National College of Ireland

Optimization of Load Balancing in Fog Computing using Bacterial Colony Optimization algorithm

MSc Research Project
Cloud Computing

Shalini Vaibhav
Student ID: 21196354

School of Computing
National College of Ireland

Supervisor: Prof. Yasantha Samarawickrama

National College of Ireland
Project Submission Sheet
School of Computing

Student Name:	Shalini Vaibhav
Student ID:	21196354
Programme:	Cloud Computing
Year:	2023
Module:	MSc Research Project
Supervisor:	Prof. Yasantha Samarawickrama
Submission Due Date:	14/08/2023
Project Title:	Optimization of Load Balancing in Fog Computing using Bacterial Colony Optimization algorithm
Word Count:	7903
Page Count:	21

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 speci ed in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	Shalini Vaibhav
Date:	14th 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 su cient to keep a copy on computer.	☐

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

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

Optimization of Load Balancing in Fog Computing using Bacterial Colony Optimization algorithm

Shalini Vaibhav
21196354

Abstract

Fog computing is one of the most important paradigm shifts, transforming data processing through enhanced e ciency and adaptability, achieved by extending computational capabilities to the edge of the network. Load balancing, which includes distributing the computational demand over a number of fog nodes to improve resource utilisation and system e ectiveness, is one of the most important issues in fog computing. This research aims to tackle load balancing in fog computing by using the Bacterial Colony Optimisation (BCO) technique. The goal of this project is to implement an e ective, and exible load balancing method speci cally designed for fog computing settings. Simulated tests were used to validate the suggested approach and performance metrics were obtained. BCO outperforms Round-Robin (RR) algorithm and Throttle Load Balancing (TLB) algorithm, with average improvements of roughly 56.85% in latency, 51.83% in makespan, and 33.77% in cost across various node con gurations.BCO demonstrates superior performance compared to RR and TLB algorithms, it achieves an improvement in the latency of approximately 56.85%, 51.83% in makespan, and 33.77% in cost across diverse node con gurations.

1 Introduction

Fog computing is a decentralised computational approach that extends cloud capabilities to the edge of the network, enabling computation, storage, and networking services close to data sources, diverging from the traditional centralised data centre paradigm. Both the studies by Kumar et al. (2019) and Chakraborty (2019) evaluate both technologies and come to the conclusion that fog computing, rather than moving all data to the cloud, o ers a more exible structure and improved data processing capabilities by making e cient use of limited network resources. A signi cant number of interconnected devices are organised into fog nodes in fog computing, which are in charge of performing computational tasks on behalf of the connected devices, the architecture of fog network Mahmud et al. (2018) is shown in Figure 1.

It makes handling and analysing data from Internet of Things (IoT) devices and other edge devices easier because there is no need to send the data to a distant data centre or cloud. As a result,performs e ciently in terms of latency, power consumption, capital and operational expenses, network tra c and content distribution Mahmud et al. (2018). When real-time analysis is required or when faraway sites are involved where cloud access is scarce or inconsistent, fog computing appears to be especially bene cial.

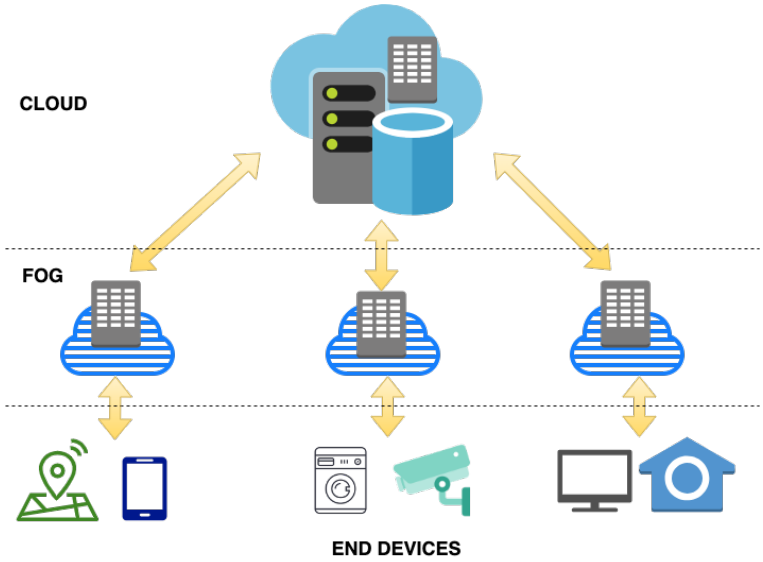

Figure 1: The architecture of Fog network Mahmud et al. (2018)

1.1 Background and Motivation

With the growing need for IoT, enormous amounts of data are produced, placing heavy load on fog nodes. Load balancing, which is the distribution of load among fog nodes, emerges as a crucial process that improves resource utilisation and task response times. In order to manage the exponential growth of data, avoid network congestion, and avoid overloading fog nodes, which could result in performance degradation and signi cant service interruptions, load balancing is essential. The complex environment and limited hardware capabilities of fog devices present special di culties for load balancing in fog computing. The key to improving Quality of Service (QoS) metrics including resource utilisation, throughput, cost-e ciency, response time, performance, and energy consumption is load balancing, which is a key characteristic of fog networks Kashani et al. (2020). Additionally, it reduces the possibility of load imbalances, preventing potential overloading while leaving other fog nodes underutilised.

Numerous research studies highlight the complex nature of load balancing within fog computing due to the variety of fog servers, client mobility, and increasing network tra c.Conventional load balancing techniques are ine ective for meeting the QoS requirements of the fog domain because their computations become unreliable. A wide range of load balancing approaches, including heuristic, meta-heuristic, probabilistic, graph theory-based, and hybrid solutions, have been presented in comprehensive studies and practical implementations in an e ort to overcome this di culty. In fog computing, e cient load balancing emerges as a key element in improving network speed, scalability, and stability. It has the ability to reduce energy consumption by evenly distributing workloads across fog nodes by maximising the utilisation of processing and storage resources and minimizing response times.

Despite many investigations and practical implementations, there is still a need to improve load balancing e ectiveness because of the rising needs of IoT and rising data size. There have been many di erent load balancing systems proposed, taking into factors like

latency, bandwidth, deadlines, cost, security, execution time, and responsiveness. The importance of finding a better solution is discussed in Singh, Kumar, Sharma and Nayyar (2020) and Kashani et al. (2020). Furthermore, Kashani et al. (2020) emphasises the continued requirement to use optimisation methods like BCO to find the optimal solutions.

The motivation behind this research is to evaluate the Bacterial Colony Optimisation algorithm's effectiveness in load balancing, assuring efficient task allocation, and avoiding both underloading and overloading situations. This strategy aims to maximise resource utilisation, improving overall efficiency of fog computing. As discussed in Batra et al. (2022) and Kashani et al. (2020), the field of load balancing comprises a wide range of approaches, including heuristic, meta-heuristic, probabilistic, graph theory-based, and hybrid techniques. In fog computing, optimisation strategies have the potential to enhance load balancing, resulting in better resource usage and increased performance. While well-known algorithms like Particle Swarm Optimisation (PSO), Artificial Bee Colony, and fuzzy logic-based load balancing have made beneficial contributions, they also have intrinsic advantages and disadvantages that have been thoroughly discussed in Kashani et al. (2020) and Batra et al. (2022). This goal of this research is to identify the extent to which BCO algorithm can further improve several QoS parameters such as latency, energy consumption, makespan and cost for a smooth functioning fog network.

Bacterial Colony Optimization is a bio-inspired optimization technique, imitates the chemotaxis, intercellular communication, and self-adjusting feeding patterns of bacterial colonies. Through extracellular chemicals, it mimics how colonies move, communicate, and seek out nutrients. These principles are used by this algorithm to produce potential solutions and identify the best fit for a particular task. Its capacity to improve load balancing emphasises its viability. The goal in this study is to examine and compare this strategy with the existing load balancing approaches.

1.2 Research Question

The above research problem motivates the following research question: [**What is the extent of the improvement in load balancing optimization achieved by using the Bacterial Colony Optimization algorithm in fog computing, especially in terms of critical performance measures like latency, energy consumption, makespan, and cost? Furthermore, how does it compare to other optimization algorithms, across different node configurations?**].

This study is organised as follows in the sections that follow: Section2, Related Work, examines previous studies, evaluates their main goals, and outlines their projected contributions. Section 3 describes the research methods in more detail and explains the flow of Bacterial Colony Optimisation algorithm to improve load balancing in fog computing. The design specification is presented in Section 4, Implementation is covered in Section 5, and evaluation is covered in Section 6. The study comes to a conclusion in Section 7, which summarises the conclusions, implications, and directions for further investigation.

2 Related Work

According to various studies and prior related work, fog computing has a signi cant potential, but it also faces a lot of challenges that need to be tackled, including resource allocation, privacy and security, job scheduling, control and management. Load balancing is an important element that can enhance the QoS measures such as resource utilisation, throughput, price, response time, performance, and energy consumption.

2.1 Understanding Fog Computing, its architectures, scope, challenges and future work

Naha et al. (2018), Mouradian et al. (2018), and Mukherjee et al. (2018) provide an extensive overview of fog computing, a distributed computing model that extends cloud computing to the network edge. Naha et al. (2018) provides a thorough review of the newly developing issue of fog computing. For academics and industry professionals interested in the subject, studying the history of fog computing, its essential traits, and the numerous architectures and technologies that make it possible is a signi cant resource. The authors also emphasise security, privacy, load balancing, and energy e ciency as requirements and challenges of fog computing. These concerns, which involve resource scheduling as one of the key issues, can help focus future research in this topic. Overall, the state of the art in fog computing is reviewed in-depth in this research. Although the authors provide a thorough theoretical framework for understanding the advantages and di culties of fog computing, the paper lacks actual examples of how fog computing has been successfully deployed in many sectors.

The research by Mouradian et al. (2018) is signi cant since it provides a thorough analysis of fog computing. Using a simple set of criteria, the study carefully evaluates the state of the art, taking into account both the structural designs and algorithms included into fog systems. The challenges and opportunities for more research in the eld are brie y discussed in the paper, with a focus on load balancing in fog computing. The authors present a clear framework for evaluation criteria, but they do not provide a thorough assessment of how well the current fog computing architectures and technologies meet these criteria.

Mukherjee et al. (2018) discusses various elements of fog computing, including its limitations and potential prospects. The authors identify one such problem as resource management, which is particularly complex given that fog computing consumes di erent resources such as processing power, storage capacity, and network bandwidth in real-time. E ective load balancing solutions, as well as advances in job scheduling and resource allocation, are required. Obtaining interoperability across the multiple diverse devices and systems used in fog computing can be quite di cult. The researchers also note prospects for future growth, such as its use in emerging applications like healthcare, smart cities, and transportation { which could potentially result in new business models and revenue streams.

Key challenges for fog computing include concerns about security and privacy, operational challenges, service-related challenges, and software architecture challenges. Fog computing raises security and privacy issues because of the distributed architecture and the vast device network. Operational problems include e ective management and dependability assurance for various devices. A crucial service consideration is ensuring resource e ciency and excellent service quality. The successful integration and broad acceptance

of fog computing across diverse applications depend on overcoming these challenges.

2.2 Load Balancing in Fog Computing and its challenges

According to the paper Kashyap and V (2022), load distribution in the context of fog computing becomes crucial with the growth in network-level tra c. Inadequate load balancing can result in unfavourable outcomes such delays in processing, responsiveness, and security standards, which lowers the overall quality of services. The research articles also explore the challenges of load balancing in fog computing, including the severely con ned environments and constrained hardware capabilities of fog devices. The publications highlight the need for more research to address the obstacles and limitations that come with load balancing in the context of fog computing.

A thorough analysis of load balancing in fog computing is included in the study Chandak and Ray (2019). The paper addresses load balancing issues in fog computing along with o oading techniques, dynamic load distribution, and e ective resource management. Additionally, it assesses other existing load balancing techniques, such as fog cluster-based load balancing. The analysis, however, falls short of rating the e ectiveness of these various load balancing solutions in various fog computing scenarios, as well as their advantages and disadvantages.

Multiple researchers present a thorough examination of load balancing approaches in fog computing. Kashani et al. (2020) and Saroa 2021 provide systematic reviews of load balancing methods in fog computing, whereas Chandak and Ray (2019) and Abbasi et al. (2018) provide surveys of load balancing algorithms utilised in fog computing. The research papers examine the importance of load balancing in the context of fog computing and how it might improve QoS metrics such as resource utilisation, throughput, costs, response time, e ciency, and energy consumption. The studies also identify a variety of load balancing algorithms and methods, including fundamental, exact, approximate, and hybrid solutions, and provide a thorough analysis of their bene ts and drawbacks. The articles also examine the current di culties and anticipated developments in the eld of fog computing load balancing approaches.

Singh, Sharma and Kumar (2020) addresses the issue that traditional algorithms do not work well in the fog zone and proposes a fuzzy logic-based load balancing solution due to the severely constricted environment and limited hardware capabilities of fog devices. Rani (2022) proposes a unique hybrid model for load balancing in smart grids that uses throttled, round-robin, and particle swarm optimisation techniques. The importance of fog computing in addressing cloud computing's latency, privacy, and congestion challenges is emphasised in the article. The research examines many types of fog-based scheduling algorithms but does not provide any gaps or comparisons between these strategies. It also looks at how fog computing helps end users by bringing computation, storage, and networking capabilities closer to them.

2.3 State-Of-the-Art and potential solutions

In the eld of fog computing, load balancing is an important area of research and development. The problem of load balancing in this domain is being addressed by numerous active studies and practical applications.

Multiple studies have examined the load balancing techniques utilised in fog computing. Batra et al. (2022) concisely describes and compares load balancing strategies such

as heuristic, meta-heuristic, probabilistic, graph theory-based, and hybrid. Round-robin and source IP hash are two load balancing algorithms that are explored and compared in Kumar et al. (2019). Xu et al. (2018) recommends a heuristic virtual machine scheduling technique for load balancing in fog-cloud computing. Overall, these publications provide a comprehensive review of the present state-of-the-art in fog computing load balancing, covering the several methodologies, measures, and assessment strategies used.

The study Rehman et al. (2019) highlights how fog computing may improve the performance of smart building applications by minimising latency. The proposed technique is evaluated using simulations and compared with alternative scheduling methods. The results show that the Min-Min algorithm outperforms alternative techniques in terms of makespan and resource utilisation. Six fog nodes from six distinct locations were used in the study, and the microgrids were linked to measure the energy consumption on these fog nodes. The ideal method queues up jobs that take more time to finish and completes those that take less time. Based on tasks that have been performed and those that have been added, the list is updated. Despite not being addressed in this work, fog computing's use in smart buildings poses security and privacy issues.

Throttled, Round Robin (RR), and First Fit (FF), three load balancing algorithms designed to provide effective resource allocation in fog computing, are compared in depth in the study Ahmad et al. (2019). The evaluation is based on factors including price, processing time, and response time. The results show that, in terms of cost optimisation, the suggested approach performs better than other algorithms; however, the RR and Throttled algorithms perform better in terms of processing time and response time. Despite these insightful observations, the research might benefit from an in-depth review of the performance metrics.

The paper Kashani et al. (2020) gives an in-depth analysis of load balancing strategies in fog computing. In order to provide a thorough classification and review of existing approaches, along with their advantages and disadvantages, they divide these techniques into approximation, exact, fundamental, and hybrid methods. The analysis presents a favourable picture of the field. Additionally, Kashani et al. (2020) discusses a variety of research issues and persistent concerns, offering practical solutions. Although many other heuristic and meta-heuristic methods have been used, the study introduces optimisation techniques such as Bacterial Colony Optimisation as optimal solution for future work. The extensive comparison and assessment of all load balancing approaches provides a strong insight and direction for future study.

Heuristic algorithms are methods for problem-solving that rely on trial and error or actual experience to produce solutions. These methods are not always accurate but are nevertheless regarded as good. Meta-heuristic algorithms, on the other hand, are optimisation procedures that draw inspiration from human or natural behaviour. It is more complicated than the heuristic approach, but it is thought to provide a solid solution. Bacterial Colony Optimisation method, the suggested best solution algorithm in Kashani et al. (2020), is employed for this research; it is a meta-heuristic algorithm. BCO is an algorithm that has been developed to simulate bacteria's behaviour with swarm intelligence, simplifying the optimisation process and introducing novel communication methods to boost efficiency (Revathi et al. (2019) and Niu and Wang (2012)). The introduction and functionality of the BCO algorithm are covered in further detail in Section 4.4.

3 Methodology

3.1 Experimental Setup and Procedure

The research methodology included numerous crucial procedures, each of which contributed to this research. The following part explains these major steps, demonstrating the systematic approach used in this study:

1. Problem Formulation: The first critical step in this research was to develop a clear and straightforward issue description for optimising load balancing in fog computing systems. A thorough study of the related work helped in identifying the issues that fog computing face as well as potential areas for development. Following that, outlining the objectives and scope of the study was very important, learning about the Bacterial Colony Optimization algorithm and the primary performance indicators used to evaluate the efficiency of load balancing algorithms. With a solid foundation for the research by developing a well-structured problem formulation, it helped to proceed with the succeeding stages in a focused and purposeful manner.

2. Data Collection: The next important step was to collect the necessary data for this research. A publicly available dataset, Cloud-Fog computing dataset, was selected as useful for this research and was so utilised. This dataset is available on Kaggle, and the source paper for this dataset Nguyen et al. (2019) is licenced under the Creative Commons Attribution 4.0 International (CC BY 4.0) licence, which governs its use, allowing for the sharing and adaptation of datasets for any purpose with proper citation. The dataset contains 13 nodes, containing 10 fog nodes and 3 cloud nodes. It consists of seven files, each of which corresponds to a different amount of tasks, beginning with 40 and increasing in increments of 40 tasks until it reaches 280 tasks.

3. Simulation Tool Used - iFogSim: In this research, the tests were performed by setting up the simulation environment for the suggested load balancing methods in fog computing. To address load balancing issues in the fog network and improve service quality, numerous researchers have carried out experiments in the field of fog computing. Many simulators made specifically for fog computing have been developed to help with these investigations. A significant example is iFogSim, an open-source application created to study and simulate scenarios including fog computing, edge computing, and the Internet of Things Gaurav (2018) and Margariti et al. (2020). In addition to resource allocation guidelines, communication methods, and evaluation standards, iFogSim has a variety of functionalities. The simulation toolkit iFogSim, which evolved from CloudSim for cloud computing, expands on its capabilities by adding specialised models and modules for simulating fog computing systems. This Java-based simulator expertly mimics a wide range of systems and applications, including smart home automation, video streaming, and natural language processing. On GitHub, the iFogSim project is openly accessible and provides access to its source code, documentation, and example programs. The project has received a lot of attention and currently has 44 forks and 80 stars.

   The open-source simulator iFogSim provides a valuable platform for modelling and testing resource management methods in IoT, edge, and fog computing settings. Researchers can experiment with different settings and policies to optimise resource

utilisation and application performance. The Application Module, Device Module, Communication Module, and Resource Module are the four major components of iFogSim.

The Application Module specifies the workflow and processing requirements for each job in the application. The Device Module represents physical devices like sensors, actuators, and fog nodes. The Communication Module defines the communication architecture and network topology. Meanwhile, the Resource Module simulates hardware resources such as the CPU, RAM, and storage. These modules together simulate the behaviour of a fog computing environment.To enable successful load balancing in the fog computing environment, iFogSim dynamically adapts the load balancing algorithm during the simulation based on the current workload and fog node performance. This adaptive technique ensures that the application performs optimally.

Given the comprehensive functionalities of iFogSim and the financial constraints associated with establishing a real fog computing infrastructure, using iFogSim as a simulation tool becomes a smart alternative for studying fog computing load balancing solutions. The key components of iFogSim as shown in Figure 2 are:

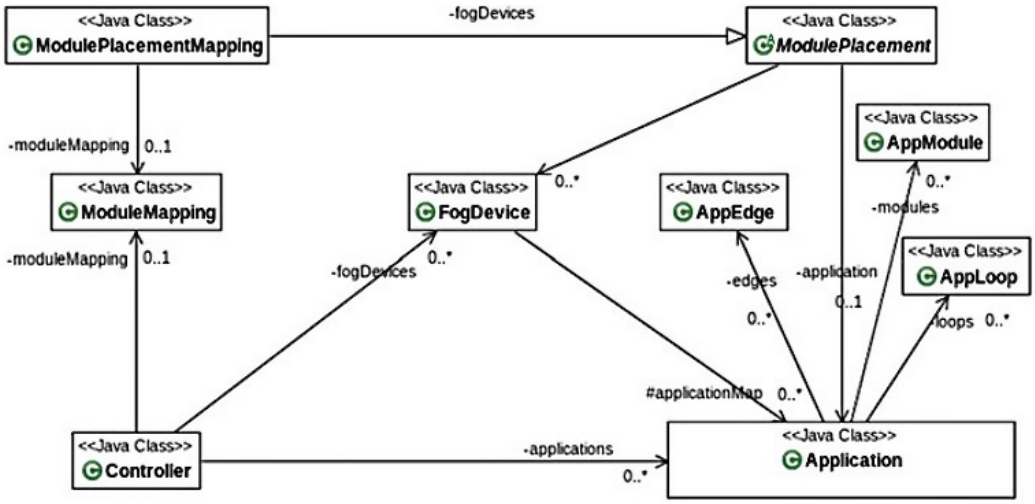

Figure 2: Fundamental classes of iFogSim Gupta et al. (2017)

- Fog Device: - The capabilities of these nodes, which include compute, storage, memory, and downlink and uplink bandwidths, are comparable to those of data centre servers.
- Sensors: - Similar to a sensor used in the Internet of Things (IoT), there is an entity of the sensor class in the simulator. It essentially detects the sensors' output properties as well as the tuples arrival rate.
- Actuator: - This class is mostly used to carry out the operations depending on the arrival of tuples from modules that are present in the application.
- Tuple: - In addition to acting as a layer for data streams in the architecture, this serves as the basic method of communication among the fog.

- Application: - This information determines how the application modules will be scheduled or placed on the fog device.

4. Algorithm Implementation and Integration To improve load balancing in fog computing, implementation of the BCO method into the process of assigning jobs to fog nodes is done. The adaptability and capability of the BCO algorithm to manage dynamic and unpredictable settings make it well-suited for fog computing environments.

   The load balancing method comprises evaluating the available resources and network conditions of fog nodes and assigning jobs to them accordingly. Using the BCO technique, we can optimally distribute workload among fog nodes, enhancing resource utilisation and overall system e ciency. The ow diagram Niu and Wang (2012) of BCO algorithm is shown in Figure 3.

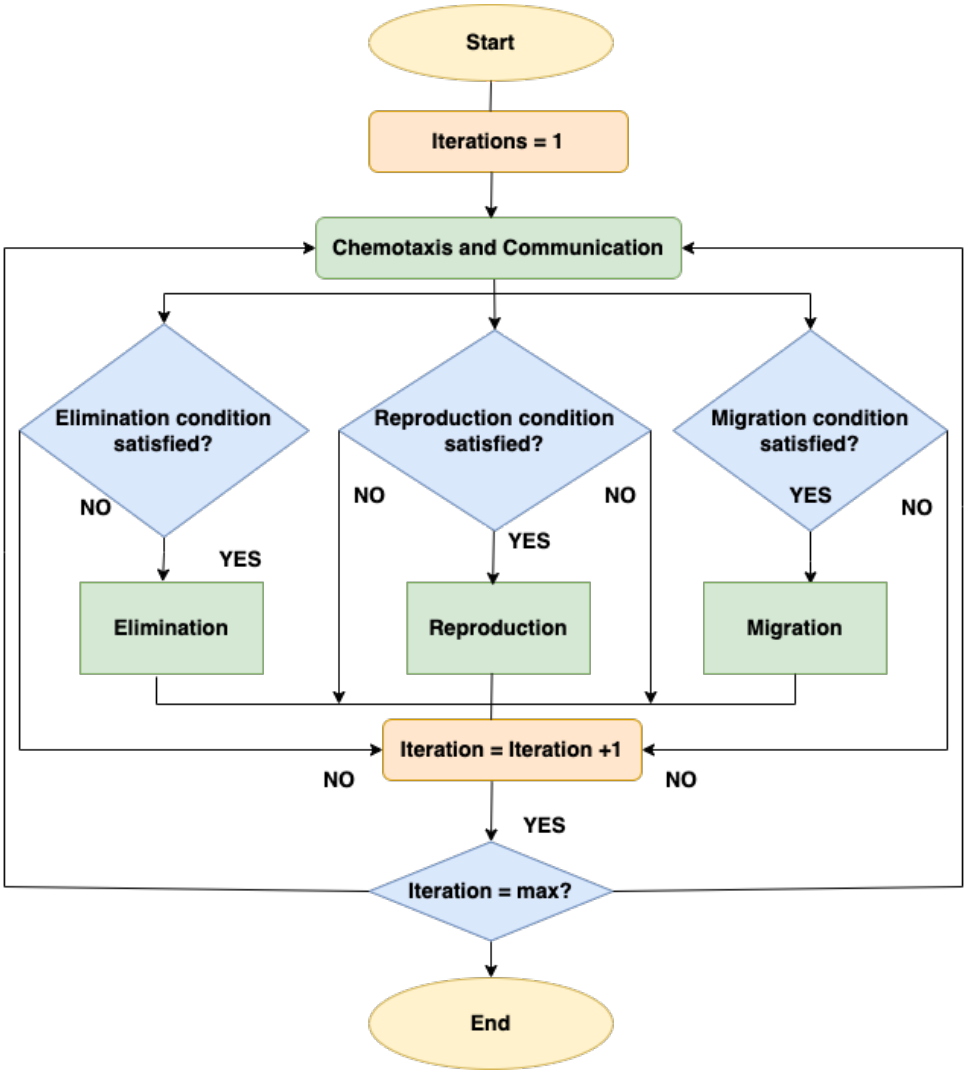

Figure 3: The ow diagram of Bacterial Colony Optimization algorithm Niu and Wang (2012)

The steps for implementing the Bacterial Colony Optimization algorithm for load balancing are discussed as follows:

- Step 1: Initialization: Set up the fog nodes, tasks, and associated parameters. Set the number of iterations, population size, and other algorithm-speci c parameters.
- Step 2: Generate Initial Population: Initialize the population of bacterial colonies, with each colony representing a potential solution. Assign tasks to fog nodes at random or according to the load balancing strategy.
- Step 3: Chemotaxis and Communication: Chemotaxis and communication is used to simulate bacterial migration towards a nutrition gradient. Each bacterium's position in the search space is updated based on its current position and the concentration of the nutrient (tness value).
- Step 4: Reproduction: Based on their tness levels, select parent bacteria for reproduction. To develop o spring bacteria, use reproduction operators such as mutation. - Replace less t bacteria in the population with their o spring.
- Step 5: Migration: Migration allows moving of bacterial colonies around the search space. This deliberate movement promotes diverse investigation, enabling the algorithm to e ectively investigate new areas and maybe identify better solutions.
- Step 6: Elimination: Eliminate some bacteria based on the elimination criterion to create space for new bacteria.
- Step 7: Termination: Repeat steps 3-6 for a speci ed number of iterations or until a termination condition is met. Terminate the algorithm when the maximum number of iterations is reached, or a satisfactory solution is found.
- Step 8: Select Best Solution: After the termination condition is met, select the best solution (bacterial colony) based on the tness value.

5. Performance Metrics The BCO algorithm's performance is measured using the following metrics:

    - Latency: The amount of time it takes to process a cloudlet from submission to completion.
    - Energy Consumption: The overall amount of energy utilised by fog devices.
    - Makespan: The overall time required to execute all cloudlets on their allocated fog devices.
    - Cost: The total cost of running cloudlets on fog devices, taking into account execution time and device utilisation cost.

6. Comparison with Existing Algorithms In this research, the load balancing performance metrics result obtained from Bacterial Colony Optimization algorithm is compared with the below two load balancing algorithms:

    - Round-Robin algorithm: Round Robin is a simple load balancing mechanism that assigns work to fog nodes in a Fist Come First Serve basis. It ensures that each node receives an equal proportion of tasks, supporting a relatively fair workload distribution. However, RR lacks adaptation to dynamic workload variations and node heterogeneity, resulting in suboptimal performance in terms of latency, energy consumption, and makespan.

- Throttled Load Balancing algorithm: Throttled Load Balancing (TLB) is a load balancing method used in fog computing settings to optimise energy consumption. It focuses on dynamically altering fog node processing speeds according on workload and energy availability. TLB tries to create a balance between energy e ciency and job completion time by reducing node operational frequency.

4 Design Specification

4.1 Techniques

The techniques used in this research includes the following:

- BCO Algorithm Customization: The BCO algorithm was adapted to the load balancing requirements of fog computing environments. The chemotaxis, reproduction, migration, and communication components have been ne-tuned to optimise resource management and reduce response time.
- Integration with iFogSim: The BCO algorithm has been integrated into iFogSim, an open-source fog computing simulator, to allow for realistic fog computing environment simulations. iFogSim provides the infrastructure for fog nodes, cloud nodes, and cloudlets (tasks).
- Performance Metrics Selection: The research identi es the relevant performance metrics, such as response time, job completion rate, resource utilisation, and energy consumption, to evaluate the e ciency of the BCO-based load balancing approach.
- Simulation Requirements: In order to run simulations e ectively, the dataset citation of fog nodes with certain properties, cloud nodes, and cloudlets representing tasks is used. The dataset is produced using actual fog computing conditions with 10 fog nodes.

4.2 Architecture Configuration

The architecture con guration consists of specifying the fog computing environment, fog nodes, cloud nodes, and their features. The following elements will be considered:

- Fog Nodes: Several fog nodes with varied computational capacity, memory, and bandwidth have been modelled. The properties of these fog nodes has been determined from a realistic dataset re ecting physical fog computing resources.
- Cloud Nodes: Cloud nodes will represent centralised cloud resources. These nodes have higher computing capacity and bandwidth than fog nodes but will be limited in quantity to encourage fog-based processing.
- Fog-Cloud Connectivity: Communication linkages between fog nodes and cloud nodes are constructed to simulate communication in a fog computing architecture.
- Cloudlets (Tasks): Using real-world fog computing workloads, cloudlets representing tasks are formed. The computing requirements and processing times for these cloudlets di er.

4.3 Simulation Configuration

The simulation con guration consists of setting up the simulation environment and settings to test the BCO-based load balancing strategy. The BCO method is integrated into iFogSim, an open-source fog computing simulator. iFogSim will provide the infrastructure required for simulating fog nodes, cloud nodes, and cloudlets in a realistic manner.

Parameter	Specification
System architecture	x86
Operating system	Linux
Virtual Machine	Xen
Cost Per Storage	0.001

Table 1: System Speci cations

4.4 Bacterial Colony Optimization

The BCO algorithm is implemented in this study to optimise load balancing in fog computing. BCO is inspired by fundamental bacterial life cycle behaviours such as chemotaxis, elimination, reproduction, migration, and communication. Chemotaxis allows bacteria to travel towards a gradient of nutrient concentrations, and it is coupled with communication in BCO to produce a single model. To move and interact inside the optimisation process, the bacterial colonies undertake runs and tumbles. The BCO algorithm is divided into ve submodels: reproduction, elimination, migration, chemotaxis, and communication. Each submodel has its own set of optimisation rules and processes.

Chemotaxis and communication are both incorporated into a single framework in the BCO technique. An agent-environment-rule schema serves as the foundation for the entire bacterial lifecycle model. The arti cial bacterial lifecycle model within BCO is made up of ve di erent submodels: communication, migration, chemotaxis, elimination, and reproduction. Each of these submodels has unique rules and processes designed for optimisation.

Bacterial colonies move and communicate using runs and tumbles in the chemotaxis and communication model of the BCO algorithm. The following equations can be used to describe this behavior (1)(2):

$$Position_T = Position_i(T \quad 1) + R_i \quad (Ru_{Info}) \quad R \quad _i \tag{1}$$

$$Position_T = Position_i(T \quad 1) + R_i \quad (Tumb_{Info}) \quad R \quad _i \tag{2}$$

The elimination and reproduction model in BCO allocates an energy level to various bacterial colonies based on their search capacities, helping in making decisions regarding these processes. This energy level (Li) determines the elimination and reproduction activities, which leads to the selection of subsequent actions as shown below :

if $L_i > L_{given}$, and $i \ 2$ healthy, then $i \ 2$ Candidate$_{repr}$;
if $L_i < Lgiven$, and $i \ 2$ healthy, then $i \ 2$ Candidate$_{eli}$;
if $i \ 2$ unhealthy, then $i \ 2$ Candidate$_{eli}$:

Another critical component of the BCO optimisation process is migration. It prevents getting stuck in poor solutions by allowing a change in location within a speci ed range. The migration process is dependent on precise parameters, and if those requirements are met, bacteria migrate to new random locations inside the search space. BCO is capable of dealing with dynamic and unpredictable settings, making it suited for load balancing in fog computing. The migration is determined by a set of criteria, which, when satis ed, causes it to move to a new, random location, it can be de ned as :

$$\text{Position}_i(T) = \text{rand} \quad (\text{ub} \quad \text{lb}) + \text{lb} \tag{3}$$

Where rand is a random number between 0 and 1, and lb, ub are the position's lower and upper boundaries, respectively.

The versatility and capability of the BCO algorithm to handle dynamic settings make it an attractive alternative for load balancing optimisation in fog computing. Aiming to demonstrate the usefulness of the BCO-based technique in attaining load balancing objectives in fog computing systems through empirical evaluation and comparison with other algorithms.

During the research the in uential factors were identi ed and after running simulation tests, the below parameters were tuned for the optimization of load balancing process.

Parameter	Value
Bacteria Population Size	100
BCO Iterations	100
Step size for movement	0.1
Replication rate	0.1

Table 2: BCO Parameters

5 Implementation

The implementation part of the research on load balancing in fog computing using the Bacterial Colony Optimisation (BCO) method includes developing a simulation model to evaluate the performance of the BCO algorithm in a fog computing environment. This section details the nal step of implementation, including the outputs produced, tools used, and simulation model developed.

5.1 Development of a Simulation Model

The simulation model was created using the iFogSim simulator, an open-source tool designed primarily for simulating fog computing, edge computing, and Internet of Things

(IoT) scenarios. iFogSim o ers a versatile and adaptable framework for representing fog computing infrastructures, devices, applications, and scheduling algorithms. The major components of the model were fog nodes, cloud nodes, cloudlets, and the BCO-based load balancing algorithm.

5.2 Outputs Produced

The simulation model generated a realistic fog computing system with many fog nodes and cloud nodes interconnected via communication channels. MIPS, RAM, uplink bandwidth, downlink bandwidth, hierarchy level, and cost factors were used to characterise the fog nodes. Cloud nodes represented centralised cloud resources with increased computational power.

Workload (Cloudlets): The model generated a collection of cloudlets (tasks) with varied processing lengths, le sizes, memory requirements, and utilisation models. In the fog computing environment, these cloudlets were employed to simulate processing activities.

Load Balancing Algorithm Based on BCO: The BCO-based load balancing algorithm was developed and included into the iFogSim simulator. The algorithm was modi ed to optimise cloudlet allocation to fog nodes based on processing capacity, available resources, and workload characteristics.

5.3 Tools and Languages Used

- iFogSim: The iFogSim simulator was used to create the simulation model and evaluate the BCO-based load balancing algorithm. It provides the necessary functionality for modelling fog computing environments, fog nodes, cloud nodes, and cloudlets.

- Java Programming Language: The Java programming language was used to create the simulation model and the BCO-based load balancing method. The object-oriented characteristics and libraries of Java aided in the creation of a modular and extendable simulation framework.

5.4 Final Stage of the Implementation Process

- Dataset Integration: The simulation model included a realistic dataset comprising information about fog computing resources and features. This dataset was used to con gure fog node properties such as MIPS, RAM, bandwidth, and prices.

- Cloudlet Generation: Cloudlets were created using a distinct dataset including real-world fog computing workloads. Each cloudlet was given its own set of characteristics, such as processing length, le size, memory requirements, and utilisation models.

- Customization of the BCO Algorithm: The BCO algorithm was modi ed and ne-tuned to meet the special requirements of load balancing in the fog computing environment. To improve the load balancing process, chemotaxis, reproduction, migration, and communication parameters were optimised.

- Execution of the Simulation Model: The simulation model was run with several experimentation situations to evaluate the performance of the BCO-based load balancing strategy. To evaluate the algorithm's performance under various scenarios, each scenario included variable fog node setups and workload conditions.
- Performance metrics collection: Relevant performance indicators such as latency,energy consumption, makespan, and cost were collected throughout simulation runs. These data revealed the algorithm's e ciency and e cacy at load balancing in fog computing.

The implementation phase of the project focused on creating a simulation model with the iFogSim simulator and integrating the BCO-based load balancing algorithm. The simulation model established a realistic fog computing environment, generated cloudlets, and ran the BCO algorithm for load balancing. The results comprised a well-con gured fog computing environment, cloudlets simulating real-world tasks, and performance statistics gathered during simulation runs. The Java programming language and iFogSim were the primary tools used during the implementation phase. The customised BCO algorithm and simulation model provided vital insights into the e ciency and e ectiveness of load balancing in fog computing settings.

6 Evaluation

This section provides a thorough evaluation of the load balancing algorithm in fog computing. BCO algorithm has been implemented, and its performance metrics are compared to those of other load balancing algorithms such as RR and TLB algorithm. The assessment focuses on critical performance indicators such as latency, energy usage, makespan, and cost. The experiments were carried out in a simulated fog computing environment, which closely resembled a real-world condition.

6.1 Performance Metrics 1 : Latency

Latency is an important statistic in fog computing since it shows the time delay between task submission and job completion. Lower latency means faster task processing and less communication overhead. In terms of latency, BCO outperforms RR and TLB.A substantial latency decrease of roughly 52.63% compared to RR and 41.94% compared to TLB is attained by BCO with 5 fog nodes. Even with 10 nodes, BCO continues to outperform RR and TLB by roughly 56.19% and 54.27%, respectively. As the number increases to 15 nodes, BCO shows its expertise in latency optimization by cutting latency by about 47.18% compared to TLB and 47.83% compared to RR. With 20 nodes, the pattern continues, with BCO achieving latency reductions of roughly 42.55% compared to RR and 45.45% compared to TLB. When compared to Round-Robin algorithm, BCO e ectively optimised task assignments based on colony behaviour, resulting in lower latency. TLB achieved reduced latency than RR due to its ne-grained task distribution mechanism as shown in Figure 4.

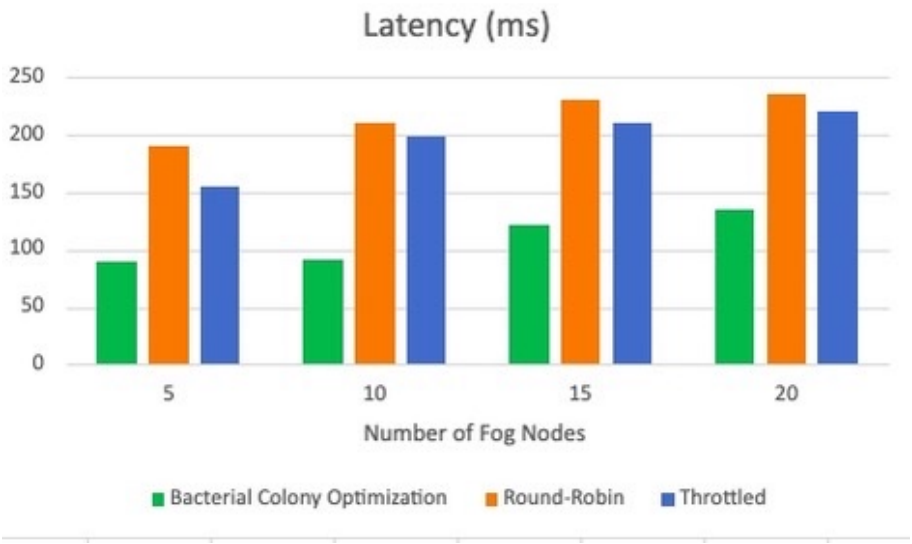

Figure 4: Latency obtained using BCO, Round-Robin and Throttled load balancing algorithms

6.2 Performance Metrics 2 : Energy Consumption

Considering fog computing is dependent on resource-constrained devices, energy consumption is a critical challenge. We assessed the total energy usage for executing the cloudlets across all fog nodes in each method. TLB outperformed both BCO and RR in terms of energy e ciency. TLB showed approximately 17.18% lower energy consumption compared to BCO and 14.55% lower energy consumption compared to RR. TLB's dynamic throttling method optimised resource utilisation and decreased idle times, resulting in lower total energy usage, even with increasing number of nodes as shown in Figure 5.

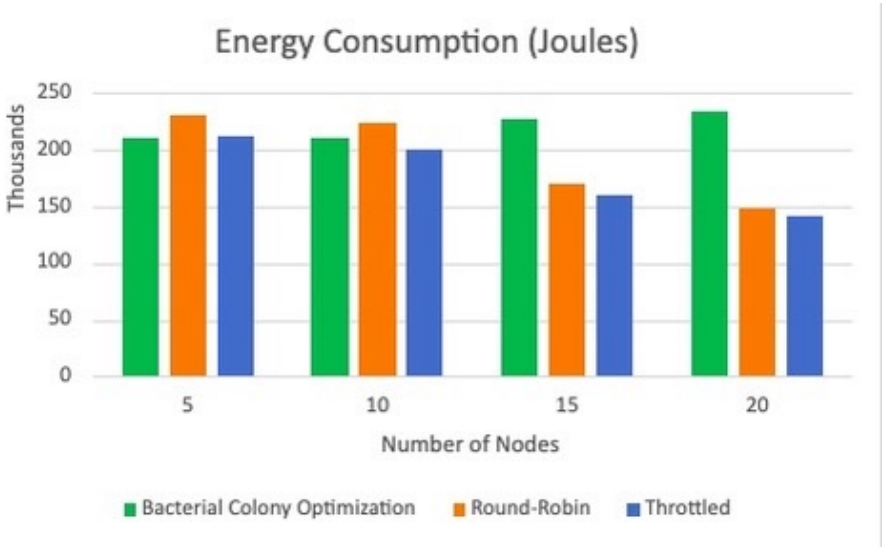

Figure 5: Energy Consumption obtained using BCO, Round-Robin and Throttled load balancing algorithms

6.3 Performance Metrics 3 : Makespan

Makespan is the entire amount of time required to execute all jobs in the workload. A shorter makespan suggests better work allocation and a shorter overall execution time. BCO outperformed RR and TLB with 5 nodes by a margin of 52.36% and 41.03%, respectively. This pattern persisted as the number of nodes rose: BCO with 10 nodes showed a 54.50% improvement over RR and a 42.35% improvement over TLB, a 39.30% improvement over RR and a 36.65% improvement over TLB, and nally a 37.19% improvement over RR and a 20.51% improvement over TLB with 20 nodes. Among the three methods, the BCO algorithm had the shortest makespan (as shown in Figure 6) due to its intelligent and dynamic load distribution.

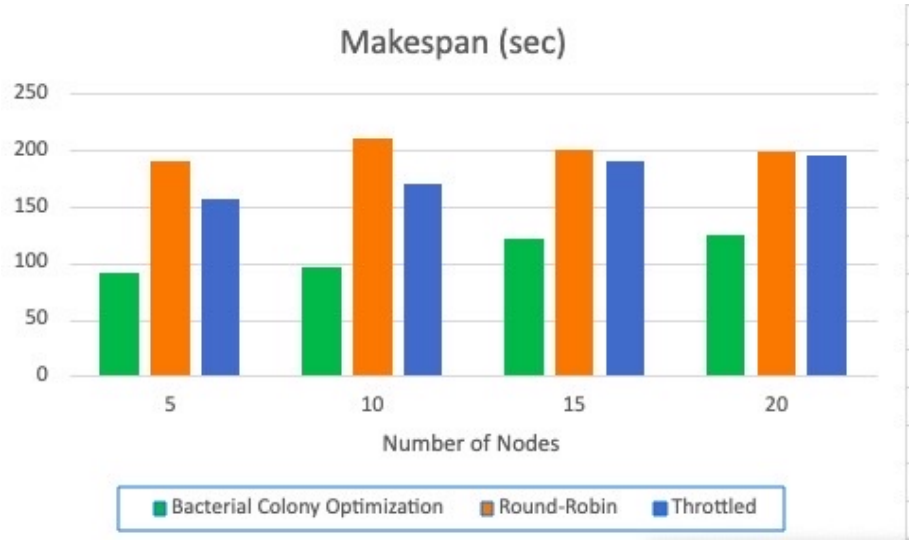

Figure 6: Makespan obtained using BCO, Round-Robin and Throttled load balancing algorithms

6.4 Performance Metrics 4 : Cost

Cost is an essential consideration for both providers and customers of fog computing services. It takes into account a number of elements, including resource utilisation, task execution time, and communication costs. In comparison to RR and TLB, BCO demonstrated lower load balancing costs. BCO showed cost reduction by 51.69% when compared to RR and TLB with 5 nodes. With more nodes, BCO was able to outperform the other algorithms, showing cost reductions of 43.90% compared to RR and 45.93% compared to TLB with 10 nodes, 38.17% compared to RR with 15 nodes, and 40.72% compared with TLB with 20 nodes. The BCO's colony-based optimisation and TLB's energy-aware method signi cantly decreased the overall cost of fog computing and outperformed Round-Robin algorithm in terms of cost as shown in Figure 7.

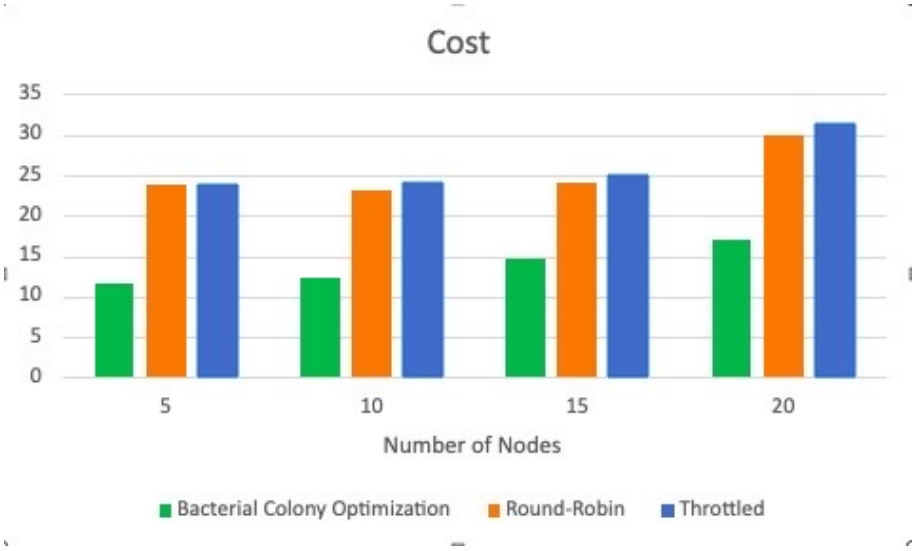

Figure 7: Cost obtained using BCO, Round-Robin and Throttled load balancing algorithms

6.5 Discussion

The use of the BCO algorithm in comparison to RR and TLB algorithms in the eld of fog computing load balancing results in signi cant performance gains, which are discussed. The study included evaluating each algorithm's scalability as the number of fog nodes increased.

The unique capability of BCO to mimic the behavior of bacterial colonies could be the reason for the constantly decreased latency it has been able to accomplish. BCO reduces execution latency by dynamically altering task assignments using colony-inspired methods. This intelligent allocation directly lowers communication overhead and increases overall task processing speed due to its ne-grained job distribution.

The ability of BCO in reducing energy consumption results from a complex balance between exploring and using the solution space. The algorithm's approaches for replication and removal, along with its dynamic adaptation, let it optimize resource usage. The approach is based with the guiding principles of BCO, which aim to imitate the adaptable and e ective characteristics of bacterial colonies. However, when compared with TLB and RR, BCO showed more energy consumption. The complex load balancing method used by BCO, which relies on chemotaxis, can result in frequent task migrations and more energy consumption. Better energy e ciency is observed in TLB due to its ne-grained throttling mechanism and the straightforward job allocation in RR. Further parameter adjustment could reduce energy consumption for BCO while maintaining a balance between energy e ciency and other performance measures.

The reduction of the makespan demonstrated by BCO highlights its pro ciency in balancing and allocating workloads. BCO intelligently distributes jobs to fog nodes via colony-based optimization, reducing the amount of time needed for task execution. This supports a theory that BCO's replication of bacterial movement and communication patterns results in e cient load distribution and, eventually, shorter makespan.

BCO algorithms shows cost-e ectiveness results, outperforming RR and TLB substantially. This accomplishment results from the innate capacity of BCO to optimize resource consumption and task distribution, hence reducing operating costs. The as-

sumption that these enhancements result from biological analogies of BCO ts well with the optimization variables that are based on concepts of e ective foraging and group behavior.

BCO improved performance in latency reduction, makespan reduction, and cost-e ectiveness can be linked to its biologically-inspired optimization parameters. The algorithm's exibility, resource optimization, and dynamic task distribution processes are consistent with the theories that support its biological roots. The overall performance improvement seen in BCO compared to RR and TLB highlights its potential as a promising load balancing approach in the fog computing environment. However, while choosing a load balancing method, individual use cases and needs must be considered, as the e ectiveness of each solution varies based on workload characteristics and fog infrastructure design.

7 Conclusion and Future Work

In this study, we used the Bacterial Colony Optimisation (BCO) technique to optimise load balancing in fog computing settings and it shows promising results. The goal was to improve overall system performance by e ciently allocating cloudlets to fog nodes based on their processing capacities, available resources, and workload characteristics. We learned a lot about the usefulness of the BCO method for load balancing by implementing and evaluating a simulation model in the iFogSim framework. By constantly changing the allocation of cloudlets to fog nodes, the BCO algorithm successfully optimised the overall performance of the fog computing environment.

The ndings of this research reveal that BCO provides signi cant bene ts in terms of latency reduction, demonstrating its capacity to speed up job processing and reduce communication overhead. Additionally, BCO demonstrates outstanding performance in lowering makespan, demonstrating e ective workload distribution and job execution. TLB's energy-e cient design outperforms BCO in terms of energy consumption, showing the complexity-energy trade-o .

To achieve e ective load balancing in a dynamic and resource-constrained fog computing environment, many performance measures must be carefully balanced. Although BCO provides impressive gains in latency, makespan, and cost-e ectiveness, future study should concentrate on optimizing its energy use while utilizing its distinctive features.

7.1 Future Work

The BCO-based load balancing method can be evaluated in real-world fog computing systems to validate its e ectiveness and scalability. Implementing the algorithm in a real-world fog infrastructure and comparing its performance to other load balancing algorithms would yield practical insights. Applying mathematical model could also be another good direction for future work, it could o er real-time adaptability and predictive insights.

Exploring the possibilities of integrating the BCO algorithm with other load balancing methodologies, such as genetic algorithms or ant colony optimisation, can lead to the development of hybrid algorithms that capitalise on the strengths of each technique. With the help of this research, we have gained a deeper understanding of load balancing optimization and created an opportunity for future investigation into algorithmic improvements and useful implementations in actual fog computing scenarios.

References

Abbasi, S. H., Javaid, N., Ashraf, M. H., Mehmood, M., Naeem, M. and Rehman, M. (2018). Load stabilizing in fog computing environment using load balancing algorithm, *Broadband and Wireless Computing, Communication and Applications*.

Ahmad, N., Javaid, N., Mehmood, M., Hayat, M., Ullah, A. and Khan, H. A. (2019). Fog-cloud based platform for utilization of resources using load balancing technique, *in* L. Barolli, N. Kryvinska, T. Enokido and M. Takizawa (eds), *Advances in Network-Based Information Systems*, Springer International Publishing, Cham, pp. 554{567.

Batra, S., Anand, D. and Singh, A. (2022). A brief overview of load balancing techniques in fog computing environment, *2022 6th International Conference on Trends in Electronics and Informatics (ICOEI)*, pp. 886{891.

Chakraborty, M. (2019). Fog computing vs. cloud computing, *ArXiv* $\mathbf{abs/1904.04026}$.

Chandak, A. and Ray, N. K. (2019). A review of load balancing in fog computing, *2019 International Conference on Information Technology (ICIT)*, pp. 460{465.

Gaurav, K. (2018). ifogsim: An open source simulator for edge computing, fog computing and iot, *Retrieved May* $\mathbf{16}$: 2019.

Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K. and Buyya, R. (2017). ifogsim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, *Software: Practice and Experience* $\mathbf{47}$(9): 1275{1296.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/spe.2509*

Kashani, M. H., Ahmadzadeh, A. and Mahdipour, E. (2020). Load balancing mechanisms in fog computing: A systematic review.

Kashyap, V. and V, A. K. (2022). Load balancing techniques for fog computing environment: Comparison, taxonomy, open issues, and challenges, *Concurrency and Computation: Practice and Experience* $\mathbf{34}$.

Kumar, V., Laghari, A., Karim, S., Shakir, M. S. and Brohi, A. A. (2019). Comparison of fog computing & cloud computing, *International Journal of Mathematical Sciences and Computing* .

Mahmud, R., Kotagiri, R. and Buyya, R. (2018). *Fog Computing: A Taxonomy, Survey and Future Directions*, Springer Singapore, Singapore, pp. 103{130.

Margariti, S. V., Dimakopoulos, V. V. and Tsoumanis, G. (2020). Modeling and simulation tools for fog computing | a comprehensive survey from a cost perspective, *Future Internet* $\mathbf{12}$(5): 89.
URL: *http://dx.doi.org/10.3390/ 12050089*

Mouradian, C., Naboulsi, D., Yangui, S., Glitho, R. H., Morrow, M. J. and Polakos, P. A. (2018). A comprehensive survey on fog computing: State-of-the-art and research challenges, *IEEE Communications Surveys Tutorials* $\mathbf{20}$(1): 416{464.

Mukherjee, M., Shu, L. and Wang, D. (2018). Survey of fog computing: Fundamental, network applications, and research challenges, *IEEE Communications Surveys Tutorials* **20**(3): 1826{1857.

Naha, R. K., Garg, S., Georgakopoulos, D., Jayaraman, P. P., Gao, L., Xiang, Y. and Ranjan, R. (2018). Fog computing: Survey of trends, architectures, requirements, and research directions, *IEEE Access* **6**: 47980{48009.

Nguyen, B. M., Thi Thanh Binh, H., The Anh, T. and Bao Son, D. (2019). Evolutionary algorithms to optimize task scheduling problem for the iot based bag-of-tasks application in cloud{fog computing environment, *Applied Sciences* **9**(9).
URL: *https://www.mdpi.com/2076-3417/9/9/1730*

Niu, B. and Wang, H. (2012). Bacterial colony optimization, *Discrete Dynamics in Nature and Society* **2012**: 1{28.

Rani, S. (2022). Analytic vision on fog computing for e ective load balancing in smart grids, *Transactions on Emerging Telecommunications Technologies* **33**(2): e3855.
URL: *https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.3855*

Rehman, S., Javaid, N., Rasheed, S., Hassan, K., Zafar, F. and Naeem, M. (2019). Min-min scheduling algorithm for e cient resource distribution using cloud and fog in smart buildings, *in* L. Barolli, F.-Y. Leu, T. Enokido and H.-C. Chen (eds), *Advances on Broadband and Wireless Computing, Communication and Applications*, Springer International Publishing, Cham.

Revathi, J., Eswaramurthy, V. P. and Padmavathi, P. (2019). Bacterial colony optimization for data clustering, *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)* pp. 1{4.

Singh, S. P., Kumar, R., Sharma, A. and Nayyar, A. (2020). Leveraging energy-e cient load balancing algorithms in fog computing, *Concurrency and Computation: Practice and Experience* **34**.

Singh, S. P., Sharma, A. and Kumar, R. (2020). Design and exploration of load balancers for fog computing using fuzzy logic, *Simul. Model. Pract. Theory* **101**: 102017.

Xu, X., Liu, Q., Qi, L., Yuan, Y., Dou, W. and Liu, A. X. (2018). A heuristic virtual machine scheduling method for load balancing in fog-cloud computing, *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing, (HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)* pp. 83{88.