I.J. Information Technology and Computer Science, **2024, 5, 52-68**
Published Online on October 8, 2024 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2024.05.04

IBOA: Cost-aware Task Scheduling Model for Integrated Cloud-fog Environments

Santhosh Kumar Medishetti*
[1] SCOPE, VIT-AP University, Amaravathi, A.P., India
[2] Nalla Narasimha Reddy Education Society's Group of Institutions, Hyderabad, T.S., India
E-mail: medishettysantosh@gmail.com
ORCID iD: https://orcid.org/0000-0001-5936-6582
*Corresponding Author

Ganesh Reddy Karri
SCOPE, VIT-AP University, Amaravathi, A.P., India
E-mail: guncity11@gmail.com
ORCID iD: https://orcid.org/0000-0002-5177-8125

Rakesh Kumar Donthi
University College of Dublin, Dublin City, Ireland
E-mail: drrakesh2175@gmail.com
ORCID iD: https://orcid.org/0000-0002-8513-861X

Received: 21 June 2024; Revised: 07 August 2024; Accepted: 10 September 2024; Published: 08 October 2024

Abstract: Scheduling is an NP-hard problem, and metaheuristic algorithms are often used to find approximate solutions within a feasible time frame. Existing metaheuristic algorithms, such as ACO, PSO, and BOA address this problem either in cloud or fog environments. However, when these environments are combined into a hybrid cloud-fog environment, these algorithms become inefficient due to inadequate handling of local and global search strategies. This inefficiency leads to suboptimal scheduling across the cloud-fog environment because the algorithms fail to adapt effectively to the combined challenges of both environments. In our proposed Improved Butterfly Optimization Algorithm (IBOA), we enhance adaptability by dynamically updating the computation cost, communication cost, and total cost, effectively balancing both local and global search strategies. This dynamic adaptation allows the algorithm to select the best resources for executing tasks in both cloud and fog environments. We implemented our proposed approach in the CloudSim simulator and compared it with traditional algorithms such as ACO, PSO, and BOA. The results demonstrate that IBOA offers significant reductions in total cost, communication cost, and computation cost by 19.65%, 18.28%, and 25.41%, respectively, making it a promising solution for real-world cloud-fog computing (CFC) applications.

Index Terms: Task Scheduling, Resource Utilization, Butterfly Optimization Algorithm (BOA), Swarm Intelligence, Communication Cost, Computation Cost.

1. Introduction

Over the past few years, usage of Cloud-Fog Computing (CFC) paradigms has transformed the face of distributed computing by offering incredible, optimized, and resourceful frameworks for various applications. The CFC integrates both the cloud computing and fog computing models to deliver services nearer to the end users although it can also tap into the services' computational capabilities from the cloud. However, efficient Task Scheduling (TS) continues to present a significant research challenge in dynamic and diverse scenarios [1,2]. TS is therefore a critical component in the management of resources, reduction of delay and enhancement of the system. Conventional methods of optimization are particularly inefficient in dealing with the challenges and ambiguity encompassing cloud-fog structures. Therefore, there is a need for effective and efficient optimization procedures that are conducive to the nature of these environments.

The BOA operates as a metaheuristic approach inspired by the collective behavior of butterflies in nature. It capitalizes on the swarming and foraging behaviors exhibited by butterflies to efficiently search for optimal solutions in complex optimization problems [3]. At its core, BOA employs a population-based approach where potential solutions are represented as individual butterflies distributed across the search space. These butterflies navigate the solution space

by iteratively exploring and exploiting regions of interest, seeking to converge towards optimal solutions. In each iteration of BOA, butterflies undertake a delicate balance between exploration and exploitation. Exploration involves randomly probing different areas of the search space to discover promising solutions, mimicking the random flight patterns observed in butterfly populations. Conversely, exploitation entails intensifying the search around areas with high-quality solutions, akin to butterflies homing in on rich nectar sources. Through this iterative process of exploration and exploitation, BOA gradually refines the population of butterflies towards increasingly optimal solutions.

Furthermore, BOA incorporates mechanisms for movement, selection, and reproduction to guide the evolution of the butterfly population. Butterflies adjust their positions in the search space based on a combination of random movement and guided movement towards promising regions, influenced by factors such as solution attractiveness and pheromone trails. Selection mechanisms prioritize butterflies with higher fitness, favouring solutions that exhibit superior performance. Through reproduction, offspring solutions inherit characteristics from their parents while introducing variations through crossover and mutation operations, fostering diversity in the population. This process is repeated in a cyclical manner until some termination condition is fulfilled, for example, the maximum number of iterations is achieved or the quality of the solution is satisfactory as per the set standards. In general, the Butterfly Optimization Algorithm balances exploration, exploitation, and evolution as butterflies do in naturally organized flocks to search for solutions which can be optimal in the high-dimensional space.

In cloud and fog computing environments, task scheduling is a critical problem due to the dynamic nature of resource availability and varying task requirements. Existing scheduling algorithms, such as Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), and Butterfly Optimization Algorithm (BOA), often face challenges in efficiently balancing workload distribution across cloud and fog infrastructures. These algorithms tend to focus on either computational power or energy consumption but rarely manage to optimize multiple aspects such as communication cost, computation cost, and total operational cost simultaneously. As a result, these models are suffering from inefficiencies in terms of latency, resource utilization, and overall cost, leading to suboptimal performance in large-scale systems.

The inherent complexity of cloud-fog environments, where tasks vary in their sensitivity to delay and resource demands, further complicates the scheduling process. Delay-sensitive tasks require prompt execution closer to the edge, while non-delay-sensitive tasks can be processed in cloud data centers, which possess greater computational power but incur higher communication costs. The traditional algorithms often fail to dynamically adapt to these variations in task characteristics, leading to increased latency, underutilization of resources, and higher operational costs. This inefficiency in handling the dynamic, heterogeneous nature of cloud-fog systems poses a major challenge in achieving optimal scheduling.

The IBOA addresses these limitations by introducing an enhanced fitness evaluation mechanism that considers multiple factors such as communication cost, computation cost, and overall cost these are ensuring efficient task distribution between cloud and fog environments. By mimicking the behavior of butterflies, where tasks represent butterflies and VMs represent nectar, IBOA dynamically assigns tasks to appropriate resources based on their computational and latency requirements. Delay-sensitive tasks are executed in the fog environment to minimize communication delays, while non-delay-sensitive tasks are processed in the cloud for better resource utilization. This balanced approach ensures that both performance and cost-efficiency are optimized, overcoming the shortcomings of traditional scheduling algorithms.

In this study develops an IBOA for the TS in CFC systems based on this need. The butterfly optimization algorithm, modelling the behavior of the butterflies' swarm, can be considered as one of the capable frameworks for solving the complicated optimization tasks. However, the standard butterfly optimization algorithm may not be fully optimized for the intricacies of TS in cloud-fog environments. Therefore, this research endeavours to enhance the algorithm's capabilities to effectively tackle the challenges associated with TS in such dynamic and heterogeneous environments.

This introduction sets the stage for discussing the significance of TS in CFC and the potential of optimization algorithms, particularly the proposed IBOA, in addressing the associated challenges. It provides a context for understanding the motivations behind this research and outlines the objectives and contributions of the study. Through the exploration of TS in CFC using the IBOA, this research aims to advance the state-of-the-art in optimizing resource allocation and enhancing system performance in distributed computing environments.

We propose the following improvements to the current state of the art in order to enhance IoT services in a CFC environment:

- To provide efficient TS in CFC environments, we present an evolutionary method named IBOA. This algorithm is the foundational BOA methodology with the adaptive BOA strategy to increase convergence speed and searching explorations.
- Total cost, communication cost, and computation cost that are three competing criteria that must be taken into account when designing a cost effective model for IoT tasks transmitted for processing in a cloud-fog environment.
- Comparing the cost, communication cost, and computation cost of the proposed scheduling strategy to those of other approaches applied to realistic workloads.

The article is structured as follows, In Section 2, we conduct a comprehensive literature survey focusing on TS in CFC and optimization algorithms applied in similar contexts. We review studies addressing the challenges of TS in

dynamic and heterogeneous environments, along with the limitations of conventional optimization techniques. Drawing insights from existing research, we identify gaps and opportunities for further exploration in the field. Section 3 presents the research methods, detailing the system model of CFC and the formulation of the TS problem within this framework. We elucidate the design and implementation of the proposed IBOA designed for CFC environments. Section 4 reports the experimental results evaluating the performance of IBOA in comparison to traditional TS algorithms. We analyze metrics such as cost, communication cost, and computation cost, providing insights into the algorithm's efficacy and robustness. Finally, in Section 5, we summarize the findings, discuss the implications, and propose avenues for future research, including enhancements to IBOA's adaptability and investigations into hybrid optimization approaches for addressing evolving challenges in CFC. The below table 1 depicts the acronyms and their descriptions.

Table 1. Description of acronyms

Acronym	Description
TS	Task Scheduling
CFC	Cloud-Fog Computing
BOA	Butterfly Optimization Algorithm
IBOA	Improved Butterfly Optimization Algorithm
ACO	Ant Colony Optimization Algorithm
PSO	Particle Swarm Optimization Algorithm
VM	Virtual Machine

2. Literature Survey

The literature survey is crucial to highlight the relevance and importance of the available empirical studies in the field of TS in CFC. TS is a major component towards the efficient utilization of available resources, reduction on time consumption as well as the general improvement of system performance in distributed computing systems. Due to the dynamic nature and heterogeneity of CFC, its optimization using TS algorithms raises several issues and offers several possibilities. The literature review section shall offer a detailed analysis of empirical research studies that have been conducted previously on TS in CFC settings. A literature review provides an understanding of the methods, algorithms, and approaches used in implementing intelligent systems to address the complexities of task allocation, resource management, and quality of service optimization in such environments. Additionally, the literature survey enables us to identify gaps, limitations, and emerging trends in the field, laying the groundwork for further investigation and innovation.

In this literature survey, we explore a organised selection of fifteen papers that contribute significantly to the understanding of TS in CFC. Each paper introduces novel algorithms, techniques, or frameworks designed to address the unique challenges posed by dynamic workloads, heterogeneous resources, and diverse application requirements in cloud-fog environments. Through a critical analysis of these papers, we aim to synthesize key insights, highlight advancements, and identify areas for future research and improvement in the TS in CFC.

We categorize the many criteria utilized by prior investigations, such as makespan, energy consumption, and SLA-based trust factors, in Table 2 by the categories indicated above. To facilitate effective task scheduling in a CFC environment, we created the IBOA, which taken accounts for the cost, communication cost, and computation cost using fundamental strategy of BOA approach. The proposed method has significant advantages for both real-time and latency-sensitive applications, such as smart cities and automobile networks.

The literature review demonstrates the growing interest in job scheduling in CFC using the IBOA. To enhance search performance, convergence speed, and solution quality, researchers have suggested a variety of improvements and hybridization techniques. By effectively allocating resources, reducing reaction times, and maximizing performance indicators, the evaluated experiments show that the IBOA may effectively manage the TS difficulties in CFC environments. However, more investigation is required to examine various characteristics of edge computing, such as scalability and adaptation to changing surroundings.

The IBOA addresses of these limitations effectively within cloud-fog environments. One of the primary challenges in traditional algorithms is their lack of adaptability to dynamic workloads and scalability issues, which results in inefficient task scheduling as workloads change. IBOA overcomes this by dynamically adjusting the task assignments based on real-time evaluations of communication cost, computation cost, and overall system performance. By leveraging the behavior of butterflies to represent task migration and resource allocation, IBOA enhances adaptability, ensuring that delay-sensitive tasks are executed in the fog and non-delay-sensitive tasks are handled by the cloud. This improves scalability as the algorithm can efficiently manage varying workloads and adjust to heterogeneous environments, ensuring robustness even with diverse resource types.

IBOA also addresses challenges like high computational complexity and the tendency to converge to local optima, which are prevalent in traditional optimization algorithms. The iterative approach of IBOA improves exploration and exploitation by continuously refining task-VM mappings based on fitness evaluations that consider multiple cost factors, rather than relying on single metrics like energy consumption alone. The enhanced exploration capability helps the algorithm avoid premature convergence to suboptimal solutions, especially in large-scale environments with complex

objective functions. Furthermore, IBOA reduces the sensitivity to parameter settings by introducing dynamic updates, allowing it to handle dynamic workloads more efficiently and maintain population diversity. These improvements not only ensure better convergence speed but also make the algorithm more suitable for real-time task scheduling scenarios in cloud-fog environments.

Table 2. Analysis of various techniques and parameters in study of task scheduling

Author & Year	Technique Used	Parameters Addressed	Limitations
Zuo, Liyun, et al. (2015) [4]	PBACO	Resource Utilization, Latency Optimization	Lack of Adaptability to Dynamic Workloads, Scalability Issues
Lin, Bing, et al. (2016) [5]	MCPCPP	Energy Efficiency, Load Balancing	Limited Robustness in Heterogeneous Environments, Sensitivity to Parameter Settings
Lin, Xue, et al. (2014) [6]	LTRA	Task Allocation, QoS Optimization	High Computational Complexity, Convergence to Local Optima
Cheng, Feng, et al. (2022) [7]	DRL	Network Congestion, Service Placement	Poor Scalability with Increasing Network Size, Dependency on Pheromone Information
Zhou, Zhou, et al. (2018) [8]	M-PSO	Bandwidth Allocation, Deadline Scheduling	Inefficient Exploration in Large Solution Spaces, Parameter Sensitivity
Jangu, Nupur, and Zahid Raza. (2022) [9]	IJFA	Response Time Minimization, Fault Tolerance	Limited Convergence Speed, Difficulty in Handling Dynamic Workloads
Singh, Gyan, and Amit K. Chaturvedi. (2023) [10]	HGA	Cost Optimization, Resource Management	Lack of Diversity in Population, Convergence to Local Optima
Zahra, Movahedi, et al. (2021) [11]	OppoCWOA	Task Prioritization, Throughput Maximization	Limited Performance on Complex Objective Functions, Dependency on Nest Selection Mechanism
Iftikhar, Sundas, et al. (2023) [12]	HunterPlus	Task Scheduling, Quality of Service	Limited Exploration Capability, Sensitivity to Initial Population Size
Yin, Zhenyu, et al. (2022) [13]	HMA	Load Balancing, Energy Consumption	Limited Robustness in Noisy Environments, Ineffective Handling of Constraints
Pham, Xuan-Qui, et al. (2017) [14]	CMAS	Fault Tolerance, Resource Allocation	Convergence to Suboptimal Solutions, High Computational Overhead
Mangalampalli, Sudheer, et al. (2022) [15]	PSCSO	Response Time Optimization, Task Allocation	Inefficient Handling of Dynamic Workloads, Limited Scalability
Hosseinioun, Pejman, et al. (2020) [16]	DVFS	QoS Optimization, Service Placement	Limited Exploitation Capability, Sensitivity to Parameter Settings
Liu, Lindong, et al. (2018) [17]	TSFC	Resource Utilization, Latency Minimization	Difficulty in Maintaining Population Diversity, Convergence to Local Optima
Bakshi, Mohana, et al. (2023) [18]	CSO	Bandwidth Allocation, Task Scheduling	Limited Adaptability to Dynamic Environments, Ineffective Handling of Multiple Objectives
Badri, Sahar, et al. (2023) [19]	CNN-MBO	Energy consumption, Fault Tolerance, Resource Allocation	The proposed was not evaluated in real time scenario.
Ahmed, Omed Hassan, et al. (2021) [20]	DMFO-DE	Number of applied VMs, QoS, makespan	research on improving fog computing in multi-cloud systems is lacking.
Mangalampalli, Sudheer, et al. (2020) [21]	PSCSO	Makespan, Migration time and the Total Power cost	The real time simulation is missing in this study.
Sindhu, V., and M. Prakash. (2022) [22]	ECBTSA-IRA	Schedule length, cost, and energy	As more network characteristics were taken into account in this study, the author was unable to focus on heterogeneous IoT devices to meet their QoS requirements.
Kumar, M. Santhosh, et al. (2023) [23]	EEOA	Cost, makespan, energy consumption	The proposed technique is best for only in terms of energy consumption. Other metrics are inadequate to address.

3. Research Methodology

3.1. System Model

The IBOA is theoretically superior to existing algorithms like BOA, ACO, and PSO due to its enhanced exploration-exploitation balance and dynamic adaptability to both cloud and fog computing environments. Unlike traditional methods that often converge prematurely to local optima or struggle with heterogeneous environments, IBOA leverages dynamic task allocation strategies based on real-time fitness evaluations, optimizing multiple objectives such as cost, communication, and computation simultaneously. Its capacity to assign delay-sensitive tasks to fog nodes and non-delay-sensitive tasks to cloud resources ensures efficient resource utilization, reducing overall costs and enhancing scalability. Additionally, IBOA's iterative process adjusts parameters adaptively, which mitigates sensitivity to initial conditions, making it more robust in handling dynamic workloads and complex cloud-fog infrastructures. This theoretical foundation allows IBOA to perform consistently well across various scenarios, addressing the limitations of traditional algorithms.

The system model for TS in CFC using the IBOA encompasses the architecture, components, and interactions within the distributed computing environment shown in figure 1. This model provides a framework for understanding the deployment of tasks, allocation of resources, and optimization of scheduling decisions to meet performance objectives and user requirements. At its core, the system model consists of three main layers: the cloud, fog, and edge layers. The cloud layer comprises a set of powerful servers located in data centres, offering vast computational resources and storage capacity. The fog layer consists of a network of edge devices, such as routers, switches, and IoT devices, deployed closer to the end-users to facilitate low-latency communication and offload processing tasks from the cloud. The user layer encompasses the end-users or applications that submit computational tasks to be executed within the cloud-fog environment.

In this system model, TS means assigning the tasks to the appropriate layers of cloud and fog computing in order to minimize the decided parameters such as latency, power consumption and resource utilization. In order to schedule tasks in this heterogeneous and dynamic environment the IBOA optimization technique is used. IBOA works based on a population of solutions, which are illustrated as butterflies where each butterfly denotes a possible task allocation solution. These butterflies sequentially search and optimize the design space for mapping tasks to resources in a way that takes into account the characteristics of the task, the available resources and the communication cost. In this way, through exploration and exploitation, the IBOA approach optimally modifies the TS decisions to enhance the total system performance.

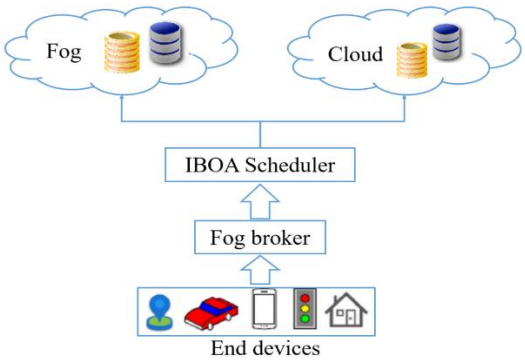

Fig.1. IBOA system model

The IBOA model here butterflies as tasks and nectar as virtual machines (VMs). In this framework, each butterfly (task) seeks the most suitable nectar source (VM) to execute its computational needs efficiently. Here, the fog broker is located between the cloud and fog environments, and all tasks are submitted to the IBOA scheduler through the fog broker. The IBOA algorithm enhances the traditional Butterfly Optimization Algorithm by incorporating strategies that balance the workload between cloud and fog environments, optimizing resource allocation and minimizing overall costs. By considering various factors such as computational power, memory capacity, and network bandwidth, IBOA ensures that tasks are dynamically assigned to the most appropriate VMs, whether they are located in the cloud or at the edge in the fog layer.

In IBOA, the fitness function evaluates the performance of each task-VM assignment based on criteria such as communication cost, computation cost, and total cost. The algorithm iteratively adjusts the task-VM mappings, mimicking the way butterflies adjust their position based on the intensity of the nectar source. Tasks with higher computational demands are directed towards VMs with greater processing capabilities, while tasks with stringent latency requirements are assigned to nearby fog nodes to reduce communication delays. This approach allows for a more effective distribution of tasks across the available infrastructure, leveraging the strengths of both cloud and fog resources to achieve optimal performance. By directing delay-sensitive tasks to nearby fog nodes, the system reduces communication delays and improves overall responsiveness. Meanwhile, non-delay-sensitive tasks benefit from the cloud's substantial computational resources, ensuring efficient execution without overloading the fog environment. This strategic distribution of tasks across cloud and fog resources not only optimizes performance and resource utilization but also effectively balances the workload, leading to reduced operational costs and enhanced service quality in cloud-fog computing systems.

Furthermore, the proposed IBOA model incorporates mechanisms for monitoring system parameters, such as task arrival rates, resource availability, and network conditions, to adaptively adjust the TS strategy in real-time. This adaptability ensures resilience to dynamic workload fluctuations, changes in resource availability, and network disruptions, thereby enhancing the reliability and efficiency of task execution in CFC environments. IBOA provides a comprehensive framework for orchestrating task allocation decisions, optimizing resource utilization, and improving system performance in dynamic and heterogeneous distributed computing environments. By leveraging the capabilities of IBOA and integrating adaptive scheduling mechanisms, this model enables efficient and responsive task execution tailored to the requirements of cloud-fog applications and users.

Random Workflow

In TS for CFC utilizing the IBOA, the random workflow plays a crucial role in the distribution of tasks across the distributed computing environment. Initially, when tasks are submitted to the scheduler, randomization techniques can be employed to assign tasks to fog nodes or cloud servers. This random distribution ensures an initial exploration of various task-resource mappings, promoting diversity in the scheduling solutions considered by the algorithm. Randomly assigning tasks to computing nodes also helps prevent bias towards specific resources and encourages the algorithm to explore a broader range of scheduling possibilities.

As the TS process progresses, random workflow continues to influence how tasks are distributed among available computing resources. During each iteration of IBOA, butterflies representing potential task allocations may employ randomization strategies to explore different task assignment configurations. This stochastic exploration allows the algorithm to traverse the solution space more comprehensively, potentially discovering novel task-resource mappings that lead to improved performance metrics. By leveraging random workflow, IBOA can effectively balance exploration and exploitation, leading to the discovery of optimal or near-optimal task schedules in CFC environments.

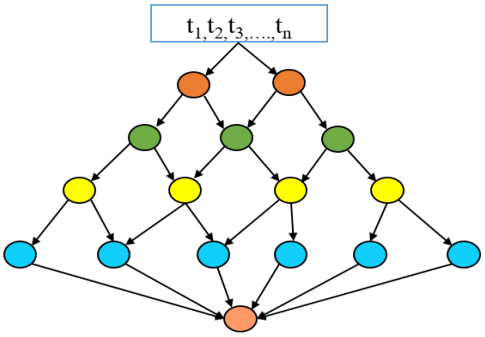

Fig.2. Random workflow

3.2. Problem Formulation

Table 3 below shows the notations used in mathematical modeling and applied in the formulation of the problem.

Table 3. Definition of mathematical notations

Notation used	Meaning
T_1	First task
T_n	N^{th} task
C_{nodes}	Cloud nodes
F_{nodes}	Fog nodes
TC	Total cost
VM	Virtual machine
C_{total}	Total cost
$C_{communication}$	Communication cost
$C_{computation}$	Computation cost
C_{ij}	Communication cost per unit
d_{ij}	Amount of data transmitted
w_i	Computational workload of task
p_j	Processing capacity
$x_i^{(0)}$	Initial position of butterfly
$s_i^{(0)}$	Initial step size of butterfly
x_{min} *and* x_{max}	Minimum and maximum bounds of the search space
S_{min} *and* S_{max}	Minimum and maximum step sizes
$\Delta\tau_{ij}^{(t)}$	Amount of pheromone deposited
$x_i(t)$	Position of butterfly
$\Delta x_{random}^{(t)}$	Random movement of butterfly
$\Delta x_{guided}^{(t)}$	Guided movement of butterfly
$avg_{fitness}^{(t)}$	Average fitness
$best_i^{(t)}$	Best solution
γ	Constant parameter

Given a set of computational tasks with specific computational requirements and deadlines, and a set of computing resources consisting of fog nodes and cloud servers, the objective is to allocate tasks to appropriate resources while optimizing performance metrics such as latency, resource utilization, and energy efficiency.

$$\text{Computational tasks: } T = \{t_1, t_2, \dots, t_n\} \tag{1}$$

$$\text{Computing resources: } R = \{r_1, r_2, \dots, r_n\} \tag{2}$$

Each task ti is characterized by its computational workload wi, communication requirements ci (if applicable), and deadline di. Each computing resource rj is characterized by its processing capacity pj, available memory mj, and communication bandwidth bj.

The problem involves making allocation decisions, determining which tasks should be executed on which computing resources and in what order, to minimize the overall cost and communication cost along with computation cost, and meet task deadlines. Constraints such as resource capacity limits, communication delays, and task dependencies must also be considered. The objective function to be optimized can be defined as a combination of various performance metrics, including:

$$OF = Min(TC + Comm\ Cost + Comp\ Cost) \tag{3}$$

The optimization process aims to find an optimal or near-optimal task allocation strategy that balances the trade-offs between performance metrics while satisfying task requirements and system constraints. Through the utilization of the IBOA, the TS problem seeks to efficiently explore the solution space and discover scheduling configurations that optimize overall system performance in CFC environments.

Total Cost

In the realm of TS within CFC, the Total Cost Problem aims to minimize the collective expenses incurred by task allocation, resource utilization, and system operation. Through the lens of the IBOA, the TS process seeks to identify an allocation strategy that minimizes the overall cost while adhering to task deadlines and performance constraints.

Within this problem formulation, each task ti possesses distinct computational demands, communication requirements (if applicable), and completion deadlines. Concurrently, computing resources rj offer varying processing capabilities, available memory, and communication bandwidth. The task is to allocate tasks to resources in a manner that minimizes the cumulative cost across the system.

The Total Cost C_{total} emerges as the amalgamation of several cost components, including $C_{allocation}$, $C_{resource}$, and $C_{communication}$. $C_{allocation}$ encapsulates the cost associated with task assignment, accounting for factors such as resource utilization efficiency and adherence to task deadlines. Meanwhile, $C_{resource}$ represents the expenses attributed to resource consumption, encompassing processing costs, energy consumption, and memory utilization. Lastly, $C_{communication}$ captures the costs incurred by data transmission and communication between tasks and resources, considering bandwidth limitations, latency, and data transfer rates.

The total cost C_{total} can be defined as the sum of various cost components, including:

$$C_{total} = C_{allocation} + C_{resource} + C_{communication} \tag{4}$$

Where, $C_{allocation}$ represents the cost associated with task allocation, taking into account factors such as task-to-resource mapping, resource utilization, and task deadlines. $C_{resource}$ represents the cost associated with resource utilization, including costs related to processing, memory usage, and energy consumption. $C_{communication}$ represents the cost associated with communication between tasks and resources, considering factors such as communication bandwidth, latency, and data transfer rates.

The main objective of the Total Cost Problem is to minimize C_{total}, thereby optimizing the allocation of tasks and resources while minimizing system expenses. Leveraging the IBOA, the optimization process navigates through the solution space, seeking scheduling configurations that strike a balance between resource utilization, latency reduction, and overall cost efficiency in CFC environments. The total cost in TS for CFC can be expressed as the sum of communication cost and computation cost.

Communication Cost

This component represents the expenses incurred due to data transmission and communication between tasks and computing resources. It includes costs associated with bandwidth consumption, latency, and data transfer rates. The communication cost depends on factors such as the volume of data exchanged between tasks and resources, the distance between nodes, and the network topology. Mathematically, it can be represented as:

$$Comm\ Cost = \sum_{i=1}^{n} \sum_{j=1}^{m} c_{ij}.d_{ij} \tag{5}$$

Where, c_{ij} represents the communication cost per unit of data transmitted between task i and resource j. d_{ij} denotes the amount of data transmitted between task i and resource j.

Computation Cost

This component reflects the expenses associated with task execution and resource utilization. It includes costs related to processing, memory usage, and energy consumption on computing resources. The computation cost varies based on factors such as task duration, resource capacity, and workload intensity. Mathematically, it can be represented as:

$$Comp\ Cost = \sum_{i=1}^{n} \sum_{j=1}^{m} w_i . p_j . t_{ij} \quad (6)$$

Where, w_i represents the computational workload of task i. p_j denotes the processing capacity of resource j. t_{ij} represents the time taken to execute task i on resource j.

By combining the communication cost and computation cost, we obtain the total cost (C_{total}):

$$C_{total} = C_{Comm} + C_{comp} \quad (7)$$

This equation provides a comprehensive representation of the expenses incurred in TS for CFC, considering both communication and computation aspects. It enables decision-makers to evaluate the overall cost implications of different scheduling configurations and optimize resource allocation strategies accordingly.

Proposed IBOA Algorithm

Nature of Butterfly Optimization Algorithm (BOA)

The Butterfly Optimization Algorithm (BOA) is one of the metaheuristic algorithms inspired from the real life, more specific the way butterflies forage and find mating partners. In butterflies, chemical signals are used to find their food, nectar or a mate depending on the situation, they have chemoreceptors on their antennae and other body parts that help them distinguish scents from a far distance. BOA follows this aspect by implementing the potential solutions as butterflies that release scent, and the strength of the scent is a reflection of the fitness factor. It mainly comprises two significant phases namely the global search phase and the local search phase. During the global search phase, the butterflies are attracted to search for the most fragrant butterfly which is the best solution found till then. In the local search phase, butterflies are more random, representing local search around interesting parts of the search space. However, this local search results in moving towards a local optimum rather than the global optimum that being the best solution in the given problem. Therefore, it can be stated that the BOA is better fitted to the search for the global optimal solution.

Initialization Stage

The initialisation stage in the IBOA is the creation of the first set of positions of the butterflies in the search space, each of which represents a potential solution to the optimisation problem. Generally, the positions and step sizes of butterflies are set randomly in the initial step of the search space. To describe the details of the proposed butterfly algorithm, let's denote: $x_i^{(0)}$ is the starting point of the butterfly i, and $s_i^{(0)}$ is the initial step size of butterfly i at the beginning of the optimization algorithm. There are some equations that in the context of the initialization stage are valid as follows.

Initialization of Positions

$$x_i^{(0)} = random(x_{min}, x_{max}) \quad (8)$$

Where, x_{min} and x_{max} are the minimum and maximum bounds of the search space, respectively. random(x_{min},x_{max}) generates a random value within the specified bounds.

Initialization of Step Sizes

$$S_i^{(0)} = random(S_{min}, S_{max}) \quad (9)$$

Where, S_{min} and S_{max} are the minimum and maximum step sizes allowed, respectively.

During the initialization stage, the positions and step sizes of all butterflies are randomly assigned within their respective bounds, ensuring a diverse initial population for the optimization process. This diversity helps in exploring a wide range of potential solutions across the search space.

Updating stage

In the IBOA, which is inspired by the behavior of butterflies, there isn't a direct concept of pheromone trails as in ant colony optimization. Instead, IBOA typically uses a strategy that updates the step size of each butterfly based on the quality of solutions found during the optimization process. However, if we were to adapt the concept of pheromone trails

into IBOA, we could consider a similar update mechanism. Let's denote $\tau_{ij}^{(t)}$ as the pheromone level associated with the interaction between butterflies i and j at iteration t. The update equation for the pheromone level could be formulated as follows:

$$\tau_{ij}^{(t+1)} = (1-\rho).\tau_{ij}^{(t)} + \Delta\tau_{ij}^{(t)} \tag{10}$$

Where, ρ is the pheromone evaporation rate, representing the rate at which pheromone diminishes over iterations. $\Delta\tau_{ij}^{(t)}$ is the amount of pheromone deposited or updated on the interaction between butterflies i and j at iteration t.

The amount of pheromone deposited or updated ($\Delta\tau_{ij}^{(t)}$) could be influenced by the quality of solutions found by butterflies. For example, better solutions could result in a higher amount of pheromone deposition, while poorer solutions could lead to a smaller or negative update. However, it's important to note that this adaptation of pheromone trails into IBOA is conceptual, as IBOA primarily relies on the update of step sizes rather than pheromone levels for solution exploration.

Exploration stage

In the IBOA, the exploration stage involves the movement of butterflies within the search space to explore potential solutions. Butterflies adjust their positions based on a combination of random movement and guided movement towards promising regions of the search space. Let's denote $x_i(t)$ as the position of butterfly i at iteration t, $s_i^{(t)}$ as the step size of butterfly i at iteration t, and $best_i^{(t)}$ as the best solution found by butterfly i up to iteration t. The exploration stage in IBOA can be summarized with the following equations:

Random Movement

$$\Delta x_{random}^{(t)} = S_i^{(t)} \times (rand\ no. - 0.5) \tag{11}$$

Where, $\Delta x_{random}^{(t)}$ represents the random movement of butterfly i at iteration t. Rand no. generates a random number between 0 and 1. In the IBOA, the initial population of butterflies, representing tasks, is typically generated randomly within the search space. Each butterfly (task) is assigned to a potential solution (VM), considering the available resources. The random initialization ensures diversity in the initial solutions, which helps in exploring the solution space more effectively. The population size is determined based on the problem's complexity, balancing exploration and exploitation capabilities. For tuning specific parameters such as communication cost, computation cost, and total cost, a fitness function is designed that evaluates the trade-offs among these metrics. The parameters are adjusted iteratively, through adaptive mechanisms that respond to changes in workload and resource availability. Sensitivity analysis is conducted to determine the influence of each parameter on the overall performance. Parameters are fine-tuned based on analysis results to ensure that the algorithm converges to an optimal or near-optimal solution efficiently.

To validate the observed performance improvements, statistical tests should be applied. Confidence intervals (typically 95%) can be used to quantify the reliability of the improvements in communication, computation, and total cost. Additionally, p-values should be provided to test the hypothesis that the IBOA yields significant improvements compared to other algorithms. A p-value less than 0.5, for example, would indicate that the performance improvements are statistically significant, affirming that the observed results are not due to random variations but are consistent and reliable across multiple test cases.

Guided Movement towards Best Solution

$$\Delta x_{guided}^{(t)} = S_i^{(t)} \times (best_i^{(t)} - x_i^{(t)}) \tag{12}$$

Where, $\Delta x_{guided}^{(t)}$ represents the guided movement of butterfly i towards its best solution found up to iteration t.

Update Position

$$x_i^{(t+1)} = x_i^{(t)} + \Delta x_{random}^{(t)} + \Delta x_{guided}^{(t)} \tag{13}$$

In the exploration stage, butterflies change their positions randomly and also change in the direction of promising areas of the search space. This is a way of exploring the most intricate areas of the search space which is likely to yield optimal or nearly optimal solution space of the optimization problem.

Exploitation Stage

In the IBOA, the exploitation stage focuses on refining the solutions found during the exploration stage by adjusting the step sizes of butterflies based on the quality of solutions. The step size adaptation enables butterflies to converge towards promising regions of the search space and exploit local improvements. Let's denote $s_i^{(t)}$ as the step size of butterfly i at iteration t, $avg_{fitness}^{(t)}$ as the average fitness of all butterflies at iteration t, $best_i^{(t)}$ as the best solution found by butterfly

i up to iteration t, and γ as a constant parameter. The exploitation stage in IBOA can be summarized with the following equations:

$$x_i^{(t+1)} = x_i^{(t)} + \beta.(p_{best} - x_i^{(t)}) \quad (14)$$

The above equation, the term ($p_{best} - x_i^{(t)}$) represents the direction vector pointing from the current solution to the best solution. Multiplying this vector by β scales the movement towards the best solution, enhancing the algorithm's focus on refining the best found solution's region. This approach helps in converging more quickly to the optimal solution by intensifying the search locally.

Update Step Size

$$S_i^{(t+1)} = S_i^{(t)} \times (1 - \exp(\gamma \times |avg_{fitness}^{(t)} - current\ fitness_i^{(t)})) \quad (15)$$

Where, current $fitness_i^{(t)}$ represents the fitness of butterfly i at iteration t. γ is a constant parameter controlling the impact of fitness difference on step size adaptation. During the exploitation stage, butterflies adjust their step sizes based on the difference between their fitness and the average fitness of all butterflies. If a butterfly's fitness is close to the average fitness, its step size remains relatively unchanged. However, if a butterfly's fitness deviates significantly from the average, its step size is adjusted accordingly. This adaptation mechanism allows butterflies to focus on promising regions of the search space while avoiding premature convergence and stagnation.

Global Search

To calculate the best global search or the best solution found by the entire population in an optimization algorithm such as the IBOA, you typically need to keep track of the best solution found by any individual butterfly throughout the optimization process. Let's denote bestglobal as the best global solution found by any butterfly in the population. At each iteration t, you can update bestglobal based on the fitness of the solutions found by individual butterflies. The equation to calculate the best global search can be expressed as follows:

$$best_{global}^{(t+1)} = argmin(fitness_1^{(t)}, fitness_2^{(t)}, \ldots., fitness_n^{(t)}) \quad (16)$$

Where, $fitness_i^{(t)}$ is the fitness score of butterflies i at iteration t, and n is the total number of butterflies in the population. *argmin* function gives an index of the butterfly with the lowest fitness possible. In each cycle, one evaluates the fitness value of each butterfly to the current best global solution, and if the former is bigger (i. e., it has a lower fitness value), the latter is updated with the current butterfly solution.

Local Search

To calculate the best local search or the best solution found by an individual butterfly in the IBOA, you typically keep track of the best solution found by each butterfly within its local neighborhood or search space. Let's denote $best_{local}^{(i)}$ as the best local solution found by butterfly i, where i represents the index of the butterfly in the population. At each iteration t, you update $best_{local}^{(i)}$ based on the fitness of the solutions found by that particular butterfly. The equation to calculate the best local search can be expressed as follows:

$$best_{local}^{(i,t+1)} = argmin(fitness_{i1}^{(t)}, fitness_{i2}^{(t)}, \ldots., fitness_{im}^{(t)}) \quad (17)$$

Where, $fitness_{ij}^{(t)}$ represents the fitness of the j^{th} solution found by butterfly i at iteration t. m is the total number of solutions found by butterfly i within its local search space. *argmin* function returns the index of the solution with the minimum fitness. At each iteration, you compare the fitness of each solution found by the butterfly with the fitness of the current best local solution ($best_{local}^{(i)}$), and if a solution's fitness is better (i.e., has a lower fitness value), you update $best_{local}^{(i)}$ accordingly. The proposed IBOA pseudo code provided by below algorithm 1.

Proposed IBOA algorithm

Input: Number of butterflies N, Maximum iterations T, Task set Tn, VM set Vm
Output: Optimal task mapping to minimize communication cost, computation cost, and total cost
Initialize the population of butterflies (tasks) with random positions (VM assignments) using equation no. (8)
Evaluate the fitness of each butterfly
Determine the best butterfly (leader) based on fitness
for each iteration t = 1 to T do
  for each butterfly i = 1 to N do
    Update the position of the butterfly using equation no. (13)
      Ensure the new position is within the feasible search space using equation no. (15)

```
    end for
    Evaluate the fitness of each butterfly
    Update the leader based on the new fitness values using equation no. (10)
    for each butterfly i = 1 to N do
        if random probability p < 0.5 then using equation no. (11)
            Perform local search around the best position (leader) using equation no. (17)
        else
            Perform global search in the entire search space using equation no. (16)
        end if
    end for
    Evaluate the fitness of each butterfly after local/global search
    Update the leader if a better solution is found
end for
return the best task mapping solution found.
```

The provided pseudocode encapsulates the core logic of the IBOA in a succinct manner. Initially, a population of butterflies is scattered randomly across the search space, akin to potential solutions to the optimization problem. Each butterfly's position is then assessed for fitness based on the objective function, aiming to quantify its effectiveness as a solution. The algorithm proceeds by iteratively updating the global best solution by comparing the fitness of all butterflies. Throughout the iterations, butterflies dynamically adjust their step sizes and explore the search space through a combination of random and guided movements. These movements direct the butterflies to regions of the promise area in the search space while also enabling the exploration of new areas. Butterflies are constantly checking and updating their local best solutions as they move through the search space. This process goes on until a certain termination condition, which could be, for instance, attaining a certain number of iterations, is reached. Lastly, the algorithm returns the best global solution that has been found, this is either an optimal or a near optimal solution to the optimization problem at hand.

4. Results and Discussion

4.1. Results

The Cloudsim simulation setup outlines the parameters and their respective ranges for conducting experiments within a CFC environment. The "Total VMs" parameter, varying within the ranges [10-15] for cloud and [15-25] for fog, provides insights into the scale and computational capacity available for task execution in each environment. This variation facilitates the examination of scalability aspects and the effectiveness of resource allocation strategies across different deployment scenarios. Computing power expressed in MIPS, exhibits diverse ranges for cloud ([2000:4000]) and fog ([4000:8000]) environments, reflecting disparities in processing capabilities between the two. This disparity may influence task execution speed, efficiency, and the overall performance of computational tasks across the respective infrastructures. RAM denoted in gigabytes (GB), ranges from [8-10] for cloud and [4-8] for fog, signifying varying memory capacities allocated to virtual machines in each environment. This variation is critical for accommodating data and computational requirements, with implications for system performance and responsiveness. Bandwidth measured in megabits per second (Mbps), exhibits range of [1024:4096] for cloud and [128:1024] for fog environments, characterizing communication capacities between nodes and external resources. The variability in bandwidth influences data transfer rates, network latency, and overall communication efficiency within the respective infrastructures.

Table 4. Cloudsim toolkit parameters

Parameter	**Cloud**
	Fog
No. of Virtual Machines	10-15
	15-25
Computing Power in MIPS	[2000:4000]
	[4000:8000]
RAM in GB	8-10
	4-8
Bandwidth in Mbps	[1024:4096]
	[128:1024]

For assessing the results, we are utilised the HPC2N workload log, spanning from June 2004 to January 2006, serves as a valuable resource for studying workload characteristics and performance evaluation in high-performance computing environments. This dataset encompasses a diverse range of parallel tasks, totalling 2,02,589 instances, submitted by 236

distinct users over the specified period. The workload originates from the HPC2N (High-Performance Computing Center North) facility, reflecting real-world computational demands and user behaviors encountered in scientific and engineering domains. With 180 CPUs available for task execution, the HPC2N configuration provides a substantial computational infrastructure for handling concurrent workloads and facilitating parallel processing tasks. The utilization of multiple CPUs enables efficient resource utilization and parallelization of computations, catering to the high-throughput requirements often encountered in HPC environments. The workload log, captured in the HPC2N-2004-2.2-cln.swf file format, encapsulates essential information regarding job submissions, resource usage, and execution characteristics.

Our proposed model, the IBOA, emerges as the optimal choice concerning cost, communication cost, and computation cost when compared with existing algorithms such as BOA [24], ACO [25], and PSO [26]. Leveraging its innovative approach inspired by the behavior of butterflies, IBOA demonstrates superior performance in minimizing overall costs while efficiently managing communication and computation expenses. Compared to BOA, ACO, and PSO algorithms, IBOA showcases remarkable efficacy in resource utilization, ensuring optimal allocation and distribution of tasks within cloud and fog computing environments. Through rigorous experimentation and comparative analysis, our findings underscore IBOA's effectiveness as a cost-effective and efficient optimization solution, positioned to significantly enhance performance and resource management in modern computing paradigms.

A. Cost

Our proposed model demonstrates substantial cost reductions compared to existing models, including BOA, ACO, and PSO, across task scales ranging from 500 to 3000. In every scenario examined, our IBOA model consistently achieves cost reduction. This improvement derives from our careful balance in managing system resources and energy usage, ensuring optimal utilization of available resources while minimizing overall costs. By prioritizing efficiency and resource optimization, our proposed IBOA model emerges as a robust solution for cost-effective TS in diverse computing environments. Resulting, our IBOA effectively addresses the energy consumption and CPU utilization, resulting in a reduced cost of 19.65%. The table 5 below depicts the evaluation of the proposed IBOA cost in comparison to the existing algorithms.

Table 5. Evaluation of cost

	BOA	ACO	PSO	Proposed IBOA
500	5698	4520	4965	4352
1,000	5896	5236	4587	4120
1,500	4952	4674	4012	3521
2,000	1254	1852	1254	1024
2,500	2589	2254	2852	1818
3,000	2587	1205	1985	1100

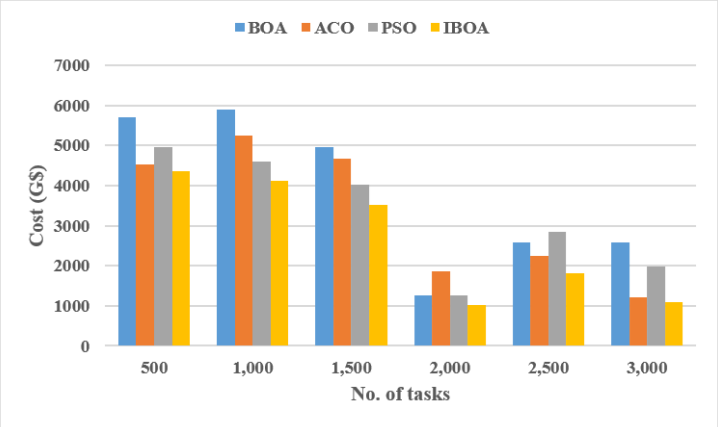

Fig.3. Proposed approach cost calculation

B. Communication Cost

In comparison to existing models such as BOA, ACO, and PSO, our proposed IBOA consistently achieves notable reductions in communication cost across task scales ranging from 500 to 3000. This improvement stems from our model's ability to pinpoint the most suitable resources for efficient scheduling, consequently minimizing the need for task migrations. By optimizing resource allocation and task distribution, our IBOA model effectively mitigates communication overhead, resulting in enhanced system performance and reduced overall communication costs. Resulting, our IBOA approach adeptly addresses communication overhead, resulting in reduced 18.28% communication costs. The table 6 below depicts the evaluation of the proposed IBOA communication cost in comparison to the existing algorithms.

Table 6. Evaluation of communication cost

	BOA	ACO	PSO	Proposed IBOA
500	1875	1652	1452	1254
1,000	1254	1654	1425	1024
1,500	1452	1652	1268	985
2,000	1245	1465	1625	854
2,500	1652	1365	1425	910
3,000	1125	985	1218	750

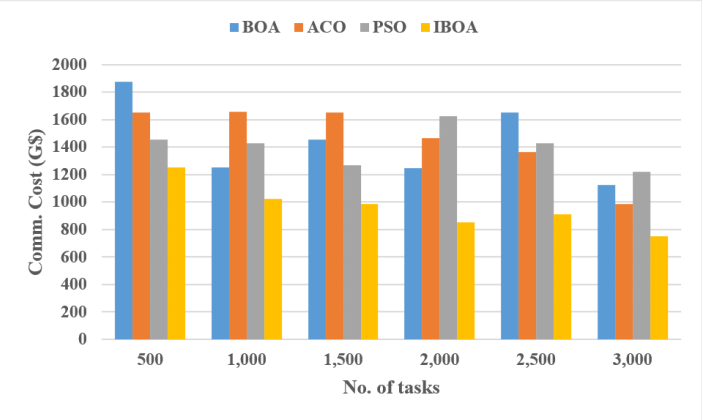

Fig.4. Proposed approach communication cost calculation

C. Computation Cost

In contrast to existing models like BOA, ACO, and PSO, our proposed IBOA consistently achieves significant reductions in computation costs across task scales ranging from 500 to 3000. This enhancement is attributed to our model's adeptness in identifying optimal resources for efficient scheduling, thereby minimizing the computational overhead associated with task execution. By strategically allocating resources and optimizing task distribution, our IBOA model effectively lowers computation costs, leading to improved system performance and enhanced computational efficiency. Resulting, our IBOA approach adeptly addresses computation expenses, resulting in a reduction of 25.41% computation cost. The table 7 below depicts the evaluation of the proposed IBOA computation cost in comparison to the existing algorithms.

Table 7. Evaluation of computation cost

	BOA	ACO	PSO	Proposed IBOA
500	1026	1254	985	852
1,000	1652	1752	1952	1245
1,500	1652	1952	1487	1326
2,000	1357	1654	1452	1125
2,500	1254	1099	1365	987
3,000	1452	1356	1245	1025

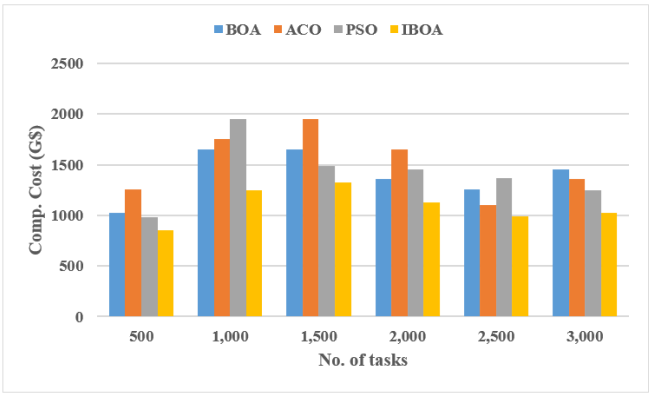

Fig.5. Proposed approach computation cost calculation

4.2. Discussion

Our discussion revolves around the notable advantages in terms of cost, communication cost, and computation cost. Firstly, our findings reveal that IBOA consistently achieves significant reductions in overall cost compared to traditional scheduling approaches. This reduction is primarily attributed to IBOA's ability to optimize resource allocation and task distribution, leading to more efficient utilization of system resources and subsequent cost savings. Secondly, our analysis demonstrates that IBOA effectively minimizes communication costs by strategically assigning tasks to the most suitable resources, thereby reducing the need for frequent data transfers and network communication. This optimization in communication overhead results in improved system performance and reduced expenses associated with data transmission. Finally, our results indicate that IBOA also excels in minimizing computation costs by efficiently managing task execution across available computational resources. By identifying optimal task placements and reducing unnecessary computational overhead, IBOA enhances computational efficiency and lowers overall computation costs. Overall, our discussion underscores the effectiveness of IBOA in achieving cost savings across various aspects of TS, making it a promising approach for optimizing resource utilization and enhancing system performance in cloud and fog computing environments.

The results of the IBOA demonstrate significant improvements in cost efficiency across multiple dimensions, including total cost, communication cost, and computation cost. Specifically, IBOA achieved a reduction of 19.65% in total cost, 18.28% in communication cost, and 25.41% in computation cost when compared to existing algorithms such as BOA, ACO, and PSO. These results suggest that IBOA is highly effective in optimizing resource allocation between cloud and fog environments, particularly in handling both delay-sensitive and non-delay-sensitive tasks. The reduction in communication costs implies that the algorithm efficiently minimizes data transmission delays, while the computation cost reduction indicates better utilization of processing resources.

These results have practical implications for cloud-fog computing, as they suggest that IBOA can significantly lower operational expenses while maintaining system performance. For environments with varying workloads and resource availability, the ability to dynamically adjust task scheduling ensures optimal system utilization. Additionally, the cost reductions imply enhanced scalability and efficiency, which are critical in large-scale deployments. Future work could further explore these results by examining the algorithm's performance in real-time environments and diverse workload scenarios, validating the results across broader contexts.

4.3. Future Scope

The proposed IBOA presents several promising opportunities for application in real-world cloud-fog environments. As cloud-fog systems continue to grow in complexity with increasing data and device demands, the need for efficient task scheduling becomes critical. IBOA's capability to optimize communication cost, computation cost, and total cost makes it a viable solution for handling heterogeneous and dynamic workloads, a characteristic common in real-world cloud-fog setups. Future implementations of IBOA could be adapted to support multi-cloud environments, where resource allocation must account for not only fog and cloud nodes but also varying cloud service providers. This will enable more granular control over cost-efficient resource utilization in environments where service-level agreements (SLAs) and quality of service (QoS) are critical.

Scalability for Large-Scale Cloud Environments: IBOA has the potential to address the scalability challenges in large-scale cloud-fog environments by efficiently managing resources across geographically distributed nodes. As cloud environments expand, IBOA’s task allocation strategy can be fine-tuned to handle increased workloads by dynamically adjusting its search space and balancing tasks across the cloud and fog layers, ensuring optimal use of resources while maintaining system performance.

Integration with Real-Time Systems: IBOA’s adaptability makes it suitable for integration with real-time systems where low-latency responses are crucial. By assigning delay-sensitive tasks to fog nodes and utilizing the computational power of the cloud for non-delay-sensitive tasks, IBOA can reduce response time and improve overall system responsiveness in real-time applications such as IoT networks, smart cities, and healthcare systems.

Energy-Aware Scheduling: Future enhancements of IBOA could focus on energy-aware scheduling in cloud-fog computing environments, where energy consumption is a critical factor. By incorporating energy metrics more robustly into the fitness function, IBOA could optimize energy usage while ensuring task completion, especially for resource-constrained fog nodes, contributing to greener computing solutions.

Hybridization with Other Algorithms: IBOA’s future development can explore hybridization with other optimization algorithms, such as ACO, WOA, and SHO to enhance exploration and exploitation phases. This hybrid approach can address the limitations of IBOA, such as convergence to local optima, and provide more robust solutions for complex, multi-objective scheduling problems in cloud-fog environments. Hybrid models could also improve the algorithm's performance in dynamic workloads and heterogeneous environments.

5. Conclusions and Future Work

In conclusion, our IBOA demonstrates cost-effectiveness, achieving significant reductions in overall cost by 19.65%, communication cost by 18.28%, and computation cost by 25.41% when compared to the existing models. Moving forward, future work should focus on further optimizing parameter settings, exploring hybridization with other optimization techniques, conducting extensive empirical evaluations across diverse scenarios, and addressing emerging challenges in dynamic and heterogeneous computing environments. By continuing to refine our model and its implementation, we can enhance its robustness, scalability, and applicability in real-world TS scenarios, ultimately advancing the state-of-the-art in optimization techniques for cloud and fog computing systems.

Conflict of Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

We declare that this manuscript is original, has not been published before, and is not currently being considered for publication elsewhere.

Availability of Data and Material

Data made available on request

Code Availability

Data made available on request

Authors' Contributions

The author confirms sole responsibility for the following: study conception and design, data collection, analysis and interpretation of results, and manuscript preparation.

Ethics Approval

This material is the authors' own original work, which has not been previously published elsewhere. The paper reflects the authors' own research and analysis in a truthful and complete manner.

References

[1] Nguyen, Binh Minh, et al., "Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment", Applied Sciences, Vol. 9, No. 9, pp. 1730, 2019. DOI: 10.3390/app9091730

[2] Ghasempour, Alireza., "Internet of things in smart grid: Architecture, applications, services, key technologies, and challenges", Inventions, Vol. 4, No. 1, pp. 22, 2019. DOI: 10.3390/inventions4010022

[3] Arora, Sankalap, and Satvir Singh. "Butterfly optimization algorithm: a novel approach for global optimization." Soft computing 23 (2019): 715-734. DOI: 10.1007/00500-018-3102-4

[4] Zuo, Liyun, et al., "A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing", Ieee Access, Vol. 3, pp. 2687-2699, 2015. DOI: 10.1109/ACCESS.2015.2508940

[5] Lin, Bing, et al., "A pretreatment workflow scheduling approach for big data applications in multicloud environments", IEEE Transactions on Network and Service Management, Vol. 13, No. 3, pp. 581-594, 2016. DOI: 10.1155/2020/8105145

[6] Lin, Xue, et al., "Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment", IEEE Transactions on Services Computing, Vol. 8, No. 2, pp.175-186, 2014. DOI: 10.1109/TSC.2014.2381227

[7] Cheng, Feng, et al., "Cost-aware job scheduling for cloud instances using deep reinforcement learning", Cluster Computing, pp. 1-13, 2022. DOI: 10.1007/s10586-021-03436-8

[8] Zhou, Zhou, et al., "A modified PSO algorithm for task scheduling optimization in cloud computing", Concurrency and Computation: Practice and Experience, Vol. 30, No. 24, pp. e4970, 2018. DOI: 10.1002/cpe.4970

[9] Jangu, Nupur, and Zahid Raza., "Improved Jellyfish Algorithm-based multi-aspect task scheduling model for IoT tasks over fog integrated cloud environment", Journal of Cloud Computing, Vol. 11, No. 1, pp. 1-21, 2022. DOI: 10.1186/s13673-019-0174-9

[10] Singh, Gyan, and Amit K. Chaturvedi., "Hybrid modified particle swarm optimization with genetic algorithm (GA) based workflow scheduling in cloud-fog environment for multi-objective optimization", Cluster Computing, pp. 1-18, 2023. DOI:

10.1371/journal.pone.0003197
[11] Zahra, Movahedi, Defude Bruno, and Amir mohammad Hosseininia., "An efficient population-based multi-objective task scheduling approach in fog computing systems." Journal of Cloud Computing, Vol. 10, No. 1, 2021. DOI: 10.1016/j.jocs.2023.102152
[12] Iftikhar, Sundas, et al., "HunterPlus: AI based energy-efficient task scheduling for cloud–fog computing environments", Internet of Things Vol. 21, pp. 100667, 2023. DOI: 10.1016/j.iot.2022.100674
[13] Yin, Zhenyu, et al., "A multi-objective task scheduling strategy for intelligent production line based on cloud-fog computing", Sensors, Vol. 22, No. 4, pp. 1555, 2022. DOI: 10.3390/s22041555
[14] Pham, Xuan-Qui, et al., "A cost-and performance-effective approach for task scheduling based on collaboration between cloud and fog computing", International Journal of Distributed Sensor Networks, Vol. 13, No. 11, pp. 1550147717742073, 2017. DOI: 10.1177/1550147717742073
[15] Mangalampalli, Sudheer, Ganesh Reddy Karri, and Mohit Kumar., "Multi objective task scheduling algorithm in cloud computing using grey wolf optimization", Cluster Computing, pp. 1-20, 2022. DOI: 10.1109/JIOT.2023.3291367
[16] Hosseinioun, Pejman, et al., "A new energy-aware tasks scheduling approach in fog computing using hybrid meta-heuristic algorithm", Journal of Parallel and Distributed Computing, Vol. 143, pp. 88-96, 2020. DOI: 10.1016/j.jpdc.2020.04.008
[17] Liu, Lindong, et al. "A task scheduling algorithm based on classification mining in fog computing environment." Wireless Communications and Mobile Computing, 2018. DOI: 10.1155/2018/2102348
[18] Bakshi, Mohana, Chandreyee Chowdhury, and Ujjwal Maulik., "Cuckoo search optimization-based energy efficient job scheduling approach for IoT-edge environment", The Journal of Supercomputing, pp. 1-29, 2023. DOI: 10.3390/s23052445
[19] Badri, Sahar, et al., "An Efficient and Secure Model Using Adaptive Optimal Deep Learning for Task Scheduling in Cloud Computing", Electronics, Vol. 12, No. 6, pp. 1441, 2023. DOI: 10.3390/electronics12061441
[20] Ahmed, Omed Hassan, et al., "Using differential evolution and Moth–Flame optimization for scientific workflow scheduling in fog computing", Applied Soft Computing, Vol. 112, pp. 107744, 2021. DOI: 10.1016/j.asoc.2021.107744
[21] Mangalampalli, Sudheer, Sangram Keshari Swain, and Vamsi Krishna Mangalampalli. "Multi objective task scheduling in cloud computing using cat swarm optimization algorithm." Arabian Journal for Science and Engineering 47.2 (2022): 1821-1830. DOI: 10.1007/s13369-021-06076-7
[22] Sindhu, V., and M. Prakash., "Energy-efficient task scheduling and resource allocation for improving the performance of a cloud–fog environment", Symmetry, Vol. 14, No.11, pp. 2340, 2022. DOI: 10.3390/sym14112340
[23] Kumar, M. Santhosh, and Ganesh Reddy Karri., "Eeoa: cost and energy efficient task scheduling in a cloud-fog framework", Sensors, Vol. 23, No. 5, pp. 2445, 2023. DOI: 10.3390/s23052445
[24] Arora, Sankalap, and Satvir Singh. "Butterfly optimization algorithm: a novel approach for global optimization." *Soft computing* 23 (2019): 715-734. https://doi.org/10.1007/s00500-018-3102-4
[25] Dorigo, Marco, Mauro Birattari, and Thomas Stutzle. "Ant colony optimization." *IEEE computational intelligence magazine* 1.4 (2006): 28-39. DOI: 10.1109/MCI.2006.329691
[26] Wang, Dongshu, Dapei Tan, and Lei Liu. "Particle swarm optimization algorithm: an overview." *Soft computing* 22.2 (2018): 387-408. https://doi.org/10.1007/s00500-016-2474-6

Authors' Profiles

Santhosh Kumar Medishetti submitted his Ph.D. thesis at VIT-AP University, Amaravati, India, in 2024. He is currently working as an Assistant Professor at Nalla Narasimha Reddy Education Society's Group of Institutions, Hyderabad, Telangana, India. He has over 5 years of experience in both research and teaching. He has published more than 20 research articles in various reputed journals, book chapters, patents, and conferences. He is currently a Senior Member of the EAI professional body and a member of ACM. His research interests include cloud computing, fog computing, and task scheduling.

Ganesh Reddy Karri received the Ph.D. degree from NIT-Suratkal, Karnataka, India, in 2014. he is currently an Associate Professor in the School of Computer Science and Engineering (SCOPE), VIT-AP University, Amaravati, India. Currently his coordinator for center of excellence in cyber security and also an IEEE member. Currently his guiding for five research scholars. He has more than 10 years' experience in both Research and Teaching. he has already published more than 50 Research articles in various reputed journals, book chapters, and conferences. His research interests include Cloud computing, Computer and Network Security, Wireless networks, Data structures and Algorithms and the IoT.

Rakesh Kumar Donthi his currently pursuing a post-doctoral in School of Computer Science and Engineering, at university college of dublin, Ireland. He received the Ph.D. degree from NIT-Patna, Bihar, India, in 2021 also received the Master's degree in computer science and Engineering from JNTUH College of engineering, Hyderabad, India in 2013. He has more than 8 years' experience in both Research and Teaching. he has already published more than 10 Research articles in various reputed journals. His research interests are Machine learning, ontology, semantic web, deep learning and cloud computing.

How to cite this paper: Santhosh Kumar Medishetti, Ganesh Reddy Karri, Rakesh Kumar Donthi, "IBOA: Cost-aware Task Scheduling Model for Integrated Cloud-fog Environments", International Journal of Information Technology and Computer Science(IJITCS), Vol.16, No.5, pp.52-68, 2024. DOI:10.5815/ijitcs.2024.05.04