Devising an Ideal Network Slicing Ring Fencing Ratio for Cloud Energy Infrastructure

MSc Research Project
Cloud Computing

Vinay Sriram Iyer
Student ID: X23203595

School of Computing
National College of Ireland

Supervisor: Yasantha Samarawickrama

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Vinay Sriram Iyer
Student ID:	X23203595
Programme:	Cloud Computing
Year:	2024
Module:	MSc Research Project
Supervisor:	Yasantha Samarawickrama
Submission Due Date:	12/12/2024
Project Title:	Devising an Ideal Network Slicing Ring Fencing Ratio for Cloud Energy Infrastructure
Word Count:	7477
Page Count:	24

Signature:	Vinay Sriram Iyer
Date:	12th December 2024

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.	☐

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

AI Acknowledgement Supplement

[MSc Research Project]

[Devising an Ideal Network Slicing Ring Fencing Ratio for Cloud Energy Infrastructure]

Your Name/Student Number	Course	Date
Vinay Sriram Iyer/x23203595	MSc in Cloud Computing	12/12/2024

This section is a supplement to the main assignment, to be used if AI was used in any capacity in the creation of your assignment; if you have queries about how to do this, please contact your lecturer. For an example of how to fill these sections out, please click here.

AI Acknowledgment

This section acknowledges the AI tools that were utilized in the process of completing this assignment.

Tool Name	Brief Description	Link to tool
N/A	N/A	N/A

Description of AI Usage

This section provides a more detailed description of how the AI tools were used in the assignment. It includes information about the prompts given to the AI tool, the responses received, and how these responses were utilized or modified in the assignment. **One table should be used for each tool used**.

[N/A]	
[N/A]	
[N/A]	[N/A]

Evidence of AI Usage

This section includes evidence of significant prompts and responses used or generated through the AI tool. It should provide a clear understanding of the extent to which the AI tool was used in the assignment. Evidence may be attached via screenshots or text.

Additional Evidence:

[N/A]

Additional Evidence:

[N/A]

Devising an Ideal Network Slicing Ring Fencing Ratio for Cloud Energy Infrastructure

Vinay Sriram Iyer
X23203595

Abstract

Disparate differences in total energy consumption by heterogeneous architectures often magnify through designed network topologies in cloud energy data centers that contrast sharply against the actualised energy consumption. A preliminary correlation can be drawn by how network operators design high-quality 'Network Slices' composed of logical ring fences akin to Radio Access Networks(RAN's) through end-to-end network slicing that reduces the network 'churn' often associated in large losses of energy consumption in cloud energy infrastructures. However, certain limitations still exist with significant cutoffs in workload resource specific requirements and the actualised energy consumption drawn from 'Network Slices'. This paper proposes an ideal network slicing to ring fencing ratio through an Iterative Heuristic Energy-Aware Non-Convex equation that optimises cloud architectural management through optimal convergence in energy consumption with refined computational execution time. Through subsequent iterative testing against a consistent convergence criterion, the Iterative Heuristic Energy-Aware Non-Convex equation reported an efficiency gain of 91.46% in total energy consumption when benchmarked against a Heuristic AUGMENT Non-Convex Algorithm(Hossain and Ansari. achieved efficiency gain of 63.89%) and Priority Selection Offloading Algorithm(Anajemba et al. achieved efficiency gain of 58.6%). By benchmarking speed in convergence with the overall energy consumption and refined computational execution time, a network-slicing ring-fencing ratio ensures a balance in idealised energy consumption is met between the network architect, operator and client.

Keywords - network topologies, energy efficiency, Network Slices ring fences, convergence criterion, execution time

1 Introduction

Novel heterogeneous cloud topologies currently offering resource and workload resource characteristics often fail to satisfy the quality of service (QoS) requirements sought by end-users. Some of these infrastructural requirements that vary in traffic volume include CPU utilisation, network transmission and disk bandwidth which is a substantial point of concern for network operators to design application workload models according to Saxena and Sivalingam (2022). Studies such as (Zhang, Kosta and Mogensen; 2023) have shown that network operators can establish designs for workload resource efficient algorithms by optimising such application specific parameters. Yet, certain offsets with overall resource specific requirements still exist while optimising paradigms for architectural scenarios.

Network operators tending to client demand for real-world applications desire differing system specific requirements that result in large offsets of total energy consumption. Clearly, an optimised solution for cloud energy models is desirable to network operators on balancing application workload and workload resource specific requirements while improving Quality of Service(QoS).

1.1 Motivation and Background

By defining high quality efficiently allocated ’network slices’ in independent 5G networks, network architects are well aware of establishing designs for cloud topologies with the required key performance indicators shown in studies such as (Lorincz et al.; 2024). Like a Radio Access Network(RAN), a ’Network Slice’ can be likened to an independent topology drawn by ring-fencing techniques that constructs logical ring fences aiming on balancing optimal resource allocation while idealising energy consumption. While ring-fencing techniques may achieve a close-to-ideal energy efficiency, similar abnormalities exist with lowered computational execution time or suboptimal resource coverage discussed in (Masoudi et al.; 2022). A better approach for cloud topologies is to ’map’ resources efficiently in ’high-quality’ network slices with improved ring fences by optimising a computational resource, fence and overall slice ratio as much as possible. While studies such as (Hossain and Ansari; 2021) and (Anajemba et al.; 2020) have described Non-Orthogonal Multiple Access(NOMA) clustering applied to network slices with increased channel gain and efficient computational offloading priority respectively, certain cutoffs associated with disparate energy consumption still exist for cloud-centric architectures. As both algorithms closely model the network slicing to ring fencing ratio through convergence in energy consumption, an iterative heuristic energy-aware non-convex equation in this paper can prove advantageous by minimising the network ’churn’ as much as possible in distributed, heterogeneous architectures.

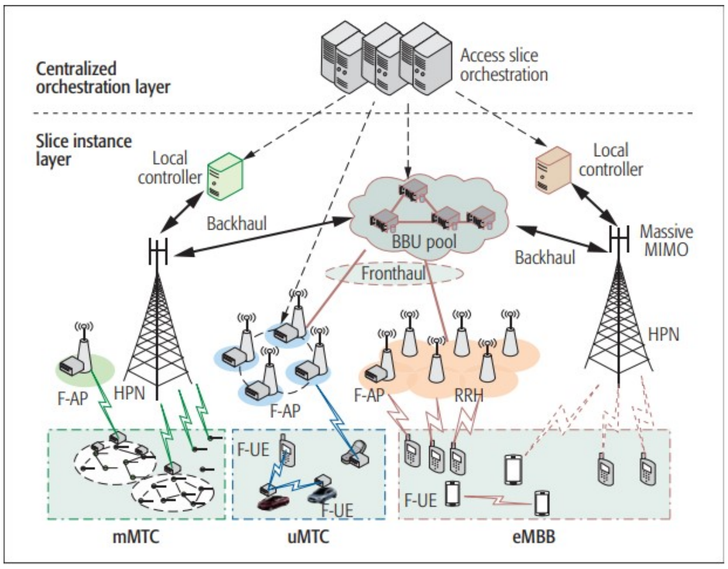


Figure 1: Centralised orchestration-based f-RAN network access slicing in (Xiang et al.; 2017).

1.2 Project Specification

By designing a solution specific uneven network slicing ring fencing architecture starting from the root 'network slice', network operators should be able to maintain a high level of interaction with clients seeking user end services with minimised energy consumption and task execution time. Based on the above requirement, this paper shall aim on determining an answer to the research question below:

1.2.1 Research Question

An investigation has been conducted in this paper to determine a solution to the energy efficiency problem by providing a goal for idealised energy consumption through the Iterative Heuristic Energy-Aware Non-Convex Equation for Cloud Energy Infrastructure:

"How well can cloud energy infrastructures optimise energy efficiency through convergence in energy consumption in a network slicing ring fencing ratio described by the iterative heuristic energy-aware non-convex equation? Can network architects benefit from such an equation when designing scalable cloud energy infrastructures for network operators to user centric services?"

1.2.2 Research Objective

The following experiments in this research paper shall address and validate the research question above:
Objective 1: Preprocessing a dataset that contain the performance metrics of 1750 Virtual Machines(VM's) attached to fast storage area network (SAN) devices and slower Network Attached Storage (NAS) devices.
Objective 2: Incorporating the uneven network slicing ring fencing architecture for the desired iterative heuristic energy-aware non-convex equation, a heuristic AUGMENT non-convex algorithm and a priority selection offloading algorithm through a simulation environment by evaluating performance through convergence in energy consumption.
Objective 3: Assessing scalability for these three algorithmic programs in a simulation environment and real-world testing by ensuring the iterative heuristic energy-aware non-convex equation idealises speed in convergence with refined execution time as an optimised solution.

1.2.3 Report Structure

The rest of this paper is arranged as follows: In Section 2, an overview of the literature review shall examine the total energy consumption that incorporates iterative, heuristic and energy-aware specifications against the inadequacies of the heuristic AUGMENT non-convex algorithm and priority selection offloading algorithm. Section 3 shall describe the derivation of the iterative heuristic energy-aware non-convex equation along with a description of the respective simulation environments. Section 4 shall outline the algorithmic programs and convergence criterion for optimising sensitivity in energy consumption. Section 5 shall provide a description of the incorporated algorithms in their environmental simulation setups critical for benchmarking. The last two sections shall examine the experimental results derived as a result of the research objectives along with a summary of the research paper's objectives and recommendations for future work to be useful for academia and the cloud computing community.

2 Related Work

A critical analysis of the derivation towards the iterative heuristic energy-aware non-convex optimisation equation is necessary for evaluation by ensuring speed and value in convergence of total energy consumption of the cloud model. The following sections shall touch upon the major components necessary for implementation. In the first subsection, the energy optimisation problem shall first be discussed in it's formulation that jointly allocates communication like task-to-VM mapping and computation like task scheduling. In subsection 2.2, the benefit of achieving close-to-ideal optimisation with latency constraints along with the drawbacks of the heuristic AUGMENT non-convex equation shall be discussed here. Subsection 2.3 shall refine the overall resource allocation with offloading along with the drawbacks of the priority offloading equation that iteratively derives the refined total energy consumption equation here.

2.1 Overview of the Non-Convex Optimisation problem

Issues with Network Slicing Parameters: Transforming high-quality network slices from designed scalable network topologies cannot closely optimise energy efficiency from conceptualisation as issues exist with heterogeneous service requirements. Further discussion in (Chien et al.; 2020) and (Kwak et al.; 2017) raises discussion with ultra-reliable, low-latency communication(uRLLC) traffic influencing inter-edge traffic without accounting for heterogeneous extreme service requirements and fault-delay tolerance through downstream service quality which further impacts overall performance.
Network Slices utilising resource-intensive computational network resources in smaller modular architectures against slices with distributed workloads might ensure inconsistencies in striking a balance with idealised energy consumption and poor computational offloading. (Foukas et al.; 2017) raises discussion on the poor flexibility of end-to-end network slicing to jointly allocate parameters as network resources.

The Open Industry-Wide SLA standard and 'Functional Splitting': By fulfilling an ideal end-to-end slice that is fine-grained but conforms to a strict SLA might suffer from interoperability issues and requirements that vary from vendor to vendor. The required parameters incorporated in the model should account for adaptability while not compromising on service quality. Designing network functions with the intention of 'functional splitting' such as implemented caching in (Sriram et al.; 2019) could incorporate the required bandwidth for reducing the tradeoffs in overall energy consumption. However, the end-to-end delay is still present such as content access delay and cache induced congestion.

The Non-Convex Optimisation Equation: To idealise a holistic cloud energy framework that fulfills the open-industry wide standards, the optimisation problem should account for joint optimisation, communication and computation with SLA requirements that are non-vendor centric. The idealisation of a non-convex expression that integrates a central cloud, network slice and ring fence is insufficient without accounting for latency constraints. The drawbacks of NOMA clustering and the incorporation of latency to increase the search space of the derived optimisation equation shall be discussed in the next section.

2.2 Overview of the Non-Convex Heuristic Equation

Leveraging Desired Parameters for Non-Convex Optimisation: Recasting the non-convex optimisation equation into a convex form by joint optimisation can be leveraged with certain parameters such as energy-aware constraints to facilitate a near-optimal solution according to (Hossain and Ansari; 2021). However, present metrics such as datarate or throughput incorporated with communication techniques between network slices may be insufficient to cope with transmit power according to (Buzzi et al.; 2016).

Increased Carbon Footprint of Central Cloud Models: While uRLLC traffic in heterogeneous services offloads high-compute intensive tasks according to (Mao et al.; 2016), an absence of a centralised cooling cloud unit presents large inadequacies to consistently refine energy consumption through iteration. Disparities in carbon-based energy still exist with mission-critical devices incorporating heterogeneous services through central cooling cloud units, with user-level services subjected to inadequate computational data rate according to (Fehske et al.; 2011) and (Auer et al.; 2011).

Incorporating QoS and the Impact of NOMA: Network operators tasking 'high-quality' slices to clients may consider 'shaving' off unnecessary slices to mitigate their exponential rise in energy consumption or task execution time. While NOMA clustering proposes user grouping based on the maximum delay tolerance of slices, inconsistent power levels leads to unfair resource allocation. 'High-Quality' network slices incorporating NOMA cluster sizes with larger $N_{\max}$ values such as the heuristic AUGMENT non-convex algorithm discussed in (Hossain and Ansari; 2021) results in computational overhead with aggregated network slices. Network operators should be able to consider the specific frameworks that only contribute to optimal energy efficiency.

The Heuristic Non-Convex Equation: To balance the associated requirements with efficient network slicing, end-to-end latency requirements ensures the required performance levels are met for cloud energy infrastructures. Incorporating $D_{\min}$ and $D_{\max}$ in the non-convex equation ensures baseline performance levels are set to a defined threshold and ring-isolation efficiency is balanced for the non-convex heuristic energy consumption equation.

2.3 Overview of the Iterative Heuristic Energy-Aware Non-Convex Equation:

Priority Offloading: Modeling an energy consumption equation on the basis of end-to-end latency constraints seems sufficient to meet QoS in an optimised solution desiring speed in convergence with refined execution time. However, high-compute computational network resources suffers from poor capacity to facilitate communication with increased aggregation and complexity of resource allocation. Computational network resources need to evaluate tasks through constant inter-VM communication which mitigates tradeoffs in energy consumption.

Incorporation of Dynamic Voltage Scaling(DVS): The joint allocation of heavier workloads in resource computational network resources through heterogeneous architectures with communication still remains an issue.

Chip design technique accounts for energy consumption should be accounted for the non-convex heuristic energy consumption equation with the correct architectural changes in place.

The Iterative Heuristic Energy-Aware Non-Convex Optimisation Equation: Induced delay with computational offloading between Smart Communicating Devices(SCD's) discussed in (Tao et al.; 2017) and (Anajemba et al.; 2020) and computational network resources may lead to a significant increase in energy consumption through subsequent iterative testing for the priority selection offloading algorithm. Incorporating Dynamic Voltage Scaling(DVS) while solving multiuser offloading shall focus solely on energy consumption through the iterative heuristic energy-aware non-convex optimisation equation. Overall, the iterative heuristic energy-aware non-convex equation aims to address holistic optimal energy-efficient cloud infrastructures to be fit for aggregation and academia.

2.4 Research Niche

2.4.1 Summary of Literature Review

The related research papers that represent the basis of the energy consumption equation are compared based on their framework, scenario, advantages, and limitations in table 1.

Table 1: Summary of Reviewed Papers

References	**Framework**	**Scenarios**	**Advantages**	**Limitations**
(Chien et al. 2020)	Algorithm to adjust capacity.	Augmented Reality (AR).	Avoid over-provisioning with latency constraints.	Bottleneck for uRLLC through enhanced mobile broadband(emBB) services.
(Kwak et al. 2017)	Bandwidth slicing algorithm.	Video Streaming Services.	Increase transmit power and quality.	Poor delay-tolerance.
(Foukas et al. 2017)	5G mobile network architecture.	Specialized sensors.	Enhanced deployment.	Poor flexibility in outside network criteria.
(Sriram et al. 2019)	'Functional Splitting'.	5G mobile streaming.	Minimises power consumption by cache induced access delay.	Increased congestion.
(Hossain & Ansari 2021)	NOMA	Telemedical surgeries.	Reduces needless allocations.	Communication impairment with aggregated clusters.
(Buzzi et al. 2016)	Next generation (5G).	Drones.	Scaled ubiquitous connectivity.	Increased communication capacity.
(Fehske et al. 2011)	Green Communication Mobile Technologies.	Broadband subscriptions.	Increased penetration.	Increased carbon footprint in data centers.
(Auer et al. 2011)	Energy efficient evaluation framework.	3G mobile computing.	Independent power from traffic load.	LTE rollout with low traffic workloads.
(Anajemba et al. 2020)	Group offloading for multi-access MEC.	IoT smart devices.	Faster data processing through DVS.	Poor QoS without end-to-end delay constraints.
(Tao et al. 2017)	Energy minimisation problem.	Real-time mobile cloud computing.	Offloading on energy consumption.	Lack of energy-aware chip design techniques.

3 Methodology

Based on the considerations identified in the literature analysis, the iterative heuristic energy-aware non-convex optimisation equation aims on optimising the network slice - ring fence - network resource energy efficiency and is described through a mathematical outline and description of said associated components. Next, the preprocessing steps necessary for extracting relevant computational network resource data from the GWA-T-12 Bitbrains dataset was described altogether in 3.1. In 3.2, Assessing performance and evaluating scalability of the incorporated algorithmic programs through the uneven network slicing ring fencing architectures was summarised by CloudSim Calheiros et al. (2011) environment and real-world AWS testing respectively. Determining the captured metrics relevant to the techniques and outputs produced at the final stage of implementation shall provide a key idea of research and results in 3.3.

3.1 Proposed Iterative Heuristic Energy-Aware Non-Convex Optimisation Equation

The derivation of the total energy consumption equation that idealises the network slice - ring fence - network resource ratio for cloud energy infrastructure aims on optimising energy efficiency or E_{total} through convergence and optimal resource allocation of all respective components while satisfying the necessary constraints.

Non-Convex Optimisation Equation: The total energy consumption of the network is expressed as the sum of energy consumption through the central cloud model, network slice, ring fence and computational network resource to derive the non-convex optimisation equation. The equation accounted for CPU processing and energy consumption of network resources including inter-network slice communication that factors in bandwidth according to (Masoudi et al.; 2022). Centralised cooling associated with the core computational tasks is provided as well. Based on the associated requirements, the energy consumption is presented as:

$$E_{\text{total}} = \frac{1}{\zeta_{\text{CC}}}\left(P^{\text{CC}}_{\text{compute}} + P^{\text{CC}}_{\text{cool}}\right)T + \frac{P^{\text{CC}}_{\text{proc}}}{\zeta_{\text{CC}}}\sum_{s\in S}\sum_{u\in U_s} d_{\text{CPU},u,rf} + \sum_{\text{NS}\in\text{Network}}\sum_{\text{RF}\in\text{NS}}\left(\frac{1}{\zeta_{\text{RF}}}P^{\text{RF}}_{\text{static}}T + \sum_{u\in U_s} E^{\text{NR}}_{u,rf}\right)$$
$$+ \sum_{s\in S}\sum_{s'\in S} B_{s,s'}P^{\text{comm}}_{s,s'}T + \sum_{j\in S}\sum_{i\in\mathcal{U}_j}\left(\alpha_i^j D^j_{\max} + \beta_i^j D^j_{\min}\right) + \tau\sum_{s\in S}\sum_{u\in U_s} S^3 t_{u_i}. \tag{1}$$

where ζ_{CC} is the power efficiency of the central cloud, T is the total operation time of all network slices, with $P^{\text{CC}}_{\text{compute}}$ and $P^{\text{CC}}_{\text{cool}}$ representing the computational power and cooling power consumption of the cloud respectively. $\frac{1}{\zeta_{\text{RF}}}$ and $P^{\text{RF}}_{\text{static}}$ represent the efficiency and static power consumption of the ring fence respectively where $\sum_{\text{RF}\in\text{NS}}\left(\frac{1}{\zeta_{\text{RF}}}P^{\text{RF}}_{\text{static}}T\right)$ represents the total energy consumption of ring fences assigned to a network slice. $\sum_{u\in U_s} E^{\text{NR}}_{u,rf}$ is the energy consumption of a network resource u within each ring fence rf and $\frac{P^{\text{CC}}_{\text{proc}}}{\zeta_{\text{CC}}}\sum_{s\in S}\sum_{u\in U_s} d_{\text{CPU},u,rf}$ accounts for the energy consumed due to the CPU processing of all network resources within each ring fence rf, considering processing power consumption $P^{\text{CC}}_{\text{proc}}$. $\sum_{s\in S}\sum_{s'\in S} B_{s,s'}P^{\text{comm}}_{s,s'}T$ represents the energy consumption due to communication between different network slices s and s', considering bandwidth $B_{s,s'}$ and power consumption $P^{\text{comm}}_{s,s'}$ over time T.

The search space of the iterative heuristic energy-aware non-convex optimisation equation was increased by incorporating QoS, latency constraints and energy-aware scheduling through DVS as discussed in (Hossain and Ansari; 2021) and (Anajemba et al.; 2020). $\sum_{s \in S} \tau S^3 t_{u_i}$ incorporates the power consumption of the CPU based on the dynamic voltage scaling technique, where $\sum_{s \in S} \tau S^3 t_{u_i}$ is the parameter. τ is the coefficient subject to chip design, S is the computational speed of the CPU, and t_{u_i} is the task execution time for user u_i. $\sum_{j \in S} \sum_{i \in U_j} \alpha_i^j D_{\max}^j$ and $\sum_{j \in S} \sum_{i \in U_j} \beta_i^j D_{\min}^j$ incorporates the minimum and maximum latency constraints for users allocated to slices respectively.

3.1.1 GWA-T-12 Bitbrains Dataset

The GWA-T-12 Bitbrains Dataset contains the network performance metrics of 1750 VM's including information about resource usage like CPU, memory, disk, network data from fast storage network(fastStorage) and slower network attached storage(Rnd) devices. For the relevant preprocessing in Python, the relevant resource performance metrics specific to energy consumption in the GWA-T-12 Bitbrains dataset such as CPU usage, memory usage, disk usage, and network I/O are provided in steps below:
Understanding the Data Structure: Two dataset traces labelled fastStorage and Rnd that consist of the network performance metrics of 1250 VM's and 500 VM's connected to fast storage area network (SAN) devices and slower network attached storage (NAS) devices were loaded from a specified directory with the separator `';\t'` properly splitting the data in the dataset. With libraries like Pandas for data manipulation, the required parameters were renamed in the preprocessed program.
Filtering the relevant network performance metrics: The overall resource computational load was aggregated for each relevant trace per hour for a network computational resource as the mean resource usage can fluctuate significantly as compared to daily usage with energy consumption. With non-numeric values in the 'Timestamp' column, CPU core values rounded to either 0 or 1 and extracted relevant time-series data, irrelevant fields were filtered out and normalisation with consistent scaling is ensured.
Loading and structuring the data for CloudSim: Through proper concatenation of the relevant Dataframe, the preprocessed dataset from each network storage device is converted into a format as preprocessed dataset programs. With columns like timestamp, CPU, memory, disk, and network usage, the script saved the cleaned dataset in CSV files labelled as preprocessed_fastStorage.csv and preprocessed_rnd.csv for inputs in CloudSim.

3.2 CloudSim and AWS Real-World Testing

3.2.1 CloudSim

CloudSim is a flexible, extensible simulation framework that provides benefits through time effectiveness where network researchers and practitioners can simulate, model, assess, and evaluate cloud-based applications for heterogeneous real-world cloud energy infrastructures. CloudSim was written in Java for this project for extensively simulating the incorporated algorithmic programs through the network slicing ring fencing architecture. Key components of the CloudSim tool include the Datacenter Characteristics, DatacenterBroker, NetworkTopology, Virtual Machines, and Cloudlets. For the incorporated network slicing ring fencing architecture, network slices, ring fences and computational network resources are defined as datacenters, hosts inside datacenters and VM's respectively.

A cooling constraint central to the desired total energy consumption equation was simulated indirectly in the architecture to model initial energy constraints for each algorithmic program.

1. **Datacenter**: Represents the physical cloud resources which are later assigned as 'Network Slices,' discussed during the design specification. Incorporated datacenters or physical machines(PM's) are ring fences or hosts inside datacenters.
2. **DatacenterBroker**: Represents an intermediary between the cloud user and cloud provider where the stated goal is to reduce QoS through overall optimal response time.
3. **NetworkTopology**: Necessary information on the network configuration and topology for the incorporated network slicing ring fencing architecture which is discussed in the design specification.
4. **Hosts**: Allocated the performance metrics central to the simulation including CPU, memory, disk, and network bandwidth.
5. **Virtual Machines**: Modeled resource-specific information analogous to the multi-objective optimisation energy approach in Choudhary and Perinpanayagam (2022). The simulated resource definitions such as CPU, memory, storage and network bandwidth were allocated to each network resource inside each ring fence.
6. **Cloudlets**: Created the tasks performed in datacenters or 'Network Slices' to virtual machines.

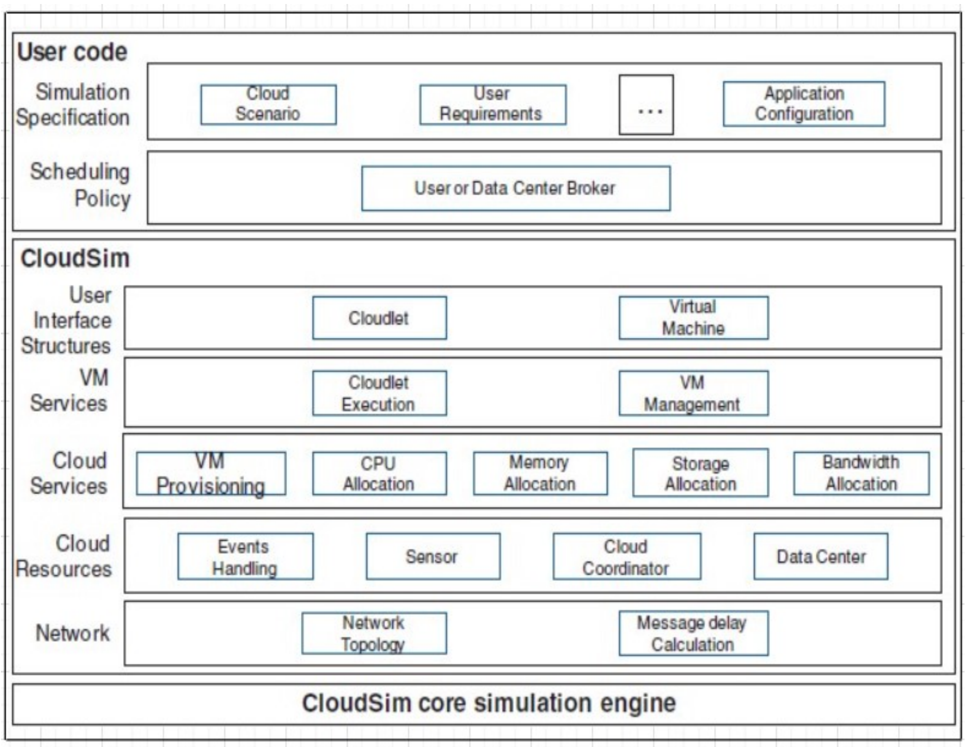


Figure 2: Layered CloudSim architecture that incorporates the uneven high-quality network slicing ring fencing architecture as per Calheiros et al. (2011).

3.2.2 AWS Real-World Testing

Architectural Incorporation:

The differences in energy consumption with convergence and resource allocation of network slices can be more efficiently highlighted with two uneven multi-scale architectures differing in complexity and size. By designing denser, uneven 'aggregated' architectures with high-quality network slices that accommodate ring fences with a larger number of hosts, network architects benefit greatly from the efficiency gain in scalability testing by seeking observations with differences in energy metrics. Clients seeking heterogeneous requirements such as deployed mobile virtual network functions(VNF's) described in Mesodiakaki et al. (2021) can benefit greatly with energy-efficient models spread across aggregated slices. For AWS testing, Java was the primary language for the simulation. The various network slices, ring fences with their computational network resources was mimicked through their logical components which are virtual private clouds with subnets, security groups and elastic compute cloud(EC2) instances. AWS Lambda served as the counterpart to CloudSim IDE by evaluating each algorithmic program against each other. AWS CloudWatch correspondingly provided the output of key performance metrics collected in real-time to validate the convergence of the desired iterative heuristic energy-aware non-convex optimisation equation. An AWS S3 Bucket served as a means of storage for the output of the algorithmic programs collected at the end of each Lambda Test event for minutely examining the performance metrics and respective logs.

3.3 Performance Metrics

To efficiently reduce tradeoffs between energy consumption and system efficiency that satisfies cloud energy model requirements, optimising paradigms such as convergence in energy consumption should closely balance user and system requirements with overall refined computational network resource execution time. Some important metrics involved include task completion time, energy consumption, total energy consumption with cooling, response time, iteration time and average iteration time in the table below:

Table 2: Performance Metrics Description and Justification

Performance Metric	**Description**	**Justification**
Task Completion Time	Time taken for overall task completion for each algorithmic program	Evaluates and assesses each algorithmic program's ability to handle tasks efficiently.
Energy Consumption	Measured the total energy used during task execution, inter-network resource communication and offloading for each algorithm.	Highlights the energy consumption of each algorithmic program.
Cooling Power Consumption	Cooling constraint incorporated indirectly to maintain thermal stability for each network slice.	Highlights the energy constraints associated with inefficient energy usage imposed by inter-slice communication and associated components.
Total Energy Consumption with Cooling	Total energy usage from all logical components of the simulation.	A holistic view of energy consumption for evaluating performance and assessing scalability for each algorithmic technique.
Iteration Time	Time taken for one iteration of the algorithm.	Gauges the overall computational efficiency for each algorithmic customisation.
Average Iteration Time	Average time taken through the number of iterations required for each algorithm to converge.	Compares the computational time for each algorithm by ensuring a fair criteria for convergence.
Convergence Time	Overall time taken for the algorithm for convergence.	Ensuring overall effectiveness of the network slicing ring fencing ratio with convergence speed.

4 Design Specification

4.1 Proposed Architectural Diagrams

Two architectural diagrams representing the implementation of the uneven network slicing ring fencing architecture are provided below:

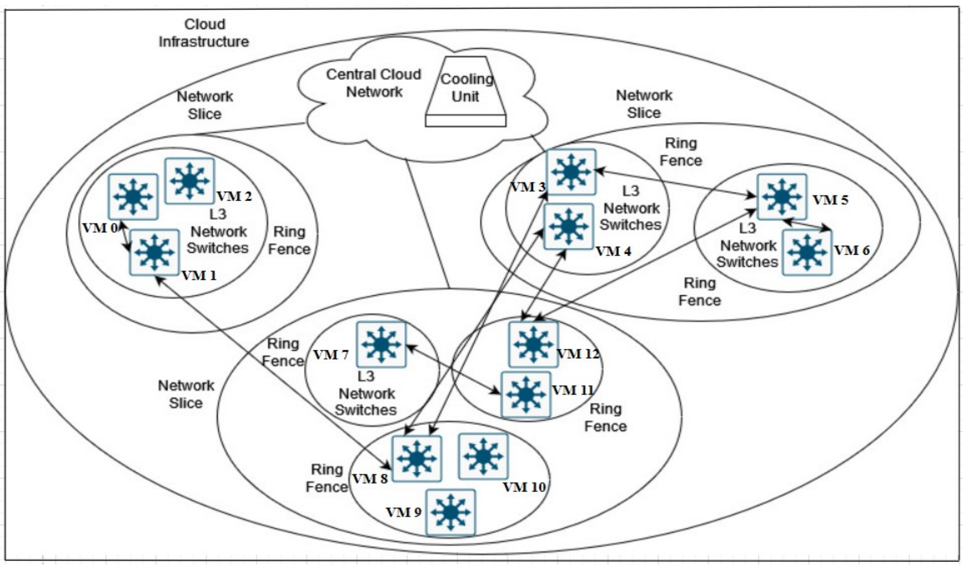


Figure 3: A simplistic representation of an uneven high-quality network slicing ring fencing architecture. The equation simulated different configurations of network slices with a varying number of ring fences and network resources assigned dynamically depending on the user requirements.

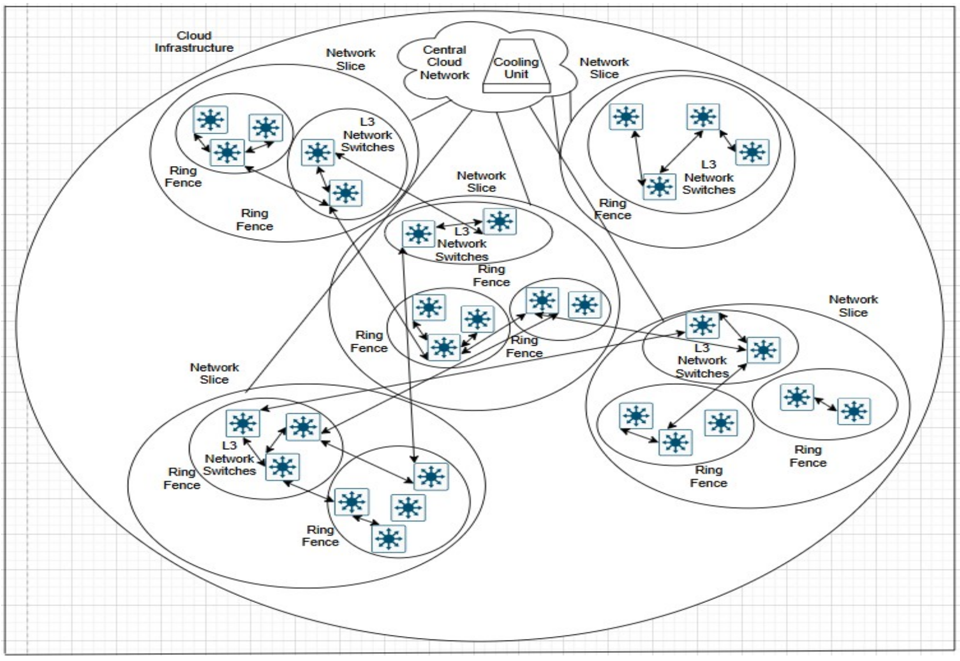


Figure 4: An aggregated representation of an uneven high-quality network slicing ring fencing architecture. The equation was utilised to simulate an aggregated number of network slices and respective components by assessing increased resource utilisation.

4.2 Algorithmic Incorporation

4.2.1 Iterative Heuristic Energy-Aware Non-Convex Algorithm

Algorithmic customisations: The following equation was translated in Java code by making use of the equation and it's respective parameters.
Since the iterative heuristic energy-aware non-convex algorithm has an existing large search space with the evaluation of it's architectural components and specifications as detailed in Anajemba et al. (2020), incorporating additional discrete variables like task scheduling and task-to-VM mapping helped tested the threshold of comparison against other algorithmic programs.

Table 3: Discrete variables for increased complexity

Performance Metric	Description	Justification
taskExecutionTime	Facilitated task scheduling for a VM based on the chip design technique.	Assesses and refines the execution time by energy-aware scheduling.
commBandwidth	Classified tasks with bandwidth due to communication from network slice-to-slice.	Models the total network energy consumption for each slice's energy consumption according to Masoudi et al. (2022).

4.2.2 Heuristic AUGMENT Non-Convex Algorithm

Algorithmic customisations: The heuristic AUGMENT non-convex algorithm primarily utilised a search approach with the sorted network slices in ascending order of maximum latency with an intention on minimising energy consumption. Aspects like NOMA clustering to UE's and inter VM-VM communication modeled the incorporated program to determine the degree of convergence in energy consumption.

Table 4: Discrete variables for an increased search space

Performance Metric	Description	Justification
dynamicBandwidth	Adjusted for current load and task requirements with dynamically adjusted bandwidth based on a task size factor.	Can maximise energy efficiency in downstream networks as described in Abuajwa and Mitani (2024).
bandwidthPenalty	Induced latency-based penalties for disparate bandwidth scenarios.	Can simplify complex network relationships and improve the performance through power allocation according to Huang et al. (2019).

4.2.3 Priority Selection Offloading Algorithm

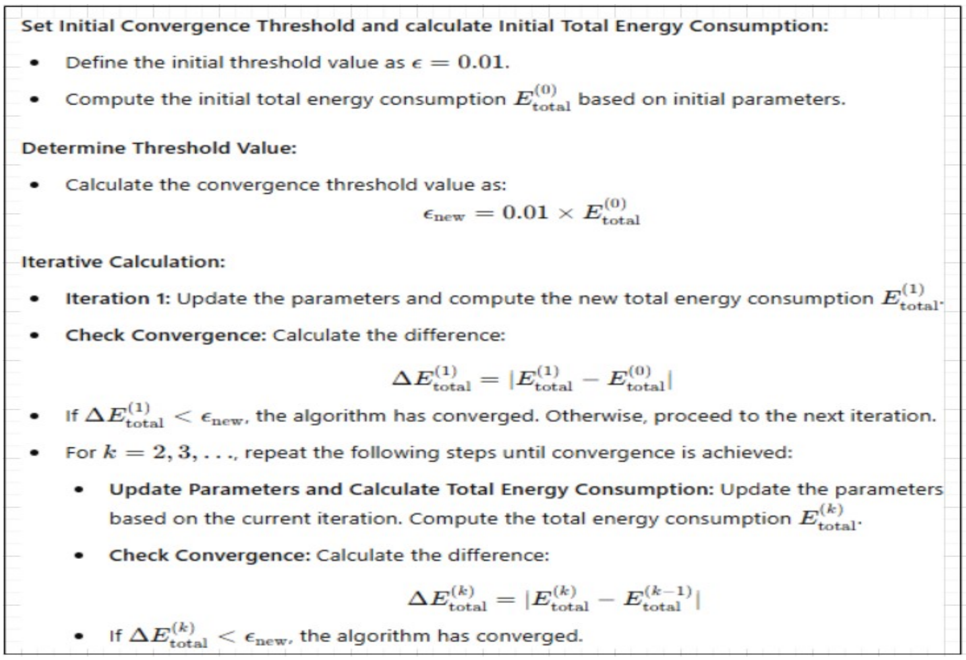
Algorithmic customisations: By focusing on the criticality of improving throughput through latency, channel quality with offloading computational tasks between SCD's is a major focus of the priority selection offloading algorithm as described in (Anajemba et al.; 2020). With CloudSim, SCD's and offloaded tasks was simulated as VM and cloudlet behavior.

Table 5: Discrete variables for an increased complexity

Performance Metric	Description	Justification
latencyPenalty	Task execution was greatly prioritised based over response time constraints after computing the local execution energy and transmission energy.	Can cope with the multi-access characteristics of heterogeneous networks by inducing minimised energy consumption as described in Zhang et al. (2016).

4.2.4 Consistent Convergence Criterion:

For each algorithm, a consistent convergence criterion ensured a fair comparison of effectiveness by assessing the performance of each algorithm with the environmental conditions. This single convergence threshold value ϵ denoted as 0.01 was set as 1% of the initial total energy consumption to refine the sensitivity of each algorithmic program. By comparing tests with varied parameter values in the re-run algorithmic programs through iterative testing, the defined threshold ensured the desired energy consumption stabilises over a certain threshold through convergence.
Steps for Evaluation:



Set Initial Convergence Threshold and calculate Initial Total Energy Consumption:

- Define the initial threshold value as $\epsilon = 0.01$.
- Compute the initial total energy consumption $E^{(0)}_{\text{total}}$ based on initial parameters.

Determine Threshold Value:

- Calculate the convergence threshold value as:

$$\epsilon_{\text{new}} = 0.01 \times E^{(0)}_{\text{total}}$$

Iterative Calculation:

- **Iteration 1:** Update the parameters and compute the new total energy consumption $E^{(1)}_{\text{total}}$.
- **Check Convergence:** Calculate the difference:

$$\Delta E^{(1)}_{\text{total}} = |E^{(1)}_{\text{total}} - E^{(0)}_{\text{total}}|$$

- If $\Delta E^{(1)}_{\text{total}} < \epsilon_{\text{new}}$, the algorithm has converged. Otherwise, proceed to the next iteration.
- For $k = 2, 3, \ldots$, repeat the following steps until convergence is achieved:
  - **Update Parameters and Calculate Total Energy Consumption:** Update the parameters based on the current iteration. Compute the total energy consumption $E^{(k)}_{\text{total}}$.
  - **Check Convergence:** Calculate the difference:

$$\Delta E^{(k)}_{\text{total}} = |E^{(k)}_{\text{total}} - E^{(k-1)}_{\text{total}}|$$

  - If $\Delta E^{(k)}_{\text{total}} < \epsilon_{\text{new}}$, the algorithm has converged.

Figure 5: Step-by-step method for ensuring the optimised solution within precision limits.

5 Implementation

The implementation of the research techniques on evaluating performance and assessing scalability against all algorithmic programs were primarily carried out to demonstrate the efficacy in energy efficiency through the network slicing ring fencing equation. Through an environmental setup, CloudSim is a vital part of this experimental verification that gauged all sources specific to energy consumption and processed execution time. In addition to fulfilling all the prerequisite criteria, a CloudSim IDE setup benchmarked speed in convergence with refined execution time of all energy sources specific to the algorithmic programs in scalable, architectural setups. The incorporated algorithmic programs and preprocessed dataset programs were primarily written in Java with Java17 and CloudSim SDK as managed dependencies for real-world AWS setup and testing, which shall be discussed later in this section.

5.1 Tools and Technologies

The following tools and technologies were utilised in evaluating and benchmarking the iterative heuristic energy-aware non-convex optimisation equation against the respective algorithmic techniques:

- **Eclipse IDE 2024-03 (4.31.0)**
- **CloudSim Toolkit 3.0.13**
- **Python 3.12.2**
- **Java17**
- **NetworkSlicing_VPC**
- **NetworkSlice_Subnet**
- **RingFence_SecurityGroup**
- **NetworkResource_VM**
- **AggregatedLambdaNetworkSlicing**
- **AggregatedMetricsMonitor**
- **AggregatedMetricsLogs**

CloudSim Configurational Settings: This research paper utilised CloudSim 3.0.3 toolkit to benchmark each algorithmic program against each other for the desired final energy consumption set by the desired convergent limit ϵ. Various procedures like energy-aware latency-constraint scheduling, NOMA non-convex clustering for network slicing, computational task offloading can be efficiently evaluated through algorithmic performance and speed in a simulating environment. Specifications for the tests were conducted on a 11th Gen Intel(R) Core(TM) i5 processor and Windows 11 Pro platform.

Table 6: Configurational settings for CloudSim 3.0.3 IDE

Parameter	**Property**	**Value**
Datacenters	Size	1000
	RAM (MB)	64
	MIPS	5
	BW (mbps)	100
	pesNumber	1
Virtual Machines	MIPS	5
	RAM (MB)	256
	BW (mbps)	1000
	Storage (MB)	100000
Cloudlets	File Size	100
	Output Size	100

5.2 Algorithmic Implementation with CloudSim

5.2.1 Iterative Heuristic Energy-Aware Non-Convex Algorithm

The defined objective specific energy consumption equation of this research paper described as an algorithm satisfied the degree of convergence most efficiently against the compared algorithmic techniques in the step-by-step implementation below.

Algorithm 1 The Iterative Heuristic Energy-Aware Non-Convex Pseudocode

```
//Step 1: Initialize the Parameters
Set simulation environment with parameters for energy consumption with simulation start and end times.

//Step 2: Create Datacenters and Initialize Datacenter Broker
for each network slice in network slices do
        Define hosts with Processing Elements (PEs), RAM, Bandwidth, storage with
        hosts. Configure DatacenterCharacteristics and Datacenter.
end for
for each ring fence in a network slice do
        Define VMs with MIPS, RAM, Bandwidth, Image Size, Number of PEs, VMM,
        and add VMMs to the broker's list.
end for

//Step 3: Load Cloudlets from Preprocessed Dataset and Submit Corresponding
VMs and Cloudlets
Read data from preprocessed CSV files fastStorage and rnd.
for each data row do
        Extract CPU usage, cores, and other metrics. Create Cloudlets with Length,
        PEs, Input and Output sizes and add the Cloudlets to the broker's list.
end for
Submit VM list and Cloudlet list to the broker.

//Step 4: Simulate Inter-VM Communication
for each predefined VM-to-VM communication pair do
        Simulate bandwidth usage, compute and aggregate communication energy
        consumption.
end for
for each isolated VM do
        Add idle energy consumption to total energy.
end for

//Step 5: Start CloudSim Simulation and Collect the Simulation Results
Start the simulation to execute all Cloudlets. Stop the simulation once all Cloudlets are processed and retrieve
processed Cloudlets from the broker with the displayed results (execution time, start time, finish time).

//Step 6: Calculate Energy Consumption
Compute and cooling energy: $pComputeCC + pCoolCC/zetaCC$.
Static power energy: $pStaticRF/zetaRF$.
Processing power energy: $pProcCC \times CPUTime$.
Communication energy: $commPower \times commBandwidth$.
Latency and DVS energy: $\tau \cdot \text{cpuSpeed} \cdot (\text{dynamicScalingFactor})^3 \cdot \text{CPUTime}$.
Add cooling overhead: $CoolingPower = TotalEnergy \times CoolingFactor$.

//Step 7: Log and Write Results
Print energy consumption results: Total energy, cooling power, and total energy with cooling and the detailed
results to an output file with the terminated CloudSim environment.
```

5.2.2 Heuristic AUGMENT Non-Convex Algorithm

The following non-convex algorithm was proposed to incorporate network-slice user aware grouping for NOMA with resource allocation in (Hossain and Ansari; 2021). With the components incorporated through the network slicing ring fencing architecture, the energy consumption was simulated to model the iterative heuristic energy-aware non-convex algorithm described earlier.

Algorithm 2 The Heuristic AUGMENT Non-Convex Algorithmic Pseudocode

```
// Step 1: Initialize Simulation Environment
Set up CloudSim environment and configure start and end times.

// Step 2: Create Datacenters and Initialize Broker
for each network slice do
    Define hosts with PEs, RAM, Bandwidth, and Storage. Configure Datacenters and
    DatacenterCharacteristics with Datacenter through VM and Cloudlet Assignments..
end for
for each ring fence in all slices do
    Define VMs with MIPS, RAM, Bandwidth, Image Size, and PEs. Add VMs to broker's list.
end for

// Step 3: Load Cloudlets from Preprocessed Dataset and Submit Corresponding VMs
and Cloudlets
Read task data from preprocessed CSV files fastStorage and rnd.
for each data row in dataset do
    Extract CPU usage, cores, and other metrics. Create Cloudlets with Length, PEs, Input, Output
sizes and add Cloudlets to the broker's list.
end for

// Step 4: Simulate Inter-VM Communication
for each predefined VM-to-VM communication pair do
    dynamicBandwidth = bandwidth * (1 + randomFactor).
    bandwidthPenalty = penaltyFactor, if dynamicBandwidth <threshold.
                     = 1.0, otherwise.
    communicationEnergyCost = bandwidth * energyCostMultiplier.
    transmissionPower: totalEnergy += bandwidthPenalty * transmissionPowerBase.
    totalEnergy += communicationEnergyCost.
    totalEnergy += bandwidthPenalty * communicationEnergyCost * adjustmentFactor.
    totalEnergy += transmissionPowerBase * adjustmentFactor.
end for
for each isolated VM do
    totalEnergy += idleEnergyConsumption.
end for

// Step 5: Execute Simulation and Collect Results
Start the simulation to execute all Cloudlets. Retrieve processed cloudlets from the broker with the
displayed results at the end of the simulation (e.g., execution time, start time, finish time).

// Step 6: Calculate Total Energy Consumption
totalEnergy += processingPower * CPUTime.
CoolingPower = totalEnergy * CoolingFactor.
Add cooling overhead to total energy.

// Step 7: Log and Write Results
Print energy consumption results: Total energy, cooling power, and total energy with cooling with the
detailed results to an output file with the terminated CloudSim environment.
```

5.2.3 Priority Selection Offloading Algorithm

To tie in with the network slicing ring fencing architecture, offloading priority based on resource availability closely models energy consumption to be benchmarked for this algorithmic program.

Algorithm 3 The Priority Selection Offloading Algorithmic Pseudocode

```
// Step 1: Initialize the Simulation Environment
Initialize the CloudSim environment with one user. Configure simulation start and end times.

// Step 2: Create Datacenters (Representing Network Slices)
for each datacenter do
    Define hosts with PEs, RAM, Bandwidth, Storage, and configure DatacenterCharacteristics.
end for

// Step 3: Create Datacenter Broker with created Virtual Machines (VMs)
Instantiate a broker to manage VM and Cloudlet assignments.
for each VM do
    Define VMs with MIPS, RAM, Bandwidth, Image Size, and Number of PEs and add
    VMs to the broker's list.
end for

// Step 4: Load Cloudlets from Preprocessed Dataset
Read task data from preprocessed CSV files.
for each row in dataset do
    Extract task metrics like CPU usage, cores, input size and output size.
    Create Cloudlets and add them to the broker's list.
end for

// Step 5: Simulate Offloading and Communication
for each VM in the list
    for each Cloudlet do
        localTime = cloudletLength / vmMIPS.
        localEnergy = localTime * energyCostFactor.
        transmissionEnergy = transmissionPower * cloudletLength / systemChannelGain.
        latencyPenalty = latencyFactor * (localExecutionTime - transmissionEnergy).
        localEnergyConsumption = localEnergyConsumption + latencyPenalty.
        Local Execution and Offloading Energies:
        if localEnergy <transmissionEnergy then
                Add Offloading Energy to Total Energy.
                totalEnergy += transmissionEnergy. Submit the Cloudlet
                to the broker for offloading.
        else
                Add Local Execution Energy to Total Energy:
                totalEnergy += localEnergyConsumption
        end if
    end for
end for

//Step 6: Simulate VM-to-VM communication based on network slicing ring fencing
architecture
for each VM-to-VM connection do
    communicationEnergy += bandwidth * energyMultiplier.
    Add communication energy to Total Energy.
end for
for each isolated VM do
    Add constant idle energy consumption to Total Energy.
end for

//Step 7: Start CloudSim Simulation and calculate Energy Consumption with Results.
Start and stop the simulation to process and execute all Cloudlets.
coolingPower = totalEnergy * coolingFactor.
totalEnergyWithCooling = totalEnergy + coolingPower.
Print Energy consumption results: Total Energy, Cooling Power and Total Energy with Cooling for the
desired results to an output file with the terminated CloudSim environment.
```

5.3 CloudSim Implementation and AWS Environmental Setup

To achieve the experimental objectives on benchmarking each algorithmic program on the basis of assessed scalability and evaluated performance against a consistent convergence criterion, CloudSim and AWS real-world testing were incorporated as simulation environments. Through the creation of simulation environments, cloud data centers can be setup to understand all energy estimates where the iterative heuristic energy-aware non-convex algorithmic program proved to be most efficient for all experiments. The architectural diagram representing the implementation of the aggregated uneven network slicing ring fencing architecture in CloudSim and AWS were provided in the configuration manual respectively.

6 Evaluation

The following section examined three major experimental results conducted on the basis of optimising the network slicing ring fencing ratio through the iterative heuristic energy-aware non-convex equation against a convergence threshold. The first experiment evaluated the results obtained by preprocessing computational network resource metrics from fast storage network(fastStorage) and slower network attached storage(Rnd) devices into their respective preprocessed files. The next section dealt with the limitations of results derived by the initial incorporated algorithmic programs. The third section critically evaluated the results obtained by the aggregated scalable program counterparts in CloudSim and AWS real-world testing. The ideal iterative heuristic energy-aware non-convex equation to be adapted as a model based on the results of the experiments shall be provided in the last subsection.

6.1 Experiment 1: Preprocessing 'GWA-T-12' Bitbrains

By the consideration of URLLC according to (Farreras et al.; 2024), preprocessing datasets of computational network resource data with an allocation of nodes is a good starting point for this section.

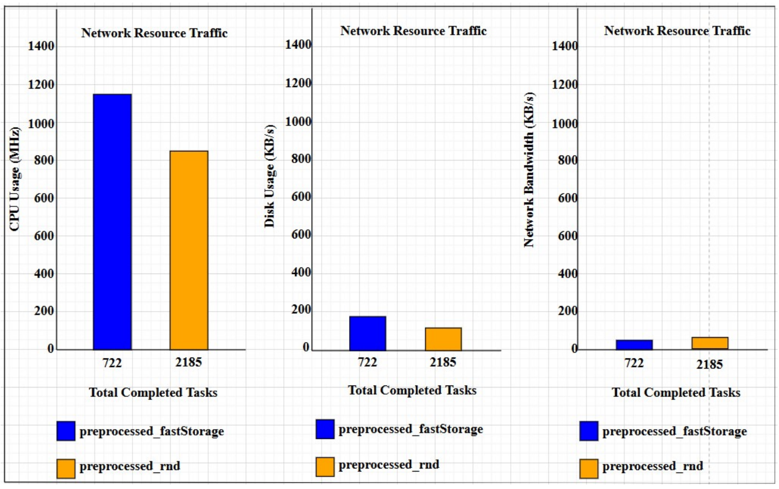


Figure 6: Computational Network Resource Data for the Preprocessed Dataset 'GWA-T-12 Bitbrains'.

The following experiment was conducted on CloudSim 3.0.3 toolkit with Python 3.12.2. The network resource metrics to be analysed in this section are CPU_Usage, Disk_IO, Net_Band and Memory_Usage that correspond to CPU Usage, Disk Usage, Network Bandwidth and Memory Usage in Figure 6. preprocessed_fastStorage.csv resulted in a CPU and memory usage of around 1172 MHz and 559 MB respectively while preprocessed_rnd.csv resulted in a CPU and memory usage of 815 MHz and 508 MB respectively. By reporting a 43.87% decrease in CPU execution with an efficiency gain of memory utilisation by 9.12% in figure 6 aggregated workloads with distributed computational network resources can adopt preprocessed_rnd.csv for lower computational task execution time. Despite possessing 1463 less tasks, preprocessed_fastStorage.csv cannot be ruled out for preprocessing as it is more efficiently suited for heavier workloads distributed in heterogeneous architectures.

With an improved network bandwidth of 25.93% for preprocessed_rnd.csv as compared to preprocessed_fastStorage.csv, the network bandwidth should increase exponentially with a larger number of computational network nodes for those slices, according to (Farreras et al.; 2024). The iterative heuristic energy-aware non-convex equation can idealise scenarios with network resources possessing greater computational offloading and task distribution per network resource. This largely aligns with the preliminary goal in this paper to preprocess datasets on scalable, distributed architectural infrastructures such as the aggregated network slicing ring fencing architecture in Experiment 3.

6.2 Experiment 2: Evaluating Baseline Simulation Performance

The following two experiments were also conducted on CloudSim 3.0.3 toolkit with Java 17. Normalising speed through a convergence threshold has demonstrated novel approaches such as Zhang et al. (2023) which describes a deep layer reinforcement NOMA learning algorithm in 5G network slicing that ensures optimisation in the guise of task offloading. A convergence threshold can help idealise the approach of iterative calculation to convergence in energy consumption for the respective algorithms which is the goal in this section.

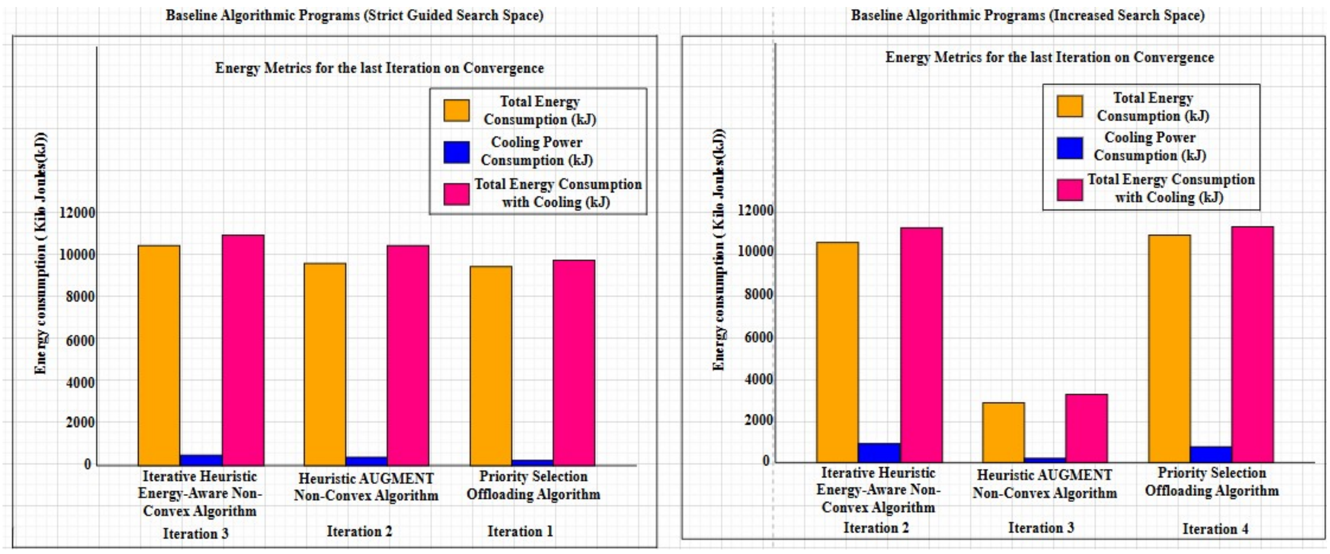


Figure 7: Energy Metrics for the Baseline Algorithmic Programs with Strict Guided Search Space and Increased Search Space'.

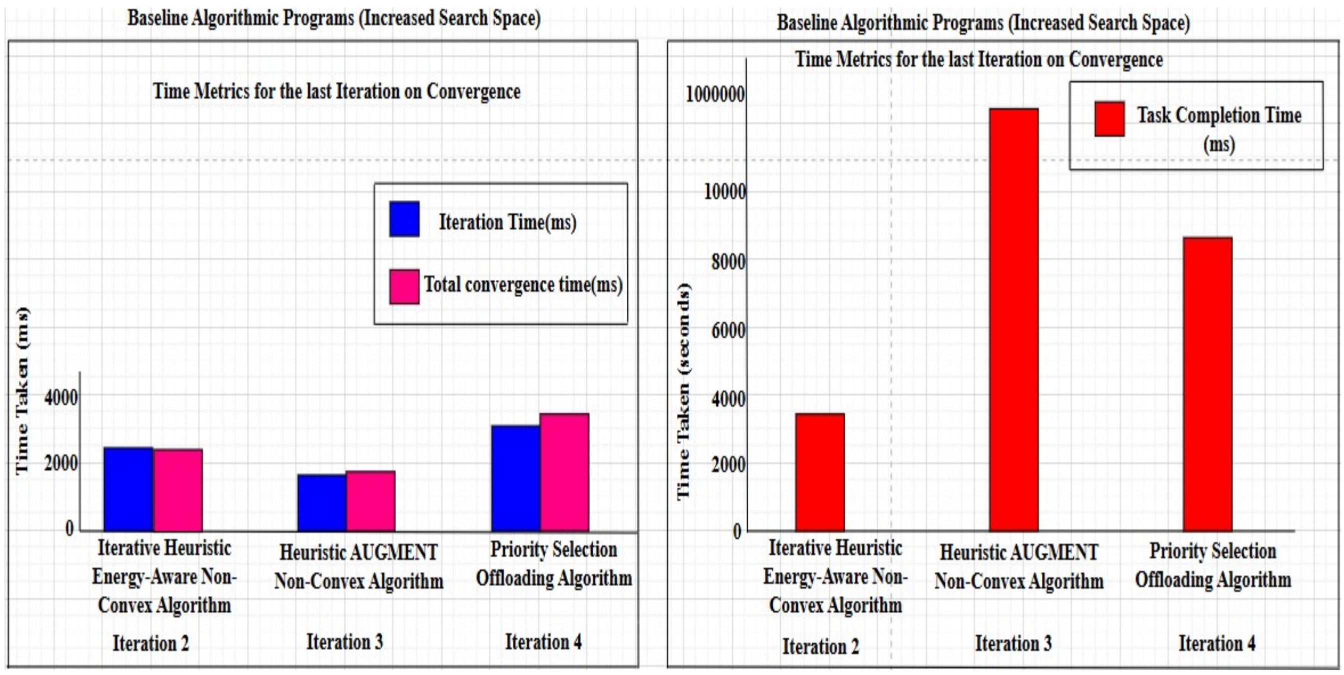


Figure 8: Time Metrics for the Baseline Algorithmic Programs with Strict Guided Search Space and Increased Search Space'.

The heuristic AUGMENT non-convex algorithm and priority selection offloading algorithm reported energy metrics of 10355.299 kJ and 9909.403 kJ respectively when iterating for convergence in energy consumption by figure 7. Additionally, the iterative heuristic energy-aware non-convex algorithmic approach took three iterations while the other algorithmic approaches took two or less in figure 7. By ensuring the same system requirements, the new baseline heuristic AUGMENT non-convex algorithm and priority selection offloading algorithm reported a total energy consumption at around 10680.219 kJ, 2464.300 kJ and 10738.070 kJ respectively for the last iteration in figure 7 with a larger search space such as dynamic bandwidth penalty and delay specific penalties explored in (Zhang, Kosta and Mogensen; 2023). This indicated an efficiency loss and gain of 76.20% and 8.36% for the heuristic AUGMENT non-convex algorithmic program and priority selection offloading algorithm respectively when compared against the restricted guided algorithmic equivalent. Akin to total energy consumption, similar disparities for the heuristic AUGMENT non-convex algorithm and priority selection offloading algorithm were reported at 990102.83 seconds and 8888.34 seconds for task completion time in figure 8. Large disparities in total energy consumption and task completion time for the heuristic AUGMENT non-convex algorithmic program with an increase in iterations for the priority selection offloading algorithm points to initial inadequacies to idealise the desired ratio.

While the iterative heuristic energy-aware non-convex program reported an efficiency gain of 3.38% from 11054.202 kJ to 10680.219 kJ in the same manner, the consistency in sensitivity for total energy consumption mostly fell within acceptable limits. Thus, precision tied to an optimised solution can be linked to the iterative heuristic energy-aware non-convex algorithm which achieved speed in convergence through consistency for network architects to design aggregated cloud models with varied system requirements.

6.3 Experiment 3: Aggregating Scalability through Real-World Applicability

The parameters of the baseline incorporated programs were updated to reflect the aggregation of the uneven network slicing ring fencing architecture on CloudSim 3.0.3 and us-east-1(North Virginia Region) on AWS. Three Lambda test events named AggregatedIterLambdaTest, AggregatedHeurLambdaTest and AggregatedPriLambdaTest were run on 'AggregatedLambdaNetworkSlicing' Lambda respectively towards the iterative heuristic energy-aware non-convex algorithmic program, heuristic AUGMENT non-convex algorithmic program and priority selection offloading algorithm. Custom dashboards for the energy metrics were created at the end of each simulation.

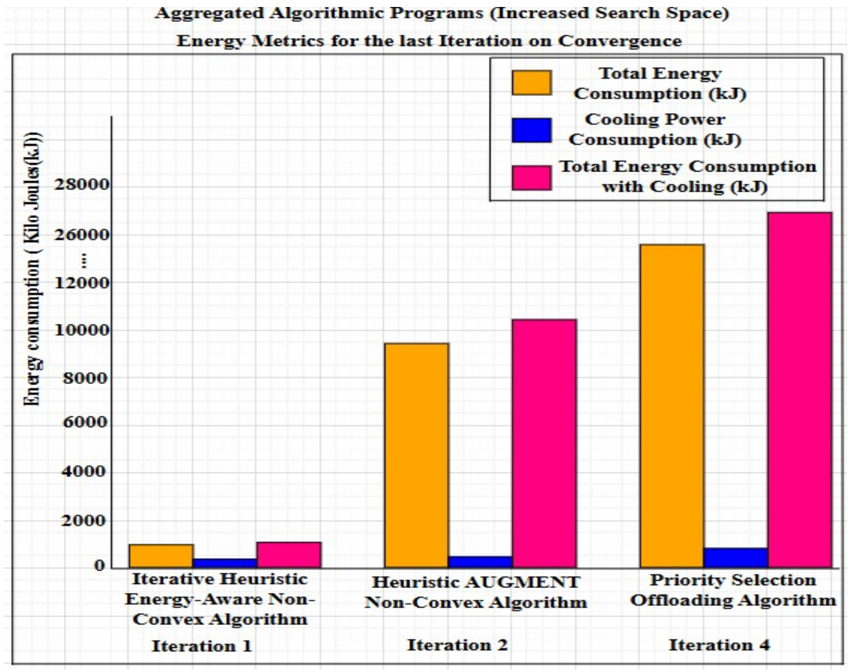


Figure 9: Energy Metrics for the Baseline Algorithmic Programs with Strict Guided Search Space and Increased Search Space'.

The iterative heuristic energy-aware non-convex algorithmic program, heuristic AUGMENT non-convex algorithmic program and priority selection offloading algorithmic program all reported total energy consumption of 912.958 kJ, 10168.261 kJ and 26845.584 kJ in figure 9. The priority selection offloading algorithm reported a 150% increase in total energy consumption in figure 9 while the heuristic AUGMENT non-convex algorithm reported an efficiency gain of 1.81% in figure 9. However, the reported convergence in two iterations as compared to three iterations makes it less favourable as compared to the iterative heuristic energy-aware non-convex algorithmic equation.
The iterative heuristic energy-aware non-convex algorithmic program fulfills the level of consistency with an efficiency gain of 91.45% by total energy consumption that proves advantageous as compared to the heuristic AUGMENT non-convex algorithm and priority selection offloading algorithm described in (Hossain and Ansari; 2021) and (Anajemba et al.; 2020). With an inherently larger search space such as energy-aware scheduling and latency constraints, the iterative heuristic energy-aware non-convex equation can closely model the network slicing ring fencing ratio for heterogeneous uneven network slicing ring fencing architectures.

6.4 Discussion

From the results of the experiments, it is clear that the iterative heuristic energy-aware non-convex equation is more efficiently suited for heterogeneous architectures with varied user and system requirements depending on the need of the client. By preprocessing dissimilar datasets that evaluates performance and assesses scalability through simulation environments, the optimisation equation demonstrated consistency in convergence by energy consumption. Discretisation for the heuristic AUGMENT non-convex algorithm and priority selection offloading algorithm resulted in limitations with disparate results. With an increased search space defined by an optimised solution, the iterative heuristic energy-aware non-convex optimisation equation ensures it's suitability for aggregated network slicing ring fencing architectures as well as smaller uneven architectures with resource-heavy workloads.

7 Conclusion and Future Work

The main purpose of this research was to benchmark the iterative heuristic energy-aware non-convex optimisation equation that refines convergence with energy consumption with computational execution time to idealise the network slicing ring fencing ratio. By evaluating baseline performance and assessing scalability against a consistent convergence criterion, the iterative heuristic energy-aware non-convex equation was validated as a stable, optimised solution for heterogeneous architectures. Incorporating discretisation such as latency constraints and DVS through energy-aware scheduling for the optimisation equation proved advantageous in contrasting architectural systems. The same experimental objectives for the heuristic AUGMENT non-convex algorithm and priority selection offloading algorithm resulted in largely disparate performance metrics that might violate SLA constraints. Designing dense uneven network slicing ring fencing architectures can exponentially raise costs if proper cost management of allocated resources is not ensured by the client. Adopting AWS Auto-Scaling for non-critical aggregated workloads is one way of mitigating high costs for carbon footprint reduced data centers. Meaningful recommendations for the future include adopting least privilege access with IAM policies and addressing security gaps through vulnerability assessments.

Acknowledgment: This research paper utilizes the GWA-T-12 Bitbrains dataset which contains the performance metrics of 1,750 VMs from a distributed datacenter managed from Bitbrains, which specializes in managed hosting enterprises. The dataset is available below at: https://web.archive.org/web/20240921001601/http://gwa.ewi.tudelft.nl/datasets/gwa-t-12-bitbrains
Please refer to the following CCGrid 2015 paper for further details:
Research Work -https://web.archive.org/web/20240921001601/http://gwa.ewi.tudelft.nl/fileadmin/pds/trace-archives/grid-workloads-archive/reports/gwa-t-12/workloadCCGrid2015.pdf
Additionally, please refer to the following acknowledgements as well:
"The work is supported by the National Basic Research Program of China under grant No. 2011CB302603 and No. 2014CB340303, by the Veni@Large (11881), NWO KiEM KiESA and NWO COMMiT COMMiSSiONER."
Grid Workloads Archive -
https://web.archive.org/web/20240909201754/http://gwa.ewi.tudelft.nl/

References

Abuajwa, O. and Mitani, S. (2024). Dynamic resource allocation for energy-efficient downlink NOMA systems in 5G networks, *Heliyon* **10**(9).

Anajemba, J. et al. (2020). Optimal cooperative offloading scheme for energy efficient multi-access edge computation, *IEEE Access* **8**: 53931–53941.

Auer, G. et al. (2011). How much energy is needed to run a wireless network?, *IEEE Wireless Communications* **18**(5): 40–49.

Buzzi, S. et al. (2016). A survey of energy-efficient techniques for 5g networks and challenges ahead, *IEEE Journal on Selected Areas in Communications* **34**(4): 697–709.

Calheiros, R. N., Ranjan, R., Beloglazov, A., Rose, C. A. F. D. and Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, *Software: Practice and Experience* **41**(1): 23–50.
URL: *https://doi.org/10.1002/spe.995*

Chien, H.-T. et al. (2020). End-to-end slicing with optimized communication and computing resource allocation in multi-tenant 5g systems, *IEEE Transactions on Vehicular Technology* **69**(2): 2079–2091.

Choudhary, R. and Perinpanayagam, S. (2022). Applications of Virtual Machine Using Multi-Objective Optimization Scheduling Algorithm for Improving CPU Utilization and Energy Efficiency in Cloud Computing, *Energies* **15**(23).

Farreras, M. et al. (2024). Generation of a network slicing dataset: The foundations for ai-based b5g resource management, *Data in Brief* **55**: 110738.

Fehske, A. et al. (2011). The global footprint of mobile communications: The ecological and economic perspective, *IEEE Communications Magazine* **49**(8): 55–62.

Foukas, X. et al. (2017). Network slicing in 5g: Survey and challenges, *IEEE Communications Magazine* **55**(5): 94–100.

Hossain, M. A. and Ansari, N. (2021). Energy aware latency minimization for network slicing enabled edge computing, *IEEE Transactions on Green Communications and Networking* **5**(4): 2150–2159.

Huang, K., Wang, Z., Zhang, H., Fan, Z., Wan, X. and Xu, Y. (2019). Energy efficient resource allocation algorithm in multi-carrier noma systems, *2019 IEEE 20th International Conference on High Performance Switching and Routing (HPSR)*, pp. 1–5.

Kwak, J. et al. (2017). Dynamic network slicing and resource allocation for heterogeneous wireless services, *2017 IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, IEEE, pp. 1–5.

Lorincz, J., Kukuruzović, A. and Blažević, Z. (2024). A comprehensive overview of network slicing for improving the energy efficiency of fifth-generation networks, *Sensors* **24**(10): 3242.

Mao, Y., Zhang, J. and Letaief, K. (2016). Dynamic computation offloading for mobile-edge computing with energy harvesting devices, *IEEE Journal on Selected Areas in Communications* **34**(12): 3590–3605.

Masoudi, M. et al. (2022). Energy-optimal end-to-end network slicing in cloud-based architecture, *IEEE Open Journal of the Communications Society* **3**: 574–592.

Mesodiakaki, A. et al. (2021). 5G-COMPLETE: End-to-end Resource Allocation in Highly Heterogeneous Beyond 5G Networks, *2021 IEEE 4th 5G World Forum (5GWF)*, IEEE, pp. 412–417.

Saxena, S. and Sivalingam, K. M. (2022). Slice admission control using overbooking for enhancing provider revenue in 5g networks, *NOMS 2022-2022 IEEE/IFIP Network Operations and Management Symposium*, pp. 1–7.

Sriram, A. et al. (2019). Joint functional splitting and content placement for green hybrid cran, *2019 IEEE 30th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, IEEE, pp. 1–7.

Tao, X. et al. (2017). Performance guaranteed computation offloading for mobile-edge cloud computing, *IEEE Wireless Communications Letters* **6**(6): 774–777. Accessed: 1 August 2024.
URL: *https://research.ebsco.com/linkprocessor/plink?id=ab850cc6-e715-3ac5-ab6e-4b657e94f6d3*

Xiang, H., Zhou, W., Daneshmand, M. and Peng, M. (2017). Network slicing in fog radio access networks: Issues and challenges, *IEEE Communications Magazine* **55**(12): 110–116.

Zhang, K. et al. (2016). Energy-efficient offloading for mobile edge computing in 5g heterogeneous networks, *IEEE Access* **4**: 5896–5907.

Zhang, W., Kosta, S. and Mogensen, P. (2023). A scheduling method for tasks and services in iiot multi-cloud environments, *2023 19th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)*, IEEE, pp. 293–300.

Zhang, Y. et al. (2023). Joint resource allocation and task offloading for hybrid noma-assisted mec network with network slicing, *2023 25th International Conference on Advanced Communication Technology (ICACT)*, pp. 164–170.