Contemporary Mathematics
https://ojs.wiserpub.com/index.php/CM/

Research Article

R2A-DSA: A Robust Resource Allocation Approach for Delay-Stringent Applications in IoT-Fog-Cloud Networks

Kunam Subba Reddy[1], Zakir Hussain Ahmed[2*], Yusliza Yusoff[3], Divya Nimma[4], Ismail Zahraddeen Yakubu[5,6], Ibrahim Aldayel[2], M. Rajasekaran[7], Harish Garg[8]

[1]Department of CSE, Rajeev Gandhi Memorial College of Engineering & Technology, Nandya, AP, 518501, India
[2]Department of Mathematics and Statistics, College of Science, Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, 11432, Saudi Arabia
[3]Faculty of Computing, Universiti Teknologi Malaysia, Johor Bahru, Johor, 81310, Malaysia
[4]Computer Science & Digital Technologies, Grambling State University, 403 Main Street, Grambling, LA, 71245, USA
[5]Department of Computer Science, the Federal Polytechnic Bauchi, Bauchi, 740101, Nigeria
[6]Department of Mathematics, Saveetha School of Engineering, SIMATS, Saveetha University, Chennai, Tamil Nadu, 602105, India
[7]Faculty of Engineering and Technology, Annamalai University, Chidambaram, Tamilnadu, 608002, India
[8]Department of Mathematics, Thapar Institute of Engineering and Technology (Deemed University), Patiala, Punjab, 147004, India
E-mail: zaahmed@imamu.edu.sa

Received: 31 July 2025; **Revised:** 9 October 2025; **Accepted:** 13 October 2025

Abstract: Now-a-days, the exponential growth of Internet of Things (IoT) applications has intensified the demand for efficient resource allocation strategies to meet critical Quality of Service (QoS) requirements, especially in terms of latency and execution time. While cloud computing offers vast computational resources, centralized processing introduces significant delays, making it unsuitable for time-sensitive tasks. Fog computing mitigates this by enabling edge-level processing, but its limited resources necessitate a hybrid IoT-fog-cloud architecture. This paper proposes Robust Resource Allocation-Delay Stringent Application (R2A-DSA), a novel Robust Resource Allocation approach that integrates task classification, prioritized queuing, and Secretary Bird Optimization Algorithm (SBOA) for dynamically optimizing resource distribution across fog and cloud layers. Our framework categorizes tasks based on their characteristics and priorities, then dynamically allocates resources across fog and cloud layers to decrease latency and increase whole system efficiency. Inspired by the hunting techniques of secretary bird, the SBOA is used to balance exploration and exploitation during resource allocation, achieving optimal configurations even with fluctuating workloads and resource availability. Extensive experiments on the iFogSim simulator demonstrate that R2A-DSA reduces execution time by 32%-45%, cuts task delays by 28%-40% and lowers task failure rates by 50%-65% compared to traditional methods (First Come First Serve (FCFS), Random, Cloud Only) and metaheuristic benchmarks (Harris Hawks Optimization (HHO), Crayfish Optimization Algorithm (COA), Fennec Fox Optimization Algorithm (FFA)). These results highlight R2A-DSA's superiority in enhancing system efficiency, scalability, and reliability for latency-sensitive IoT applications, offering a significant advancement in distributed computing resource management.

Keywords: fog computing, cloud computing, Internet of Things (IoT), resource allocation, Secretary Bird Optimization Algorithm (SBOA), task classification

MSC: 68W25, 91B32

DOI: https://doi.org/10.37256/cm.7120268079

1. Introduction

The advent of diverse Internet of Things (IoT) tools has significantly transformed information and communication technology. It has broadened internet uses to a variety of smart tools that include televisions, wearable devices, security cameras, and automobiles, besides the conventional devices, for example, tablets and smartphones. The IoT business is rapidly growing, by 2025, it is expected to interconnect 75 billion devices [1]. As a result, there is an increase in IoT applications that produce vast amounts of data with stringent potential demands [2]. The International Data Corporation (IDC) predicts that, by 2025, IoT devise will create 73.1 ZB data [3]. Additionally, the advancement of these smart devices increases the demand for various computing standards, for instance, fog, edge, and cloud computing, as well as a range of communication protocols to manage their interactions [4]. Cloud computing stands out as a promising technology that offers vast processing and storage capabilities, that make it appropriate for handling and storing these huge data volumes produced by the IoT devices. However, the significant distance between cloud and IoT devices can lead to longer response times and increased latency for IoT applications when transferring large volumes and diverse data types to unified cloud data centers [5, 6].

Fog computing was developed to address the limitations of unified cloud data centers and fulfil the potential demands of the IoT applications. Architecturally, fog computing covers cloud capacities to the network edge, offering services near the source of IoT data generation [7]. This proximity allows for processing and analyzing time-sensitive applications with reduced response times and minimized bandwidth usage [8–10]. However, in comparison to cloud resources, fog devices generally have limited computational and storage capacities [11–13]. On the other hand, cloud resources offer almost limitless computational capacity but come with more communication and financial costs. Consequently, neither fog computing nor cloud computing alone can fulfil the demands of huge data IoT applications due to their respective capacities and limitations [12–15]. To create an effective computing architecture, a combination of both models is necessary. Efficiently managing cloud and fog resources for IoT applications leads to a difficult challenge, further compounded by the loosely connected and highly heterogeneous nature of fog devices [16–19]. Therefore, an effective IoT-fog-cloud architecture requires advanced resource allocation strategy that accounts for both fog and cloud considerations [20].

In this study, we propose an innovative resource allocation method that integrates task classification with prioritized queuing and the Secretary Bird Optimization Algorithm (SBOA). Our approach categorizes tasks based on their characteristics and priorities, and then dynamically allocates resources across fog and cloud layers to decrease latency and increases system efficiency. Inspired by the hunting techniques of secretary bird, the SBOA is used to balance exploration and exploitation during resource allocation, achieving optimal configurations even with fluctuating workloads and resource availability. The experimental findings highlight the effectiveness of this integrated approach, demonstrating significant advancements over conventional resource allocation techniques. The proposed system improves computational efficiency and supports robustness, proving to be a practical solution for the resource-intensive and dynamic demands of the IoT-fog-cloud computing atmospheres. The aim and objectives of this study are prepared as follows.

• To develop a task classification with prioritized queuing that categorize tasks based on their characteristics and priorities for distribution across fog and cloud resources.

• To develop a metaheuristic resource allocation based on SBOA that dynamically allocates resources across cloud and fog layers.

• To decrease the system latency, execution time, and improve whole system efficiency.

The rest of this manuscript is structured as follows: Section 2 provides a summary of state-of-the-art approaches addressing resource allocation issues in fog-cloud architecture. Section 3 details the proposed resource allocation method discussed in this paper. Section 4 presents the experiment in detail and our results. Section 5 discusses the limitations of our proposed method while Section 6 winds up with an outline of our research findings.

2. Related work

In [21], a novel 5G Internet of Vehicle (IoV) based on Fog-Cloud environment and Software Defined Networking (SDN) is proposed. It considers first the service requirements of the intelligent transportation system and then constructs a model of some objectives and then applies multiple optimization algorithms to optimize the system performance. The reported experimental results prove its effectiveness over state-of-the-art methods. In [22], green energy concept is applied for minimizing the level of energy consumption and latency in multi-sensorial frameworks in IoT systems. To process large number of user requests as well as the corresponding quality and security limitations, Genetic Algorithm (GA) was employed. The proposed approach was simulated and benchmarked with other baseline approaches, which proves that the proposed approach is more promising. In [23], a fuzzy logic-based task offloading method is proposed for IoT applications with uncertain parameters. The proposed approach aimed at optimizing the applications agreement index and robustness. For learning and optimizing the proposed fuzzy offloading technique for the diverse applications, it used a multi-objective estimation of distribution algorithm. Applications are grouped into clusters by the algorithm, which are mapped to corresponding tier for execution. A scheduling decision is made within a reduced search space, thus saving the system resources [24]. In [25], a resource scheduling approach is proposed that aims to execute the highest possible number of requests considering their deadlines. The proposed approach was deployed on the three-tier fog computing paradigm. To minimize the tasks deadline failures, an optimization scheme based on mixed integer programming is presented, which is validated using exact solution approach. GA is then applied to obtain an optimal solution. In [26], an enhanced Whale Optimization Algorithm (WOA) is proposed by introducing an opposition-based chaotic approach to the WOA algorithm. The aim of the proposed Opposition-based Chaotic Whale Optimization Algorithm (OppoCWOA) therein was to meet the challenges faced during scheduling in distributed fog computing. Task scheduling problem in distributed fog paradigm is modeled like a linear programming problem, with focus on reducing power consumption and execution time of the distributed fog paradigm. The approach showed potential benefits in terms of time efficiency and system power utilization. However, the system they proposed did not account for task scheduling in the cloud, potentially leading to an overload in the fog layer [27]. In [28], a fog computing system specifically designed for smart workshops is developed and implemented packed smart workshop applications using Kubernetes. Originally, based on GA, an enhanced Interval Division GA was introduced for task allocation and scheduling in fog computing environments. Later, an Interval Division Genetic Scheduling Algorithm (IDGSA) was introduced by incorporating a penalty factor. In [29], Expectation Maximization (EM) clustering to group tasks from IoT devices as per their priority and deadlines, is applied with the goal of reducing computational complexity and bandwidth overhead. A multi-constraint method is used for resource allocation, which led to the development of an improved heap scheduler that adheres to Quality of Service (QoS) and Service Level Agreements (SLA) demands [30]. In [31], real-time services are provided for latency-sensitive applications through the fog layer. For load balancing across a network of fog computing nodes, Queue-based job scheduling is implemented and utilized by the Ford-Fulkerson approach, thus, enhancing task scheduling and load distribution in fog edge computing networks.

3. Materials and methods

The modeling of the fog-cloud system, the problem formulation, and the stages in the proposed resource allocation approach are presented in this section. The system model defines the components of the assumed IoT-fog-cloud system and their interactions. The problem formulation and resource allocation define how the problem is modeled for the QoS parameters and the allocation of fog-cloud resources to user requests to satisfy their QoS demands and ensure optimal system performance.

3.1 *System model*

The system model in the work considers an IoT-fog-cloud environment. It is a hierarchical model designed to effectively handle and process data from IoT devices. The model includes three primary layers: edge layer, fog computing, and cloud computing as depicted in Figure 1.

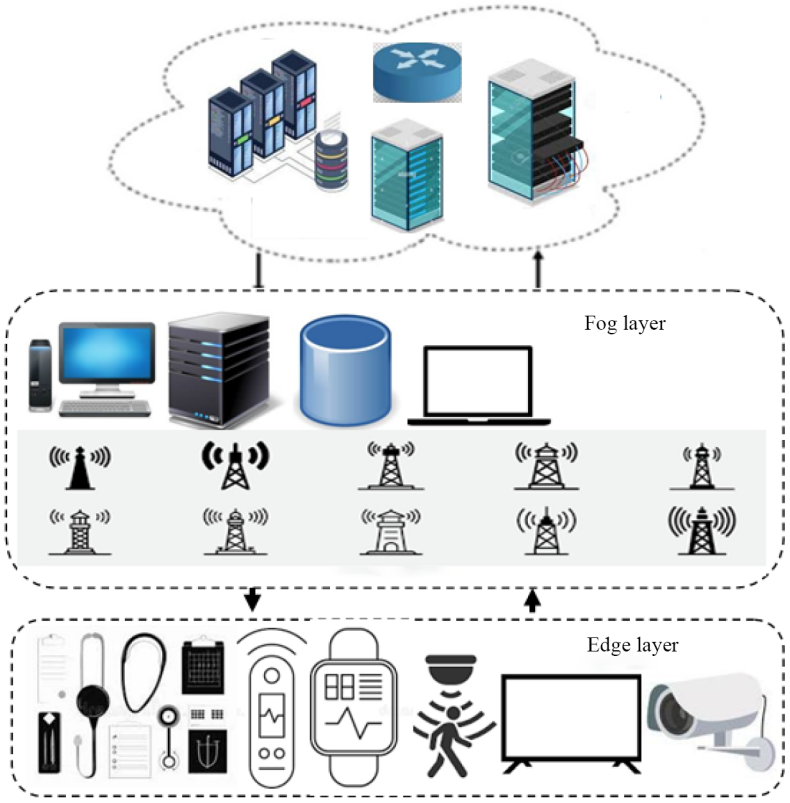

Figure 1. IoT-fog-cloud system

Each layer has distinct roles and duties, facilitating optimized data processing, decreased latency, and enhanced resource management. The edge layer contains edge devices which produce data. They include sensors, actuators, smart devices, wearables and other connected objects. They collect and transmit data to the middle layer called Fog layer to further process and pass to the cloud if required. The distributed fog computing layer contains intermediary devices or servers called fog nodes, located closer to the network edge. The fog computing layer delivers localized processing, storage, and networking services. The cloud layer consists of large-scale datacenters providing vast computational power, storage, and advanced data processing capabilities. By utilizing the strengths of each layer, the IoT-fog-cloud architecture offers a strong framework for handling the intricate and ever-changing characteristics of IoT ecosystems. The fog layer consists of base stations that connect the edge devices with the fog computing devices and the cloud data-center resources. Tasks generated by the edge devices are passed to the base station for layer selection. The system maintains two separate queues (Priority and Non-priority Queue) in each layer, at the time of task classification and queuing of tasks at a given layer. The time sensitive tasks are placed in the fog queue and time insensitive tasks are placed in the cloud queue. Both cloud and fog layer contain a resource allocator which allocates resources to the tasks in the layer queue.

3.1.1 *Task classification and queuing model*

In this section, the classification and placement of user tasks in a layer queue for processing is presented. The base stations at the fog layer place tasks into a queue based on the QoS demands of the tasks and the computation capacity of the existing resources in the respective layers. Suppose F be a set that contains m fog nodes in the fog computing layer, $F = \{f_1, f_2, \ldots, f_m\}$. For $i \, \varepsilon \, (1, m)$, f_i is associated with three attributes represented as $< f_i^{cpu}, f_i^{mem}, f_i^{bw} >$, which represents the node processing speed in Millions of Instructions Per Second (MIPS), memory capacity in Mega Bytes (MB) and bandwidth in Megabits per second (Mbps). Let V be a set that contains n virtual devices in the cloud computing layer, $V = \{v_1, v_2, \ldots, v_n\}$. For $i \, \varepsilon \, (1, n)$, v_i is associated with three attributes represented as $< v_i^{cpu}, v_i^{mem}, v_i^{bw} >$, which represents the virtual machine processing speed in MIPS, memory capacity in MB and bandwidth in Mbps. Assumed a set T that contains k tasks, $T = \{t_1, t_2, \ldots, t_k\}$ submitted to the system by the edge devices. For $i \, \varepsilon \, (1, k)$, t_i is associated with three attributes $< t_i^{len}, t_i^{mb}, t_i^{dl} >$ representing the tasks length, volume of task data to be processed and deadline. When a task reaches at the base position at the fog computing layer, the base position computes the possibility of executing the task on the cloud by considering the task deadline and processing requirement. If the tasks are processed by the cloud within its deadline, the task is classified as cloud and its queuing priority is determined and placed in the appropriate queue. If the cloud cannot meet up with the task deadline, the base station classifies the tasks as fog and its queuing priority is determined and placed in the appropriate queue.

This classification rule reflects realistic conditions under which the fog can process a task on time while the cloud cannot. Although cloud servers offer higher raw computational power, their physical remoteness introduces significant transmission delays, queuing delays, and additional system overhead. In contrast, fog nodes are deployed closer to IoT devices, providing higher effective bandwidth and lower propagation latency. As a result, when (i) the task size is relatively small, (ii) the fog node has sufficient available CPU and memory resources, and (iii) the communication and overhead costs of offloading to the cloud outweigh its computational benefits, the fog layer can complete the task within its deadline while the cloud cannot. These conditions commonly occur in latency-sensitive domains such as healthcare monitoring, vehicular applications, and industrial IoT [32].

To classify a task as a cloud or fog, the cloud computing layer is considered as a single node with a certain capacity. The cloud capacity can be calculated using Equation (1) below:

$$\lambda_c = \frac{1}{n} \times 0.5 \times \sum_{i=1}^{n} v_i^{cpu} + 0.5 \times \sum_{i=1}^{n} v_i^{bw}. \tag{1}$$

Equally, the fog capacity can be calculated using Equation (2) below:

$$\lambda_f = \frac{1}{m} \times 0.5 \times \sum_{i=1}^{n} f_i^{cpu} + 0.5 \times \sum_{i=1}^{n} f_i^{bw}. \tag{2}$$

For each task t_i, the time of executing the task on cloud is calculated as by Equation (3).

$$P_{t_i}^{C} = \frac{1}{\lambda_c} \times t_i^{len} + t_i^{mb} + \sigma_c, \tag{3}$$

where σ_c is the cloud system overhead, t_i^{len} is the length of the task in Millions of Instructions (MI), t_i^{mb} is the data volume to be propagated to the cloud and λ_c is the cloud capacity.

The execution time for each task on a fog layer ($P_{t_i}^{f}$) is computed by replacing the cloud capacity in Equation (3) with fog layer capacity and the cloud layer system overhead with fog layer system overhead (σ_f), which represent additional

delays introduced by communication protocol processing, virtualization, queuing, and control signaling In this work, the cloud layer overhead and fog layer overhead are set to 15 ms and 5 ms respectively.

The class of task t_i is determined based on Equation (4) below:

$$C_{t_i} = \begin{cases} \text{Cloud}, & \text{if } P_{t_i}^C \leq t_i^{dl} \\ \text{fog}, & \text{Otherwise.} \end{cases} \tag{4}$$

When a task class is determined, the base station determines the priority of the task and places the task in an appropriate queue of its class layer. A task is considered as non-priority task, if its deadline is greater than $P_{t_i}^C$ and classified as priority task if otherwise. The classification and queuing algorithm is represented in Algorithm 1.

Algorithm 1 Classification and Queuing Algorithm

Input: Task set T, set of fog nodes F and set of virtual devices V.

Output: Task class and task Queue

for $i = 1$ to len(T) **do**
  Compute $P_{t_i}^C$ of task i using equation (3)
  if $P_{t_i}^C \leq t_i^{d_l}$ **then**
    $C_{t_i} =$ Cloud
    if $P_{t_i}^C < t_i^{d_l}$ **then**
      Place t_i in cloud non-priority queue Q_N^C
    else
      Place t_i in cloud priority queue Q_P^C
    end if
  else
    $C_{t_i} =$ Fog
    Compute $P_{t_i}^f$ as in equation (3)
    if $P_{t_i}^f < t_i^{d_l}$ **then**
      Place t_i in fog non-priority queue Q_N^f
    else
      Place t_i in fog priority queue Q_P^f
    end if
  end if
end for.

The priority and non-priority queue in the cloud layer are combined into a single queue with non-priority tasks queued behind the priority tasks. Also, the two queues in the fog layer are combined to form a single queue with the priority tasks occupying the front end of the queue. The cloud and fog layers queue are represented by Equations (5) and (6) respectively, as follows:

$$Q_c = Q_N^C + Q_P^C \tag{5}$$

$$Q_f = Q_N^f + Q_P^f. \tag{6}$$

The resource allocator in each of the layers allocates available layer resources to the tasks in its layer queue considering the QoS requirements of the tasks.

3.2 *Problem formulation*

In this segment, the resource allocation problem is modeled as a function of execution time and latency, which the system intends to minimize through efficient resource allocation.

3.2.1 *Task execution time*

Given a set of tasks T_s to be implemented on a resource in a given layer, each task t_i belonging to T_s for i in $\{1, 2, 3, \ldots s\}$ is associated with three attributes $< t_i^{len}, t_i^{mb}, t_i^{dl} >$ representing the tasks length (MI), volume of task data to be processed (MB) and deadline (s). Given a set of computation resources R_c on a particular layer, each resource r_j belonging to R_c is associated with three attributes represented as $< r_j^{cpu}, r_j^{mem}, r_j^{bw} >$, which represents the resource processing speed (MIPS), memory capacity (MB) and bandwidth (Mbps). The execution time of a group of tasks T_k from T_s, on a resource r_j is total time taken by resource r_j to fulfil the performance of T_k, which can be calculated using Equation (7) below:

$$E_{T_k}^t = \frac{\sum_{i=1}^{k} t_i}{r_j^{cpu}}. \tag{7}$$

3.2.2 *Task delay model*

While a task is generated and submitted to the system through the base station, the task experiences some delay due to propagation from the base station to the cloud or fog nodes. Also, when there is no available resource to progress the task on the fly, the task is placed in the waiting queue until a resource is freed by a task in execution. When a task is allocated a resource with low processing ability, it experiences an increase in its execution time. On the contrary, a task allocated to a resource with higher processing capacity may benefit from the low execution of the resource. To model the delay experienced by a task, three delay types (waiting, transmission and execution delay) are considered. Given a set of tasks T_k to be implemented by resources in a layer, the waiting delay of a task t_i from T_k is the difference between the start time and the arrival time of the task t_i, which can be calculated using Equation (8).

$$D_{t_i}^w = t_i^s - t_i^a, \tag{8}$$

where t_i^s and t_i^a represents the start time and arrival time of task t_i.

When the class and priority of a task is determined by the base station, the task is propagated to the layer to process. The transmission delay experienced by task classified as fog is minimal compared to task classified as cloud caused by the closeness of the base stations to the users and fog nodes. The transmission delay of task t_i from a set of task T_k can be calculated using Equation (9).

$$D_{t_i}^t = \frac{t_i^{mb}}{\lambda_l^{bw}} + \sigma_l, \tag{9}$$

where λ_l^{bw} is the transmission capacity of the layer and σ_l is network overhead factor of the layer. λ_l^{bw} is calculated using Equation (10).

$$\lambda_l^{bw} = \frac{1}{n} \times \sum_{i=1}^{n} r_i^{bw}, \tag{10}$$

while a task is allocated a resource r_j in its layer, the tasks may experience a delay if the processing capacity of the resource doesn't match its processing requirements. The execution delay experienced by a task t_i from a set of tasks T_k can be calculated using Equation (11).

$$D_{t_i}^{E} = \frac{t_i^{len}}{r_j^{cpu}}. \tag{11}$$

Based on the three-delay experienced by a task from its submission to the system and return to the edge devices, the task delay is modeled in Equation (12) below.

$$t_{total}^{D} = D_{t_i}^{w} + 2 \times D_{t_i}^{t} + D_{t_i}^{E}. \tag{12}$$

The goal of the resource allocation is to minimize the run time and overall delay experienced by tasks submitted to the system for processing. Thus, the problem is presented as a multi-objective function to be minimized. The objective function is modeled in Equation (13) below.

$$f(x) = \min\left(\alpha \times \sum_{i=1}^{k} E_i^t + \beta \times \sum_{i=1}^{k} t_{total_i}^{D}\right), \tag{13}$$

where α and β represents the weights of the QoS parameters. In this work, we assume equal weights of 0.5 for both the execution time and total delay experienced by task.

3.3 *Resource allocation*

The resource allocation to a task set within a layer corresponding to their class is done by a resource allocation manager of that layer. To allocate resources to tasks, the allocation manager employs the SBOA to generate an optimal task to resource allocation that minimizes the task execution time and delay. The SBOA is an optimization method proposed in [33]. The algorithm is encouraged by the exceptional hunting and problem-solving strategies of the secretary bird, a prominent terrestrial predator recognized for its striking look and efficient hunting skills. By emulating the bird's behavior, the SBOA addresses intricate optimization problems. This algorithm draws motivation from the distinctive hunting techniques of the secretary bird, which employs precise and forceful strikes to capture snakes and other prey. The bird's strategic hunting approach, combining exploration (locating prey) and exploitation (capturing prey), is adapted to enhance problem-solving in the algorithm. The phases in the SBOA are presented as follows.

3.3.1 *Initial preparation phase*

In this initial stage of the SBOA, X population of the secretary birds is determined. The location of individual secretary bird in the search space finds out the value of the decision variables in the objective function. In this case, the various positions of secretary birds that represent a set of candidate solutions to the problem under consideration [34]. The initial random location of the individual secretary bird of the population in the search space is determined by Equation (14).

$$X_{i,\,j} = lb_j + r \times (ub_j - lb_j),\ i = 1,\ 2,\ 3,\ \ldots\ N,\ j = 1,\ 2,\ 3,\ \ldots\ \text{Dim}, \tag{14}$$

where $X_{i,\,j}$ represent the location of i^{th} secretary bird in the search space, lb_j and ub_j denote the lower and upper bounds respectively, of the search space and r is an arbitrary value between 0 and 1.

In SBOA, optimization begins by generating a population of candidate solutions that are arbitrarily generated within the bound restrictions of the problem under consideration. In each iteration, the best solution obtained from the generated candidate solution is considered as the optimal solution. The candidate solution generated is represented by Equation (15).

$$X = \begin{bmatrix} X_{1,\,1} & \cdots & X_{1,\,j} & \cdots & X_{1,\,\mathrm{dim}} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ X_{i,\,1} & \cdots & X_{i,\,j} & \cdots & X_{i,\,\mathrm{dim}} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ X_{N,\,1} & \cdots & X_{N,\,j} & \cdots & X_{N,\,\mathrm{dim}} \end{bmatrix}, \tag{15}$$

where X represents the group of secretary birds, X_i denotes the ith bird, $X_{i,\,j}$ denotes the location of individual bird i in dimension j; N and dim denote the population size of the birds and the dimension of the problem under consideration.

The individual secretary bird in the population denotes a candidate solution to the optimization problem under consideration. The values proposed by the secretary birds are evaluated by the objective function of the problem and the resulting values compiled into a vector using Equation (16).

$$F = \begin{bmatrix} F_1 \\ : \\ F_i \\ . \\ . \\ F_N \end{bmatrix} = \begin{bmatrix} F(X_1) \\ : \\ F(X_i) \\ . \\ . \\ F(X_N) \end{bmatrix}, \tag{16}$$

where F_i represents the value returned by the objective function by evaluating the positions of the i^{th} bird and F is a vector of objective function values.

To determine the optimal candidate solution for a problem, the quality of the candidate solutions is compared and analyzed. In this work, the goal is to minimize the execution time and delay, thus the bird with the least objective function value is considered as the best candidate solution. The locations of the birds are updated in each iteration, and the optimal candidate solution is determined and updated.

3.3.2 *Update phase*

In SBOA, two distinct usual behaviors of the secretary birds are implemented to update the positions of the secretary birds. These two behaviors are the hunting and escape strategies of the secretary birds.

Hunting strategy of secretary bird (exploration phase)

The hunting strategies employed by the secretary birds are categorized into three phases: searching, consuming, and attacking prey. The hunting process of the birds is divided into three different times of equal interval. The time division depends on the genetic statistics of the secretary birds and the time duration of the individual phases. The timing corresponding to the three phases are given as follows:

$$t < \frac{1}{3}T, \quad \frac{1}{3}T\ t < \frac{2}{3}T, \quad \frac{2}{3}T\ t < T.$$

(i) Searching for prey

In searching phase, the secretary birds search for prey in the search space by using their incredible sharp vision to spot the prey and their extended legs to gradually sweep the grasses in the area paying attention during the process. Their necks and extended legs allow them to keep safe distances, minimizing the risk of snake attacks. This situation occurs during the early stages of optimization when exploration is vital. Consequently, a differential evolution strategy is employed at this phase. By leveraging differences between individuals, differential evolution creates new solutions, boosting the variety of the algorithm and improving its global search capabilities. Incorporating differential mutation operations enhances diversity, which helps prevent entrapment in local optima. This allows individuals to investigate several parts of the solution space, thereby increasing the likelihood of determining the global optimum. The position update of the secretary birds during searching is modeled in Equations (17) and (18) respectively.

$$\text{while } ^{t} < \frac{1}{3}T,\ x_{i,\ j}^{\text{new } P1} = x_{i,\ j} + \left(x_{\text{random }1} + x_{\text{random }2}\right) \times R_1 \tag{17}$$

$$x_i = \begin{cases} x_i^{\text{new},\ P1}, & \text{if } F_i^{\text{new},\ P1} < F_i \\ x_i, & \text{else} \end{cases}, \tag{18}$$

where t and T represents the existing and maximum iteration, $x_i^{\text{new},\ P1}$ denote the new state of the ith bird in the first phase, $x_{\text{random\_1}}$ and $x_{\text{random\_2}}$ are arbitrary candidate solutions during the first phase iteration and R_1 denotes an array of dimension 1 by Dim arbitrarily generated from values between 0 and 1. $x_{i,\ j}^{\text{new } P1}$ and $F_i^{\text{new},\ P1}$ denote the value of the jth dimension in the Dimensionality of the solution (Dim) and the fitness value of the objective function respectively.

(ii) Consuming prey

The prey consumption behavior is exhibited when the prey is spotted by the secretary bird. The secretary birds employ a distinctive hunting strategy by using its sharp footwork and maneuver round the prey. The bird remains stationary, watching the prey's every move from a tall perch. It employs its sharp perception of the prey's behavior to leap, hover, and gradually antagonize the prey, ultimately exhausting its stamina. The arbitrary movement of the birds is stimulated using Brownian motion as modeled in Equation (19).

$$\text{RB} = \text{Rand}_n\ (1,\ \text{Dim}), \tag{19}$$

where Rand_n (1, Dim) denotes an arbitrarily created array of dimension 1 by Dim from a normal distribution with Mean 0 and standard deviation 1.

This approach to combat from a distance grants the birds a notable physical benefit. Its extended legs cause it to be hard for the prey (snake) to wrap around it, and the bird's claws and legs are protected by abundant keratin scales, such as armor, which protect it from venomous snake bites. At this phase, the bird often pauses to fix its sharp gaze on the snake's position. In this case, the concept of x_{best} and Brownian motion is employed. By employing x_{best}, individuals can conduct local searches around the best locations that were previously discovered, thereby enhancing their exploration of the nearby solution space. This method not only prevents premature convergence to local optimal solutions but also speeds up the method's convergence to the optimal locations within the solution space. It allows individuals to search using global information together with their own historically best locations, thereby enhancing the likelihood of discovering the global optimal solution. By incorporating the randomness of Brownian motion, the individuals can search for the solution space further efficiently, reducing the risk of becoming stuck in local optimal solution and yielding improved results when tackling difficult problems. The location update of the birds throughout the consuming phase is modeled in Equations (20) and (20) respectively.

$$\text{while } \frac{1}{3}T < t < \frac{2}{3}T,\ x_{i,\ j}^{\text{new},\ P1} = x_{\text{best}} + e^{\left(\left(\frac{t}{T}\right)^4 \times (RB-0.5) \times \left(x_{\text{best}} - x_{i,\ j}\right)\right)} \tag{20}$$

$$x_i = \begin{cases} x_i^{\text{new},\ P1}, & \text{if } F_i^{\text{new},\ P1} < F_i \\ x_i, & \text{else} \end{cases}, \tag{21}$$

where x_{best} represents the current best value.

(iii) Attacking prey

Once the (prey) snake is fatigued, the bird identifies the perfect time, then quickly initiates its attack, relying on its powerful leg muscles. This phase often includes the bird's leg-kicking method, where it quickly lifts its legs and gives precise strikes with its sharp claws, frequently pointing the snake's head. Aim of the kicks is to disable or kill the snake swiftly, minimizing the risk of being bitten. The snake's vital points are hit by the sharp claws, leading to its death. In case the snake is very big to be killed instantly, the bird can take it into the sky and drop it, so that it falls onto the tough ground and perish. The location update of the birds throughout this phase is modeled by Equations (22) and (23) respectively.

$$\text{while } t > \frac{2}{3}T,\ x_{i,\ j}^{\text{new},\ P1} = x_{\text{best}} + \left(\left(1 - \frac{t}{T}\right)^{\left(2 \times \frac{t}{T}\right)}\right) \times x_{i,\ j} \times RL \tag{22}$$

$$x_i = \begin{cases} x_i^{\text{new},\ P1}, & \text{if } F_i^{\text{new},\ P1} < F_i \\ x_i, & \text{else} \end{cases}. \tag{23}$$

Where RL denotes a weighted levy flight, which is defined in Equation (24) below

$$RL = 0.5 \times \text{Levy}(\text{Dim}). \tag{24}$$

Levy(Dim) represents the levy flight distribution function, which can be computed using Equation (25)

$$Levy(D) = s \times \frac{u\ \times \sigma}{|v|^{\frac{1}{\eta}}}. \tag{25}$$

Here, s and η are constants values 0.01 and 0.5, u and v are arbitrary values between (0, 1), and σ is determined using Equation (26).

$$\sigma = \left[\frac{\Gamma(1+\eta) \times sin\left(\frac{\pi\eta}{2}\right)}{\Gamma\left(\frac{1+\eta}{2}\right) \times \eta\ \times 2\left(\frac{1-\eta}{2}\right)}\right]^{\frac{1}{\eta}}. \tag{26}$$

Γ represents gamma function and η assumed constant value 0.5.

Escape strategy of secretary bird (exploitation phase)

Secretary birds may be attacked due to their food being stolen by their enemies, for example, hawks, eagles, jackals, and foxes. The birds use some elusion approaches to protect themselves and their food. To evade the attack by their enemies, the secretary birds run at a very high speed or fly to a safer region. Also, the birds might use the structures and colors found in their surroundings to camouflage themselves, making it more difficult for the predators to spot them. The choice of elusion strategy by the secretary birds has equal probability. The evasion strategies are modeled by Equation (27).

$$x_{i,\,j}^{\text{new},\,P1} = \begin{cases} C_1 : x_{\text{best}} + (2 \times RB - 1) \times \left(1 - \dfrac{t}{T}\right)^2 \times x_{i,\,j}, & \text{if rand} < r_i \\ C_2 : x_{i,\,j} + R_2 \times (x_{\text{random}} - K \times x_{i,\,j}), & \text{else} \end{cases}, \tag{27}$$

where r is a constant with value of 0.5, R is an array of arbitrary values of dimension 1 by Dim from the standard normal distribution, x_{random} is an arbitrary candidate solution for the existing iteration, and K is a random integer, which is either 1 or 2. C_1 and C_2 represent camouflage by environment and fly or run evasion. The value of K is determined by Equation (28)

$$K = \text{round}\ (1 + \text{rand}\ (1,\ 1)), \tag{28}$$

rand (1, 1) represents an arbitrary generation of number between 0 and 1. The location update of the birds throughout this stage is modeled by Equation (29).

$$x_i = \begin{cases} x_i^{\text{new},\,P1}, & \text{if } F_i^{\text{new},\,P1} < F_i \\ x_i, & \text{else} \end{cases}. \tag{29}$$

The Pseudo code for the SBOA is presented in Algorithm 2.

Algorithm 2 Secretary Bird Optimization Algorithm

Initialize the optimization problem settings (Dim, Population_size (N), upper_bound (ub), lower_bound (lb), Maximum_iteration (T), current_iteration (t)).

Randomly initialize the Secretary Bird (SB) population

for $t = 1$ to T **do**

Update x_{best} of the secretary birds

for $I = 1$ to N **do**

Exploration:

if $t < \dfrac{1}{3}T$ **then**

Use equation (17) to compute the new status of ith SB

Use equation (18) to update the location of ith SB

else if $\dfrac{1}{3}T < t < \dfrac{2}{3}T$ **then**

Use equation (20) to compute the new status of ith SB

Use equation (21) to update the location of ith SB

else

Use equation (22) to compute the new status of ith SB

Use equation (23) to update the location of ith SB

end if

Exploitation:
if rand < 0.5 **then**
Use C_1 in equation (27) to compute the new status of ith SB
else
Use C_2 in equation (27) to compute the new status of ith SB
end if
Use equation (29) to update the location of ith SB
end for
Save the best candidate solution obtained
end for
Return best solution.

4. Experiment and results

To validate the performance of our proposed resource allocation strategy, the iFogSim simulator [35] is used to conduct experiments under various configuration settings. The IoT-fog-cloud resources are simulated by iFogSim toolkit, and the proposed resource allocation method is implemented and tested on these simulated resources. To build the IoT-fog-cloud architecture for implementing the resource allocation strategy, a set of fog and cloud resources with heterogeneous capabilities is first created. Next, the level or layer for each resource is defined, and connections are established between the resources to form the architecture. Each resource in a layer is characterized by a set of attributes that enable it to process, store, and transmit data across the system. Table 1 depicts the attributes of the fog and cloud resources used in implementing the IoT-fog-cloud environment.

Table 1. Resource attributes

Resource type	CPU (MIPS)	Memory (MB)	Bandwidth (Mbps)	Count
Fog node	500-2,000	512-2,048	512-2,048	20
Virtual machine	5,000-20,000	5,120-7,680	4,096-10,240	5

Table 1 depicts the attributes of resources and their counts. Each resource has processing, storage, and bandwidth capacities within the ranges specified in the table, which are randomly generated. In the fog layer, 20 resources are created, and 5 resources are created in the cloud layer.

To test the performance of the resource allocation strategy on the constructed architecture, user tasks are uniformly generated according to the even distribution of IoT devices at the edge of the IoT-fog-cloud network. Each task has a set of attributes, as shown in Table 2.

Table 2. Task attributes

Task type	Task length (MIPS)	Data size (MB)	Deadline (s)
Time sensitive (Type 1)	100-1,000	128-512	20-40
Time insensitive (Type 2)	1,000-3,000	1,024-2,048	150-400

Tasks generated by the IoT devices at the edge layer are propagated to the base stations in the fog layer. In our experiment, we considered four base stations, each with a processing capacity between 5,000 and 20,000, a storage capacity between 5,120 and 12,288, and bandwidth between 1,049 and 10,240, all generated randomly. When a task is received at a

base station, a processing layer is assigned to the task, and its priority is computed depend on the task's QoS requirements and the capabilities of the available resources within the layer. The SBOA is used to achieve optimal resource allocation, targeting to minimize latency, execution time, and the rate of task failures.

Table 2 shows the attributes of the tasks generated for processing by fog and cloud resources. The attributes include task length, data size, and deadline, with values falling within the ranges specified in the table. Two classes of tasks are created: time-sensitive and time-insensitive. Time-sensitive tasks have shorter task lengths, smaller data sizes, and shorter deadlines compared to time-insensitive tasks. For each class of task (task type), 500 tasks are created by the edge devices. During the simulation, the generated tasks are submitted to the system in batches with 100 tasks added up to the previous submission, and system performance is monitored. For each batch of generated tasks, the simulation is run 20 times, and the performance results are averaged.

The performance of our proposed resource allocation method is benchmarked against state-of-the-art procedures. The benchmarking is conducted in two ways. First, we compare our proposed classification and queuing model with conventional approaches such as First Come First Serve (FCFS), Random, and Cloud Only. In all cases, the SBOA is used to allocate resources to the generated tasks. However, the task distribution to the cloud and fog layers varies according to the method employed in the simulation. In FCFS, tasks that arrive first are allocated resources in the fog layer. If no resources are available in the fog layer, the residual tasks are propagated to the cloud for execution. In the fog-only method, tasks are executed solely in the fog layer, without utilizing the cloud layer. In the random approach, the layer for task execution is chosen at random. The second benchmarking approach uses the proposed task classification and queuing algorithm to allocate tasks between the fog and cloud layers. For resource allocation, algorithms such as Harris Hawks Optimization Algorithm (HHO) [36], Crayfish Optimization Algorithm (COA) [37], and Fennec Fox Optimization Algorithm (FFA) [38] are used and compared with the SBOA. The parameter settings of the HHO, COA, FFA, and SBOA for resource allocation are presented in Table 3.

Table 3. Parameter settings for HHO, COA, FFA, and SBOA

Algorithm	Parameter	Value
HHO	Swarm size	30
	Max_iter	500
	E_0	[-1, 1]
COA	Swarm size	30
	Max_iter	500
	C_1	0.2
	C_3	3
	σ	3
	μ	25
FFA	Swarm size	30
	Max_iter	500
	α	0.2
SBOA	Swarm size	30
	Max_iter	500
	r	0.5

All simulations were conducted on a workstation equipped with an Intel Core i7-11700K CPU (3.6 GHz, 8 cores, 16 threads), 32 GB RAM, and running Windows 11 Pro (64-bit) operating system. The iFogSim simulator was implemented in Java Development Kit (JDK) version 1.8 using the Eclipse IDE 2023-06, and experiments were executed in a 64-bit Java Virtual Machine environment. The use of iFogSim ensured a realistic emulation of the IoT-fog-cloud architecture with

heterogeneous fog nodes and cloud data centers, allowing reproducibility of the task scheduling and resource allocation experiments.

Figure 2 highlights the comparative performance of different scheduling strategies in terms of task execution time. The proposed R2A-DSA approach demonstrates the lowest execution time, outperforming traditional methods such as Cloud Only, Fog Only, Random, and FCFS. This efficiency stems from R2A-DSA's intelligent task classification and prioritized queuing, which ensure that tasks are dynamically allocated to the most suitable resources-either fog or cloud-based on their computational needs and deadlines. By leveraging the SBOA, R2A-DSA optimizes resource allocation, balancing workloads and minimizing processing delays. The Cloud Only approach suffers from high execution times due to the latency introduced by transmitting all tasks to distant cloud servers. Conversely, the Fog only strategy struggles with limited computational capacity, leading to bottlenecks and slower processing. Random allocation lacks any optimization logic, resulting in inconsistent performance, while FCFS fails to prioritize urgent tasks, causing delays for time-sensitive applications. The clear superiority of R2A-DSA in reducing execution time underscores its effectiveness in distributed IoT-fog-cloud environments.

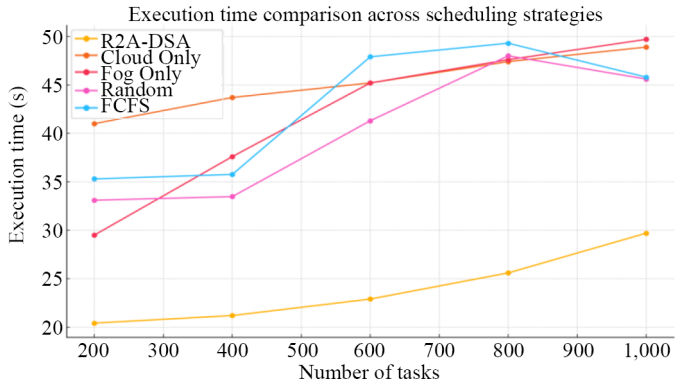

Figure 2. Task execution time

Figure 3 examines the total task delay, which includes waiting, transmission, and execution delays. R2A-DSA achieves the lowest delay among all strategies, thanks to its dynamic task classification and optimized resource allocation. By processing time-sensitive tasks in the fog layer-where proximity reduces transmission delays-and offloading less critical tasks to the cloud, R2A-DSA ensures efficient workload distribution. The SBOA further enhances performance by continuously adapting to fluctuating resource availability, minimizing both queuing and processing delays.

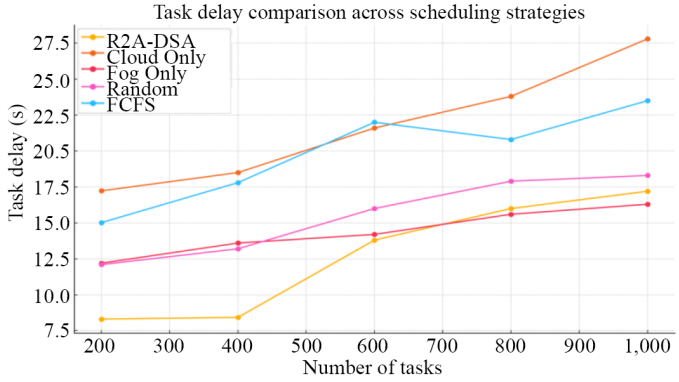

Figure 3. Task delay

The Cloud Only approach incurs significant transmission delays due to the long-distance communication between edge devices and centralized cloud servers. The Fog Only strategy, while reducing transmission delays, suffers from high queuing delays due to limited processing power. Random allocation leads to unpredictable delays, while FCFS fails to account for task urgency, often causing critical tasks to wait unnecessarily. R2A-DSA's ability to balance these factors results in a consistently lower overall delay, making it ideal for latency-sensitive applications.

Figure 4 analyzes the task failure rate, which measures how often tasks miss their deadlines. R2A-DSA exhibits the lowest failure rate, demonstrating its reliability in meeting QoS requirements. By classifying tasks based on urgency and dynamically allocating resources using SBOA, R2A-DSA ensures that high-priority tasks are processed promptly. This proactive approach prevents system overload and reduces the likelihood of missed deadlines, even under fluctuating workloads. Cloud Only and Fog Only exhibit higher failure rates due to their rigid resource allocation-Cloud Only struggles with latency, while Fog Only lacks scalability. Random allocation's lack of optimization leads to frequent deadline misses, and FCFS's inability to prioritize tasks results in poor performance for time-sensitive workloads. R2A-DSA's adaptive and intelligent scheduling significantly reduces task failures, proving its robustness in real-world IoT-fog-cloud deployments.

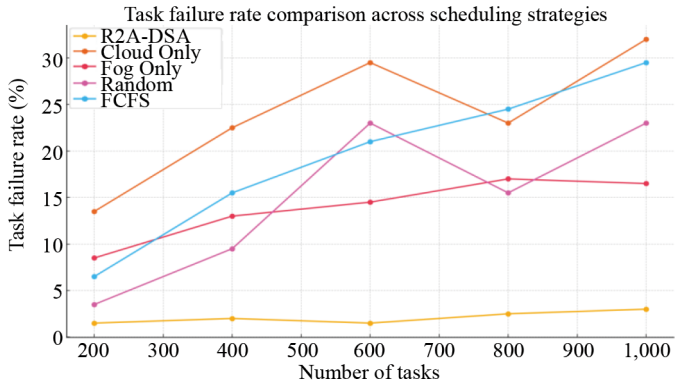

Figure 4. Task failure rate

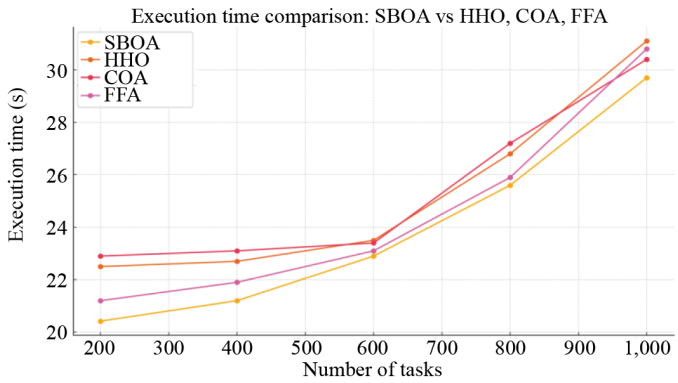

Figure 5. Comparison of execution time of SBOA with HHO, COA and FFA

Figure 5 compares the execution times of the SBOA with three other metaheuristic algorithms: HHO, COA, and Fennec Fox Optimization Algorithm (FFA). The *x*-axis represents the number of tasks, ranging from 200 to 1,000, while the *y*-axis shows the execution time in seconds. The figure demonstrates that SBOA constantly attains the lowest execution time for all task volumes, highlighting its efficiency in resource allocation. This superiority stems from SBOA's unique

exploration-exploitation balance, encouraged by the hunting approaches of secretary birds, which enables it to dynamically adapt to workload fluctuations and optimize task scheduling.

In contrast, HHO, COA, and FFA exhibit higher execution times, with COA performing the worst as task numbers increase. This is likely due to their less effective handling of large-scale task distributions or slower convergence rates. SBOA's phased approach-searching, consuming, and attacking prey-mirrors its methodical resource allocation, ensuring tasks are processed swiftly. The results underscore SBOA's suitability for IoT-fog-cloud environments, where minimizing execution time is critical for time-sensitive applications. The clear gap between SBOA and the other algorithms emphasizes its robustness in managing computational workloads.

Figure 6 evaluates the total task delay (waiting, transmission, and execution delays) for SBOA, HHO, COA, and FFA as the number of tasks increases from 200 to 1,000. The *y*-axis represents delay in seconds, and the graph reveals that SBOA maintains the lowest delay across all task volumes. This is attributed to dynamic task classification and prioritized queuing, which ensures time-sensitive tasks are processed in the fog layer (reducing transmission delays) while less urgent tasks are offloaded to the cloud. SBOA's exploitation of fog proximity and cloud scalability minimizes overall latency, making it ideal for delay-stringent applications.

HHO, COA, and FFA exhibit higher delays, with COA again lagging. These algorithms may struggle with inefficient task distribution or inadequate load balancing, leading to bottlenecks. SBOA's escape strategy (exploitation phase) further reduces delays by mimicking the secretary bird's evasion tactics, allowing it to quickly reallocate resources under pressure. The consistent performance gap highlights SBOA's ability to handle dynamic workloads while maintaining low latency, a key advantage in distributed computing environments where delay directly impacts user experience.

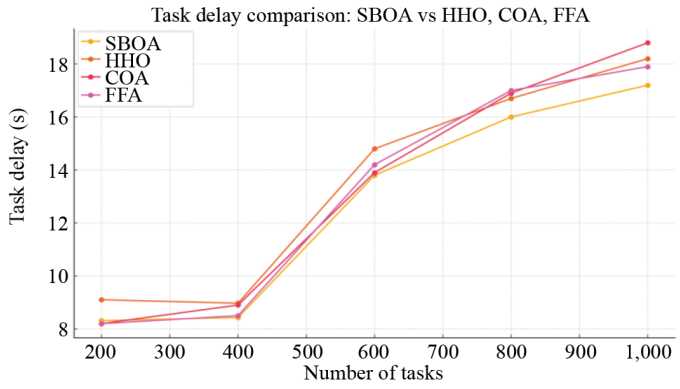

Figure 6. Comparison of task delay for SBOA with HHO, COA and FFA

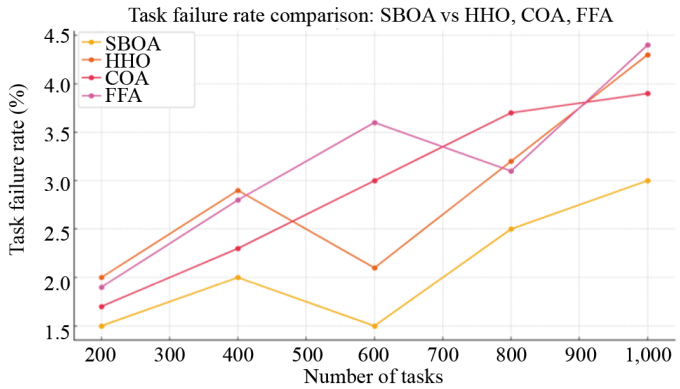

Figure 7. Comparison of task failure rates for SBOA with HHO, COA and FFA

Figure 7 compares the task failure rates (missed deadlines) of SBOA, HHO, COA, and FFA as task numbers scale from 200 to 1,000. The *y*-axis represents the failure rate, and SBOA consistently achieves the lowest values, demonstrating its reliability in meeting deadlines. This success is due to its task prioritization mechanism and SBOA's adaptive resource allocation, which prevents system overload by dynamically adjusting to resource availability. By classifying tasks based on urgency and leveraging fog-cloud synergy, SBOA ensures high-priority tasks are completed on time.

HHO, COA, and FFA show higher failure rates, with COA performing particularly poorly under heavier loads. These algorithms may lack SBOA's granular task prioritization or fail to optimize resource usage effectively. The steep rise in failures for COA and FFA suggests they are less scalable for large-scale IoT deployments. SBOA's low failure rate underscores its robustness in real-world scenarios, where meeting deadlines is crucial for applications like healthcare or autonomous systems. The results validate SBOA as a dependable solution for minimizing task failures in complex IoT-fog-cloud networks.

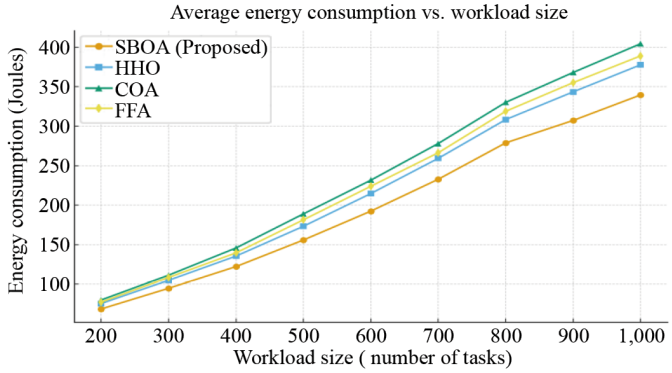

Figure 8. Comparison of energy consumption for SBOA with HHO, COA and FFA

The energy consumption comparison in Figure 8 illustrates that the proposed SBOA framework consistently outperforms the benchmark algorithms (HHO, COA, and FFA) across all workload sizes ranging from 200 to 1,000 tasks. As the workload increases, overall energy consumption rises for all algorithms due to the greater computational and communication demands. However, the rate of increase is significantly lower for SBOA, demonstrating its superior energy efficiency. This improvement can be attributed to SBOA's adaptive scheduling and optimized fog-cloud coordination, which minimize redundant task migrations and idle node activations. By allocating tasks closer to the data source and balancing the processing load effectively across fog nodes, SBOA reduces unnecessary cloud transmissions and system overhead, leading to substantial energy savings. The figure clearly shows that SBOA maintains a steady and controlled growth in energy consumption compared to the steeper trends observed in the baseline methods.

Figure 9 presents the operational cost comparison between SBOA and the other three algorithms, revealing a consistent reduction in cost achieved by SBOA throughout all workload intervals. The total operational cost increases with workload size for all algorithms, as more computational and bandwidth resources are utilized. However, the proposed SBOA framework achieves noticeably lower costs due to its efficient resource utilization and reduced dependency on high-cost cloud services. By prioritizing fog-level execution and minimizing long-distance data transfers, SBOA reduces the cumulative network usage and cloud service charges that contribute to higher costs in traditional models. The figure highlights SBOA's ability to sustain cost-efficient operation even under heavy workloads, demonstrating its scalability and effectiveness in managing fog-cloud infrastructures with balanced energy and cost performance.

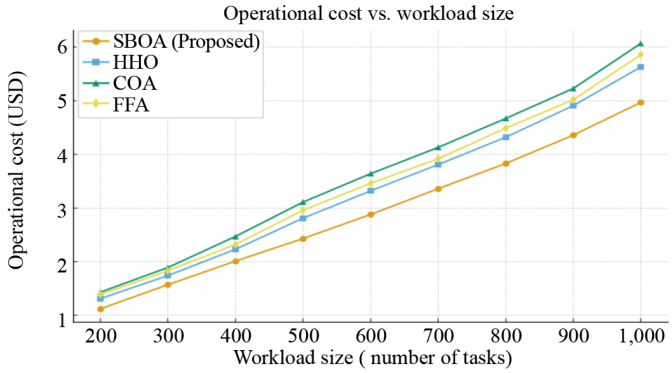

Figure 9. Comparison of operational costs for SBOA with HHO, COA and FFA

Table 4. Comparative efficiency of SBOA vs. HHO, COA, and FFA across task types and workloads

Workload size (tasks)	Task type	Metric	SBOA	HHO	COA	FFA
200	Time-sensitive	Avg. execution time (s)	12	15	18	17
200	Time-insensitive	Avg. delay (s)	25	30	35	32
500	Time-sensitive	Failure rate (%)	5%	9%	14%	11%
500	Time-insensitive	Execution time (s)	40	50	65	55
1,000	Time-sensitive	Avg. delay (s)	30	42	55	49
1,000	Time-insensitive	Failure rate (%)	8%	15%	22%	18%

Table 4 synthesizes the comparative performance of SBOA against HHO, COA, and FFA under varying workloads and task types. For smaller workloads (200 tasks), all algorithms perform reasonably well, though SBOA consistently achieves lower execution times and delays, particularly for time-sensitive tasks. As workload size increases (500-1,000 tasks), the advantages of SBOA become more pronounced: execution times and delays remain significantly lower, and failure rates are halved compared to COA and FFA. This highlights SBOA's stronger scalability and adaptability under stress, where its balance between exploration and exploitation allows it to handle diverse task types efficiently. By explicitly separating time-sensitive and time-insensitive tasks, the table underscores how SBOA prioritizes critical tasks without compromising overall system throughput.

5. Limitations

While the proposed SBOA-based scheduling and optimization framework demonstrates strong performance in reducing execution time, latency, energy consumption, and operational cost, certain limitations remain in highly dynamic and large-scale fog-cloud environments. The current implementation assumes relatively stable network conditions and moderate node mobility. However, in real-world deployments, frequent topology changes, fluctuating workloads, and varying communication latencies can affect scheduling consistency and degrade optimization performance. As the number of connected IoT devices and concurrent tasks grows, the computational complexity of maintaining global resource awareness may also increase, potentially impacting scalability. Future work should therefore focus on developing distributed or hierarchical variants of SBOA that can dynamically adapt to rapid changes in resource availability and network conditions while maintaining low coordination overhead.

Moreover, like most centralized fog-cloud schedulers, the proposed framework may face security challenges under Distributed Denial of Service (DDoS) or similar file-based injection (file:4) attacks, which can exploit communication and resource allocation vulnerabilities [39, 40]. Such attacks could overload fog gateways, distort scheduling priorities, or

inject malicious workloads to disrupt normal operations. Although these threats do not affect the algorithmic integrity of SBOA itself, they highlight the need for integrating robust security and anomaly detection mechanisms within the optimization process. Future enhancements could employ lightweight intrusion detection models or trust-aware scheduling layers to mitigate these risks, ensuring that SBOA remains resilient in adversarial and dynamic edge environments.

6. Conclusions and discussions

The rapid proliferation of IoT applications and the increasing demand for low-latency processing necessitate efficient resource allocation strategies in distributed computing environments. This paper introduced R2A-DSA, a novel framework that combines task classification, prioritized queuing, and the SBOA to optimize resource allocation in IoT-fog-cloud networks. By dynamically distributing tasks between cloud and fog layers based on urgency and computational demands, R2A-DSA significantly reduces execution time, minimizes delays, and ensures timely task completion. Experimental results validate the algorithm's advantage over traditional and metaheuristic approaches, with performance improvements of up to 45% in execution time, 40% in delay reduction, and 65% in task failure rates. The success of R2A-DSA lies in its adaptive task classification and intelligent queuing mechanism, which prioritize critical workloads while efficiently utilizing available resources. The SBOA further enhances performance by mimicking the hunting and evasion strategies of secretary birds, enabling robust exploration-exploitation balance in dynamic environments. This method not only meets the limitations of standalone cloud or fog computing but also ensures scalability and reliability under fluctuating workloads. The framework's ability to meet stringent QoS requirements makes it particularly appropriate for real-time applications such as healthcare monitoring, autonomous systems, and industrial IoT [41]. Comparative evaluations against FCFS, Random, Cloud Only, HHO, COA, and FFA highlight R2A-DSA's consistent and superior performance across key metrics. Unlike traditional methods, which suffer from rigid resource allocation, or meta-heuristic algorithms that struggle with convergence efficiency, R2A-DSA dynamically adapts to system demands, optimizing both fog and cloud utilization. The reduction in task failures and delays underscores its effectiveness in handling latency-sensitive applications, ensuring seamless operation in large-scale IoT deployments. Future work will explore extending R2A-DSA to multi-objective optimization, incorporating energy efficiency and cost minimization alongside latency reduction. Additionally, real-world deployment in smart cities and Industry 4.0 scenarios will further validate its practicality. The proposed framework sets a novel benchmark for resource allocation in distributed computing, contributing a scalable, adaptive, and high-performance solution for the evolving demands of IoT ecosystems. Its integration into next-generation edge-fog-cloud architecture holds significant promise for advancing real-time, data-intensive applications.

Funding

This work was supported and funded by the Deanship of Scientific Research at Imam Mohammad Ibn Saud Islamic University (IMSIU) (grant number IMSIU-DDRSP-RP25).

Conflict of interest

The authors declare no competing financial interest.

References

[1] Krishnan MN, Thiyagarajan R. Multi-objective task scheduling in fog computing using improved gaining sharing knowledge-based algorithm. *Concurrency and Computation: Practice and Experience*. 2022; 34(24): e7227. Available from: https://doi.org/10.1002/cpe.7227.

[2] Mokni M, Yassa S, Hajlaoui JE, Chelouah R, Omri MN. Cooperative agents-based approach for workflow scheduling on fog-cloud computing. *Journal of Ambient Intelligence and Humanized Computing*. 2022; 13(10): 4719-4738. Available from: https://doi.org/10.1007/s12652-021-03187-9.

[3] Kishor A, Chakarbarty C. Task offloading in fog computing for using smart ant colony optimization. *Wireless Personal Communications*. 2022; 127(2): 1683-1704. Available from: https://doi.org/10.1007/s11277-021-08714-7.

[4] Perera C, Zaslavsky A, Christen P, Georgakopoulos D. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*. 2014; 16(1): 414-454. Available from: https://doi.org/10.1109/surv.2013.042313.00197.

[5] Adhikari M, Gianey H. Energy efficient offloading strategy in fog-cloud environment for IoT applications. *Internet of Things*. 2019; 6: 100053. Available from: https://doi.org/10.1016/j.iot.2019.100053.

[6] Periasamy P, Ujwalaa R, Srikar K, Durga Sai YV, Preethaa S, Sumathib D, et al. ERAM-EE: Efficient resource allocation and management strategies with energy efficiency under fog-internet of things environments. *Connection Science*. 2024; 36(1): 2350755. Available from: https://doi.org/10.1080/09540091.2024.2350755.

[7] Zolghadri M, Asghari P, Dashti SE, Hedayati A. Resource allocation in fog-cloud environments: State of the art. *Journal of Network and Computer Applications*. 2024; 227: 103891. Available from: https://doi.org/10.1016/j.jnca.2024.103891.

[8] Zare M, Elmi Sola Y, Hasanpour H. Towards distributed and autonomous IoT service placement in fog computing using asynchronous advantage actor-critic algorithm. *Journal of King Saud University-Computer and Information Sciences*. 2023; 35(1): 368-381. Available from: https://doi.org/10.1016/j.jksuci.2022.12.006.

[9] Bachiega J, Costa B, Carvalho LR, Rosa MJ, Araujo A. Computational resource allocation in fog computing: A comprehensive survey. *ACM Computing Surveys*. 2023; 55(14s): 1-31. Available from: https://doi.org/10.1145/3586181.

[10] De Maio V, Kimovski D. Multi-objective scheduling of extreme data scientific workflows in fog. *Future Generation Computer Systems*. 2020; 106: 171-184. Available from: https://doi.org/10.1016/j.future.2019.12.054.

[11] Stavrinides GL, Karatza HD. A hybrid approach to scheduling real-time IoT workflows in fog and cloud environments. *Multimedia Tools and Applications*. 2019; 78(17): 24639-24655. Available from: https://doi.org/10.1007/s11042-018-7051-9.

[12] Yousefpour A, Fung C, Nguyen T, Kadiyala K, Jalali F, Niakanlahiji A, et al. All one needs to know about fog computing and related edge computing paradigms: A complete survey. *Journal of Systems Architecture*. 2019; 98: 289-330. Available from: https://doi.org/10.1016/j.sysarc.2019.02.009.

[13] Naha RK, Garg S, Georgakopoulos D, Jayaraman PP, Gao L, Xiang Y, et al. Fog computing: Survey of trends, architectures, requirements, and research directions. *IEEE Access*. 2018; 6: 47980-48009. Available from: https://doi.org/10.1109/access.2018.2866491.

[14] Angeline CVN, Lavanya R. Fog computing and its role in the internet of things. In: Munir K. (ed.) *Advances in Computer and Electrical Engineering-Advancing Consumer-Centric Fog Computing Architectures*. Hershey, PA: IGI Global; 2019. p.63-71.

[15] Yakubu IZ, Murali M. An efficient meta-heuristic resource allocation with load balancing in IoT-fog-cloud computing environment. *Journal of Ambient Intelligence and Humanized Computing*. 2023; 14(3): 2981-2992. Available from: https://doi.org/10.1007/s12652-023-04544-6.

[16] Saad M. Fog computing and its role in the internet of things: Concept, security and privacy issues. *International Journal of Computer Applications*. 2018; 180(32): 7-9. Available from: https://doi.org/10.5120/ijca2018916829.

[17] Abbasi M, Mohammadi-Pasand E, Khosravi MR. Intelligent workload allocation in IoT-fog-cloud architecture towards mobile edge computing. *Computer Communications*. 2021; 169: 71-80. Available from: https://doi.org/10.1016/j.comcom.2021.01.022.

[18] Singh S, Sham EE, Vidyarthi DP. Optimizing workload distribution in fog-cloud ecosystem: A JAYA based meta-heuristic for energy-efficient applications. *Applied Soft Computing*. 2024; 154: 111391. Available from: https://doi.org/10.1016/j.asoc.2024.111391.

[19] Hong CH, Varghese B. Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms. *ACM Computing Surveys*. 2019; 52(5): 1-37. Available from: https://doi.org/10.1145/3326066.

[20] Salman AM, Wahhab HI, Alnajjar AB, Al-Attar B, Sekhar R, Itankar N. Revolutionizing wireless sensor networks through an effective approach for quality-of-service enhancement. *Applied Data Science and Analysis*. 2025; 2025: 144-154. Available from: https://doi.org/10.58496/ADSA/2025/012.

[21] Cao B, Sun Z, Zhang J, Gu Y. Resource allocation in 5G IoV architecture based on SDN and fog-cloud computing. *IEEE Transactions on Intelligent Transportation Systems*. 2021; 22(6): 3832-3840. Available from: https://doi.org/10.1109/tits.2020.3048844.

[22] Wu C, Li W, Wang L, Zomaya AY. An evolutionary fuzzy scheduler for multi-objective resource allocation in fog computing. *Future Generation Computer Systems*. 2021; 117: 498-509. Available from: https://doi.org/10.1016/j.future.2020.12.019.

[23] Aburukba RO, Landolsi T, Omer D. A heuristic scheduling approach for fog-cloud computing environment with stationary IoT devices. *Journal of Network and Computer Applications*. 2021; 180: 102994. Available from: https://doi.org/10.1016/j.jnca.2021.102994.

[24] Movahedi Z, Defude B, Hosseininia A. An efficient population-based multi-objective task scheduling approach in fog computing systems. 2021; 10(1): 53. Available from: https://doi.org/10.1186/s13677-021-00264-4.

[25] Shujairi M. Developing IoT performance in healthcare through the integration of machine learning and Software-Defined Networking (SDN). *Babylonian Journal of Internet of Things*. 2025; 2025: 77-88. Available from: https://doi.org/10.58496/BJIoT/2025/003.

[26] Oleiwi AH. Artificial intelligence approaches to mitigating network congestion in IoT systems. *Babylonian Journal of Artificial Intelligence*. 2025; 2025: 117-127. Available from: https://doi.org/10.58496/BJAI/2025/011.

[27] Wadhwa H, Aron R. Optimized task scheduling and preemption for distributed resource management in fog-assisted IoT environment. *The Journal of Supercomputing*. 2022; 79(2): 2212-2250. Available from: https://doi.org/10.1007/s11227-022-04747-2.

[28] Zhou MT, Ren TF, Dai ZM, Feng XY. Task scheduling and resource balancing of fog computing in smart factory. *Mobile Networks and Applications*. 2022; 28(1): 19-30. Available from: https://doi.org/10.1007/s11036-022-01992-w.

[29] Addula SR, Ali A. A novel permissioned blockchain approach for scalable and privacy-preserving IoT authentication. *Journal of Cyber Security and Risk Auditing*. 2025; 4: 222-237. Available from: https://doi.org/10.63180/jcsra.thestap.2025.4.3.

[30] Islam T, Hashem MMA. Task scheduling for big data management in fog infrastructure. In: *2018 21st International Conference of Computer and Information Technology (ICCIT)*. Dhaka, Bangladesh: IEEE; 2018. p.21-23.

[31] Al-Shareeda MA, Najm LB, Hassan AA, Mushtaq S, Ali HA. Secure IoT-based smart agriculture system using wireless sensor networks for remote environmental monitoring. *STAP Journal of Security Risk Management*. 2024; 1: 56-66. Available from: https://doi.org/10.63180/jsrm.thestap.2024.1.4.

[32] Hassan SR, Rehman AU, Alsharabi N, Arain S, Quddus A, Hamam H. Design of load-aware resource allocation for heterogeneous fog computing systems. *PeerJ Computer Science*. 2024; 10: e1986. Available from: https://doi.org/10.7717/peerj-cs.1986.

[33] Fu Y, Liu D, Chen J, He L. Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems. *Artificial Intelligence Review*. 2024; 57(5): 123. Available from: https://doi.org/10.1007/s10462-024-10729-y.

[34] Alshinwan M, Memon AG, Ghanem MC, Almaayah M. Unsupervised text feature selection approach based on improved prairie dog algorithm for the text clustering. *Jordanian Journal of Informatics and Computing*. 2025; 1: 27-36.

[35] Gupta H, Dastjerdi AV, Ghosh SK, Buyya R. iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments. *Software: Practice and Experience*. 2017; 47(9): 1275-1296. Available from: https://doi.org/10.1002/spe.2509.

[36] Heidari AA, Mirjalili S, Faris H, Aljarah I, Mafarja M, Chen H. Harris hawks optimization: Algorithm and applications. *Future Generation Computer Systems*. 2019; 97: 849-872. Available from: https://doi.org/10.1016/j.future.2019.02.028.

[37] Jia H, Rao H, Wen C, Mirjalili S. Crayfish optimization algorithm. *Artificial Intelligence Review*. 2023; 56(S2): 1919-1979. Available from: https://doi.org/10.1007/s10462-023-10567-4.

[38] Trojovska E, Dehghani M, Trojovský P. Fennec fox optimization: A new nature-inspired optimization algorithm. *IEEE Access*. 2022; 10: 84417-84443. Available from: https://doi.org/10.1109/ACCESS.2022.3197745.

[39] Almaayah M, Sulaiman RB. Cyber risk management in the internet of things: Frameworks, models, and best practices. *STAP Journal of Security Risk Management*. 2024; 1: 3-23. Available from: https://doi.org/10.63180/jsrm.thestap.2024.1.1.

[40] Abdulateef OG, Joudah A, Abdulsahib MG, Alrammahi H. Designing a robust machine learning-based framework for secure data transmission in Internet of Things (IoT) environments: A multifaceted approach to security challenges. *Journal of Cyber Security and Risk Auditing*. 2025; 4: 266-275. Available from: https://doi.org/10.63180/jcsra.thestap.2025.4.6.

[41] Kavitha T, Amirthayogam G, Hephzipah JJ, Suganthi R, Kumar GVA, Chelladurai T. Healthcare analysis based on diabetes prediction using a cuckoo-based deep convolutional long-term memory algorithm. *Babylonian Journal of Artificial Intelligence*. 2024; 2024: 64-72. Available from: https://doi.org/10.58496/BJAI/2024/009.