Computers, Materials & Continua

Tech Science Press

DOI: 10.32604/cmc.2024.048833

ARTICLE

Hybrid Approach for Cost Efficient Application Placement in Fog-Cloud Computing Environments

Abdulelah Alwabel[1,*] **and Chinmaya Kumar Swain**[2]

[1]Department of Computer Sciences, College of Computer Engineering and Sciences, Prince Sattam bin Abdulaziz University, Al-Kharj, 11942, Saudi Arabia

[2]Department of Computer Science and Engineering, Institute of Management and Information Technology, Cuttack, BPUT, Odisha, India

*Corresponding Author: Abdulelah Alwabel. Email: a.alwabel@psau.edu.sa

Received: 20 December 2023 Accepted: 07 April 2024 Published: 20 June 2024

ABSTRACT

Fog computing has recently developed as a new paradigm with the aim of addressing time-sensitive applications better than with cloud computing by placing and processing tasks in close proximity to the data sources. However, the majority of the fog nodes in this environment are geographically scattered with resources that are limited in terms of capabilities compared to cloud nodes, thus making the application placement problem more complex than that in cloud computing. An approach for cost-efficient application placement in fog-cloud computing environments that combines the benefits of both fog and cloud computing to optimize the placement of applications and services while minimizing costs. This approach is particularly relevant in scenarios where latency, resource constraints, and cost considerations are crucial factors for the deployment of applications. In this study, we propose a hybrid approach that combines a genetic algorithm (GA) with the Flamingo Search Algorithm (FSA) to place application modules while minimizing cost. We consider four cost-types for application deployment: Computation, communication, energy consumption, and violations. The proposed hybrid approach is called GA-FSA and is designed to place the application modules considering the deadline of the application and deploy them appropriately to fog or cloud nodes to curtail the overall cost of the system. An extensive simulation is conducted to assess the performance of the proposed approach compared to other state-of-the-art approaches. The results demonstrate that GA-FSA approach is superior to the other approaches with respect to task guarantee ratio (TGR) and total cost.

KEYWORDS

Placement mechanism; application module placement; fog computing; cloud computing; genetic algorithm; flamingo search algorithm

1 Introduction

In 2014, Cisco proposed fog computing as a new paradigm with the aim of addressing time-sensitive applications better than with cloud computing by placing and processing tasks closer to the data sources [1]. Fog computing can curtail various costs such as networking overhead, making it a suitable platform for dealing with latency-sensitive tasks [2]. Fog computing is a distributed computing environment that expands services of cloud systems to users and data sources [3].

Nodes in this environment are preferably located only one hop at a distance from the data to be handled. Placing nodes in a close location to users allows for processing data locally in the nodes instead of transferring data to nodes in cloud systems in a remote location, which can cause extra communication overhead. Fog computing is developed as a middle layer between Internet of Things (IoT) and cloud computing and is suitable for time-sensitive application placement. The majority, nonetheless, of the fog nodes are geographically scattered with resources that are limited in terms of computing capabilities compared to cloud nodes, thus making the application placement problem more complex than that in cloud computing.

The application placement problem further complicates microservice applications. A microservice application is a set of interdependent and interconnected modules that might be processed by various computing nodes [4]. Each module executes a set of instructions to produce the appropriate output that might be communicated as an input to another module based on the dependency of data between these modules [5]. This study presents a novel mechanism of application placement for fog-cloud environments that aims to optimize the overall system utilization by efficiently reducing the cost of computation, communication, energy, and service level agreement (SLA) violations. The proposed work combines the genetic algorithm (GA) with the Flamingo Search Algorithm (FSA) to place application modules to minimize overall cost. The choice of using GA and FSA in a hybrid algorithm depends on the problem at hand and the characteristics (Complementary Strengths, Multi-objective Optimization, and Adaptability) of the algorithms being combined. If the problem exhibits feature that align well with both GA and FSA, a hybrid approach might be more effective. It is worth noting that the selection of metaheuristic algorithms for a hybrid approach is often empirical and depends on the characteristics of the optimization problem. Other metaheuristic algorithms could also be considered for our problem in future. The hybrid approach considers the cost associated with computation, communication, violation, and energy consumption to place the applications to appropriate fog or cloud nodes. The violation cost associated with the placement of applications considers the real-world penalty model adopted by various cloud service providers. The work also focuses on more number of applications to be executed before their deadline on the considered fog-cloud system.

The major contribution of the paper is as follows:

- A novel hybrid GA-FSA approach is proposed for application placement in a fog-cloud environment. This hybrid approach is intended to overcome issues such as low convergence rates and high costs.
- In the proposed model, we consider four cost types (computation, communication, violation, and energy consumption) associated with application placement.
- The model for violation cost is aligned with the existing penalty models used by the standard cloud service providers.
- The hybrid approach effectively places the applications to fog or cloud nodes based on the application parameters, which improves the task guarantee ratio (TGR) while maintaining the Quality of Service (QoS) and stability of the overall system.

The remainder of this paper is organized as follows. In Section 2, we discuss the state-of-the-art application placement in cloud and fog models, including the identified gaps in the research. The problem is formulated and discussed in Section 3. In Section 4, we present the proposed mechanism, which is evaluated and discussed in Section 5. Finally, in Section 6, we present our conclusions and future directions for further improvements in the proposed work.

2 Related Work

This section discusses the research conducted by different researchers in related areas. A novel framework was proposed to allocate workload in a fog-cloud system to minimize the power consumption while maintaining an acceptable level of service delay [6]. The proposed framework divides workload allocation into three subproblems. This framework attempts to solve this problem by employing a generalized benders decomposition (GBD) mechanism. The results demonstrated that the power consumption can be optimized within a fog-cloud system. However, they did not thoroughly investigate the complexion of workloads and resources [7].

The authors of [8] presented a mechanism to map module with an aim to place IoT applications in a fog-cloud environment. The contribution of this work was to improve resource utilization. Utilization is improved by sorting and mapping nodes to application modules according to the available capacity and requirements of these modules and nodes. Then, the mechanism maps the modules when the requirements are fulfilled. This study, however, considered only CPU and RAM to find an appropriate candidate. Therefore, the mechanism misses other crucial elements, for instance time requirements of applications.

The authors of [9] proposed a fog-supported architecture that manages both the computational and networking costs. The architecture employs an energy-aware algorithm within a fog-based datacenter. A node is rewarded or penalized according to the idleness state of the node, top frequency, and highest energy parameters. It, then, computes how many virtual machines might be placed on a node in different timeslots and according to their frequencies. The reward mechanism is utilized to minimize power consumption, which could grow aggressively.

The effectiveness of mobility on the behavior of applications was investigated by studying the problem of scheduling in fog systems [10]. The study reviewed several scheduling approaches with an aim to reduce the execution time of applications based on their characteristics. This study demonstrated that the delay-priority policy outperformed the others with regards to reducing network delay.

A service-based placement mechanism for fog computing was designed by [11] with the aim of sharing resources in an optimum way in nodes inside IoT systems. The mechanism takes in consideration both deadline and latency requirements when allocating various modules on machines. Each machine is identified by computation, memory, and storage elements. The proposed policy tries to meet QoS demands by prioritizing applications based on expected time to response.

A new technique was developed by [12] to manage applications in order to satisfy different service-delivery latency requirements in fog computing. iFogsim [13] was used to evaluate the proposed technique. The technique outperformed other approaches in the literature by assigning modules to nodes within the required deadlines.

The authors in [14] developed a quality of experience (QoE) mechanism that assigns modules to fog nodes. The mechanism prioritizes different requests to place module according to the expectations of different users. In similar fashion, a study on improving quality of experience was proposed by [15]. This approach, however, does not consider resource availability as a factor when placing modules to nodes. This can be considered a weakness in this context according to [16] in cloud-fog environments.

The authors of [17] presented an optimized placement approach for applications in fog environments based on a GA. This approach aims at reducing the costs of the computational as well as exchange of data overheads of tasks assignments while preserving an acceptable level of utilization. Another study employing a GA was carried out by [18]. This study presented a mechanism to schedule tasks with a focus on cost factor that is based on GA-based algorithm for fog-cloud systems in order to cut the cost time-aware applications.

Another scheduling mechanism was proposed by [19] that maps tasks to nodes according to deadline and frequency constraints. The approach employs an auction mechanism, where tasks that are rejected by one fog node can be accepted by another. It illustrates that the total number of executed tasks can increase as a consequence of using this mechanism.

A novel placement mechanism was presented by [20] that aims to improve throughput by focusing on the requirement of computation and networking of applications. The performance was improved by allocating the maximum number of modules to one area to curtail the networking overheads in fog computing. Experiments demonstrated that the proposed mechanism outperformed related methods in terms of throughput improvement. However, the proposed method only minimally addresses saving energy in fog environments. The authors of [21] proposed a deadline-aware mechanism that plays an intermediate role for processing tasks in a fog-cloud environment. In addition, the mechanism focuses on the efficient utilization of resources.

The authors of [22] presented a placement mechanism that takes context of applications in consideration in fog environments. This mechanism can help in reducing the latency of applications in IoT systems by mapping IoT machine-level contexts with the specifications of the nodes in a fog system. The authors of [23] employs mechanism that is based on the grey wolf optimization approach to curtail the execution costs for IoT applications in a fog environment. This improves the performance with regard to the application deadlines.

The authors of [24] proposed an algorithm called PACK that maps tasks to fog node according to locations with an aim to reduce service delay. The travel time between data sources and fog nodes are minimized by placing these parties close to each other provided the load between working nodes is balanced. Moreover, PACK ranks fog nodes based on location and reliability as factors for allocating tasks to nodes. A hybrid mechanism that merges a linear programming method with several heuristic mechanisms to optimize energy and bandwidth consumption in a fog environment was introduced by [25].

A load-balancing placement policy for fog and cloud systems on IoT platforms was presented by [26]. This work enhanced their previous work [27] by employing a software-defined network paradigm to enhance the outcome of IoT applications in fog-cloud computing. The approach employs a load-balancing mechanism to distribute tasks between fog nodes to suit the demands of expandability and time-constraint while avoiding fog nodes being overloaded. This study showed that the delay can decrease because of this approach. However, load balancing can lead to poor resource utilization and an increase in power consumption.

The authors of [28] presented a mechanism that coordinates the allocation of tasks as a chain of services [29] using a deep reinforcement learning approach. A vigorous method was employed to reduce latency; improve resource usage; and enhance productivity in fog systems. A similar approach was adopted by [30] to solve the scalability shortcomings of existing schemes for the dynamic placement of the mechanism.

The authors of [31] proposed an efficient and autonomous scheme to solve the problem of mapping IoT services to fog nodes using metaheuristic approaches with a shared parallel architecture. They employ the Archimedes optimization algorithm as a new metaheuristic approach that addresses the complexity of IoT service allocation to fog nodes as a multi-objective problem. The approach employed helped reduce the overall costs of fog systems.

The authors of [32] presented a novel application placement mechanism for fog computing that focusing on reducing power consumption. The main contribution of this paper is to reduce the number of running fog nodes which leads to a cut in the overall energy consumption of the system. However, this study pays a little attention to the cost of the proposed mechanism. The adoption of mechanism can increase the SLA violation as a result.

In the proposed approach, we formulated the cost model using four types of cost, like computation cost, communication cost, violation cost, and energy consumption cost. Many researchers have formulated the model taking at most three types of cost and we included the fourth one, i.e., energy consumption cost to make the problem more realistic. General optimization techniques are proposed by different researcher's for solving this type of problems and we tried with that along with the hybrid approach to explore the efficacy of the hybrid approach. The hybrid approach we consider here combines both GA and FSA to place the applications efficiently in fog/cloud nodes to minimize the overall cost while satisfying the deadline requirements.

3 Problem Formulation

This section defines the problem application placement in fog-cloud environment. In addition, it presents several cost-type models of application executions in this environment.

3.1 Computing Environment

The considered computing environment consists of a three-layered hierarchical structure, as displayed in Fig. 1. The uppermost layer is the cloud layer, which consists of servers that we call datacenters. The middle layer, called the fog layer, links the cloud with device layers. The lowest layer is the device layer, where user applications are generated. The fog layer resides closer to the data generation sources and addresses requests in a time-bound manner. Latency-sensitive tasks are addressed by fog servers rather than cloud servers because the cloud servers are placed distant from the user request locations. The decentralized feature of fog computing makes it more appropriate for addressing latency-sensitive tasks that can be submitted to the system in a distributed manner. The focus of this study is to use computing and communication resources in a collaborative manner to minimize system utilization costs.

3.2 Machine Environment

In this subsection, we consider a set of m computing nodes $M = \{M_1, M_2, \cdots, M_m\}$, which consists of k cloud nodes and other nodes $(m-k)$ as fog nodes. Each node in the machine environment has the following characteristics: $\mathrm{M_j} = < \mathrm{M_j^{rate}}, \mathrm{M_j^{cpu}}, \mathrm{M_j^{mem}} >$. The term M_j^{rate} represents the CPU processing rate of machine M_j, M_j^{cpu} represents the CPU capacity, and M_j^{mem} represents the memory capacity of machine M_j. Cloud nodes have more computing power than fog nodes and have higher communication latency because the cloud nodes are distant from the task submission point.

3.3 Task Environment

The set of n tasks $T = \{T_1, T_2, T_3, \ldots, T_n\}$ is submitted to the system, where each task T_i has the following characteristics: $T_i = < ins_i, cpu_i, mem_i, data_i, bw_i, d_i >$. Term ins_i represents the number of instructions for task T_i. The other terms cpu_i, mem_i, and bw_i represent the CPU, memory, and bandwidth requirements for task T_i, respectively. The term $data_i$ represents the input/output file size and d_i represents the deadline of the task. We assume that the sets of tasks/applications submitted to the system are independent of each other. When a task is submitted to the system, it is scheduled to either a cloud or fog server for execution. The execution time for a task can be computed based on the number of instructions and rate of CPU processing where the task is scheduled. Herein, we define the execution time of task T_i when allocated to machine M_j as e_{ij}, which can be calculated as follows:

$$e_{ij} = \frac{ins_i}{M_j^{rate}} \tag{1}$$

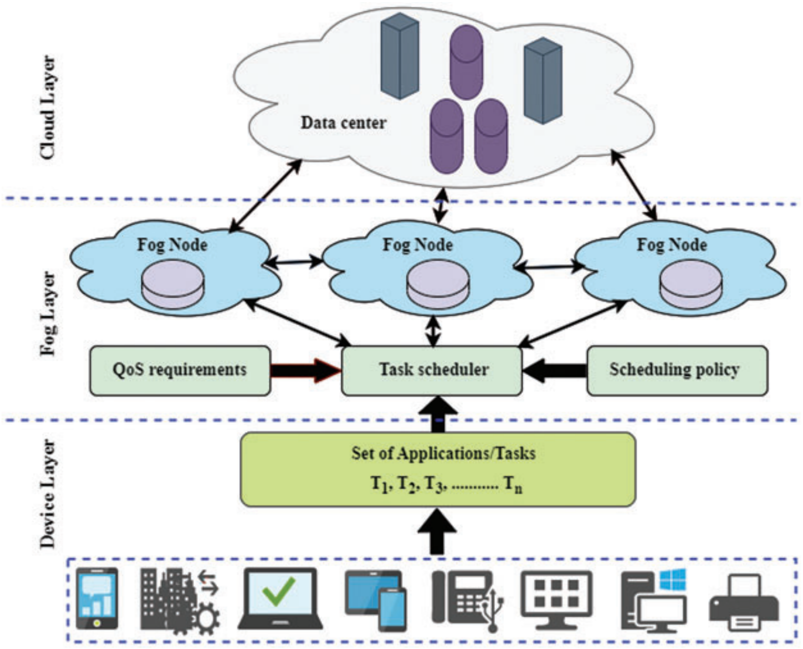

Figure 1: Proposed computing environments

3.4 Cost Model

This section defines the different cost-type models associated with fog-cloud environments.

3.4.1 Computational Cost

The primary monetary cost associated with the fog-cloud environment is computational cost. The computation cost depends on the amount of computational resources required to execute the application and execution duration [33]. Herein, we consider two computational resources: The CPU and memory. The computational cost of task T_i scheduled on machine M_j can be calculated as follows:

$$Comp_i^{cost} = \left(c_j^p \times cpu_i + c_j^m \times mem_i\right) \times e_{ij} \tag{2}$$

where c_j^p is the per-unit CPU usage cost per unit time and c_j^m is the per-unit memory usage cost per unit time for machine M_j. Here cpu_i and mem_i represent the CPU and memory requirement of the task T_i, respectively. The term e_{ij} represents the expected execution time of the task T_i when allocated to the machine $\mathrm{M_j}$. Thus, the total computational cost of n tasks can be computed as follows:

$$Comp^{cost} = \sum_{i=1}^{n} Comp_i^{cost} \tag{3}$$

3.4.2 Communication Cost

In addition to computational cost, the communication cost has a vital role in scheduling because latency is crucial for IoT application deployment [34,35]. The communication cost depends on the size of the data file involved in the task and cost of bandwidth usage per data unit of the host server. The communication cost involved in task T_i can be computed as follows:

$$Comm_i^{cost} = c_j^b \times bw_i \tag{4}$$

where c_j^b is the bandwidth-usage cost per unit of data for machine M_j. The total communication cost for all n tasks can be obtained using the following equation:

$$Comm^{cost} = \sum_{i=1}^{n} Comm_i^{cost} \tag{5}$$

3.4.3 SLA Violation Cost

The fog-cloud service provider must adhere to the SLA signed by the service provider and user. The service provider must repay compensation proportional to the violation level. Popular penalty computation methods adopted by different cloud-service providers are presented in Table 1. There are three popular methods for calculating the penalty: (a) a certain percentage of the total charge paid, (b) a fixed amount paid for different violation levels, and (c) a specific ratio of downtime [36]. In the proposed model, we design the violation cost, which depends on the violation level, i.e., the amount of time the task requires to finish its execution beyond the specified deadline. We formulate the violation cost model, which is similar to the penalty models of real cloud service providers and can be defined as

$$Violation_i^{cost} = V_i \times Penalty_i \tag{6}$$

where $\boldsymbol{Penalty_i}$ is the violation cost of task $\boldsymbol{T_i}$ for one percent of the delay violation; $\boldsymbol{V_i}$ can be obtained using the following equation:

$$V_i = \begin{cases} \dfrac{f_i - d_i}{e_{ij}} \times 100 & if \quad f_i > d_i \\ 0 & if \quad f_i \leq d_i \end{cases} \tag{7}$$

where f_i is the completion time of task T_i. To determine $V_i \in [0, 100]$, we assume that if any task requires more time beyond the deadline than its execution time, then the task must not be allowed to be executed. Considering the above formulations for penalty calculations, we design the violation cost of the system as follows:

$$Violation^{cost} = \sum_{i=1}^{n} Violation_i^{cost} \tag{8}$$

Table 1: Penalty models, reprinted with permission from [33]

Cloud provider	Calculation method	Service credit	Penalty cap
Amazon EC2	Ration of total charge	< 99.95%–10% < 99%–30%	N/A
IBM Softlayer	Ration of downtime	Each 30 min downtime, 5% of the fees	N/A

(Continued)

Table 1 (continued)

Cloud provider	Calculation method	Service credit	Penalty cap
Windows Azure	Ration of total charge	< 99.95%–10% < 99%–25%	N/A
VPS.net	Ration of downtime	10 × *downtime*	100%
Google GCE	Ration of total charge	< 99.95%–10% < 99%–25% < 99%–50%	N/A
Rackspace	Ration of downtime	Each 30 min downtime, 5% of the fees	100%
GoGrid	Ration of downtime	10 × *downtime*	100%

3.4.4 Energy Consumption Cost

In this section, we formulate the energy cost of the system under consideration to execute the set of tasks submitted to the system. The energy consumption cost depends on the total number of servers required to execute the tasks. To compute the energy consumption of each server/node (M_j), the following formula is used:

$$E_j = E_j^{static} + E_j^{dynamic} \tag{9}$$

where E_j^{static} is the energy consumed by the server/node M_j when it is ideal and $E_j^{dynamic}$ is the energy consumed by the server while executing any task. The term E_j^{static} is constant for a particular machine/server; however, the term $E_j^{dynamic}$ varies and depends on the computational resource utilization over time. Herein, we formulate the dynamic component of the energy consumption using two types of computational resources: CPU and memory utilization. This model could be extended by considering additional resources. Similar assumptions were made in [37]. Dynamic energy consumption ($E_j^{dynamic}$) is computed as follows:

$$E_j^{dynamic} = \left(\alpha \times U_j^{CPU} + \beta \times U_j^{mem}\right) \times E_j^{max} \tag{10}$$

where α and β are positive constants and $\alpha + \beta = 1$. The terms U_j^{CPU} and U_j^{mem} represent the CPU and memory utilization of the machine M_j, respectively. Herein, E_j^{max} represents the energy consumption of the machine when operating at its maximum capacity. Thus, the total energy consumption cost of the system under consideration can be computed as follows:

$$Energy^{cost} = \sum_{t=0}^{TimeDuration} \sum_{j=1}^{m} P_j \times c_j^e \tag{11}$$

where c_j^e is the energy consumption per server unit. The total cost ($Total^{cost}$) of the system is computed as follows:

$$Total^{cost} = Comp^{cost} + Comm^{cost} + Violation^{cost} + Energy^{cost} \tag{12}$$

where $Comp^{cost}$ is the computational cost, $Comm^{cost}$ is the communication cost, $Violation^{cost}$ is the violation cost, and $Energy^{cost}$ is the energy-related cost of the system under consideration.

3.5 Optimization Goal

The main objective of this study is to schedule a set of latency-sensitive independent tasks to fog/cloud servers such that the total cost is minimized. The final optimization formula for the stated problem is defined as an Mixed-Integer Linear Programming (MILP) problem and is expressed as follows:

$$\text{Minimize}(Total^{cost}), \tag{13}$$

subject to

Task Constraint:

$$\sum_{i=1}^{n} x_{ij} = 1, \forall j \in \{1, \ldots, m\} \tag{14}$$

and

CPU Constraint:

$$x_{ij} \times cpu_i \leq M_j^{cpu}, \forall i \in \{1, \ldots, n\}, j \in \{1, \ldots, m\} \tag{15}$$

and

Memory Constraint:

$$x_{ij} \times mem_i \leq M_j^{mem}, \forall i \in \{1, \ldots, n\}, \forall j \in \{1, \ldots, m\} \tag{16}$$

Binary decision variable x_{ij} is defined as

$$x_{ij} = \begin{cases} 1 & \text{if task } T_i \text{ is allocated to machine } M_j \\ 0 & \text{Otherwise} \end{cases} \tag{17}$$

Eq. (14) states that one task will be allocated to only one server and the term x_{ij} is a binary variable which takes the value 1 when task (T_i) is allocated to a machine (M_j) otherwise the value is 0. Eqs. (15) and (16) are the CPU and memory constraints for each machine, respectively. The total CPU and memory required for the set of tasks allocated to a machine must be less than the total available resources.

4 Proposed Mechanism

There are three layers in a fog-cloud computing environment. The lower layers, called the IoT device layers, are generated from the user tasks and sent to either the fog layer (middle layer) or cloud layer (top layer). The tasks submitted to the cloud layer must pass through the middle layer because the fog broker decides whether to allocate the tasks to the nodes/servers deployed in the fog layer or the cloud layer. The main functions of a fog broker are threefold: (a) request receiver, (b) resource monitoring, and (c) task scheduling. The proposed scheduling process is displayed in Fig. 2, where the requests from IoT devices are received by the fog broker unit through gateways. The fog broker analyzes the task parameters and schedules the tasks to the appropriate servers in the fog or cloud layer to minimize costs and optimize QoS.

In this case, we designed a hybrid approach by combining the GA [38] with the FSA [39], which we call GA-FSA. The proposed model efficiently schedules tasks to fog/cloud nodes to meet the QoS and minimize cost.

GA is a popular evolutionary algorithm widely used in different optimization problems [38]. This algorithm was inspired by Darwin's evolutionary principle based on gene simulations. The GA follows predefined steps such as (a) initial population generation, (b) evaluation of the fitness function for an individual population, (c) selection of the best possible populations to reproduce a new generation, (d) crossover operation for producing new generations from the parent population, and (e) mutation operation to produce a new child and avoid the algorithm from being trapped in a local optimum. These steps are repeated until the algorithm converges or the maximum number of iterations is achieved, as indicated in Fig. 3.

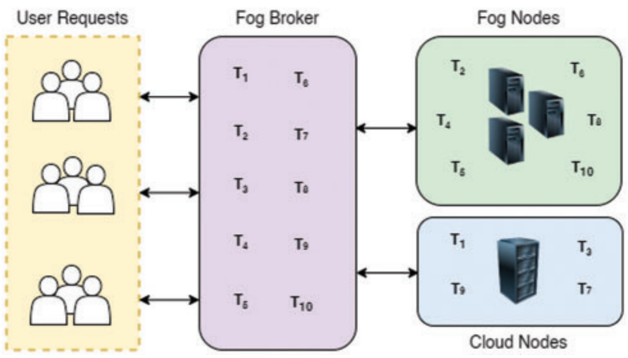

Figure 2: Proposed approach for task scheduling

Similarly the FSA was inspired by the migratory and foraging behaviors of flamingos. The FSA algorithm uses migratory and foraging behaviors to select the optimal solution, as indicated in Fig. 4.

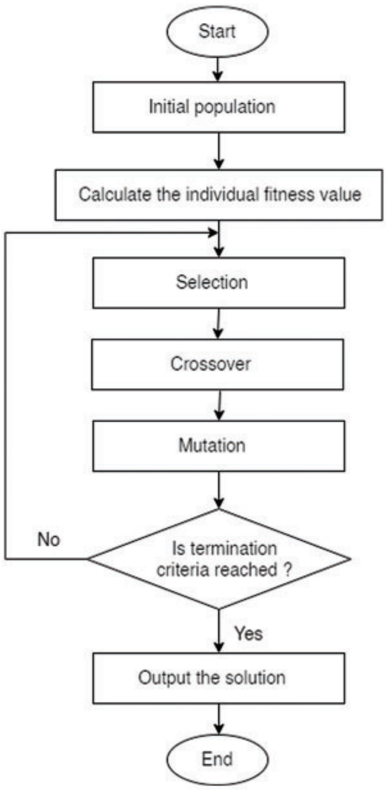

Figure 3: Flowchart for GA

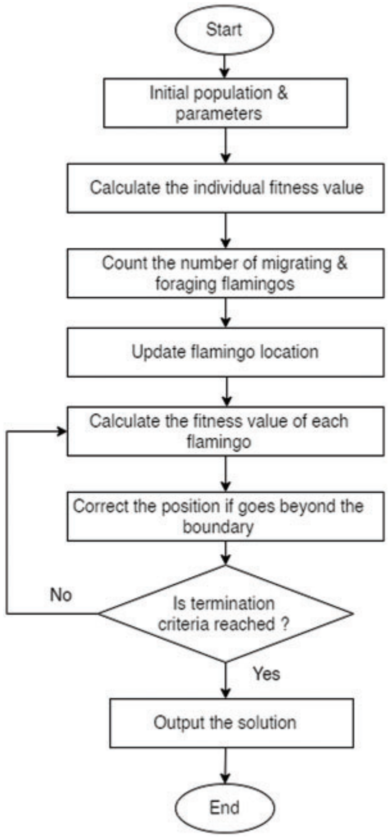
Start
Initial population & parameters
Calculate the individual fitness value
Count the number of migrating & foraging flamingos
Update flamingo location
Calculate the fitness value of each flamingo
Correct the position if goes beyond the boundary
Is termination criteria reached ?
No
Yes
Output the solution
End

Figure 4: Flowchart for FSA

In this study, we propose a hybrid algorithm called GA-FSA to effectively schedule tasks to servers in the fog and cloud layers. A flowchart of the combined approach is displayed in Fig. 5. The proposed approach follows the GA to generate the initial population and encoded form of the population, as indicated in Fig. 6. The chromosomes of the population represent the tasks, and the gene value of each chromosome is considered as the server where a task is scheduled.

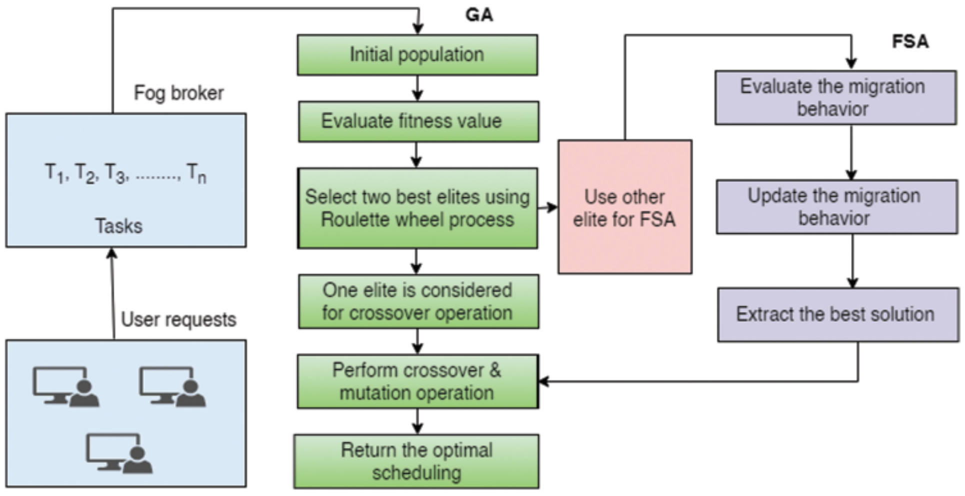
GA
Initial population
Evaluate fitness value
Select two best elites using Roulette wheel process
One elite is considered for crossover operation
Perform crossover & mutation operation
Return the optimal scheduling
Fog broker
T1, T2, T3, ..., Tn
Tasks
User requests
Use other elite for FSA
FSA
Evaluate the migration behavior
Update the migration behavior
Extract the best solution

Figure 5: Hybrid GA and FSA approach

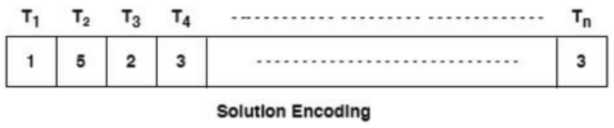

Figure 6: Individual solution encoding of GA

From the entire set of populations based on fitness values (using Eq. (12)), two best elites are chosen using the roulette wheel process. Assume, the two elites are G_1 and G_2. G_1 is used for the crossover operation through the GA. The other elite, G_2, is used by the FSA to generate the optimal solution. The best elites from GA and FSA are promoted for the two-point crossover operation (Fig. 7) and one point mutation operation (Fig. 8). The results of this series of crossover and mutation operations produce an effective task scheduling.

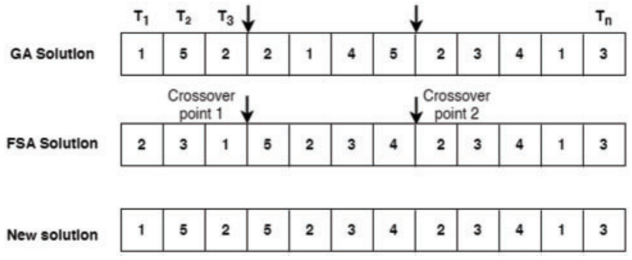

Figure 7: Illustration of two-point crossover operation

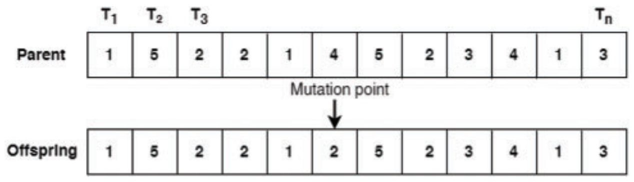

Figure 8: Illustration of one-point mutation operation

4.1 Hybrid GA-FSA Scheduling Approach

User applications are the inputs for the proposed approach. The steps involved in our hybrid approach are depicted in Algorithm 1:

Algorithm 1: Hybrid Approach (GA-FSA)

```
1.    Initialize parameter values
2.    Initialize chromosome encoding for crossover and mutation operation
3.    Evaluate fitness function using Eq. (13)
4.    if maximum iteration is reached then
5.        Perform selection operation
6.        Select two elites $G_1$ and $G_2$ based on fitness value for the next process
7.        Select elite $G_1$ for crossover operation
8.    end if
9.    while termination condition is not matched
10.       Select other elite $G_2$ for FSA
11.       Promote two best solutions $G_1$ and $G_2$ for crossover and mutation operation
12.   end while
13.   return optimal solution
```

Herein, we use a two-point crossover operation where a new solution is obtained by swapping the genes of the chromosomes, as illustrated in Fig. 7. After the crossover operation, the chromosomes undergo a single-point mutation operation where a new solution is obtained by flipping the digit of the string, as indicated in Fig. 8. Through this hybrid approach, we can obtain effective scheduling of tasks to the fog/cloud servers. The pseudocode for the proposed approach is presented in algorithm 1.

5 Experimental Evaluation

The proposed hybrid approach, GA-FSA, was implemented using Python to evaluate its efficacy against other state-of-the-art approaches. GA-FSA was compared with approaches such as the GA, Min-CCV, and Min-V mechanisms. The performances of the approaches were evaluated based on the computation cost, communication cost, energy consumption cost, SLA violation cost, total cost, makespan, and TGR. TGR is defined as follows [40]:

$$TGR = \frac{Number\ of\ tasks\ executed\ before\ deadline}{Number\ of\ admitted\ tasks} \tag{18}$$

GA is a metaheuristic approach for finding a suitable solution that supports multi-objective optimization problems [41]. Min-CCV [42] is a cost-aware scheduling approach that allocates tasks to nodes to minimize cost. Min-V [42] is a heuristic scheduling approach that minimizes delay violations.

5.1 Simulation Setup

The simulation was performed to understand the efficiency of the proposed hybrid approach by varying three parameters: The number of tasks, number of fog nodes, and number of cloud nodes. We conducted these experiments to investigate the influence of these parameters on the optimization goal stated earlier. We performed the experiments in different phases. In the first phase, we varied the number of tasks from 100 to 500 and fixed them with 20 cloud nodes and 50 fog nodes. In the second phase of the experiment, we fixed the number of tasks to be constant, i.e., 500, and the number of cloud nodes to 20. Subsequently, we performed experiments by varying the number of fog nodes from 10 to 50. The third phase of the experiment was conducted by varying the number of cloud nodes from 5 to 20, holding the number of tasks (500) and fog nodes (50) constant. The experimental settings of the simulations are listed in Table 2.

Table 2: Experimental settings

Phases	Characteristics	Parameters		
		Tasks	Fog nodes	Cloud nodes
1	Varying tasks	[100, 500]	50	20
2	Varying fog nodes	500	[20, 50]	20
3	Varying cloud nodes	500	50	[5, 20]

In the proposed method, the tasks were submitted to the system with the characteristics described in Table 3. The computing environment discussed previously consisted of fog and cloud nodes with heterogeneous characteristics. The values for the different parameters were selected randomly and set for experimental purposes. The node attributes of fog/cloud environment are listed in Table 4.

Table 3: Task settings

Parameter	Values	Unit
Size	[1028, 4280]	MI
Required CPU	[1, 4]	Integer
Required memory	[50, 200]	MB
Data size	[0.5, 2]	MB
Deadline	[500, 2500]	ms
Penalty	[0.1, 0.5]	G$%

The simulation was written in the Python programming language and conducted on an INTEL CORE i7 7th gen, CPU @ 3.0 GHz, 8 GB RAM, and 64-bit operating system. The parameter for the GA-FSA experiments was reported in Table 5.

Table 4: Node settings

Parameter	Values		Unit
	Fog node	Cloud node	
CPU processing rate	[500, 2000]	[3000, 10000]	MIPS
CPU usage cost	[0.2, 0.5]	[0.05, 0.1]	G\$/s
Memory	[200, 300]	[512, 4096]	MB
Memory usage cost	[0.01, 0.04]	[0.02, 0.06]	G\$/MB
Delay	[1, 5]	[1.5, 2.5]	ms
Bandwidth usage cost	[0.01, 0.02]	[0.05, 0.1]	G\$/MB
Static energy consumption	[1, 5]	[5, 10]	kWh
Dynamic energy consumption	[1, 3]	[1, 6]	kWh
Energy cost	0.1	0.1	G\$/kWh

Table 5: GA-FSA parameters setting

Parameter	Values
Population size	[50]
Crossover rate	[0.6]
Mutation rate	[0.015]
Number of iterations	[100]

5.2 Simulation Result by Varying the Number of Tasks

In this experiment, we varied the number of tasks from 100 to 500 and reported the results in Figs. 9 to 15. A lower makespan value implies that the proposed approach is more efficient. It can be observed that the proposed approach (GA-FSA) demonstrated a lower makespan value than the other state-of-the-art approaches for all task counts (indicated in Fig. 9). Fig. 10 indicates the TGR of each approach; in this case a higher value of TGR is better. GA-FSA provided a higher TGR value than the other approaches. Figs. 11 to 15 report the result of different costs associated with the models. In these cases, the lower the cost, the better the performance. This phenomenon was observed for the proposed approach (GA-FSA) in terms of computation, communication, violation, energy consumption, and total cost.

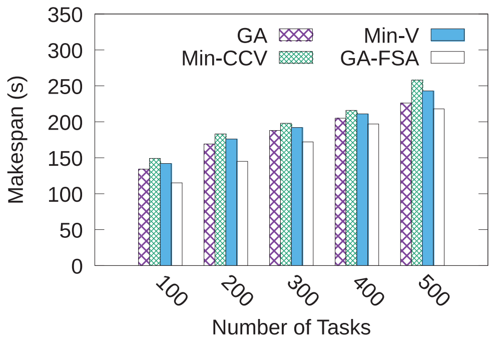

Figure 9: Makespan (Lower value is better)

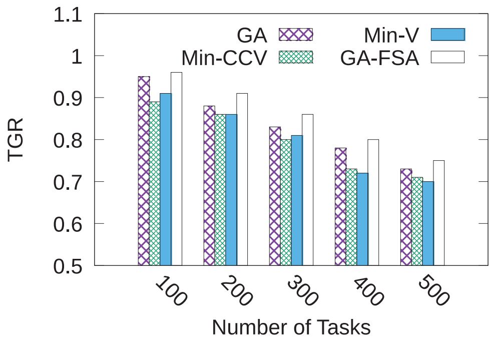

Figure 10: TGR (Higher value is better)

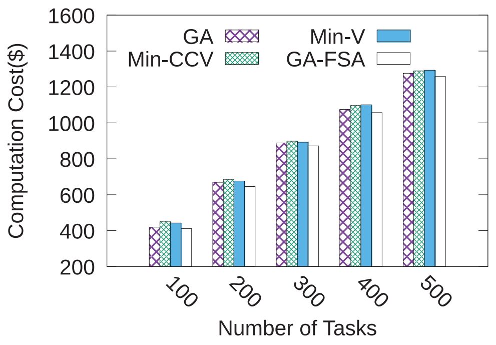

Figure 11: Computation cost (Lower value is better)

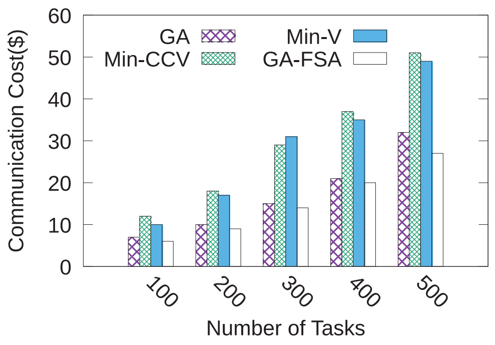

Figure 12: Communication cost (Lower value is better)

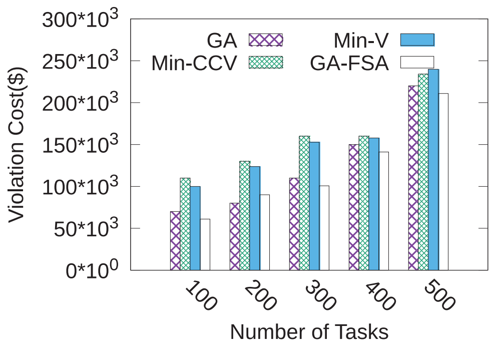

Figure 13: Violation cost (Lower value is better)

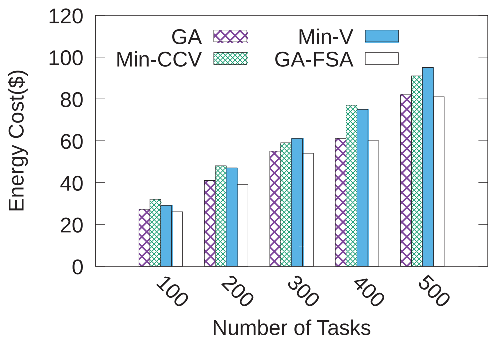

Figure 14: Energy cost (Lower value is better)

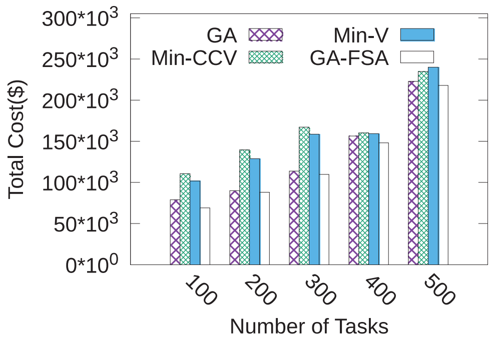

Figure 15: Total cost (Lower value is better)

5.3 Simulation Result by Varying the Number of Fog Nodes

In this case, the number of fog nodes was varied, while maintaining a constant number of tasks (500) and cloud nodes (20). As the number of fog nodes varied, the results exhibited varying trends for different performance parameters. Figs. 16 to 22 report the different parameters for the approaches. Fig. 16 indicates that the proposed approach (GA-FSA) outperformed the other approaches as its value was less compared to the others. As the number of fog nodes increased, the TGR value improved for all cases; GA-FSA had a higher value than the other approaches (as indicated in Fig. 17). The costs associated with the different models are displayed in Figs. 18 to 22. The proposed approach incurred a lower cost than the other approaches, as indicated in the different figures.

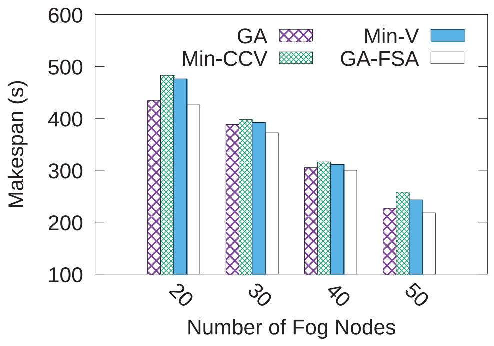

Figure 16: Makespan (Lower value is better)

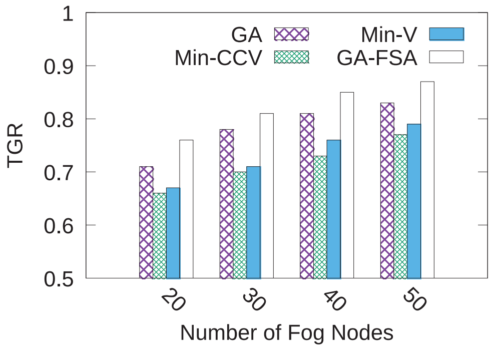

Figure 17: TGR (Higher value is better)

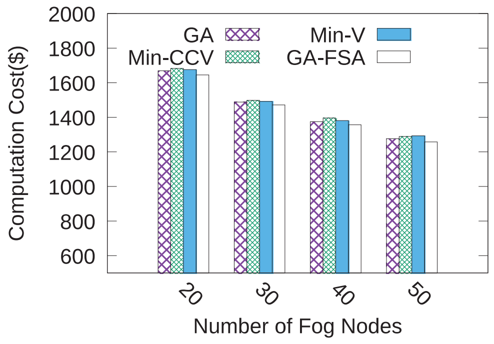

Figure 18: Computation cost (Lower value is better)

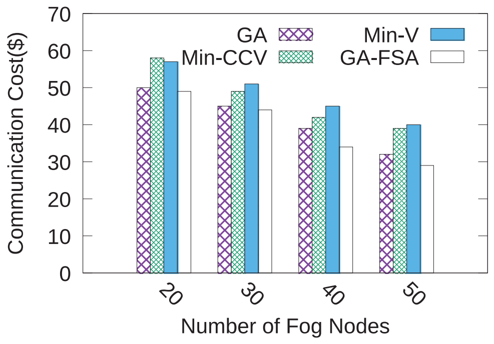

Figure 19: Communication cost (Lower value is better)

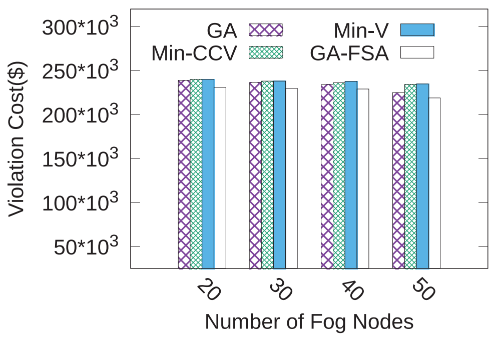

Figure 20: Violation cost (Lower value is better)

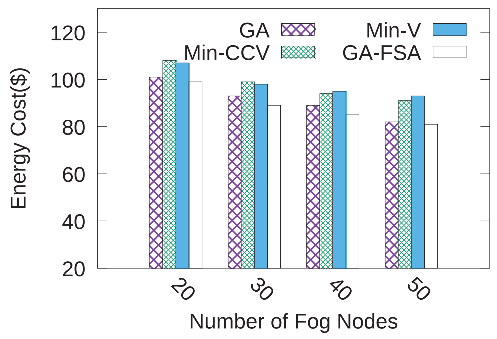

Figure 21: Energy cost (Lower value is better)

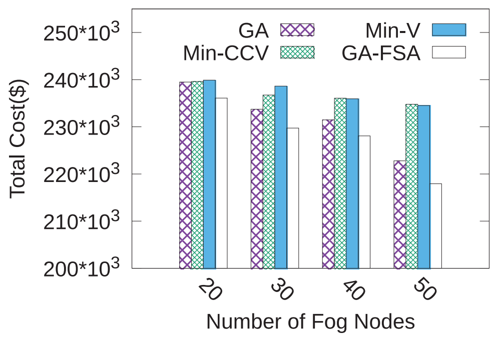

Figure 22: Total cost (Lower value is better)

5.4 Simulation Result by Varying the Number of Cloud Nodes

In this subsection, we varied the number of cloud nodes from 5 to 20, while holding the number of tasks (500) and fog nodes (50) constant. Figs. 23 to 29 report the results for different performance parameters. In this case, the communication cost varied as the cloud nodes were distant from the data source, and hence influenced the TGR. However, in all cases, GA-FSA had a higher TGR value (Fig. 24) compared to the other approaches. The cost associated with GA-FSA was also lower than that associated with the other approaches (Fig. 29).

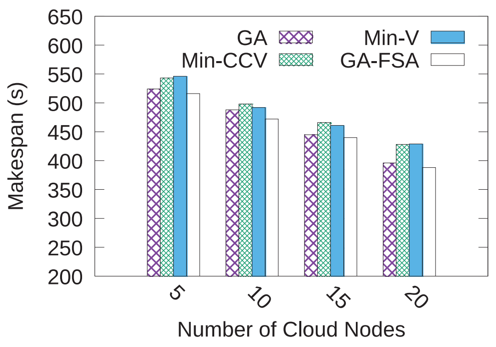

Figure 23: Makespan (Lower value is better)

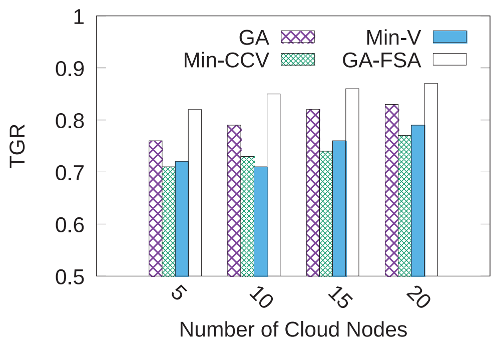

Figure 24: TGR (Higher value is better)

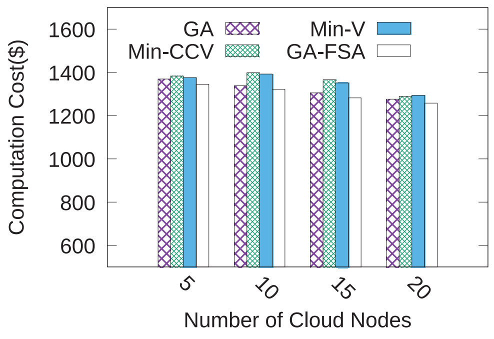

Figure 25: Computation cost (Lower value is better)

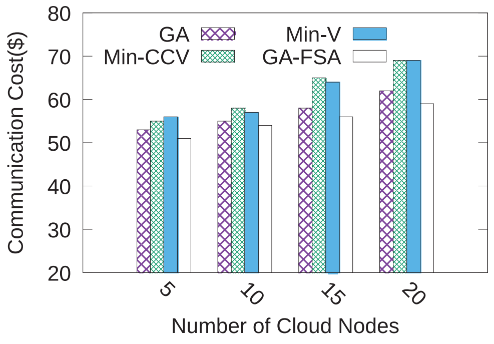

Figure 26: Communication cost (Lower value is better)

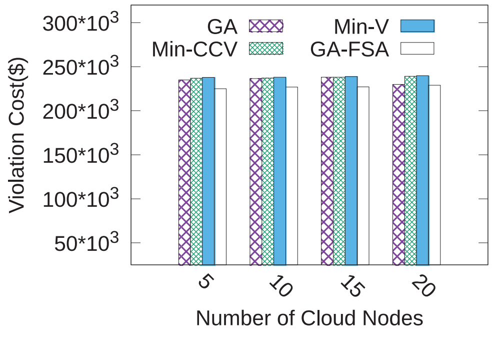

Figure 27: Violation cost (Lower value is better)

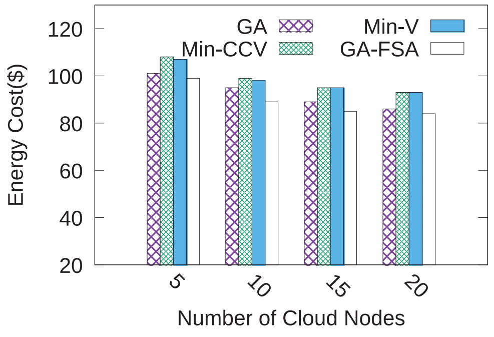

Figure 28: Energy cost (Lower value is better)

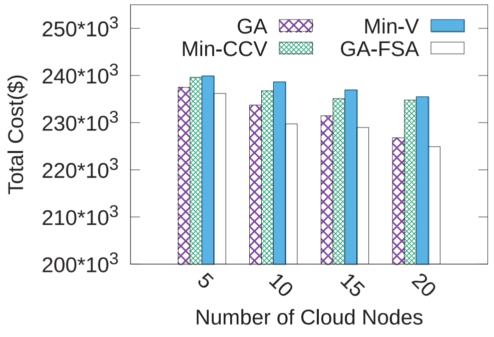

Figure 29: Total cost (Lower value is better)

In addition to reporting the results in terms of different parameters, we also report (Table 6) the running times of the different approaches. The proposed approach, GA-FSA, required less time than the other approaches. Although the time difference was small, it makes sense that by requiring less time, the proposed approach outperformed the other approaches for the different parameters.

Table 6: Running time of different approaches

Approach	Run time (seconds)
GA	21
Min-CCV	22
Min-V	23
GA-FSA	19

5.5 *Statistical Analysis of Variance (ANOVA) Results*

ANOVA [43] was employed to test the difference in terms of means between two or more cases. SPSS software was used to conduct the statistical analysis. The null hypothesis, which was presumed to be the means of the four populations, was equal. We demonstrated, mathematically, that H_0 was $\mu_1 = \mu_2 = \mu_3$. In the alternative hypothesis, we presumed that at least one of the means differed from the others. We conducted ANOVA for the synthetic datasets with $\alpha = 0.05$. Table 6 lists the results of the ANOVA test.

Table 7 demonstrates that the F-value > F was critical for the synthetic dataset, indicating that we rejected the null hypothesis. This inferred that the means of the population of the four different datasets were not equal because the *p*-value was considerably less than 0.5. Consequently, we can conclude that the performance achievement gained by GA-FSA against GA, Min-CCV, and Min-V was not by chance.

Table 7: ANOVA test results for GA-FSA, GA, Min-CCV, and Min-V for synthetic datasets

Source of variation	df	Sum of square	Mean square	F-value	*p*-value	F critical
Between groups	3	58256.05	19455.01	3.8743	0.0101	2.6497
Within groups	186	985562.98	5132.74			
Total	189	1043819.03				

6 Conclusion and Future Direction

The proposed approach, GA-FSA, outperformed the other state-of-the-art approaches (GA, Min-CCV, and min-V) with respect to different parameters (makespan, TGR, and cost). The objective of this study was to minimize cost while satisfying the QoS, which was achieved by the proposed approach. For latency-sensitive tasks (where the deadline of a task matters), the proposed approach demonstrated improved performance by reducing the different costs associated with scheduling tasks in a fog-cloud environment. In the future, we would like to study load-balancing and security issues while scheduling tasks. Real-world fog-cloud-based deployment of tasks should be studied and the efficacy of the proposed approach evaluated.

Acknowledgement: Not applicable.

Funding Statement: This study is supported via funding from Prince Sattam bin Abdulaziz University Project Number (PSAU/2024/R/1445).

Author Contributions: Study conception and design: Abdulelah Alwabel and Chinmaya Swain; analysis and interpretation of results: Chinmaya Swain; draft manuscript preparation: Abdulelah Alwabel. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data available on request from the authors.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. P. Singh, A. Nayyar, R. Kumar, and A. Sharma, "Fog computing: From architecture to edge computing and big data processing," *J. Supercomput.*, vol. 75, no. 4, pp. 2070–2105, 2019. doi: 10.1007/s11227-018-2701-2.

[2] P. Bellavista, J. Berrocal, A. Corradi, S. K. Das, L. Foschini and A. Zanni, "A survey on fog computing for the Internet of Things," *Pervasive Mob. Comput.*, vol. 52, pp. 71–99, 2019. doi: 10.1016/j.pmcj.2018.12.007.

[3] A. V. Dastjerdi and R. Buyya, "Fog computing: Helping the internet of things realize its potential," *Computer*, vol. 49, no. 8, pp. 112–116, Aug. 2016. doi: 10.1109/MC.2016.245.

[4] S. Jošilo and G. Dán, "Decentralized algorithm for randomized task allocation in fog computing systems," *IEEE/ACM Trans. Netw.*, vol. 27, no. 1, pp. 85–97, 2019. doi: 10.1109/TNET.2018.2880874.

[5] A. Samanta, Y. Li, and F. Esposito, "Battle of microservices: Towards latency-optimal heuristic scheduling for edge computing," in *Proc. 2019 IEEE Conf. Netw. Softw.: Unleash. Pow. Netw. Softw.*, pp. 223–227.

[6] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet Things J.*, vol. 3, no. 6, pp. 1171–1181, 2016. doi: 10.1109/JIOT.2016.2565516.

[7] R. K. Naha *et al.*, "Fog computing: Survey of trends, architectures, requirements, and research directions," *IEEE Access*, vol. 6, pp. 47980–48009, 2018. doi: 10.1109/ACCESS.2018.2866491.

[8] M. Taneja and A. Davy, "Resource aware placement of IoT application modules in fog-cloud computing Paradigm," in *2017 IFIP/IEEE Symp. Int. Netw. Service Manag. (IM)*, 2017, pp. 1222–1228.

[9] Z. Pooranian, M. Shojafar, P. G. V. Naranjo, L. Chiaraviglio, and M. Conti, "A novel distributed fog-based networked architecture to preserve energy in fog data centers," in *Proc.14th IEEE Int. Conf. Mob. Ad Hoc Sens. Syst.*, 2017, pp. 604–609.

[10] L. F. Bittencourt, J. Diaz-Montes, R. Buyya, O. F. Rana, and M. Parashar, "Mobility-aware application scheduling in fog computing," *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 26–35, Mar. 2017. doi: 10.1109/MCC.2017.27.

[11] O. Skarlat, M. Nardelli, S. Schulte, and S. Dustdar, "Towards QoS-aware fog service placement," in *2017 IEEE 1st Int. Conf. Fog Edge Comput. (ICFEC)*, 2017, pp. 89–96.

[12] R. Mahmud, K. Ramamohanarao, and R. Buyya, "Latency-aware application module management for fog computing environments," *ACM Trans. Internet Technol.*, vol. 19, no. 1, pp. 1–21, Feb. 2019. doi: 10.1145/3186592.

[13] 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. Exp.*, vol. 47, no. 9, pp. 1275–1296, Sep. 2017.

[14] R. Mahmud, S. N. Srirama, K. Ramamohanarao, and R. Buyya, "Quality of Experience (QoE)-aware placement of applications in Fog computing environments," *J. Parallel Distr. Comput.*, vol. 132, pp. 190–203, Oct. 2019. doi: 10.1016/j.jpdc.2018.03.004.

[15] 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. doi: 10.1109/ACCESS.2020.3003249.

[16] F. Saeik *et al.*, "Task offloading in edge and cloud computing: A survey on mathematical, artificial intelligence and control theory solutions," *Comput. Netw.*, vol. 195, no. 3, pp. 108177, Aug. 2021. doi: 10.1016/j.comnet.2021.108177.

[17] A. Brogi, S. Forti, C. Guerrero, and I. Lera, "Meet genetic algorithms in monte carlo: Optimised placement of multi-service applications in the fog," in *2019 IEEE Int. Conf. Edge Comput. (EDGE)*, 2019, pp. 13–17.

[18] T. S. Nikoui, A. Balador, A. M. Rahmani, and Z. Bakhshi, "Cost-aware task scheduling in fog-cloud environment," in *2020 CSI/CPSSI Int. Symp. Real-Time Embed. Syst. Technol. (RTEST)*, 2020, pp. 1–8.

[19] M. Louail, M. Esseghir, and L. Merghem-Boulahia, "Dynamic task scheduling for fog nodes based on deadline constraints and task frequency for smart factories," in *Proc. 11th Int. Conf. Netw. Future*, 2020, pp. 16–22.

[20] F. Faticanti, F. De Pellegrini, D. Siracusa, D. Santoro, and S. Cretti, "Throughput-aware partitioning and placement of applications in fog computing," *IEEE Trans. Netw. Serv. Manag.*, vol. 17, no. 4, pp. 2436–2450, 2020. doi: 10.1109/TNSM.2020.3023011.

[21] A. V. Postoaca, C. Negru, and F. Pop, "Deadline-aware scheduling in cloud-fog-edge systems," in *Proc. 20th IEEE/ACM Int. Symp. Cluster, Cloud Int. Comput.*, 2020, pp. 691–698.

[22] R. Mahmud, A. N. Toosi, K. Ramamohanarao, and R. Buyya, "Context-aware placement of industry 4.0 applications in fog computing environments," *IEEE Trans. Ind. Inform.*, vol. 16, no. 11, pp. 7004–7013, Nov. 2020. doi: 10.1109/TII.2019.2952412.

[23] M. Salimian, M. Ghobaei-Arani, and A. Shahidinejad, "Toward an autonomic approach for Internet of Things service placement using gray wolf optimization in the fog computing environment," *Softw. Pract. Exp.*, vol. 51, no. 8, pp. 1745–1772, Aug. 2021.

[24] T. Lähderanta *et al.*, "Edge computing server placement with capacitated location allocation," *J. Parallel Distr. Comput.*, vol. 153, no. 150055, pp. 130–149, Jul. 2021. doi: 10.1016/j.jpdc.2021.03.007.

[25] N. Godinho, H. Silva, M. Curado, and L. Paquete, "A reconfigurable resource management framework for fog environments," *Future Gener. Comput. Syst.*, vol. 133, no. 99, pp. 124–140, 2022. doi: 10.1016/j.future.2022.03.015.

[26] E. Batista, G. Figueiredo, and C. Prazeres, "Load balancing between fog and cloud in fog of things based platforms through software-defined networking," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 9, pp. 7111–7125, Oct. 2022.

[27] E. Batista, G. Figueiredo, M. Peixoto, M. Serrano, and C. Prazeres, "Load balancing in the fog of things platforms through software-defined networking," in *2018 IEEE Int. Conf. Int. Things (iThings) IEEE Green Comput. Commun. (GreenCom) IEEE Cyber, Phy. Soc. Comput. (CPSCom) and IEEE Smart Data (SmartData)* Halifax, NS, Canada, Feb. 2019, pp. 1785–1791.

[28] Y. Zhang, F. Zhang, S. Tong, and A. Rezaeipanah, "A dynamic planning model for deploying service functions chain in fog-cloud computing," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 34, no. 10, pp. 7948–7960, Nov. 2022.

[29] X. Gao, R. Liu, and A. Kaushik, "Virtual network function placement in satellite edge computing with a potential game approach," *IEEE Trans. Netw. Serv. Manag.*, vol. 19, no. 2, pp. 1243–1259, Jun. 2022. doi: 10.1109/TNSM.2022.3141165.

[30] H. Xu *et al.*, "Dynamic SFC placement scheme with parallelized SFCs and reuse of initialized VNFs: An A3C-based DRL approach," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 35, no. 6, pp. 101577, Jun. 2023. doi: 10.1016/j.jksuci.2023.101577.

[31] Z. Zhang, H. Sun, and H. Abutuqayqah, "An efficient and autonomous scheme for solving IoT service placement problem using the improved Archimedes optimization algorithm," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 35, no. 3, pp. 157–175, Mar. 2023. doi: 10.1016/j.jksuci.2023.02.015.

[32] A. Alwabel and C. K. Swain, "Deadline and energy-aware application module placement in fog-cloud systems," *IEEE Access*, vol. 12, pp. 5284–5294, 2024. doi: 10.1109/ACCESS.2024.3350171.

[33] C. K. Swain, B. Gupta, and A. Sahu, "Constraint aware profit maximization scheduling of tasks in heterogeneous datacenters," *Computing*, vol. 102, no. 10, pp. 2229–2255, Oct. 2020. doi: 10.1007/s00607-020-00838-1.

[34] X. Q. Pham, N. D. Man, N. D. T. Tri, N. Q. Thai, and E. Huh, "A cost- and performance-effective approach for task scheduling based on collaboration between cloud and fog computing," *Int. J. Distrib. Sens. Netw.*, vol. 13, no. 11, pp. 155014771774207, Nov. 2017. doi: 10.1177/1550147717742073.

[35] B. M. Nguyen, H. Thi Thanh Binh, T. The Anh, and D. Bao Son, "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment," *Appl. Sci.*, vol. 9, no. 9, pp. 1730, Apr. 2019.

[36] X. Y. Yuan, H. Y. Tang, Y. Li, T. Jia, T. C. Liu, and Z. H. Wu, "A competitive penalty model for availability based cloud SLA," in *2015 IEEE 8th Int. Conf. Cloud Comput.*, 2015, pp. 964–970.

[37] A. Uchechukwu, K. Li, and Y. Shen, "Improving cloud computing energy efficiency," in *2012 IEEE Asia Pacific Cloud Comput. Cong. (APCloudCC)*, 2012, pp. 53–58.

[38] C. Su, Y. Gang, and C. Jin, "Genetic algorithm based edge computing scheduling strategy," in *2021 4th Int. Conf. Data Sci. Inform. Technol.*, 2021, pp. 130–134.

[39] Z. H. Wang and J. H. Liu, "Flamingo search algorithm: A new swarm intelligence optimization algorithm," *IEEE Access*, vol. 9, pp. 88564–88582, 2021. doi: 10.1109/ACCESS.2021.3090512.

[40] C. K. Swain and A. Sahu, "Interference aware workload scheduling for latency sensitive tasks in cloud environment," *Computing*, vol. 104, no. 4, pp. 925–950, Apr. 2022. doi: 10.1007/s00607-021-01014-9.

[41] B. V. Natesha and R. M. R. Guddeti, "Adopting elitism-based genetic algorithm for minimizing multi-objective problems of IoT service placement in fog computing environment," *J. Netw. Comput. Appl.*, vol. 178, no. 5, pp. 102972, Mar. 2021. doi: 10.1016/j.jnca.2020.102972.

[42] F. Hoseiny, S. Azizi, M. Shojafar, and R. Tafazolli, "Joint QoS-aware and cost-efficient task scheduling for fog-cloud resources in a volunteer computing system," *ACM Trans. Internet Technol.*, vol. 21, no. 4, pp. 1–21, Nov. 2021.

[43] K. E. Muller and B. A. Fetterman, "Testing hypothesis in multiple regression," in *Regression and ANOVA: An Integrated Approach Using SAS Software*, 1st ed. Cary, NC, USA: John Wiley & Sons, Inc., 2003, vol. 5, pp. 70–92.