sci.*, vol. 7, no. 3, pp. 278–285, Nov. 1974.

[30] E. K. Burke et al., *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques*. New York, NY, USA: Springer, 2014, doi: 10.1007/978-1-4614-6940-7.

[31] S. Moulik, Z. Das, R. Devaraj, and S. Chakraborty, "SEAMERS: A semi-partitioned energy-aware scheduler for heterogeneous MulticorE real-time systems," *J. Syst. Archit.*, vol. 114, Mar. 2021, Art. no. 101953.

[32] S. Moulik, R. Devaraj, and A. Sarkar, "HEALERS: A heterogeneous energy-aware low-overhead real-time scheduler," *IET Comput. Digit. Techn.*, vol. 13, no. 6, pp. 470–480, Nov. 2019.

[33] Q. Jiang, X. Xin, L. Yao, and B. Chen, "METSM: Multiobjective energy-efficient task scheduling model for an edge heterogeneous multiprocessor system," *Future Gener. Comput. Syst.*, vol. 152, pp. 207–223, Mar. 2024.

[34] A. Khiat, M. Haddadi, and N. Bahnes, "Genetic-based algorithm for task scheduling in fog–cloud environment," *J. Netw. Syst. Manage.*, vol. 32, no. 1, p. 3, 2024.

[35] S. A. Murad, Z. R. M. Azmi, A. J. M. Muzahid, M. M. H. Sarker, M. S. U. Miah, M. K. B. Bhuiyan, N. Rahimi, and A. K. Bairagi, "Priority based job scheduling technique that utilizes gaps to increase the efficiency of job distribution in cloud computing," *Sustain. Comput., Informat. Syst.*, vol. 41, Jan. 2024, Art. no. 100942.

[36] S. A. Murad, Z. R. M. Azmi, A. J. M. Muzahid, M. K. B. Bhuiyan, M. Saib, N. Rahimi, N. J. Prottasha, and A. K. Bairagi, "SG-PBFS: Shortest gap-priority based fair scheduling technique for job scheduling in cloud environment," *Future Gener. Comput. Syst.*, vol. 150, pp. 232–242, Jan. 2024.

[37] P. Banerjee, S. Roy, A. Sinha, M. M. Hassan, S. Burje, A. Agrawal, A. K. Bairagi, S. Alshathri, and W. El-Shafai, "MTD-DHJS: Makespan-optimized task scheduling algorithm for cloud computing with dynamic computational time prediction," *IEEE Access*, vol. 11, pp. 105578–105618, 2023.

[38] M. Zakarya, A. A. Khan, M. R. C. Qazani, H. Ali, M. Al-Bahri, A. U. R. Khan, A. Ali, and R. Khan, "Sustainable computing across datacenters: A review of enabling models and techniques," *Comput. Sci. Rev.*, vol. 52, May 2024, Art. no. 100620.

[39] S. Wang, Z. Ding, and C. Jiang, "Elastic scheduling for microservice applications in clouds," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 1, pp. 98–115, Jan. 2021.

[40] M. Zakarya, L. Gillam, K. Salah, O. Rana, S. Tirunagari, and R. Buyya, "CoLocateMe: Aggregation-based, energy, performance and cost aware VM placement and consolidation in heterogeneous IaaS clouds," *IEEE Trans. Services Comput.*, vol. 16, no. 2, pp. 1023–1038, Mar. 2023.

[41] P. Hosseinioun, M. Kheirabadi, S. R. Kamel 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.

[42] P. Pirozmand, A. A. R. Hosseinabadi, M. Farrokhzad, M. Sadeghilalimi, S. Mirkamali, and A. Slowik, "Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing," *Neural Comput. Appl.*, vol. 33, no. 19, pp. 13075–13088, Oct. 2021.
[43] X. Fu, Y. Sun, H. Wang, and H. Li, "Task scheduling of cloud computing based on hybrid particle swarm algorithm and genetic algorithm," *Cluster Comput.*, vol. 26, no. 5, pp. 2479–2488, Oct. 2023.

HAMEED HUSSAIN received the bachelor's degree in information technology from Gomal University, Dera Ismail Khan, Pakistan, in 2007, and the M.S. and Ph.D. degrees in computer science from the COMSATS Institute of Information Technology (CIIT), Pakistan, in 2009 and 2017, respectively. He is currently an Active Researcher. He is the author of several international publications. His research interests include optimization, machine learning, fog and edge computing, the Internet of Things, algorithms real-time systems, resource allocation, and load balancing in high-performance computing.

MUHAMMAD ZAKARYA (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Surrey, Guildford, U.K. He is currently an Assistant Professor with the Faculty of Computing and Information Technology (FCIT), Sohar University, Oman, and the Department of Computer Science, Abdul Wali Khan University Mardan (AWKUM), Pakistan. His research interests include cloud computing, mobile edge clouds, the Internet of Things (IoT), performance, energy efficiency, algorithms, and resource management. He has a deep understanding of theoretical computer science and data analysis. Furthermore, he also has a deep understanding of various statistical techniques, which are largely used in applied research. His research has appeared in several international conferences, journals, and transactions of repute. He is a TPC Member of a few prestigious international conferences, including CCGrid, GECON, ICCCI, FIT, and UCC. He is an Associate Editor of IEEE Access, *Journal of Cloud Computing* (Springer), and *Journal of Cluster Computing* (Springer). He is the Program Director of the iFuture: a leading Research Group, AWKUM, which has research collaboration with the CLOUDS Laboratory, The University of Melbourne, Australia, and the IoT Laboratory, Cardiff University, U.K.

AHMAD ALI received the Ph.D. degree from Shanghai Jiao Tong University, China. He is currently a Postdoctoral Researcher with the Department of Computer Science and Engineering, Shenzhen University, China. His current research interests include deep learning, big data analytics, data mining, urban computing, cloud computing, and fog computing. He is a Reviewer of various SCI-journals, including *Information Sciences*, *Multimedia Tools and Applications*, *Neural Computing and Applications*, IEEE Access, *IEEE Internet of Things Magazines*, and *Journal of Healthcare Engineering*.

AYAZ ALI KHAN received the Ph.D. degree in computer science from Abdul Wali Khan University, Pakistan. He is currently an Assistant Professor with the Department of Computer Science, University of Lakki Marwat, Pakistan. His research interests include cloud computing, mobile edge clouds, the Internet of Things (IoT), performance, energy efficiency, algorithms, and resource management. He has a deep understanding of theoretical computer science and data analysis. Furthermore, he also has a deep understanding of various statistical techniques, which are largely used in applied research. His research has appeared in several international conferences, journals, and transactions of repute.

MOHAMMAD REZA CHALAK QAZANI received the Bachelor of Engineering degree in manufacturing and production from the University of Tabriz, Tabriz, Iran, in 2010, the master's degree in robotic and mechanical engineering from Tarbiat Modares University, Tehran, Iran, in 2013, and the Ph.D. degree in modeling and simulation of a motion cueing algorithm using prediction and computational intelligence techniques from the Institute for Intelligent Systems Research and Innovation (IISRI), Deakin University, Australia, in 2021. He is currently with the Department of Computer Science and Information Systems, Murray State University, Murray, KY, USA, and an Assistant Professor with the Faculty of Computing and Information Technology, Sohar University, Oman. He is also an Alfred Deakin Post-Doctoral Research Fellow with IISRI, Deakin University. His current research interests include model predictive control, motion cueing algorithms, and soft computing controllers.

MAHMOOD AL-BAHRI received the Ph.D. degree from Saint Petersburg State University of Telecommunications, Saint Petersburg, Russia. He has held prestigious academic positions at renowned institutions. While pursuing the Ph.D. degree, he was a Lecturer with the Department of Communication Networks and Data Transmission, Saint Petersburg State University of Telecommunications, from 2017 to 2019. Following that, he was an Assistant Professor with the University of Nizwa, Oman, from 2019 to 2020. Since 2020, he has been an Assistant Professor with the Faculty of Computing and Information Technology, Sohar University. His research interests include computer systems, networks, the Internet of Things, and info-communication devices.

MUHAMMAD HALEEM received the M.S.C.S. degree from the Department of Computer Science, Abdul Wali Khan University, Mardan, Pakistan. He is currently an Assistant Professor with the Department of Computer Science, Faculty of Engineering, Kardan University, Kabul, Afghanistan. His research interests include the Internet of Things, machine learning, cloud computing, particle swarm optimization, power systems, and data analytics.