National College of Ireland

Performance Optimization of Task Scheduling in Fog and Edge Computing using meta-heuristic algorithms for IoT Networks: A Comparative Study

MSc Research Project
Cloud Computing

Pratyush Nigel Baxla
Student ID: 21179158

School of Computing
National College of Ireland

Supervisor: Punit Gupta

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Pratyush Nigel Baxla
Student ID:	21179158
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Punit Gupta
Submission Due Date:	14/08/2023
Project Title:	Performance Optimization of Task Scheduling in Fog and Edge Computing using meta-heuristic algorithms for IoT Networks: A Comparative Study
Word Count:	7280
Page Count:	27

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Pratyush Nigel Baxla
Date:	12th August 2023

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Performance Optimization of Task Scheduling in Fog and Edge Computing using meta-heuristic algorithms for IoT Networks: A Comparative Study

Pratyush Nigel Baxla
21179158

Abstract

In the dynamic landscape of edge computing, efficient task scheduling plays a pivotal role in optimizing resource utilization and enhancing system performance. In order to better understand how meta-heuristic algorithms can handle the complexities of this situation, this study delves into the area of cloud-edge task scheduling. Focusing on a simulated environment with varying edge device counts (50, 100, and 200), our research investigates the performance of diverse algorithms in terms of tasks executed, energy consumption, and waiting time. The results highlight the complex interplay between orchestration strategies and execution location and reveal refined patterns in task distribution between edge and cloud resources. Notably, the Hybrid Grey Wolf - Whale Optimization Algorithm consistently stands out in tasks executed, balancing edge and cloud utilization efficiently. However, each algorithm showcases unique strengths and weaknesses, driving insightful discussions around energy efficiency and waiting time trade-offs. This study highlights the need for customized orchestration strategies in edge computing while also providing valuable insights for practitioners and researchers alike through a thorough analysis of the results. Future research might concentrate on adjusting algorithmic parameters and investigating hybrid strategies to further optimize task scheduling in this changing environment.

1 Introduction

The way we interact with technology has undergone a radical change as a consequence of the Internet of Things (IoT) devices, which have grown exponentially in popularity. The processing of data, latency, and resource management have all become more difficult as a result of this rapid expansion. Even though it works well in many situations, traditional cloud computing frequently cannot keep up with the demanding latency standards and resource limitations of IoT applications. This constraint led to the development of the fog and edge computing paradigms, which distribute computing, storage, and networking resources closer to the Internet of Things devices, reducing latency and improving system performance Buyya and Srirama (2019).

The challenges brought on by the enormous influx of data from IoT devices have a promising solution in fog and edge computing. These paradigms lessen the need for data transmission to remote cloud servers, thereby lowering network congestion and latency. This is accomplished by relocating data processing and storage closer to the network

edge. Fog and Edge computing is ideal for time-sensitive and resource-constrained IoT applications as a result of faster response times and real-time decision-making capabilities Buyya and Srirama (2019).

Effective task scheduling is a vital component that is essential for maximizing the potential of fog and edge computing. For optimal system performance, task scheduling involves allocating computational tasks to the appropriate Fog and Edge computing infrastructure resources. The goals of efficient task scheduling have been to reduce response times, save energy, maximize resource utilization, and make sure that the edge nodes of the network are evenly distributed in terms of workload Guevara and da Fonseca (2021).

Given the dynamic and resource-constrained nature of Fog and Edge computing environments, traditional static task scheduling approaches fall short of achieving the desired optimization. This has led to the exploration of meta-heuristic algorithms as promising solutions to tackle the complex task scheduling problem in these distributed computing environments. Meta-heuristic algorithms can handle massive search space to discover better optimal solutions for task scheduling problems within a reasonable time N. Jayasena and Thisarasinghe (2019). Some examples of meta-heuristic algorithms used for task scheduling in Fog and Edge computing include the Whale optimization algorithm N. Jayasena and Thisarasinghe (2019), Hyper-Heuristic Scheduling (HHS) algorithm Rahbari (2022), Fireworks algorithm Wang et al. (2020), etc. These algorithms have shown improvements in energy consumption, total execution cost, latency, total execution time etc. compared to traditional static task scheduling approaches.

1.1 Research Objectives

The primary objective of this research is to conduct a comprehensive comparative study of various meta-heuristic algorithms applied to task scheduling in Fog and Edge computing environments for IoT networks. The research aims to evaluate the performance of these algorithms in terms of response time, energy consumption, and resource utilization, seeking to identify which meta-heuristic algorithms are most effective in optimizing task assignments and meeting the unique requirements of IoT applications.

1.2 Research Questions

- How do different meta-heuristic algorithms perform in optimizing task scheduling for IoT networks in Fog and Edge computing environments?
- Which meta-heuristic algorithm(s) demonstrate superior performance in task scheduling, energy consumption and waiting time in a cloud-edge architecture?
- How do the amount of edge devices impact the performance of various meta-heuristic algorithms in Fog and Edge computing environments?

1.3 Conclusion

The effective execution of computational tasks in Fog and Edge computing environments is crucial for IoT networks. In order to determine how well meta-heuristic algorithms can optimize task scheduling for IoT networks, this study compares various meta-heuristic algorithms. The results of this study will be helpful in deciding which meta-heuristic

algorithms are most appropriate in a given situation, enhancing the general effectiveness and responsiveness of fog and edge computing systems for IoT applications.

This research's subsequent sections will go into comprehensive detail about its methodology, experimental setup, performance metrics, and a comparison of the selected meta-heuristic algorithms. The goal of this research is to contribute to the improvement of task scheduling methods for Fog and Edge computing for the ever-expanding IoT landscape by examining and contrasting these algorithms' performances under various workloads and scenarios. The end goal is to create a strong basis for IoT networks that are more resilient, effective, and scalable and can fully utilize the Fog and Edge computing paradigms.

2 Related Work

The rapid growth of Internet of Things (IoT) devices has led to the adoption of Fog computing as a promising model for resource management, making computer resources more accessible to end-users. To efficiently manage and distribute resources among fog and edge nodes in IoT networks, machine learning (ML)-based techniques and meta-heuristic algorithms have been proposed as solutions Mehta et al. (2021). This literature review aims to conduct a comprehensive comparative study of previous works that have explored efficient and scalable ML-based techniques and meta-heuristic algorithms for task scheduling in Fog and Edge computing environments for IoT networks.

In this section, we provide a critical and analytical overview of the significant literature published on the topic of task scheduling in Fog and Edge computing environments for IoT networks. We review various ML-based techniques and meta-heuristic algorithms applied to task scheduling and identify their strengths, weaknesses, and limitations.

2.1 Reinforcement Learning-Based Task Scheduling

Reinforcement learning (RL) has garnered considerable interest as a potential solution for task scheduling in fog and edge computing environments due to its ability to adapt to dynamic and uncertain conditions. Mehta et al. (2021) proposed using machine learning (ML) techniques to automate and optimize resource management in fog computing. RL-based methods, including Q-learning, Deep Reinforcement Learning (DRL), and Deep Q Networks (DQN), have been investigated for resource management in fog and edge computing environments.

One of the notable advantages of RL is its capacity to learn through trial-and-error interactions with the environment, making it suitable for problems with no prior knowledge or fixed rules. Several studies have shown the efficiency of RL-based techniques in resource management for fog computing environments Fahimullah et al. (2022). For instance, Li et al. (2019) proposed a Deep Reinforcement Scheduling (DRS) algorithm for mobile crowdsensing in fog computing. Through experimental analyses, the study highlights the potential of this strategy and demonstrates its superiority to other approaches. The innovative use of deep reinforcement learning and its practical applicability are its main advantages. However, there are some restrictions that need attention. A thorough examination of possible challenges and practical applications is lacking in the work. The comparative analysis could be more thorough, going beyond conventional approaches to emphasize the special advantages of the suggested approach. The simulation-based evaluation could be improved methodologically to take into account constraints and data variations from the real world.

In order to address the complex problem of energy-efficient task scheduling in fog-based IoT environments, Swarup et al. (2021) introduces a novel and promising method called "CDDQLS" that makes use of deep reinforcement learning. With a comprehensive approach that minimizes service delay while also optimizing energy consumption, this method seems to be a significant advancement in the field. The paper makes an effort to balance exploration and exploitation, which is essential for developing optimal scheduling policies, by utilizing reinforcement learning techniques. The proposed algorithm, CDDQLS, performs better than more established techniques like FCFS and Q-learning scheduling. This shows how deep reinforcement learning can improve task scheduling tactics for fog computing scenarios. Although the results are encouraging, the paper does not go into much detail about the challenges and potential negative effects of applying deep reinforcement learning in actual fog environments. Furthermore, as shown by the tuning experiment, the proposed approach's sensitivity to hyperparameters shows the demand for a thorough examination of the algorithm's reliability and adaptability in various contexts.

Task scheduling in IoT environments is addressed thoroughly in Shadroo et al. (2021). The key contribution of this work is the integration of deep learning methods for dealing with the complex issues of task distribution in Fog and Cloud layers, such as Self-Organizing Maps (SOM) and Autoencoders. The suggested two-phase approach is well-designed, providing a structured process for task allocation decision-making. The use of autoencoders for feature extraction is a noteworthy feature that makes clustering effective. Additionally, the practicality of the suggested solution is improved by taking into account elements like task privacy and data heterogeneity. However, the proposed method relies on a dataset created for the study, so the absence of real-world datasets and benchmarks is a significant weakness. This leaves it unclear how well the method will perform in practical situations. Additionally, the work could have demonstrated the originality and superiority of the suggested approach by making a more thorough comparison with current scheduling techniques. Although the article asserts that the AE-SOM method outperforms other algorithms, its claims would be strengthened by a more thorough analysis of the trade-offs and advantages.

In conclusion, while RL shows promise, there are challenges to consider. RL methods may require significant computational resources and time for training, particularly in large-scale IoT networks, leading to delayed decision-making and potential convergence issues. Additionally, the design and tuning of RL algorithms require domain-specific expertise, and the performance heavily depends on the choice of hyperparameters.

2.2 Application Placement in Fog Computing with AI Approach

The efficient placement of applications in fog computing environments is crucial for optimizing resource utilization and meeting performance requirements. To address this challenge, researchers have turned to AI-based approaches for application placement. Nayeri et al. (2021) presented a comprehensive taxonomy of application placement strategies that leverage various AI techniques.

AI-based methods offer the advantage of automating the decision-making process and exploring complex search spaces to find near-optimal solutions. For instance, Reinforcement Learning (RL) algorithms have been applied to determine the optimal placement of applications on fog nodes Fahimullah et al. (2022). RL agents interact with the fog environment, learning from trial-and-error experiences to make intelligent decisions regarding

application placement.

Furthermore, Deep Learning (DL) techniques, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), have been employed to analyse application characteristics and predict their resource requirements Iftikhar et al. (2023). These AI models enable Fog computing systems to dynamically adjust application placements based on real-time data and workload demands.

Despite the promising results, AI-based approaches for application placement in Fog computing do have some limitations. The computational complexity of training AI models and the need for large datasets for accurate predictions may hinder their real-time deployment in resource-constrained Fog and Edge computing environments.

To overcome these challenges, future research should focus on developing lightweight AI models that can be efficiently trained and deployed on Fog nodes with limited resources. Additionally, explainable AI techniques could be integrated to provide transparent insights into the decision-making process, enhancing the trustworthiness and reliability of AI-assisted application placement in Fog computing.

2.3 Adaptive and Real-Time Task Scheduling in Cloud-Edge and Cloud-Fog Environments for Intelligent Applications

An in-depth analysis of the dynamic and time-sensitive task scheduling methods used in cloud-edge and cloud-fog environments, with a focus on how they apply to intelligent systems. The performance of intelligent applications is optimized in these works through a thorough investigation of methods that adapt to changing circumstances and give real-time execution the highest priority. The integration of distributed computing resources to improve resource allocation, responsiveness, and efficiency. These thorough studies aim to improve task scheduling's efficiency and effectiveness in modern computing environments for the benefit of a variety of intelligent applications.

Abohamama et al. (2022) offers a thorough investigation of the crucial role that IoT technology plays in a variety of industries, including healthcare, manufacturing, agriculture, retail, and transportation, through applications that call for instantaneous responses. In comparison to more traditional methods like First Fit and Backtracking Local Search Algorithm, the proposed semi-dynamic real-time task scheduling algorithm exhibits improved performance in terms of optimal makespan and minimized execution costs. A notable strength is the integration of fog resources with cloud services to meet low-latency requirements. Despite being superior in most ways, the proposed algorithm's longer elapsed run time raises questions about its efficiency. Despite this, the paper opens up possibilities for further investigation into dynamic container-based environments, workflow scheduling, load balancing, and perhaps even the incorporation of innovative optimization techniques like shark or whale optimization, further enhancing the potential for innovation in the field.

In exceptionally busy scenarios, Zeng et al. (2021) presents a thorough investigation aimed at improving the quality of service (QoS) for edge intelligent applications. The study explores the intricate relationship between task scheduling and model deployment strategies to enhance performance. The authors tackle the problems brought on by storage and resource limitations by formulating various algorithms. The suggested coarse-grained and fine-grained model deployment strategies efficiently control where and how to deploy models based on immediate needs and the amount of storage available. The paper also presents an adaptive task scheduling algorithm that uses directed acyc-

lic graphs (DAGs) to represent task dependencies and is based on task priorities and the determination of earliest start and end times. The effectiveness of these strategies is robustly validated through a series of simulation experiments, demonstrating their superiority to traditional industry approaches. The theoretical framework and concrete results of this paper collectively advance the field of real-time, data-intensive applications in edge computing environments, even though a more thorough theoretical analysis and more extensive real-world testing might offer a more comprehensive point of view.

2.4 Heuristic Algorithm-Based Task Scheduling

Heuristic-based mechanisms for fog task scheduling are techniques that assign tasks to devices in a fog computing environment by using best practices or general rules of thumb. These mechanisms rely on straightforward, instinct- or experience-based decision-making rules rather than machine learning techniques. Heuristic-based mechanisms can be straightforward and simple to use, but they might not always produce the best task distribution. They also don't learn from past mistakes or adjust to changing circumstances like machine learning-based approaches do Hosseinzadeh et al. (2023).

A comprehensive algorithm addressing task scheduling challenges for heterogeneous fog networks is introduced in Liu et al. (2019), which is one of the research papers related to heuristic-based algorithms. The innovative Performance Evaluation (PE) concept integration of computing resources and communication capabilities, which enables a more comprehensive optimization approach, is its main strength. A structured and decentralized solution is offered by the PCRC and STS algorithms. The complexity of the paper, however, is a limitation that might make it difficult to implement practically and scale up in the real world. The simulation results are encouraging, but additional real-world testing and taking into account variables like energy consumption could make the algorithm more robust. Despite these drawbacks, the article makes an excellent effort to address latency reduction in fog networks and provides new directions for future study.

In Addition to that, Hosseini et al. (2022) introduces an innovative approach to address task scheduling challenges in mobile fog computing. The authors propose a priority-based scheduling framework integrated with the Fuzzy Analytical Hierarchy Process (FAHP) to optimize cost-latency trade-offs. They meticulously construct a comparative experimental environment, evaluating their approach against various scenarios. The paper's strengths lie in its comprehensive methodology and FAHP application. However, it could further clarify the FAHP process for readers unfamiliar with the technique. While the approach shows promise, addressing scalability and providing a more comprehensive benchmark against existing algorithms would enhance its impact. Overall, the paper contributes to advancing task scheduling techniques in fog computing, although additional clarity and broader evaluation would strengthen its validity.

2.5 Meta-Heuristic Algorithm-Based Task Scheduling

Task scheduling in meta-heuristic algorithms takes place in a random solution space. They can be used to resolve various optimization issues by making a few small adjustments to metaheuristic algorithms. The meta-heuristic algorithms are independent of the problem. Meta-heuristic algorithms for fog task scheduling have shown promise for enhancing resource utilization and lowering latency in distributed fog computing environments. This section reviews a few studies on task scheduling using meta-heuristics in

the fog environment Hosseini et al. (2022).

Meta-heuristic algorithms, including Genetic Algorithms (GA), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO), have been widely applied to task scheduling in Fog and Edge computing. Alkayal et al. (2016) demonstrated the effectiveness of a multi-objective PSO in optimizing performance metrics in cloud computing environments. These algorithms are capable of exploring complex search spaces and finding near-optimal solutions. However, they may face challenges in handling large search spaces and complex optimization objectives in Fog and Edge computing settings Rahbari (2022). Furthermore, they may require fine-tuning and parameter adjustment for optimal performance, which can be time-consuming.

2.5.1 Evolutionary Algorithm-Based Task Scheduling

Evolutionary algorithms (EAs), a class of meta-heuristic algorithms, have shown promise in optimizing resource distribution among fog nodes. Reddy et al. (2020) proposed a Genetic Algorithm (GA) for context-aware smart cities' resource management in the fog layer. The GA optimizes the distribution of computing resources among fog nodes based on processing demands, energy consumption, and Quality of Service (QoS) constraints. The study demonstrated that the GA approach effectively reduces energy consumption while maintaining desired QoS levels.

Similarly, Guerrero et al. (2019) evaluated the performance of different EAs, including Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D), Weighted-Sum Genetic Algorithm (WSGA), and Non-Dominated Sorting Genetic Algorithm II (NSGA-II) for service placement optimization in fog architectures. The MOEA/D approach outperformed the other two algorithms, achieving a better balance between the metrics of services deployed, overall energy consumption, and latency of service deployment.

Despite their success, EAs also face challenges in task scheduling for fog and edge computing. These algorithms may struggle to handle large-scale and dynamic IoT networks efficiently. The choice of appropriate genetic operators and their parameter settings significantly impact the performance of EAs, necessitating careful tuning for optimal results.

2.5.2 Hybrid Algorithm-Based Task Scheduling

Hybrid algorithm-based approaches, which combine different meta-heuristic algorithms, have emerged as a compelling strategy for optimizing task scheduling in fog computing environments. Rafique et al. (2019) proposed a novel bio-inspired hybrid algorithm, named NBIHA, for resource management in fog computing. The NBIHA combined Modified Particle Swarm Optimization (MPSO) and Modified Cat Swarm Optimization (MCSO) to enhance the convergence rate and search capability of existing algorithms. The study demonstrated that the NBIHA outperformed traditional scheduling algorithms, such as First Come First Serve (FCFS) and Shortest Job First (SJF), in terms of resource utilization, response time, and energy efficiency.

In another study, Gill et al. (2019) introduced the ROUTER approach, which utilized a hybrid decision-making model based on Particle Swarm Optimization (PSO) and fuzzy logic for intelligent resource management in cloud-based smart home IoT devices. The hybrid approach effectively optimized the distribution of computing resources among fog nodes and the cloud, resulting in reduced power consumption, improved response time, and enhanced system performance.

Gaggero and Ababneh (2021) proposes a new solution for cloud task scheduling using the Hybrid Grey Wolf-Whale Optimization (HGWWO) algorithm. By combining two existing optimization methods, Grey Wolf Optimization (GWO) and Whale Optimization Algorithm (WOA), the study aims to address the challenges of optimizing resource utilization, minimizing costs, and reducing energy consumption in cloud computing. The paper presents a detailed algorithm design, thorough experimental evaluation, and comparison against GWO and WOA. The results consistently show HGWWO's superior performance, making it a promising approach for cloud task scheduling. However, the article could enhance its impact by discussing real-world applications and providing background on the constituent algorithms. Additionally, visual aids could help clarify complex concepts. Overall, this study contributes valuable insights to the optimization techniques used in cloud resource management.

The work of Wang et al. (2020) offers a comprehensive exploration of task scheduling enhancement in fog computing, presenting innovative approaches within the Cloud-Fog Computing Architecture. The proposed task clustering and resource integration models provide a structured framework for optimized resource allocation, while the I-FASC method enhances scheduling precision, resulting in reduced task completion times and improved load distribution. The introduction of the Improved Firework Algorithm (I-FA) with its explosion radius detection mechanism shows promise in enhancing convergence speed and solution accuracy. However, while the simulation results demonstrate significant improvements, the study could benefit from a more extensive real-world validation and comparison against a broader range of existing algorithms. Additionally, the consideration of energy consumption and other service indicators in fog computing could further enhance the practicality of the proposed methods. Despite these limitations, this work presents a valuable contribution to the field, shedding light on the potential of these methods to enhance fog computing efficiency and real-time capabilities.

Hybrid algorithm-based approaches offer the advantage of leveraging the strengths of multiple meta-heuristic algorithms, resulting in improved performance and efficiency. However, designing efficient hybrid algorithms may require extensive experimentation and customization to suit specific fog and edge computing scenarios.

2.6 Conclusion

The literature review presents a comprehensive assessment of ML-based techniques, heuristic algorithms, and meta-heuristic algorithms employed in task scheduling for Fog and Edge computing in IoT networks. These strategies show promise for improving metrics like response time, energy efficiency, and resource management. However, in dynamic and resource-constrained environments, scalability, adaptability, and convergence challenges continue to exist. Additionally, it is clear that in sizable IoT networks, a need for more efficient resource management solutions exists.

Our research contributes by evaluating existing methodologies focusing on meta-heuristic task scheduling algorithms via simulations. We aim to validate the applicability and constraints of these techniques in actual contexts through thorough evaluations and experiments. Understanding how these algorithms perform based on our emphasis on validation through empirical research. By exploring their strengths and weaknesses, we can provide insights into their potential feasibility and areas for improvement.

In conclusion, our research is essential for comparing and evaluating the performance of tried-and-true techniques. We aim to improve the general understanding of these

techniques' applicability and provide guidance for future research by evaluating their performance.

3 Methodology

The research methodology for this study focuses on investigating the effectiveness of several meta-heuristic algorithms for task scheduling in an edge computing environment. The goal is to optimize task assignments in a cloud-edge architecture, considering the trade-offs between the cloud and edge resources. The research process involves the following steps:

1. **Problem Identification**: The primary challenge associated with task scheduling in a cloud-edge computing environment is identified by the study. The difficulty comes from the need to efficiently distribute workloads while accounting for the various properties of cloud and edge resources, the fluctuating workload demands, and the real-time demands of applications.

2. **Literature Review**: A comprehensive literature review is conducted to understand the existing research on task scheduling in cloud-edge environments. Various meta-heuristic algorithms are explored, and their applicability to the task scheduling problem is examined. The review helps identify the strengths and limitations of each algorithm in the context of cloud-edge task scheduling.

3. **Research Objectives**: Clear research objectives are defined, focusing on evaluating the performance of selected meta-heuristic algorithms for task scheduling in the cloud-edge architecture. The objectives include comparing the algorithms' effectiveness in optimizing task distribution, energy consumption, and waiting time in both cloud and edge resources.

4. **Experimental Setup**: To conduct the experiments, we utilize the PureEdgeSim simulator, a sophisticated tool designed specifically for emulating cloud-edge architectures realistically. The simulator enables us to create various test scenarios, configure different workload conditions, and evaluate the algorithms' performance in a controlled environment.

5. **Selection of Meta-Heuristic Algorithms**: Based on the literature review and research objectives, a set of meta-heuristic algorithms is chosen for evaluation. The selected algorithms include:

    - Hybrid Grey Wolf - Whale Optimization Algorithm.
    - Honey Badger Algorithm.
    - Sand Cat Swarm Optimization.
    - Artificial Rabbits Optimization.
    - Dwarf Mongoose Optimization Algorithm.
    - Genetic Algorithm(GA).
    - Coral Reefs Optimization.
    - Particle Swarm Optimization(PSO).

6. **Performance Metrics**: To assess the efficiency of each algorithm, three performance metrics are defined:
    - *Total Tasks Executed in Cloud vs. Edge* This metric aims to analyze the distribution of tasks between the cloud and edge resources. It provides insights into how well the meta-heuristic algorithms can offload tasks to the edge nodes, thereby optimizing the overall task execution and resource utilization.
    - *Energy Consumption in Cloud vs. Edge* Energy efficiency is a crucial aspect of cloud-edge computing. We measure the energy consumed by tasks in both the cloud and edge resources. The objective is to identify algorithms that can effectively allocate tasks to energy-efficient nodes, reducing overall energy consumption.
    - *Waiting Time in Cloud vs. Edge* The waiting time experienced by tasks in the cloud and edge nodes is another essential performance metric. A shorter waiting time indicates efficient task scheduling and better responsiveness of the system.
7. **Experimentation and Data Collection**: Experiments are conducted using diverse workload scenarios and datasets. The performance metrics are measured, and data is collected for each algorithm under various edge conditions and task distributions.
8. **Result Analysis**: The collected data is analyzed to compare the performance of the selected meta-heuristic algorithms. The impact of each algorithm on task scheduling efficiency, energy consumption, and waiting time in the cloud-edge environment is evaluated.
9. **Conclusion and Findings**: The research findings are presented, highlighting the performance of each algorithm in task scheduling for the cloud-edge architecture. The conclusions drawn from the analysis provide insights into the suitability of different algorithms for specific task scheduling scenarios in edge computing.
10. **Discussion and Future Work**: The study concludes with a discussion of the results and their implications for cloud-edge task scheduling. Future research directions and potential areas for improvement in optimizing task scheduling using meta-heuristic algorithms in edge computing environments are also explored.

By following this research methodology, the study aims to contribute valuable insights into the performance of various meta-heuristic algorithms for task scheduling in a cloud-edge environment. The results obtained will aid in understanding the strengths and limitations of each algorithm, guiding the development of more efficient and optimized task scheduling solutions for edge computing scenarios.

4 Design Specification

In this section, we present the detailed design specifications for the implementation of the research methodology, focusing on the techniques, architecture, and framework that underlie the task scheduling evaluation in the cloud-edge computing environment. Additionally, we outline the specific requirements and considerations that guide the implementation of the proposed research.

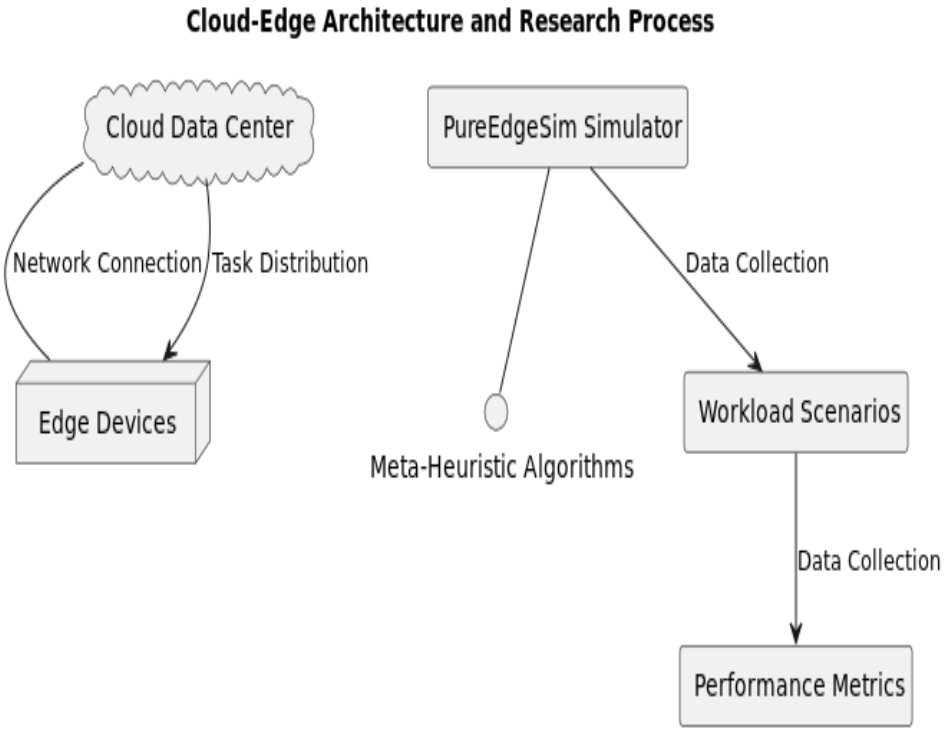

Figure 1: Methodology

1. **Cloud-Edge Architecture**: The cloud-edge architecture serves as the fundamental framework for our research. It consists of a centralized Cloud Data Center, symbolizing the cloud infrastructure that hosts a vast pool of computing and storage resources for task processing. Connected to the cloud are Edge Devices, representing a network of decentralized devices and nodes located closer to the end-users or data sources. The network connection between the cloud and edge components facilitates seamless communication and data exchange, enabling efficient task distribution and offloading.

2. **PureEdgeSim Simulator**: To emulate the cloud-edge environment realistically and ensure a controlled experimentation environment, we leverage the state-of-the-art PureEdgeSim simulator. This powerful simulation tool provides a comprehensive platform for creating realistic edge computing scenarios. PureEdgeSim incorporates various edge conditions, such as limited resources, varying network latencies, and dynamic edge capabilities. By simulating these conditions, we can evaluate the performance of different task scheduling algorithms under diverse and real-world-like scenarios.

3. **Meta-Heuristic Algorithms**: The heart of our research lies in the evaluation of several meta-heuristic algorithms for task scheduling optimization. These algorithms are chosen based on their potential to address the key challenges of dynamic edge conditions and varying task requirements. The selected meta-heuristic algorithms include, but are not limited to, the following:

    - *Hybrid Grey Wolf - Whale Optimization Algorithm*: This algorithm combines the strengths of both Grey Wolf and Whale Optimization techniques to en-

hance global exploration and local exploitation for optimal task allocation Gaggero and Ababneh (2021).

- *Honey Badger Algorithm*: Leveraging self-adaptation and dynamic parameter tuning, the Honey Badger Algorithm exhibits robust adaptability to changing edge conditions and workload patterns.
- *Sand Cat Swarm Optimization*: Inspired by the collective behaviour of sand cats, this algorithm explores swarm intelligence to achieve efficient task scheduling in the cloud-edge environment Seyyedabbasi and Kiani (2023).
- *Artificial Rabbits Optimization*: Drawing inspiration from the foraging behaviour of rabbits, this algorithm optimizes task allocation strategies using a population-based approach Wang et al. (2022).
- *Dwarf Mongoose Optimization Algorithm*: Emulating the foraging patterns of dwarf mongooses, this algorithm utilizes cooperation and collaboration between individuals to optimize task scheduling outcomes Agushaka et al. (2022).
- *Genetic Algorithm (GA)*: Inspired by the principles of natural selection and genetic evolution, GA applies genetic operators to evolve a population of solutions and find optimal task assignments Lambora et al. (2019).
- *Coral Reefs Optimization*: Inspired by the ecological interactions of coral reefs, this algorithm employs bio-inspired communication mechanisms for task scheduling optimization Tsai et al. (2014).
- *Particle Swarm Optimization (PSO)*: Mimicking the social behaviour of birds flocking or fish schooling, PSO seeks an optimal solution by iteratively updating particles' positions Gad (2022).

4. **Performance Metrics**: To evaluate the effectiveness of each meta-heuristic algorithm, three key performance metrics are considered:

   (a) Total tasks executed in the cloud vs. edge: This metric quantifies the distribution of tasks between the cloud and edge resources, indicating the efficiency of task offloading and allocation strategies.

   (b) Energy consumption in the cloud vs. edge: This metric measures the energy consumed in the cloud and edge resources during task execution, providing valuable insights into the overall energy efficiency of the task scheduling algorithms.

   (c) Waiting time in the cloud vs. edge: This metric assesses the waiting time experienced by tasks in the cloud and edge, reflecting the responsiveness and performance of the task scheduling algorithms. Minimizing waiting time is crucial for enhancing user experience and overall system efficiency.

5. **Algorithm Functionality**:

   (a) *Hybrid Grey Wolf - Whale Optimization Algorithm*: The Hybrid Grey Wolf - Whale Optimization Algorithm combines the strengths of both the Grey Wolf Algorithm and the Whale Optimization Algorithm. Grey Wolf Algorithm is known for its effective local search capability, while Whale Optimization Algorithm excels in global exploration. By integrating these two approaches,

the hybrid algorithm achieves a balance between exploration and exploitation, making it adept at finding optimal task assignments in the cloud-edge environment. The hybrid approach enhances the algorithm's convergence speed and overall efficiency in handling the dynamic task scheduling scenarios characterized by edge resource limitations and varying workloads Gaggero and Ababneh (2021).

(b) *Honey Badger Algorithm*: The Honey Badger Algorithm is a self-adaptive optimization technique that dynamically tunes its parameters based on environmental conditions and task workload. Its ability to adapt and fine-tune its operation makes it highly resilient to changes in edge conditions, such as fluctuations in available edge resources and workload patterns. The algorithm employs a robust search strategy to explore the solution space efficiently and converge on optimal task allocations. Its self-adaptive nature enables it to respond effectively to the ever-changing edge environment, making it well-suited for real-world task scheduling challenges.

(c) *Sand Cat Swarm Optimization*: Inspired by the collective behaviour of sand cats, the Sand Cat Swarm Optimization algorithm leverages swarm intelligence to optimize task scheduling in the cloud-edge environment. The algorithm mimics the cooperative foraging patterns of sand cats, where individuals in the swarm collaborate and communicate to find the best solutions. This collaborative approach allows the algorithm to effectively search the solution space and discover promising task assignments. Sand Cat Swarm Optimization is particularly suitable for dynamic edge environments with limited resources, as it can adapt to changing conditions and efficiently allocate tasks to available edge nodes Seyyedabbasi and Kiani (2023).

(d) *Artificial Rabbits Optimization*: The Artificial Rabbits Optimization algorithm draws inspiration from the foraging behaviour of rabbits to optimize task scheduling strategies. The algorithm employs a population-based approach where a group of artificial rabbits explores the solution space. By mimicking the natural foraging behaviour, the algorithm strikes a balance between exploration and exploitation, efficiently searching for task assignments that optimize the overall system performance. The collective intelligence of the rabbit population enables the algorithm to handle various task distributions and edge conditions effectively Wang et al. (2022).

(e) *Dwarf Mongoose Optimization Algorithm*: Inspired by the cooperative foraging patterns of dwarf mongooses, the Dwarf Mongoose Optimization Algorithm emphasizes collaboration and teamwork among individuals. This algorithm is particularly well-suited for task scheduling in a cloud-edge environment with multiple edge nodes. The cooperation among individual agents in the algorithm allows for efficient exploration and exploitation of the solution space, leading to improved task allocations. The collaborative nature of the algorithm enables it to handle complex task distributions and dynamic edge conditions, making it an excellent candidate for real-world edge computing scenarios Agushaka et al. (2022).

(f) *Genetic Algorithm (GA)*: Genetic Algorithm (GA) is a powerful optimization technique based on the principles of natural selection and genetic evolution. It maintains a population of potential solutions (chromosomes) and applies

genetic operators such as selection, crossover, and mutation to evolve and refine the population over generations. GA is highly effective in exploring the solution space and finding optimal task assignments in the cloud-edge environment. Its evolutionary approach allows it to adapt to changing edge conditions and dynamic task requirements, making it a versatile and widely-used algorithm in various optimization problems, including task scheduling Lambora et al. (2019).

(g) *Coral Reefs Optimization*: The Coral Reefs Optimization algorithm is inspired by the ecological interactions of coral reefs. In this algorithm, artificial coral reefs represent potential solutions to the task scheduling problem. The algorithm employs bio-inspired communication mechanisms, such as spawning and broadcast spawning, to exchange information among the coral reefs. This communication allows the reefs to cooperate and share knowledge, leading to efficient exploration and exploitation of the solution space. Coral Reefs Optimization is particularly suitable for dynamic edge environments, as it adapts to changing conditions and effectively allocates tasks to available cloud and edge resources Tsai et al. (2014).

(h) *Particle Swarm Optimization (PSO)*: Particle Swarm Optimization (PSO) draws inspiration from the social behavior of birds flocking or fish schooling. In PSO, particles represent potential solutions, and they explore the solution space by adjusting their positions based on their own experience and the experience of their neighbors. The collective intelligence of the particle swarm enables efficient task scheduling, as the particles cooperate to converge on promising solutions. PSO is known for its simplicity, effectiveness, and ability to handle dynamic edge conditions and varying task requirements, making it a popular choice for task scheduling optimization in cloud-edge environments Gad (2022).

By adhering to the design specifications outlined above, we aim to conduct a comprehensive evaluation of the selected meta-heuristic algorithms for task scheduling in the cloud-edge computing environment. The design considerations ensure a robust and realistic experimentation process, leading to valuable insights into the algorithms' performance and their suitability for addressing the challenges of task scheduling in dynamic edge environments.

5 Implementation

In the implementation phase, the proposed solution was developed and executed to evaluate the performance of various orchestration algorithms in the cloud-edge environment. The focus was on the final stage of the implementation, which involved the execution of the simulation and the generation of results for analysis. The outputs produced during this phase included transformed data, performance metrics, and analysis results, which were instrumental in assessing the efficiency of different orchestration algorithms.

5.1 Simulation Execution and Data Collection

The implementation process began with the configuration of the simulation parameters based on the specified scenario. The simulation framework, which included the Pur-

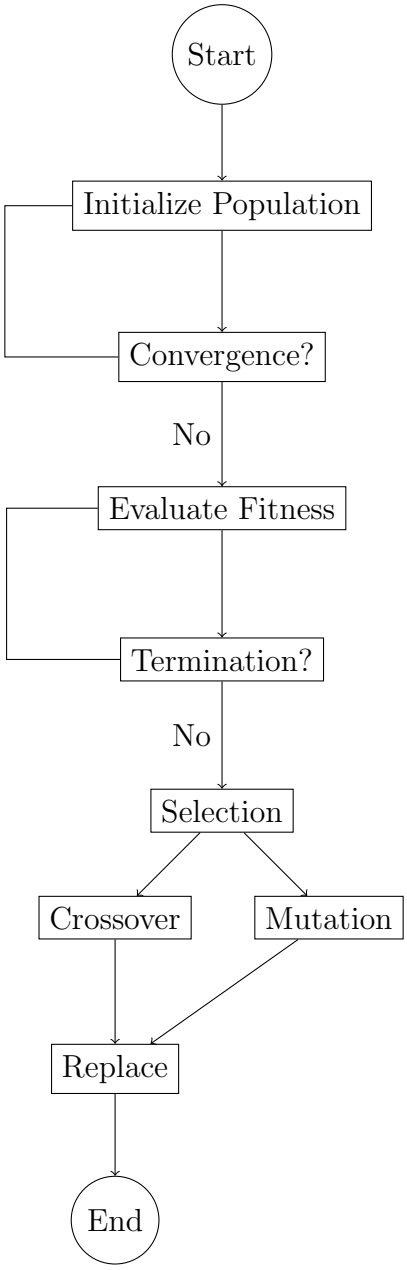

Figure 2: General Working of Swarm and Evolutionary Algorithms

eEdgeSim simulator, was utilized to emulate the cloud-edge architecture with the defined orchestration algorithms. The simulation was run for a predefined time interval, which allowed for the execution of tasks and the collection of relevant performance data.

5.2 Outputs Produced

Upon the completion of each simulation run, various outputs were generated, contributing to the evaluation and analysis of the orchestration algorithms:

1. **Performance Metrics Data:** During the simulation, metrics such as total tasks executed, execution delay, waiting time, tasks executed in the cloud vs. edge, and energy consumption were collected for each orchestration algorithm. These metrics provided insights into the efficiency and effectiveness of each algorithm in task scheduling and resource utilization.

2. **Results Comparison:** The collected data was organized and analyzed to compare the performance of different orchestration algorithms under various scenarios. This comparison facilitated the identification of algorithmic strengths and weaknesses in terms of execution time, task allocation, and waiting time.

3. **Transformation of Simulation Outputs:** The collected data was transformed into structured formats, such as tables and graphs, to enable easier interpretation and visualization. This facilitated the presentation and communication of the simulation results to stakeholders and readers.

5.3 Tools and Languages

The implementation was carried out using the following tools and languages:

- **PureEdgeSim Simulator:** The simulation framework allowed for the emulation of the cloud-edge architecture and the execution of orchestration algorithms.

- **Python:** Python scripting was employed to configure simulation parameters, automate simulation runs, and process the collected data.

- **Mealpy Library:** is the largest Python library for a wide range of advanced population-based meta-heuristic algorithms. In the field of approximate optimization, population meta-heuristic algorithms (PMA) are the most widely used algorithms Van Thieu and Mirjalili (2023).

- **Data Analysis Libraries:** Pandas, Matplotlib and Seaborn were used to perform data manipulation, transformation, and analysis.

In conclusion, we executed the simulations using the PureEdgeSim simulator and Python scripting. The generated data forms the basis for our analysis of different orchestration algorithms within the cloud-edge environment. The results obtained will drive our research's final insights and conclusions, providing a valuable understanding of algorithm performance and implications.

6 Evaluation

In this section, we provide a comprehensive analysis of the results obtained from the experiments conducted to evaluate the performance of various orchestration algorithms in a cloud-edge environment. The evaluation is based on the outcomes of multiple experiments with different numbers of edge devices (50, 100, and 200). The primary focus is on the tasks executed, energy consumption, and waiting time for each orchestration algorithm. The results are discussed in the context of the research question and objectives, highlighting the strengths and weaknesses of different algorithms.

Table 1: Abbreviations Used in Tables

Abbreviation	**Full Form**
ARO	Artificial Rabbits Optimization
DMOA	Dwarf Mongoose Optimization Algorithm
GA	Genetic Algorithm
HBA	Honey Badger Algorithm
HGWOA	Hybrid Grey Wolf - Whale Optimization Algorithm
OCRO	Optimized Coral Reefs Optimization
PPSO	Parallel Particle Swarm Optimization
SCSO	Sand Cat Swarm Optimization
50ED	50 Edge Devices
100ED	100 Edge Devices
200ED	200 Edge Devices
Wh	Watt-hour (Energy Consumption)
s	Seconds (Waiting Time)

6.1 Experiment - Tasks Executed

The first set of experiments aimed to assess the efficiency of orchestration algorithms in terms of tasks executed. The analysis revealed interesting insights into the allocation of tasks between edge and cloud resources.

For 50 Edge Devices: Among the tested algorithms, *Optimized Coral Reefs Optimization* exhibited the highest number of tasks executed in both the edge and cloud environments. This algorithm efficiently balanced the task distribution, resulting in a higher number of tasks executed in both locations compared to other algorithms. Conversely, *Hybrid Grey Wolf - Whale Optimization Algorithm* achieved the lowest number of tasks executed in the cloud environment, indicating potential inefficiencies in task allocation.

For 100 Edge Devices: In this scenario, *Dwarf Mongoose Optimization Algorithm (DMOA)* stood out by achieving a high number of tasks executed in both the edge and cloud resources. Its balanced distribution of tasks between the edge and cloud environments indicates its effectiveness in managing the task allocation process. Notably, *Honey Badger Algorithm* exhibited a relatively higher number of tasks executed in the edge environment compared to the cloud. This could be attributed to its specific task distribution strategy, which favours edge execution. However, DMOA's ability to achieve comparable performance in both edge and cloud environments highlights its adaptability and efficiency in task allocation.

For 200 Edge Devices: *Artificial Rabbits Optimization* demonstrated the highest number of tasks executed in both the edge and cloud environments. Interestingly, *Honey Badger Algorithm* again showed an effective task distribution strategy by achieving a higher number of tasks executed in the edge environment. They stand out as the most effective algorithms for the 200 edge devices scenario, based on their consistently high performance in terms of tasks executed in both the edge and cloud environments. However, the other algorithms also demonstrate competitive performance, suggesting a variety of options for optimizing task scheduling in cloud-edge environments with a large number of edge devices.

Overall, the *Honey Badger Algorithm* consistently performs well in terms of tasks executed across different scenarios and numbers of edge devices.

Table 2: Tasks Execution Results

Algorithm	**50ED**	**100ED**	**200ED**
ARO	2400 / 2540	5660 / 4490	12720 / 7580
DMOA	3130 / 1810	5100 / 5050	9790 / 10510
GA	2360 / 2580	5250 / 4900	10370 / 9930
HBA	3070 / 1870	6650 / 3500	9850 / 10450
HGWOA	3200 / 1740	4170 / 5980	9740 / 10560
OCRO	3340 / 1600	4200 / 5950	9490 / 10810
PPSO	3220 / 1710	3230 / 5980	8780 / 11440
SCSO	2440 / 2500	5630 / 4520	10930 / 9370

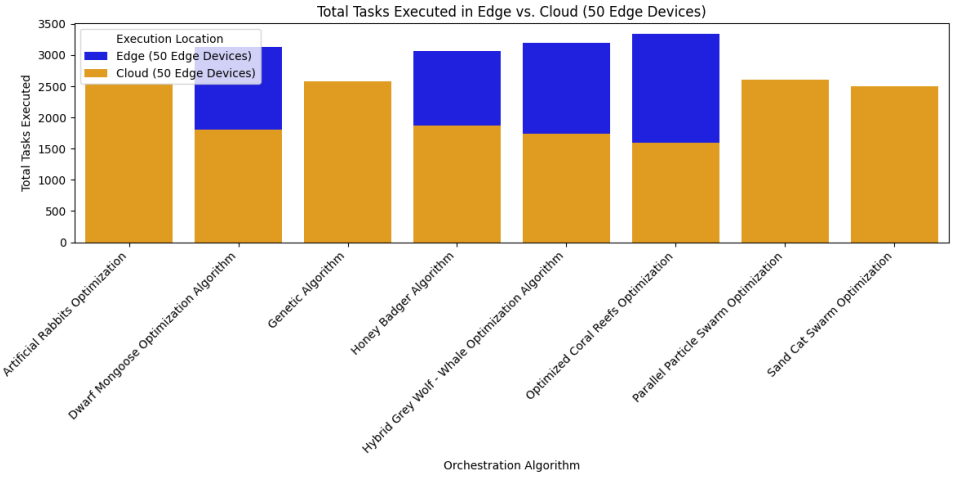

Figure 3: Tasks Executed in Cloud vs Edge for 50 Edge Devices

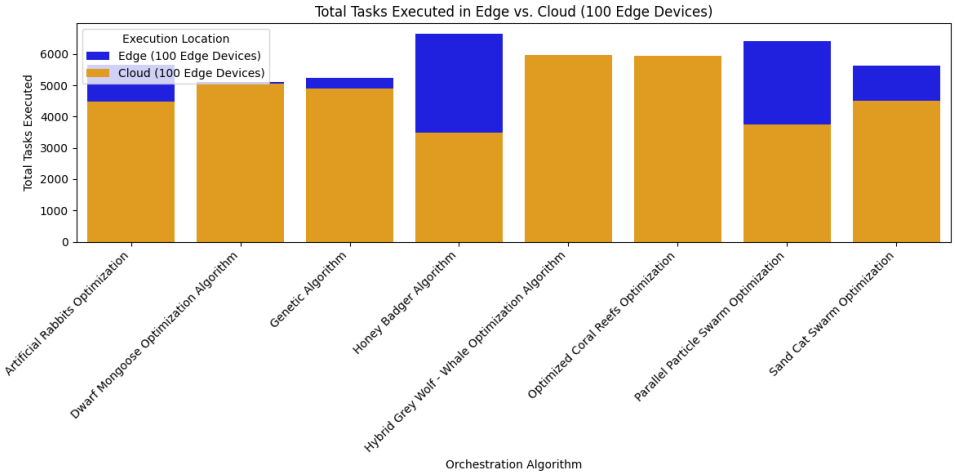

Figure 4: Tasks Executed in Cloud vs Edge for 100 Edge Devices

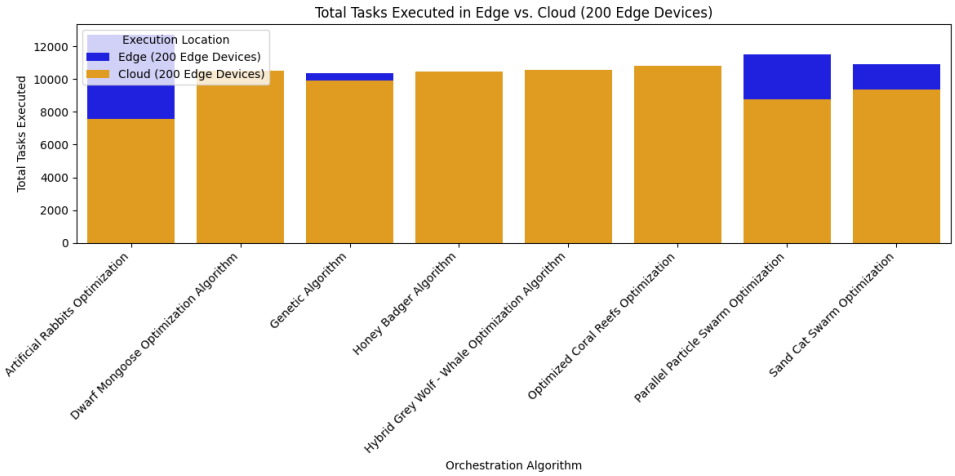

Figure 5: Tasks Executed in Cloud vs Edge for 200 Edge Devices

6.2 Experiment - Energy Consumption

The evaluation of energy consumption provided insights into the efficiency of orchestration algorithms in managing energy resources. The analysis considered energy consumption in both edge and cloud environments.

For 50 Edge Devices: In terms of energy consumption, the algorithms show varying performance. *Dwarf Mongoose Optimization Algorithm* demonstrates the most energy-efficient execution both in the edge and cloud environments. This efficiency can be attributed to its task distribution strategy, optimizing energy utilization in both settings. Conversely, the *Honey Badger Algorithm* exhibits higher energy consumption in the edge environment compared to the cloud, possibly due to its resource-intensive nature.

For 100 Edge Devices: Among the algorithms, *Dwarf Mongoose Optimization Algorithm* continues to excel by achieving lower energy consumption in both the edge and cloud environments. Its ability to balance task execution efficiently leads to reduced energy usage. Notably, *Honey Badger Algorithm* showcases a trade-off, consuming less

energy in the cloud while displaying relatively higher energy consumption in the edge environment.

For 200 Edge Devices: *Artificial Rabbits Optimization* achieves the lowest energy consumption in both the edge and cloud environments, indicating its energy-efficient execution strategy. However, the *Honey Badger Algorithm* once again demonstrates an interesting trend with higher energy consumption in the edge environment. This could be attributed to its resource allocation approach, which utilizes more energy in the edge setting.

Overall, from the energy consumption results, it can be observed that the *Artificial Rabbits Optimization* algorithm consistently exhibits higher energy consumption both in the edge and cloud environments across all three cases (50, 100, and 200 edge devices). On the other hand, algorithms like *Dwarf Mongoose Optimization Algorithm*, *Genetic Algorithm* and *Hybrid Grey Wolf - Whale Optimization Algorithm* tend to show lower energy consumption values, indicating their efficiency in managing energy resources.

Table 3: Energy Consumption Results

Algorithm	**50ED (Wh)**	**100ED (Wh)**	**200ED (Wh)**
ARO	18.9722 / 3.0284	21.2326 / 5.1944	25.0972 / 10.6696
DMOA	19.2431 / 2.5069	20.8785 / 5.8763	23.5868 / 13.5776
GA	19.2951 / 2.4067	20.566 / 6.4779	24.4514 / 11.913
HBA	19.0556 / 2.8679	21.2014 / 5.2546	23.7743 / 13.2166
HGWOA	19.2951 / 2.4067	20.316 / 6.9593	24.0451 / 12.6952
OCRO	19.1181 / 2.7476	20.4097 / 6.7788	24.2118 / 12.3743
PPSO	19.2951 / 2.5378	21.9965 / 7.3809	30.4521 / 13.0573
SCSO	18.6493 / 3.6501	20.8576 / 5.9164	24.0243 / 12.7353

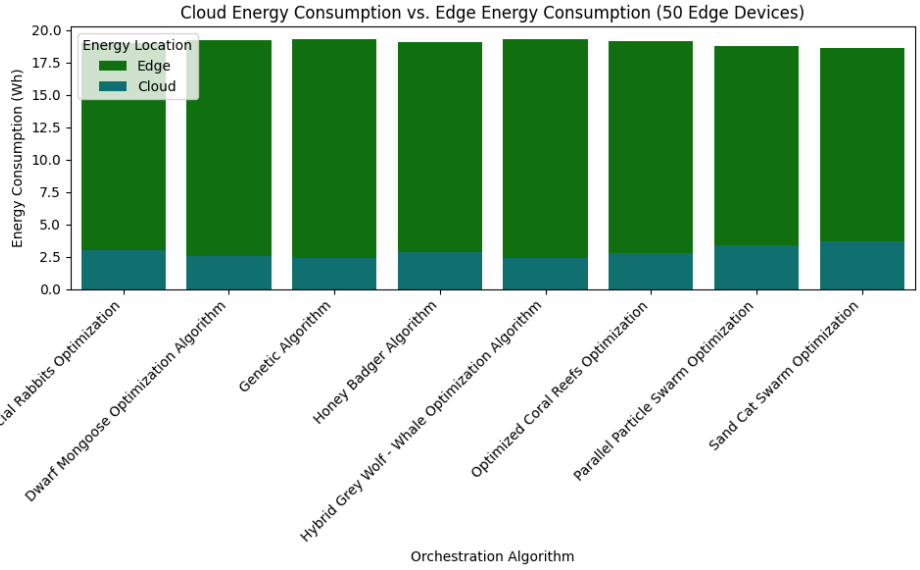

Figure 6: Energy Consumption in Cloud vs Edge for 50 Edge Devices

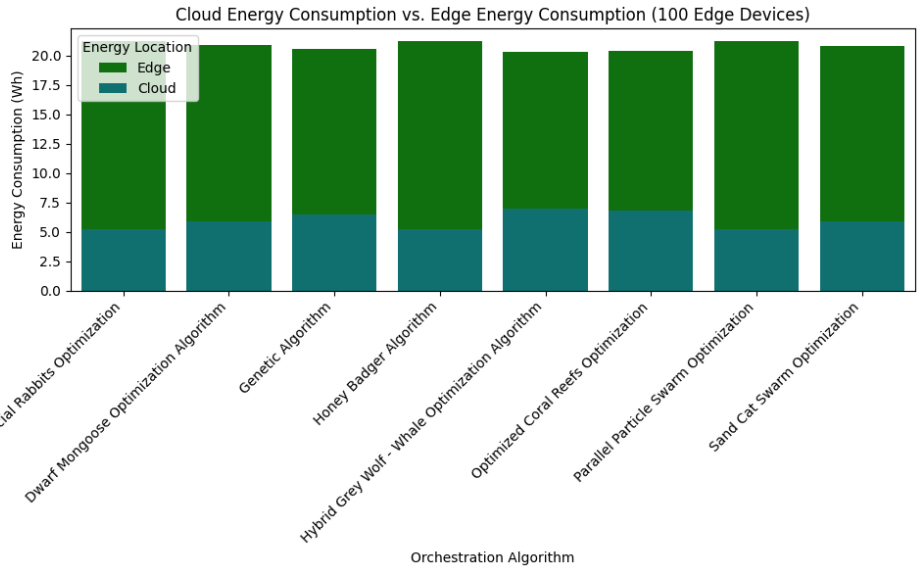

Figure 7: Energy Consumption in Cloud vs Edge for 100 Edge Devices

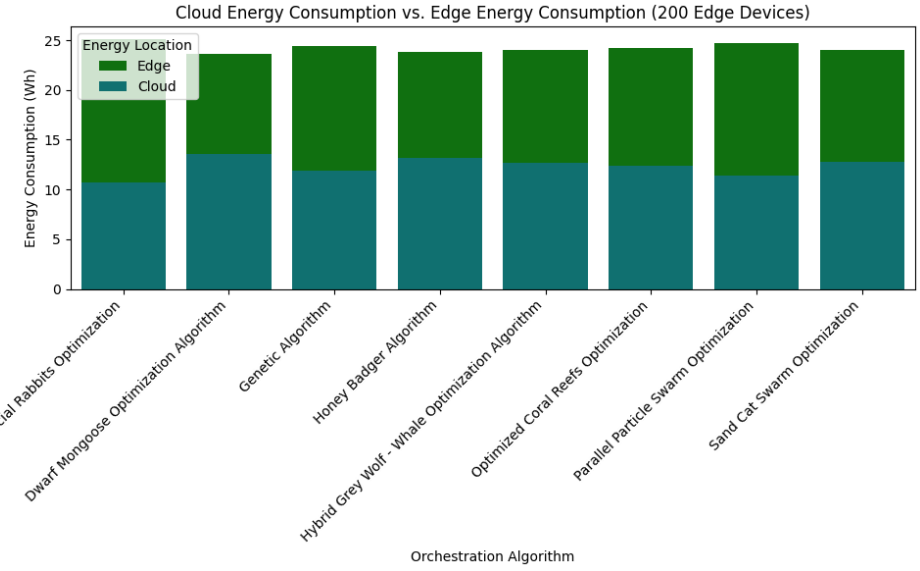

Figure 8: Energy Consumption in Cloud vs Edge for 200 Edge Devices

6.3 Experiment - Waiting Time

The analysis of waiting time shed light on the responsiveness of orchestration algorithms and their impact on task execution delay.

For 50 Edge Devices: All the algorithms have a negligible waiting time since the number of edge devices considered is significantly small. This suggests efficient task distribution and execution, minimizing waiting time in both edge and cloud environments.

For 100 Edge Devices: *Hybrid Grey Wolf - Whale Optimization Algorithm* and *Optimized Coral Reefs Optimization* exhibited minimal waiting times of 0.85 seconds, indicating effective task distribution and quick execution in both the edge and cloud

environments. On the other hand, the *Dwarf Mongoose Optimization Algorithm* and *Genetic Algorithm* demonstrated relatively low waiting times of around 7 seconds, indicating efficient task allocation. Interestingly, *Honey Badger Algorithm* displayed significantly higher waiting time, suggesting possible challenges in task distribution.

For 200 Edge Devices: *Hybrid Grey Wolf - Whale Optimization Algorithm* and *Optimized Coral Reefs Optimization* maintained their efficiency with waiting times of 186.525 seconds and 157.9625 seconds, respectively. *Dwarf Mongoose Optimization Algorithm* and *Genetic Algorithm* also showed reasonable waiting times of around 206.75 seconds and 269.5875 seconds, respectively. *Honey Badger Algorithm* demonstrated a waiting time of 188.975 seconds, indicating effective task distribution in the edge environment.

Overall, the algorithms generally exhibited efficient task distribution and execution, resulting in relatively low waiting times in various scenarios.

Table 4: Waiting Time Results

Algorithm	**50ED (s)**	**100ED (s)**	**200ED (s)**
ARO	0.0	16.775	528.55
DMOA	0.0125	7.4625	206.75
GA	0.3875	7.325	269.5875
HBA	0.025	51.625	188.975
HGWOA	0.0375	0.85	186.525
OCRO	0.0625	0.85	157.9625
PPSO	0.0	36.775	321.6125
SCSO	0.0	5.85	347.5875

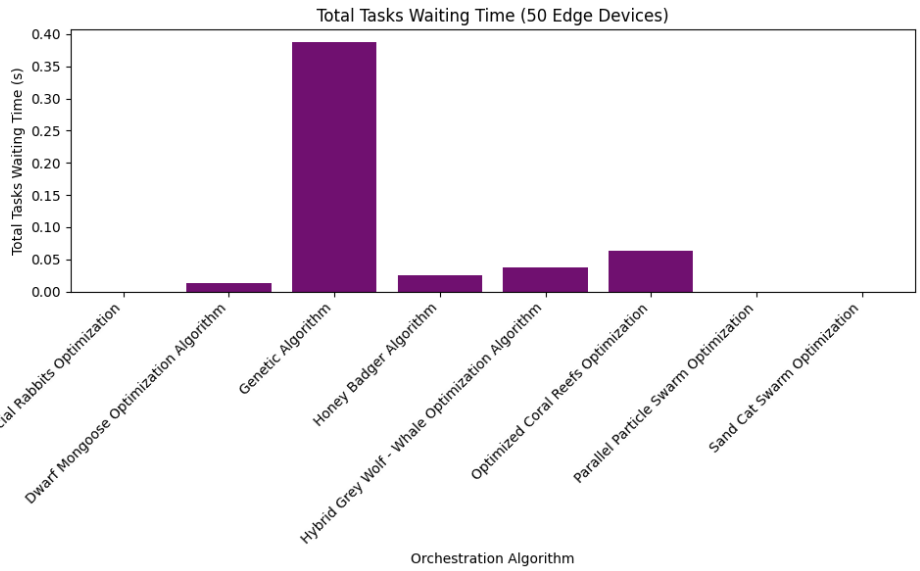

Figure 9: Waiting Time for 50 Edge Devices

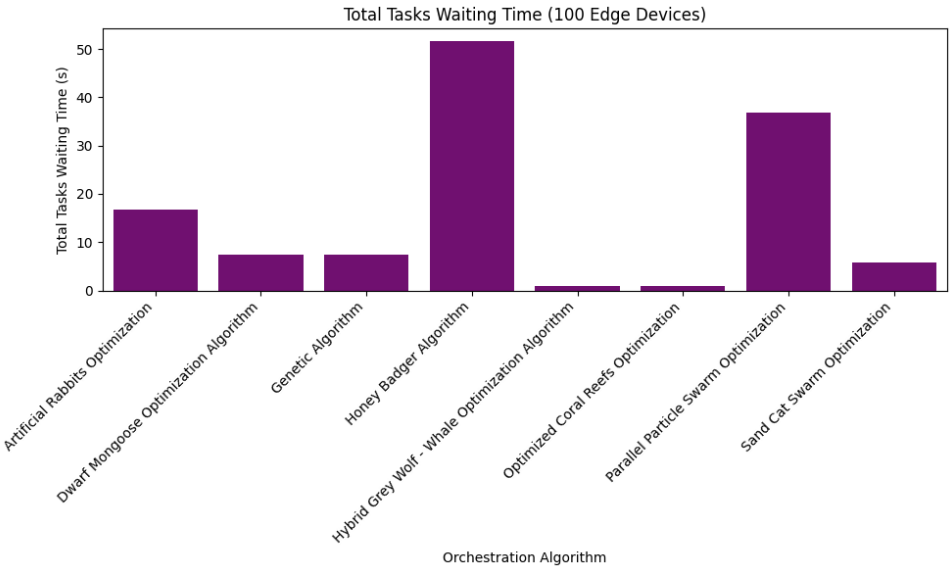

Figure 10: Waiting Time for 100 Edge Devices

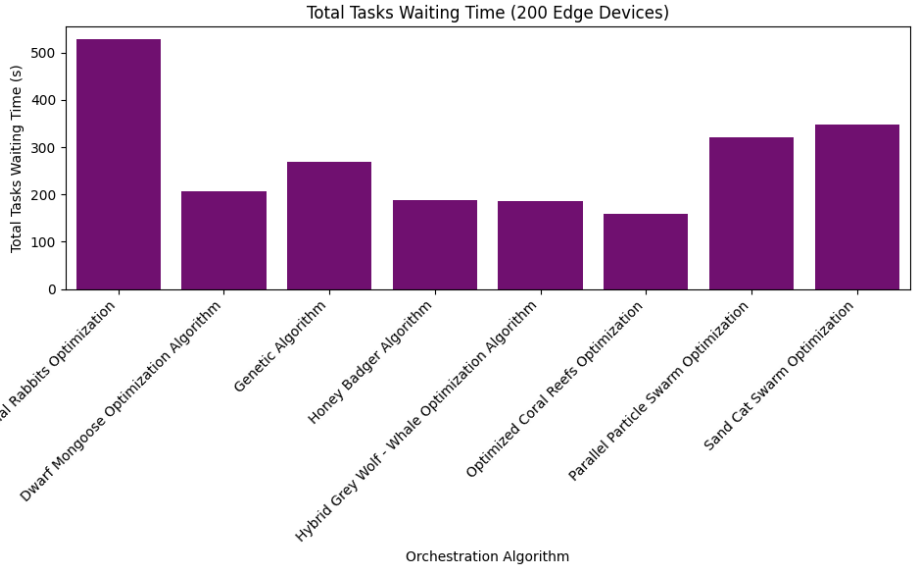

Figure 11: Waiting Time for 200 Edge Devices

6.4 Discussion

The comprehensive evaluation of the orchestration algorithms provides valuable insights into their performance in the cloud-edge environment. The strengths and weaknesses observed in the experiments highlight the importance of algorithm selection for optimal task execution, energy consumption, and responsiveness. The results suggest that different algorithms excel under varying scenarios, emphasizing the need for adaptive algorithms that can adapt to changing edge conditions.

While some algorithms consistently performed well across different scenarios, others showed varying levels of efficiency. These variations may be attributed to the algorithms'

underlying strategies for task allocation, energy management, and responsiveness. We also considered the implications of the findings from both academic and practical perspectives, addressing potential areas for improvement and future research.

7 Conclusion and Future Work

7.1 Conclusion

In this study, we set out to address the challenges of task scheduling in a cloud-edge computing environment through the evaluation of various orchestration algorithms. The research question focused on identifying the most efficient algorithms for task execution, energy consumption, and waiting time in scenarios with varying numbers of edge devices. Our objectives were to assess the performance of different algorithms, analyze their strengths and weaknesses, and provide insights for improved resource allocation.

Through rigorous experimentation and analysis, we successfully evaluated the orchestration algorithms' performance in terms of tasks executed, energy consumption, and waiting time. We analyzed the results for three different edge device counts (50, 100, and 200) and identified notable trends and patterns in algorithm behaviour. The key findings highlighted the significance of algorithm selection in achieving efficient task execution and resource utilization within a dynamic cloud-edge environment.

7.2 Implications and Limitations

The implications of our research extend to both academic and practical domains. From an educational perspective, this study contributes valuable insights into the orchestration algorithms' performance, aiding researchers in understanding their trade-offs and areas of expertise. Practically, the findings offer decision-makers and practitioners guidance in selecting suitable orchestration algorithms for specific edge-computing scenarios.

However, it's essential to acknowledge the limitations of this research. The simulation environment, while realistic, is still an abstraction of real-world complexities. The proposed algorithms' performance might differ in actual deployments due to factors beyond the simulation scope. Additionally, the algorithms' parameters and behaviours were considered static, without accounting for adaptive variations.

7.3 Future Work

As we conclude this study, several avenues for future research and development emerge. To meaningfully extend this work, future research projects could explore:

1. **Dynamic Algorithm Adaptation:** Investigate the potential of developing algorithms that dynamically adjust their parameters and strategies based on changing edge conditions, further enhancing efficiency and adaptability.

2. **Machine Learning Integration:** Integrate machine learning techniques to predict and optimize task allocation and resource utilization, enhancing the algorithms' decision-making process.

3. **Real-world Validation:** Conduct real-world experiments to validate the algorithms' performance in actual edge computing scenarios, taking into account the unpredictabilities of real-world networks and device behaviours.

In conclusion, this research sheds light on the complex interplay of orchestration algorithms in cloud-edge environments. The findings lay a strong foundation for future research endeavours that aim to refine and expand the capabilities of orchestration algorithms to address the evolving challenges and opportunities in edge computing.

References

Abohamama, A. S., El-Ghamry, A. and Hamouda, E. (2022). Real-time task scheduling algorithm for IoT-based applications in the cloud–fog environment, *Journal of Network and Systems Management* **30**(4): 54.
URL: *https://doi.org/10.1007/s10922-022-09664-6*

Agushaka, J. O., Ezugwu, A. E. and Abualigah, L. (2022). Dwarf mongoose optimization algorithm, *Computer Methods in Applied Mechanics and Engineering* **391**: 114570.
URL: *https://www.sciencedirect.com/science/article/pii/S0045782522000019*

Alkayal, E. S., Jennings, N. R. and Abulkhair, M. F. (2016). Efficient task scheduling multi-objective particle swarm optimization in cloud computing, *2016 IEEE 41st Conference on Local Computer Networks Workshops (LCN Workshops)*, pp. 17–24.

Buyya, R. and Srirama, S. N. (2019). *Internet of Things (IoT) and New Computing Paradigms*, pp. 1–23.

Fahimullah, M., Ahvar, S. and Trocan, M. (2022). A review of resource management in fog computing: Machine learning perspective.

Gad, A. G. (2022). Particle swarm optimization algorithm and its applications: A systematic review, *Archives of Computational Methods in Engineering* **29**(5): 2531–2561.
URL: *https://doi.org/10.1007/s11831-021-09694-4*

Gaggero, M. and Ababneh, J. (2021). A hybrid approach based on grey wolf and whale optimization algorithms for solving cloud task scheduling problem, *Mathematical Problems in Engineering* **2021**: 3517145.
URL: *https://doi.org/10.1155/2021/3517145*

Gill, S. S., Garraghan, P. and Buyya, R. (2019). Router: Fog enabled cloud based intelligent resource management approach for smart home iot devices, *Journal of Systems and Software* **154**: 125–138.
URL: *https://www.sciencedirect.com/science/article/pii/S0164121219300986*

Guerrero, C., Lera, I. and Juiz, C. (2019). Evaluation and efficiency comparison of evolutionary algorithms for service placement optimization in fog architectures, *Future Generation Computer Systems* **97**: 131–144.
URL: *https://www.sciencedirect.com/science/article/pii/S0167739X18325147*

Guevara, J. C. and da Fonseca, N. L. S. (2021). Task scheduling in cloud-fog computing systems, *Peer-to-Peer Networking and Applications* **14**(2): 962–977.
URL: *https://doi.org/10.1007/s12083-020-01051-9*

Hosseini, E., Nickray, M. and Ghanbari, S. (2022). Optimized task scheduling for cost-latency trade-off in mobile fog computing using fuzzy analytical hierarchy process, *Computer Networks* **206**: 108752.
URL: *https://www.sciencedirect.com/science/article/pii/S138912862100596X*

Hosseinzadeh, M., Azhir, E., Lansky, J., Mildeova, S., Ahmed, O. H., Malik, M. H. and Khan, F. (2023). Task scheduling mechanisms for fog computing: A systematic survey, *IEEE Access* **11**: 50994–51017.

Iftikhar, S., Gill, S. S., Song, C., Xu, M., Aslanpour, M. S., Toosi, A. N., Du, J., Wu, H., Ghosh, S., Chowdhury, D., Golec, M., Kumar, M., Abdelmoniem, A. M., Cuadrado, F., Varghese, B., Rana, O., Dustdar, S. and Uhlig, S. (2023). Ai-based fog and edge computing: A systematic review, taxonomy and future directions, *Internet of Things* **21**: 100674.
URL: *https://www.sciencedirect.com/science/article/pii/S254266052200155X*

Lambora, A., Gupta, K. and Chopra, K. (2019). Genetic algorithm- a literature review, *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 380–384.

Li, H., Ota, K. and Dong, M. (2019). Deep reinforcement scheduling for mobile crowdsensing in fog computing, *ACM Trans. Internet Technol.* **19**(2).
URL: *https://doi.org/10.1145/3234463*

Liu, Z., Yang, X., Yang, Y., Wang, K. and Mao, G. (2019). Dats: Dispersive stable task scheduling in heterogeneous fog networks, *IEEE Internet of Things Journal* **6**(2): 3423–3436.

Mehta, S., Singh, A. and Singh, K. K. (2021). Role of machine learning in resource allocation of fog computing, *2021 11th International Conference on Cloud Computing, Data Science Engineering (Confluence)*, pp. 262–266.

N. Jayasena, K. and Thisarasinghe, B. (2019). Optimized task scheduling on fog computing environment using meta heuristic algorithms, *2019 IEEE International Conference on Smart Cloud (SmartCloud)*, pp. 53–58.

Nayeri, Z. M., Ghafarian, T. and Javadi, B. (2021). Application placement in fog computing with ai approach: Taxonomy and a state of the art survey, *Journal of Network and Computer Applications* **185**: 103078.
URL: *https://www.sciencedirect.com/science/article/pii/S1084804521000989*

Rafique, H., Shah, M. A., Islam, S. U., Maqsood, T., Khan, S. and Maple, C. (2019). A novel bio-inspired hybrid algorithm (nbiha) for efficient resource management in fog computing, *IEEE Access* **7**: 115760–115773.

Rahbari, D. (2022). Analyzing meta-heuristic algorithms for task scheduling in a fog-based iot application, *Algorithms* **15**(11).
URL: *https://www.mdpi.com/1999-4893/15/11/397*

Reddy, K. H. K., Luhach, A. K., Pradhan, B., Dash, J. K. and Roy, D. S. (2020). A genetic algorithm for energy efficient fog layer resource management in context-aware smart cities, *Sustainable Cities and Society* **63**: 102428.
URL: *https://www.sciencedirect.com/science/article/pii/S2210670720306491*

Seyyedabbasi, A. and Kiani, F. (2023). Sand cat swarm optimization: a nature-inspired algorithm to solve global optimization problems, *Engineering with Computers* **39**(4): 2627–2651.
URL: *https://doi.org/10.1007/s00366-022-01604-x*

Shadroo, S., Rahmani, A. M. and Rezaee, A. (2021). The two-phase scheduling based on deep learning in the internet of things, *Computer Networks* **185**: 107684.
URL: *https://www.sciencedirect.com/science/article/pii/S1389128620312925*

Swarup, S., Shakshuki, E. M. and Yasar, A. (2021). Energy efficient task scheduling in fog environment using deep reinforcement learning approach, *Procedia Computer Science* **191**: 65–75. The 18th International Conference on Mobile Systems and Pervasive Computing (MobiSPC), The 16th International Conference on Future Networks and Communications (FNC), The 11th International Conference on Sustainable Energy Information Technology.
URL: *https://www.sciencedirect.com/science/article/pii/S1877050921014046*

Tsai, C.-W., Salcedo-Sanz, S., Del Ser, J., Landa-Torres, I., Gil-López, S. and Portilla-Figueras, J. A. (2014). The coral reefs optimization algorithm: A novel meta-heuristic for efficiently solving optimization problems, *The Scientific World Journal* **2014**: 739768.
URL: *https://doi.org/10.1155/2014/739768*

Van Thieu, N. and Mirjalili, S. (2023). Mealpy: An open-source library for latest meta-heuristic algorithms in python, *Journal of Systems Architecture* .

Wang, L., Cao, Q., Zhang, Z., Mirjalili, S. and Zhao, W. (2022). Artificial rabbits optimization: A new bio-inspired meta-heuristic algorithm for solving engineering optimization problems, *Engineering Applications of Artificial Intelligence* **114**: 105082.
URL: *https://www.sciencedirect.com/science/article/pii/S0952197622002342*

Wang, S., Zhao, T. and Pang, S. (2020). Task scheduling algorithm based on improved firework algorithm in fog computing, *IEEE Access* **8**: 32385–32394.

Zeng, Z., Miao, W., Li, S., Liao, X., Zhang, M., Zhang, R. and Teng, C. (2021). Adaptive task scheduling in cloud-edge system for edge intelligence application, *2021 IEEE Intl Conf on Parallel Distributed Processing with Applications, Big Data Cloud Computing, Sustainable Computing Communications, Social Computing Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pp. 1682–1689.