Internet of Things 30 (2025) 101515

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

# Energy efficient resource management for real-time IoT applications

Rolden John Fereira[a,*], Chathurika Ranaweera[a], Kevin Lee[a], Jean-Guy Schneider[b]

[a] *Deakin University, School of Information Technology, Geelong, 3220, VIC, Melbourne, Australia*
[b] *Monash University, Faculty of Information Technology, Clayton, 3800, VIC, Melbourne, Australia*

ARTICLE INFO

*Keywords:*
IoT
Edge computing
Fog computing
Convergence
Resource allocation
Node selection

ABSTRACT

The Internet of Things (IoT) has a large and rapidly expanding number of deployed devices, which leads to a significant global energy consumption footprint. Diverse IoT use cases, including smart cities, smart grids, Industry 5.0, eHealth, and autonomous vehicles, are contributing to this increase in energy consumption. Optimising energy utilisation is crucial to sustaining the exponential growth of IoT applications, which demand stringent delays and latencies measured in milliseconds and microseconds. There are additional complexities with the emergence of edge, fog, and cloud computing and the need to manage the energy consumption at all the layers. In this paper, mechanisms that can be used to minimise energy consumption within an edge–fog–cloud IoT architecture for real-time IoT applications are being proposed. We investigate mechanisms for optimal node selection, primarily focusing on minimising energy usage while adhering to the Quality of Service (QoS) requirements of various IoT requests. The mechanisms include genetic, modified genetic, and delay-aware algorithms tailored explicitly for real-time IoT applications. We evaluated the proposed mechanisms using a simulation of diverse network scenarios. The results presented in the paper provide insight into balancing processing time and energy efficiency, which are critical considerations in sustainably developing IoT applications in an edge–fog–cloud IoT architecture.

## 1. Introduction

In our increasingly connected world, the Internet of Things (IoT) is becoming an essential aspect of our daily lives, with various objects around us evolving into diverse IoT applications. By 2030, it is expected that the total number of IoT devices will surge to 30 billion [1]. To cater to this vast number of IoT devices, which are integral to diverse use cases such as eHealth, autonomous vehicles, smart grids, smart cities, Industry 5.0, and virtual reality (VR) / artificial reality (AR), a sophisticated IoT infrastructure is required. The evolution of communication and computation technologies is pivotal in establishing an advanced electronic infrastructure to accommodate this exponential growth. Cutting-edge technologies such as fifth generation (5G)/ sixth generation (6G), edge computing, fog computing, mobile computing, software-defined networks, and artificial intelligence are crucial in forming a versatile IoT architecture [2,3].

Among these technologies, edge computing has emerged as a critical component, addressing the increasing complexity and real-time demands of IoT applications. Time-sensitive standards are pivotal in enabling low-latency and high-reliability communication for real-time systems [4–7]. For instance, autonomous vehicles require ultra-low latency to process and respond to sensor data instantaneously, a requirement that traditional cloud-based models struggle to meet efficiently.

* Corresponding author.
*E-mail address:* rjfereira@deakin.edu.au (R.J. Fereira).

https://doi.org/10.1016/j.iot.2025.101515
Received 30 September 2024; Received in revised form 15 January 2025; Accepted 16 January 2025
Available online 31 January 2025

Developing a flexible, scalable, energy and cost-efficient IoT infrastructure supporting diverse real-time IoT use cases with varied QoS requirements presents a significant challenge [8–11].These challenges are amplified by the unique characteristics and limitations of emerging technologies, alongside the stringent QoS requirements of real-time applications. For example, time-critical use cases such as autonomous driving, eHealth, and smart grids demand sub-millisecond decision-making capabilities. Achieving these requirements while minimising operational costs and energy consumption remains a critical concern [12,13].

The motivation behind this research lies in addressing the growing energy demands, associated costs, and environmental impact, which drive the need for more scalable and energy-efficient IoT infrastructures that use diverse advanced computation and communication technologies [14,15]. To fully leverage the potential of these emerging technologies, a convergence of communication, computation, and caching mechanisms is essential [3,16]. This approach should be able to collectively support all the IoT use cases, moving away from a use case-centric architecture and methodology. A more integrated approach, leveraging a load-balanced edge–fog–cloud IoT architecture [2,17], is necessary to efficiently manage the diverse and real-time IoT requests while minimising energy consumption and fulfilling stringent QoS requirements.

Previous work, such as the Integer Linear Programming (ILP)-based framework [18,19], has provided valuable insights into energy-efficient node selection with a custom number of servers deployed at each node and each layer. However, the time required for solutions in real-time applications remains a limitation. Though the framework provides a good foundation to identify how different scenarios impact the most energy-optimal solution, there is a limitation of the applicability of the framework in the real-time IoT application as the time required for a solution can vary significantly. The proposed meta-heuristic mechanisms overcome this limitation by enabling faster convergence, scalability and real-time applicability while maintaining energy efficiency. Therefore, this paper investigates time-efficient heuristic approaches, including genetic algorithms, modified genetic algorithms, and delay-aware mechanisms, for energy-efficient node selection in IoT networks.

The genetic algorithm is particularly suited for solving complex optimisation problems, making it ideal for resource allocation in multi-layered IoT architectures. To further enhance its efficiency, the algorithm is modified with problem-specific adjustments for faster convergence and improved handling of IoT challenges. Further, a delay-aware mechanism is proposed to address the critical need for low-latency processing in real-time IoT applications. Together, these mechanisms ensure timely, energy-efficient processing for use cases such as autonomous vehicles and eHealth. By prioritising energy efficiency, the proposed solutions not only reduce operational costs but also align with sustainable IoT practises by minimising the carbon footprint of large-scale deployments.

This paper evaluates the proposed mechanisms in a distributed edge–fog–cloud IoT architecture that supports diverse, real-time IoT use cases. The mechanisms aim to minimise energy consumption while adhering to QoS requirements and processing incoming IoT requests with significantly lower solution times. Evaluations are conducted using a wide array of incoming IoT requests from use cases, including eHealth, smart grids, autonomous vehicles, and other diverse network configurations. The results demonstrate the practicality and effectiveness of the proposed mechanisms, with comprehensive comparisons against ILP-based frameworks.

The key contributions of this paper are (1) proposed node selection mechanisms to minimise the energy in an edge–fog–cloud IoT architecture based on genetic algorithm, modified genetic and delay-aware algorithms, (2) evaluation of the suitability of proposed mechanisms under diverse network scenarios and IoT applications with varying requirements (3) comparative analysis of ILP-based framework, genetic algorithm, modified genetic algorithm and delay-aware based mechanisms in terms of its performance, time, and complexity to identify the suitability of these mechanisms for energy-efficient real-time processing of IoT applications.

The remainder of this paper is structured as follows. Section 2 presents current trends, QoS standards for diverse IoT applications and recent research on the heuristic and meta-heuristic approaches used to optimise resource allocation in IoT. The research challenges in the node selection of a heterogeneous IoT architecture while achieving energy efficiency are further discussed in Section 2. In Section 3, a comprehensive description of proposed node selection mechanisms is presented. Section 4 presents the evaluation of the proposed mechanisms and comparative analyses of the proposed mechanisms. The concluding remarks of this paper are presented in Section 5.

## 2. Related work on heuristic approaches for resource allocation in IoT

IoT architectures primarily utilise either a single computation layer or a combination of the three layers (edge, fog, and cloud) to deliver on-demand, priority, shared, and distributed services for various IoT use cases. The IoT use cases are spread across various sectors, such as health, industry automation, manufacturing, surveillance, smart cities, and defence. Each of these use cases mostly focuses on using a single layer as the ideal computation layer based on a few network QoS and application requirements [20]. Each layer plays a crucial role in enabling the IoT use cases. The cloud layer provides distributed computing via scalable, virtualised, pooled, and flexible resources [14]. The fog layer facilitates adaptable, scalable, and virtualised management of computation, network, and repository services, much closer to the user than the cloud layer while reducing latency and preserving bandwidth [21]. The edge layer emphasises much lower latency and is much closer to the user compared to the fog layer [14,21].

Each of the diverse IoT applications has many different use cases. For example, smart grids have use cases like teleprotection, synchrophasor monitoring, SCADA, VoIP, and smart metering. The delay between devices for these use cases ranges from 10 ms to 1ms [7,22–24], and the required bandwidth is between 5–75 Mbps. Data transmission rates range from 56 kbps to 1 Mbps, while reliability ranges from 99% to 99.99%. These metrics indicate the critical nature of low-latency and high-reliability communications for maintaining grid stability and safety.

For autonomous vehicles, use cases include vehicle-to-vehicle communication and advanced traffic management. These applications demand ultra-low latency ( 1 ms) [4,5,25] and high bandwidth (512Gbps-1024Gbps) [25], with reliability reaching 99.99

%-100%. The extremely high bandwidth and low latency requirements highlight the need for real-time data exchange to ensure safety and efficiency in autonomous driving systems.

For eHealth, use cases such as real-time patient monitoring, emergency response, and remote surgery rely on a wide range of metrics. The delay between devices varies from 1 ms to 25ms [26], while end-to-end latency ranges from 1 ms to 250ms [6]. Bandwidth requirements range from 5Mbps to 2Gbps, and data transmission rates range from 1Gbps to 2Gbps, with reliability close to 100%. These metrics demonstrate the need for fast, reliable data exchange to ensure timely healthcare interventions.

The stringent QoS requirements, particularly for delay-sensitive applications such as real-time healthcare monitoring or autonomous vehicles, place significant strain on the implementation and deployment of IoT architectures. Given the limited availability of computation resources and strict communication constraints in IoT and networking architectures, it has led to the exploration of various methods for optimising resource utilisation, enhancing performance, and managing the growing energy consumption driven by the growing IoT devices [15]. Techniques range from fog node selection methods to strategies addressing shortest path computation, optimal node placement, load balancing, and improved internal communication [27–29].

Recent studies have increasingly adopted meta-heuristic algorithms for resource allocation and energy optimisation in IoT systems. These approaches offer flexibility and adaptability, making them well-suited for dynamic and complex IoT environments. One notable contribution is the integration of deep reinforcement learning with discrete cuckoo search (DCS) to enhance optimisation processes. Researchers have also delved into particle swarm optimisation (PSO) for addressing scheduling issues [30] and regret-based adaptive search algorithms for efficient decision-making in dynamic IoT environments [31].

The authors in [32] used whale optimisation algorithms for resource allocation in 5G-IoT networks, demonstrating high efficiency in handling complex resource management tasks. Adaptive fuzzy multi-objective genetic algorithms (AFMOGA) have been developed for virtual resource allocation, optimising the use of resources in cloud computing environments [33]. The evolutionary game approaches have been utilised in mobile edge computing (MEC), shedding light on the competitive dynamics and resource distribution among mobile users [34]. These contributions highlight the growing importance and effectiveness of meta-heuristic methods in optimising resource allocation and minimising energy consumption in various technological domains.

Other notable studies on resource allocation and energy minimisation include the development of heuristic algorithms for network function virtualisation (NFV) and resource allocation in edge computing [35]. Graph theory-based virtual machine (VM) placement has been proposed to reduce IoT service response times [36], alongside adaptive resource allocation methods in low-power wide area networking protocol (LoRaWAN) for IoT applications [37]. Dynamic resource and task allocation algorithms for energy minimisation in mobile cloud systems have been explored by researchers in [38]. A two-level algorithm in MEC aimed at efficient task offloading and resource utilisation is discussed in [39]. Additionally, research into heuristic approaches for resource allocation in IoT includes complex event processing (CEP) for event-driven service invocation [40] and delay-sensitive IoT applications in MEC [41]. While these approaches have shown promise, they often focus on specific layers or metrics, such as latency or energy efficiency, without addressing the comprehensive needs of multi-layered IoT architectures. Additionally, many algorithms do not adapt well to changing network conditions or scale efficiently to larger networks.

Table 1 summarises the latest research done in node selection optimisation in a layered IoT architecture. It highlights the layered selected and prioritised during the optimisation process. It also highlights various heuristics and meta-heuristics algorithms used for optimisation and the metrics controlled during the process.

The rapid increase in IoT devices has brought about a significant rise in energy demand, particularly in resource-intensive domains such as healthcare and automation. Balancing energy utilisation with operational demands within the IoT paradigm is a pressing challenge. Efficient resource utilisation and energy conservation strategies are crucial to mitigate environmental impacts. This research focuses on optimising the distribution of IoT service requests across three computational layers (edge, fog, cloud), a problem often framed as NP-hard due to its complexity.

The various research gaps identified from the brief literature review are as follows:

- Limited comparative studies: Most studies focus on one algorithm without comparing it to others.
- Lack of hybrid approaches: Few studies explore combining multiple algorithms for better results.
- Scalability issues: Many algorithms do not scale well to large networks.
- Lack of adaptability: Existing algorithms often cannot adjust to changing network conditions.
- Narrow focus on metrics: Studies usually focus on a limited set of metrics, such as latency and energy efficiency.
- Overlooked metric inter-dependencies: Relationships between different performance metrics are not well-studied.
- Reliance on simulations: Many studies test algorithms in simulated environments, not real-world scenarios.
- Ignoring hardware constraints: Few studies consider the limitations of actual IoT devices and network nodes.

Given the limitations of deterministic methodologies, such as ILP, in handling complex networks and the increased solution time with growing IoT requests, heuristic and meta-heuristic algorithms present a promising alternative. These algorithms are particularly well-suited for solving complex optimisation problems and offer several advantages:

- Handling complex problems: Meta-heuristic algorithms are great for solving complex optimisation problems.
- Efficient global search: They can perform global searches to avoid getting stuck in local optima.
- Flexibility: These algorithms can be adapted to different problems and constraints.
- Parameter tuning: They allow for fine-tuning parameters to optimise performance.
- Hybrid solutions: Meta-heuristics can be combined with other techniques for more robust solutions.
- Scalability and robustness: Hybrid approaches can scale and adapt to different network sizes and conditions.

**Table 1**
Node selection optimisation in layered IoT architecture.

| Paper | Prioritised Layer(s) | Algorithm Used | Metrics Controlled |
|---|---|---|---|
| (2022) [18,19] | Edge–Fog–Cloud | Integer Linear Programming (ILP) | Latency, Energy Efficiency |
| (2021) [42] | Edge–Fog | Heuristic and Meta-Heuristic | Computational Load, Latency |
| (2022) [30] | Edge–Fog–Cloud | Deep Reinforcement Learning (DRL) | Resource Utilisation, Latency |
| (2020) [43] | Fog-Cloud | Genetic Algorithm | Latency, Energy Efficiency |
| (2022) [44] | Fog-Cloud | Delay-Aware Task Offloading | Energy Efficiency, Delay |
| (2021) [45] | Fog-Cloud | Priority-Aware Genetic Algorithm (PGA) | Task Scheduling, Heterogeneity |
| (2021) [46] | Fog-Cloud | Massive Task Scheduling | Energy, Delay |
| (2021) [47] | Fog | Meta-Heuristic Algorithms | Energy Consumption, Delay |
| (2021) [48] | Edge–Fog–Cloud | Deep Reinforcement Learning (DRL) | Resource Utilisation, Latency |
| (2021) [49] | Edge-Cloud | Whale Optimisation Algorithm (WOA) with Simulated Annealing (SA) (WOA + SA) | Energy Consumption, Network Lifetime |
| (2022) [50] | Fog-Cloud | Heuristic and Meta-Heuristic | Computational Load, Latency |
| (2022) [51] | Edge-Cloud | Delay Optimal Schemes | Delay, Heterogeneity |
| (2022) [52] | Edge-Cloud | Regret-Based Adaptive Search | Convergence Speed, Efficiency |
| (2023) [53] | IoT–Fog–Cloud | Non-dominated Sorting Genetic Algorithm II(NSGA-II) Meta-Heuristic | Energy Consumption, Time Delay |
| (2023) [54] | Fog-Cloud | Particle Swarm Optimisation (PSO) | Network Throughput, Latency |
| (2023) [55] | Edge-Cloud | Whale Optimisation Algorithm | Service Quality, Energy Efficiency |

- Real-world success: Meta-heuristic algorithms have proven effective in practical applications.
- Improved metrics: They have been shown to improve key metrics such as latency, energy efficiency, and computational load.

In conclusion, while existing research has laid the groundwork for energy-efficient resource allocation in IoT architecture, significant gaps remain. Addressing these gaps through meta-heuristic algorithms can provide more flexible, adaptable, and efficient solutions, ultimately enhancing the performance and sustainability of IoT systems. The proposed approach evaluates and compares the performance of three approaches a genetic algorithm, a modified genetic, and a delay-aware mechanism to comprehensively address the challenges of efficient resource allocation, energy minimisation, and stringent QoS requirements in multi-layered IoT architectures. The genetic algorithm-based mechanism excels in solving complex optimisation problems, while the modified genetic-based is specifically adapted for the dynamic nature of IoT environments, offering faster convergence and better scalability. The delay-aware mechanism ensures efficient processing of time-sensitive IoT applications by prioritising nodes with minimal delay. Together, these algorithms provide a robust, adaptable, and scalable solution that fills gaps in existing methods and meets the demands of modern IoT applications. Given the challenges and the motivation, this paper investigates the applicability of genetic and modified genetic algorithms in real-time network scenarios for node selection and energy efficiency optimisation. Additionally, a delay-aware algorithm based on a latency-focused mechanism for incoming IoT requests has been developed.The effectiveness of these approaches is benchmarked against an ILP-based mechanism. The results from ILP, genetic, modified genetic, and delay-aware algorithms were comprehensively compared, demonstrating the practicality and effectiveness of the proposed mechanisms.

## 3. Proposed node selection mechanisms

Managing the intricate structure of IoT networks, coupled with the vast number of devices connected at the edge layer, poses significant challenges for efficiently operating an edge–fog–cloud IoT architecture. Activating and utilising all nodes across this architecture would result in substantial operational costs and energy consumption. To mitigate this, energy-aware node selection mechanisms are essential for identifying optimal nodes to process IoT requests, thereby minimising energy costs while supporting diverse use cases [18].

While traditional prediction-based solutions address scheduling, offloading, and resource allocation in integrated cloud–edge architectures, they often struggle to scale effectively with the increasing complexity of IoT networks. In contrast, the mechanisms proposed in this study offer scalable and efficient solutions tailored for modern IoT environments. Traditional ILP-based mechanisms, such as those discussed in [19], are effective but face scalability limitations as network size and complexity grow. The proposed mechanisms, leveraging genetic and delay-aware algorithms, are specifically designed to provide sub-millisecond solutions, making

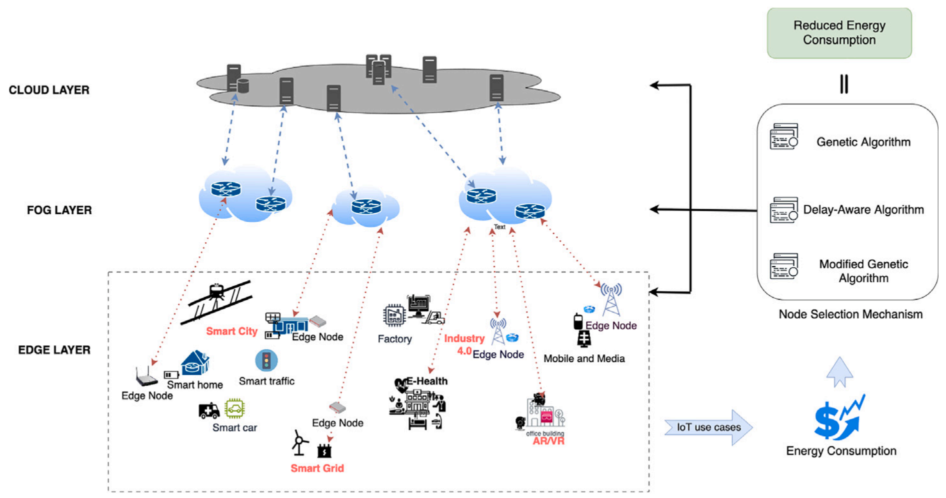

**Fig. 1.** IoT Architecture with Energy-Optimising Mechanisms.

them particularly suitable for delay-sensitive and real-time IoT applications. These mechanisms surpass traditional prediction-based methods by delivering faster, more scalable optimisation while accommodating diverse IoT request patterns.

The proposed node selection mechanisms, which are based on genetic, modified genetic, and delay-aware algorithms, aim to address the dual challenges of energy consumption and resource allocation in IoT architectures. By incorporating diverse QoS metrics and application-specific constraints, these mechanisms build upon a foundational ILP-based approach, overcoming its limitations in large-scale IoT networks. Each mechanism has been evaluated individually to highlight its unique strengths, while comparative analyses demonstrate their collective improvements in energy efficiency and resource management over the ILP baseline.

The genetic-based mechanism excels in solving complex optimisation problems through iterative global searches, ensuring energy-efficient node selection. The modified genetic-based mechanism extends these capabilities by adapting to the dynamic nature of IoT environments, offering faster convergence and enhanced scalability. The delay-aware mechanism, on the other hand, prioritises latency-sensitive applications by selecting nodes that minimise delay while meeting energy constraints. Together, these mechanisms provide a robust, adaptable solution that meets the stringent demands of modern IoT applications.

The objective of the proposed genetic algorithm, modified genetic, and delay-aware mechanisms is to minimise the energy cost associated with processing incoming IoT requests across diverse use cases. This optimisation problem is formulated with a clear objective function, focusing on energy minimisation while adhering to QoS and application-specific constraints. The ILP-based mechanism, as explored in [18,19], serves as the foundation for these proposed mechanisms, where it minimises energy costs while processing requests from IoT devices connected at the edge layer, ensuring QoS compliance.

The genetic, modified-genetic, and delay-aware mechanisms are not only compared individually with the ILP-based approach but also assessed in terms of their collaborative applicability in real-time IoT scenarios. These mechanisms are iteratively refined to integrate delay-aware decision-making with genetic algorithm principles, ensuring both delay and energy constraints are effectively addressed. By combining the strengths of these mechanisms, the proposed solution achieves significant scalability and energy optimisation while maintaining real-time processing capabilities. The proposed mechanisms are designed to operate within sub-millisecond timeframes, aligning with real-time QoS constraints of IoT applications, as suggested in the related work above. By optimising energy costs while meeting latency requirements, these mechanisms demonstrate their applicability in scenarios such as real-time health monitoring and autonomous vehicle operations.

The IoT architecture considers the varied number of servers installed at each node within the edge–fog–cloud IoT architecture and the initial utilisation of resource capacity for each node. The proposed mechanism considers the activation and deactivation of nodes. The nodes that are currently operational and being used are referred to as active nodes, whereas inactive nodes are linked to the IoT network but lack active servers. To reduce energy usage across the IoT architecture, inactive computation nodes should be deactivated [18]. The activation and operation of nodes and servers are determined by the energy consumption costs involved and diverse QoS metrics. By taking into account factors such as resource availability, connectivity, and latency, the mechanism balances the trade-off between activating an inactive node at either of the layers and using an already active one for processing all the diverse incoming IoT requests. The mechanism ensures that the constraints of diverse IoT use cases, such as resource processing capacity and latency, as well as those of the communication architecture, including connectivity and delay support, are met. The following subsections discuss different node selection mechanisms proposed and evaluated in this journal. The use of the proposed mechanism for an edge–fog–cloud IoT architecture in this journal is illustrated in Fig. 1.

### 3.1. Genetic-based mechanism

In the genetic-based mechanism, a stationary energy cost is considered for processing jobs at each node across the edge–fog–cloud layers and for activating servers at edge and fog nodes. For the cloud layer, a constant energy cost is assumed for each request, simplifying the complexity of cost calculations [56].

The objective function, outlined in Eq. (1), minimises the total energy cost across all layers while processing IoT requests from diverse use cases [19]. It includes the energy costs for activating new nodes at the edge and fog layers ($Cost_{en}$ and $Cost_{fn}$), activating servers on active nodes ($Cost_{es}$ and $Cost_{fs}$), and processing jobs at the cloud layer ($Cost_c$), where nodes are always active. Relevant constraints are incorporated as discussed in [19].

$$\begin{aligned}
&Minimise( \sum_{x\epsilon\mathcal{E}}(\boldsymbol{E_a}[x] - \boldsymbol{Loc_{e[x]}}) * Cost_{en} + \\
&\sum_{x\epsilon\mathcal{E}}\sum_{s\epsilon\mathcal{E}s}(\boldsymbol{N_{EdgeS}}[x][s] - \boldsymbol{N_{EdgeSI}}[x][s]) * Cost_{es} + \\
&\sum_{y\epsilon\mathcal{F}}(\boldsymbol{F_a}[y] - \boldsymbol{Loc_f}[y]) * Cost_{fn} + \\
&\sum_{x\epsilon\mathcal{F}}\sum_{s\epsilon\mathcal{F}s}(\boldsymbol{N_{FogS}}[x][s] - \boldsymbol{N_{FogSI}}[x][s]) * Cost_{fs} + \\
&\sum_{z\epsilon C}(\boldsymbol{C_a}[z] - \boldsymbol{Loc_{c[z]}}) * Cost_c)
\end{aligned} \tag{1}$$

where,

- Edge Layer:
  - Cost for activating/deactivating nodes: $\sum_{x\epsilon\mathcal{E}}(\boldsymbol{E_a}[x] - \boldsymbol{Loc_{e[x]}}) * Cost_{en}$
  - Cost for activating/deactivating servers: $\sum_{x\epsilon\mathcal{E}}\sum_{s\epsilon\mathcal{E}s}(\boldsymbol{N_{EdgeS}}[x][s] - \boldsymbol{N_{EdgeSI}}[x][s]) * Cost_{es}$
- Fog Layer:
  - Cost for activating/deactivating nodes: $\sum_{y\epsilon\mathcal{F}}(\boldsymbol{F_a}[y] - \boldsymbol{Loc_f}[y]) * Cost_{fn}$
  - Cost for activating/deactivating servers: $\sum_{x\epsilon\mathcal{F}}\sum_{s\epsilon\mathcal{F}s}(\boldsymbol{N_{FogS}}[x][s] - \boldsymbol{N_{FogSI}}[x][s]) * Cost_{fs}$
- Cloud Layer:
  - Cost for running nodes: $\sum_{z\epsilon C}(\boldsymbol{C_a}[z] - \boldsymbol{Loc_{c[z]}}) * Cost_c$

- $n_C$: Total node count deployed at cloud layer
- $n_F$: Total node count deployed at fog layer
- $n_E$: Total node count deployed at edge layer
- $s_e$: Highest number of edge servers that can be installed at each edge node
- $s_f$: Highest number of edge servers that can be installed at each fog node
- $s_c$: Highest number of edge servers that can be installed at each cloud node
- $\mathcal{E}$=1, …, $n_E$: Represents the collection of all nodes designated as the edge layer
- $\mathcal{F}$=1, …, $n_F$: Represents the collection of all nodes designated as the fog layer
- $\mathcal{C}$=1, …, $n_C$: Represents the collection of all nodes designated as the cloud layer
- $\boldsymbol{Loc_e}[l]$: Parameter representing the location $l$ positioned as the edge node $e$
- $\boldsymbol{Loc_f}[l]$: Parameter representing the location $l$ positioned as the fog node $f$
- $\boldsymbol{Loc_c}[l]$: Parameter representing the location $l$ positioned as the cloud node $c$
- $\mathcal{E}_s = 1, \dots, s_e$: Represents a set of edge servers that can be deployed at each edge node
- $\mathcal{F}_s = 1, \dots, s_f$: Represents a set of fog servers that can be deployed at each fog node
- $\mathcal{C}_s = 1, \dots, s_c$: Represents a set of cloud servers that can be deployed at each cloud node
- $\boldsymbol{E_a}$[e]: Boolean variable

$$\boldsymbol{E_a}[e] = \begin{cases} 1; & \text{if the } e\text{th edge node is active} \\ 0; & \text{otherwise, where } e \in \mathcal{E} \end{cases}$$

- $\boldsymbol{F_a}$[f]: Boolean variable

$$\boldsymbol{F_a}[f] = \begin{cases} 1; & \text{if the } f\text{th fog node is active} \\ 0; & \text{otherwise, where } f \in \mathcal{F} \end{cases}$$

- $\boldsymbol{C_a}$[c]: Binary variable

$$\boldsymbol{C_a}[c] = \begin{cases} 1; & \text{if the } c\text{th cloud node is active} \\ 0; & \text{otherwise, where } c \in \mathcal{C} \end{cases}$$

- $\boldsymbol{N_{EdgeS}}[e][i]$: Boolean variable

$$\boldsymbol{N_{EdgeS}}[e][i] = \begin{cases} 1; & \text{if the server at position } i\text{th} \\ & \text{of the } e\text{th edge node is active} \\ 0; & \text{otherwise, where } i \in \mathcal{E}_s \\ & e \in \mathcal{E} \end{cases}$$

- $\boldsymbol{N_{FogS}}[f][i]$: Boolean variable

$$\boldsymbol{N_{FogS}}[f][i] = \begin{cases} 1; & \text{if the server at position } i\text{th} \\ & \text{of the } f\text{th fog node is active} \\ 0; & \text{otherwise, where } i \in \mathcal{F}_s \\ & f \in \mathcal{F} \end{cases}$$

- $\boldsymbol{N_{EdgeSI}}[i][j]$: Boolean variable

$$\boldsymbol{N_{EdgeSI}}[i][j] = \begin{cases} 1; & \text{if the } i\text{th edge node initially has} \\ & \text{an active server at the } j\text{th} \\ & \text{position} \\ 0; & \text{otherwise, where } i \in \mathcal{E} \\ & j \in \mathcal{E}_s \end{cases}$$

- $\boldsymbol{N_{FogSI}}[x][y]$: Boolean variable

$$\boldsymbol{N_{FogSI}}[x][y] = \begin{cases} 1; & \text{if the } x\text{th fog node initially has} \\ & \text{an active server at the } y\text{th} \\ & \text{position} \\ 0; & \text{otherwise, where } x \in \mathcal{F} \\ & y \in \mathcal{F}_s \end{cases}$$

- $Cost_{en}$ : Cost incurred in activating the inactive node at edge layer
- $Cost_{fn}$: Cost incurred in activating the inactive node at fog layer
- $Cost_{es}$: Cost incurred in activating the inactive server at edge layer
- $Cost_{fs}$: Cost incurred in activating the inactive server at fog layer
- $Cost_{c}$: Cost incurred in processing the job at cloud layer

The flowchart of the implemented genetic algorithm approach is illustrated in Fig. 2. The algorithm begins with the creation of a feasible population that adheres to network and application QoS constraints. Each individual in the population consists of three lists representing job allocations to nodes at the edge, fog, and cloud layers. The position of each job in the list corresponds to the node where it is allocated. Population generation ensures feasibility by verifying constraints such as connectivity, resource capacity, and latency. Following the establishment of the population list, the fitness value is calculated, representing the energy cost for each population.

The next step is the main genetic algorithm loop depicted in Fig. 3. In this step, the number of generations for running the genetic loop is set. Then, the elite individuals, selected based on fitness, are preserved across generations and excluded from crossover and mutation processes. A tournament selection method is used for selecting offspring for the next generation [57]. A single-point ordered crossover, triggered based on a predefined probability, generates new offspring. Non-elite offspring are evaluated for feasibility; infeasible offspring are discarded, and the original population is retained.

Mutation, applied with a predefined probability, uses swap mutation logic [58]. The new mutants are checked against elite and feasibility criteria based on QoS constraints such as connectivity, resource capacity, and latency for each IoT request. Non-elite mutants are added to the new offspring list, and if mutants fail feasibility checks, they are discarded.

Finally, the offspring list is examined. If the list is empty, elites are added to ensure continuity. The offspring list may also be truncated to match the original population size. The entire new population's fitness is evaluated, and if the current fitness surpasses

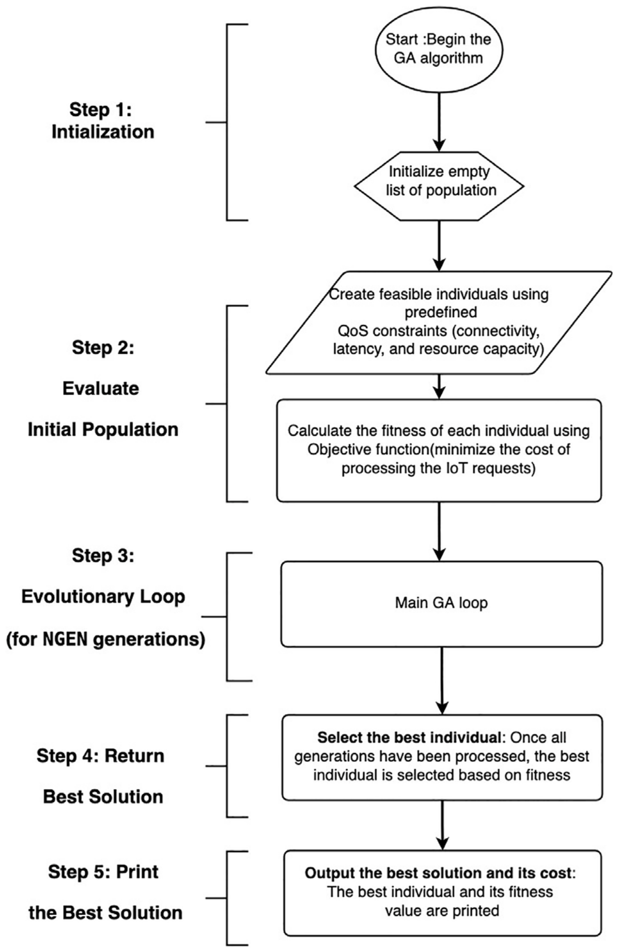
Step 1: Intialization
Start :Begin the GA algorithm
Initialize empty list of population
Step 2: Evaluate Initial Population
Create feasible individuals using predefined QoS constraints (connectivity, latency, and resource capacity)
Calculate the fitness of each individual using Objective function(minimize the cost of processing the IoT requests)
Step 3: Evolutionary Loop (for NGEN generations)
Main GA loop
Step 4: Return Best Solution
Select the best individual: Once all generations have been processed, the best individual is selected based on fitness
Step 5: Print the Best Solution
Output the best solution and its cost: The best individual and its fitness value are printed

**Fig. 2.** Genetic Algorithm-based Optimisation Mechanism Flowchart.

the previous measurements, it is updated. Should there be no improvement in fitness over the last 25 generations, the loop is exited, and the best individual is selected from the population list for display as illustrated in Fig. 2 and Fig. 3.

This mechanism selects an ideal node at the edge layer, fog layer, or cloud layer where each of the IoT requests should be processed, minimising the energy used in executing all the incoming IoT requests.

While the proposed genetic-based mechanism is evaluated for energy efficiency and latency, it is important to note that its effectiveness can be influenced by hardware constraints inherent to IoT devices. Factors such as limited processing power, memory capacity, and battery life in low-power IoT devices may impose restrictions on the algorithm's performance. For instance, the computation-heavy steps of the genetic algorithm, such as crossover and mutation, may need to be optimised or simplified for devices with minimal CPU/GPU resources. Similarly, memory constraints could limit the population size used in the genetic algorithm, potentially impacting convergence efficiency and solution quality. Future implementations will explore lightweight optimisations tailored to low-power devices to balance computational demands with energy efficiency.

### *3.2. Modified genetic-based mechanism*

The proposed modified genetic-based mechanism is extracted from the genetic algorithm mechanism described in Section 3.1 and customised with additional logic of a latency greedy approach for handling incoming IoT requests. This mechanism tunes the original mechanism to prioritise the latency demands of IoT requests. The system architecture is designed to sort incoming IoT requests into two groups based on latency requirements. Creating two groups of incoming IoT requests gives the modified genetic algorithm an upper hand in running the allocation and creating a population quickly. Thus, it reduces the time the mechanism takes to produce the solution compared to the original genetic algorithm-based mechanism proposed in Section 3.1.

The first group consists of IoT requests that meet the maximum latency threshold permissible at the fog layer. Any requests with latency demands at or below this threshold are collated into this primary group. Conversely, the remaining IoT requests that exceed the latency capability of the fog layer are assembled into a second group, targeted primarily for the edge layer.

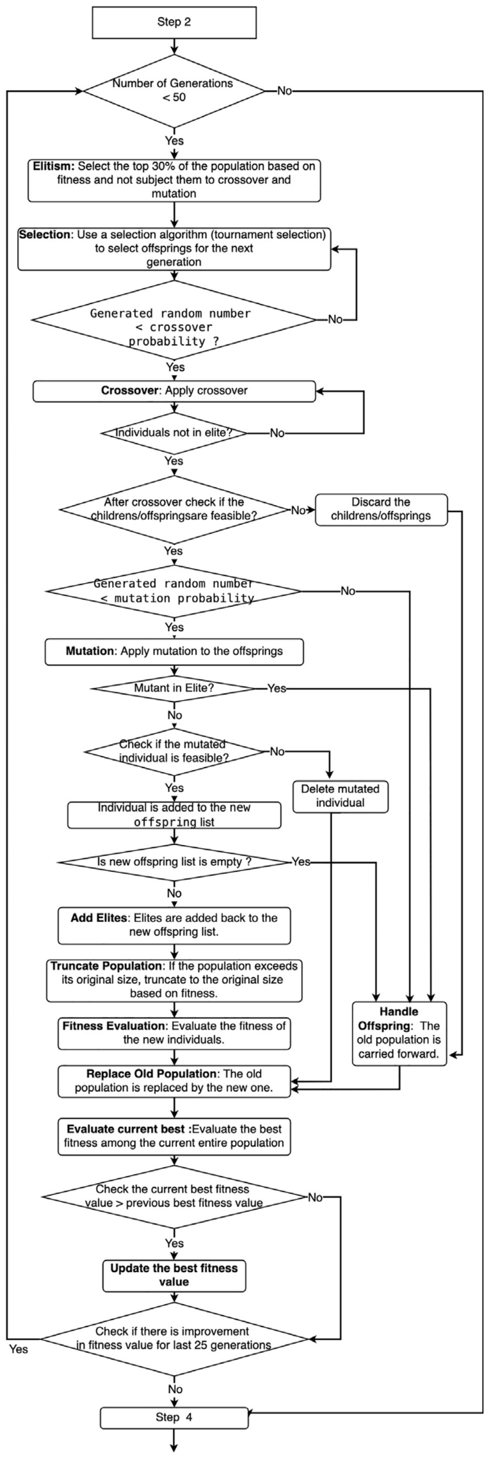
Step 2
Number of Generations < 50
No
Yes
Elitism: Select the top 30% of the population based on fitness and not subject them to crossover and mutation
Selection: Use a selection algorithm (tournament selection) to select offsprings for the next generation
Generated random number < crossover probability ?
No
Yes
Crossover: Apply crossover
Individuals not in elite?
No
Yes
After crossover check if the childrens/offspringsare feasible?
No
Discard the childrens/offsprings
Yes
Generated random number < mutation probability
No
Yes
Mutation: Apply mutation to the offsprings
Mutant in Elite?
Yes
No
Check if the mutated individual is feasible?
No
Delete mutated individual
Yes
Individual is added to the new offspring list
Is new offspring list is empty ?
Yes
No
Add Elites: Elites are added back to the new offspring list.
Truncate Population: If the population exceeds its original size, truncate to the original size based on fitness.
Fitness Evaluation: Evaluate the fitness of the new individuals.
Handle Offspring: The old population is carried forward.
Replace Old Population: The old population is replaced by the new one.
Evaluate current best :Evaluate the best fitness among the current entire population
Check the current best fitness value > previous best fitness value
No
Yes
Update the best fitness value
Check if there is improvement in fitness value for last 25 generations
Yes
No
Step 4

**Fig. 3.** Main Genetic Algorithm-based Mechanism Loop Flowchart.

During the node selection, if the fog layer cannot accommodate IoT requests from the first group, they are redirected to the edge layer and, if unsuitable, to the cloud layer. Similarly, requests from the second group, initially intended for the edge layer, are rerouted to the cloud layer if necessary. If the requests cannot be processed at the cloud layer, they are considered for the fog layer.

This strategic allocation and grouping of IoT requests are integral to the setup of the initial population within the main genetic mechanism. This methodology is explicitly illustrated in Fig. 2 and Fig. 3, showcasing the dynamic adaptation of node selection in response to varying latency requirements. The modification of the formation of two groups of incoming IoT requests takes place during the initial population creation at step 1 in Fig. 2. The rest of the steps and flowcharts stay the same as the original mechanism.

### *3.3. Delay-aware mechanism*

With the intention to further reduce the solution time, another heuristic approach is proposed. The delay-aware node selection mechanism is based on the latency greedy mechanism. In this proposed mechanism, the energy consumption at each layer is taken into account. For this investigation, it is assumed that the cloud layer consumes the most energy, followed by the edge layer, with the fog layer consuming the least. This order can be changed depending on the requirements and energy usage of each layer. Based on the scenario considered, the goal is to process the maximum number of IoT requests at the fog layer rather than the edge and cloud layers while simultaneously meeting the QoS constraints based on application and network requirements for all IoT use cases. When the IoT request arrives, they are sorted into two groups, that is, IoT requests to be processed at the fog layer initially into group one and IoT requests to be processed at the edge layer into group two, based on the maximum latency supported at the fog layer.

If the fog layer fails to process any IoT requests allocated to it initially, they would be assigned to the edge layer, and if not, then the cloud layer sequentially. If the requests initially allocated to the edge layer fail, they will be assigned to the cloud layer, and if they fail, then they will be allocated to the fog layer. At all the layers, this mechanism would always ensure that the QoS constraints of each incoming IoT request are met while they are being allocated to a node at either of the layers. The entire allocation and grouping logic is shown in Fig. 4.

## 4. Performance evaluation

The performance of the proposed mechanisms has been evaluated through simulations conducted in Python. The specific network employed for this evaluation is presented in Fig. 5. This network serves as a crucial component in the evaluation, providing a structured visualisation that aids in understanding the effectiveness of the mechanisms.

The considered network consists of 23 nodes, including 2 at the cloud layer, 5 at the fog layer, and 16 at the edge layer, shown in Fig. 5. The network consists of both inactive and active nodes. In Fig. 5, inactive nodes are depicted by grey oval shapes, while active nodes are represented by blue oval shapes. The notation “R” represents the supported resource capacity of each node. Each node is assigned a number for identification purposes. The majority of nodes are equipped with a single server. Nodes with multiple servers are illustrated in Fig. 5, such as nodes 13 and 23. The delay and the connectivity between the nodes are shown additionally in Fig. 5 on the links between the nodes.

The evaluation begins with an initial set of five incoming IoT requests and is incrementally scaled to accommodate up to 16 IoT requests. The details of these requests, originating from diverse IoT applications and use cases such as smart grid, eHealth, and autonomous vehicles, are listed in Table 2. As discussed in Section 2, the latency, the resource requirements of the diverse IoT requests incoming considered from smart grids, eHealth and autonomous vehicles fall within the defined range of acceptable performance metrics for real-time systems. For example, job numbers (3, 5, 8, 9, 10, 11, 13, 15), (6, 7, 12, 16), and (1, 2, 4, 14) represent requests incoming from smart grids or smart homes, eHealth, and autonomous vehicles, respectively. Table 2 lists essential information such as the resource capacity each IoT request demands, the latency requirements, and the origin node of each request. The job numbers considered from smart grids and smart homes comprise IoT requests from mobile devices, smart solar systems, smart meters, and SCADA devices that are used for real-time grid monitoring and efficient energy management. These requests often have stringent latency requirements and dynamic resource needs due to the critical nature of grid operations and the varying load on renewable energy sources. Similarly, the job numbers considered from eHealth include IoT requests originating from remote surgery devices that demand ultra-low latency for precise control, health data records and report retrieval applications that prioritise secure and efficient data transfer, and hospital appliances such as patient monitoring devices, which require reliable connectivity to ensure patient safety. These scenarios emphasise the importance of balancing latency, resource capacity, and energy efficiency to support life-critical applications. The job numbers associated with autonomous vehicles comprise IoT requests from diverse sensors and communication devices embedded in autonomous cars, as well as from traffic monitoring systems. These applications demand real-time processing of large volumes of sensor data to ensure safe navigation, collision avoidance, and efficient traffic management. The latency tolerance for these applications is extremely low, making it critical to evaluate the proposed mechanisms under such high-pressure, delay-sensitive conditions. A diverse range of resource and latency requirements is considered for these IoT requests to simulate realistic scenarios and obtain a more accurate, real-time estimate of the performance of the proposed mechanisms. By analysing these scenarios in detail, the evaluation provides insights into the scalability, adaptability, and efficiency of the proposed node selection mechanisms in meeting the unique demands of different IoT applications. The delay and connectivity matrices are explicitly defined and maintained in separate records while simulating the approaches in Python under different network scenarios. The last column, “Node Allocated for Respective Mechanism” in Table. 2 lists the nodes assigned to process each IoT request for the respective mechanisms, based on the network configuration shown in Fig. 5.

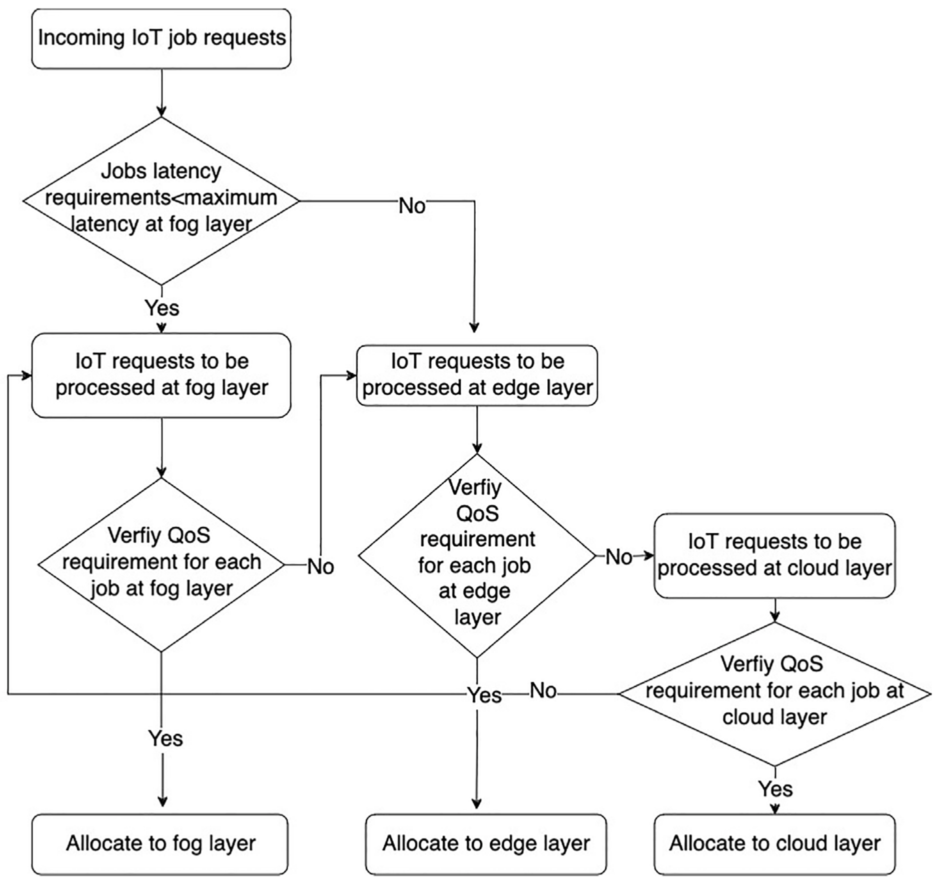
Incoming IoT job requests
Jobs latency requirements<maximum latency at fog layer
No
Yes
IoT requests to be processed at fog layer
IoT requests to be processed at edge layer
Verfiy QoS requirement for each job at fog layer
No
Verfiy QoS requirement for each job at edge layer
No
IoT requests to be processed at cloud layer
Verfiy QoS requirement for each job at cloud layer
Yes
No
Yes
Yes
Allocate to fog layer
Allocate to edge layer
Allocate to cloud layer

**Fig. 4.** Delay-aware Optimisation Mechanism.

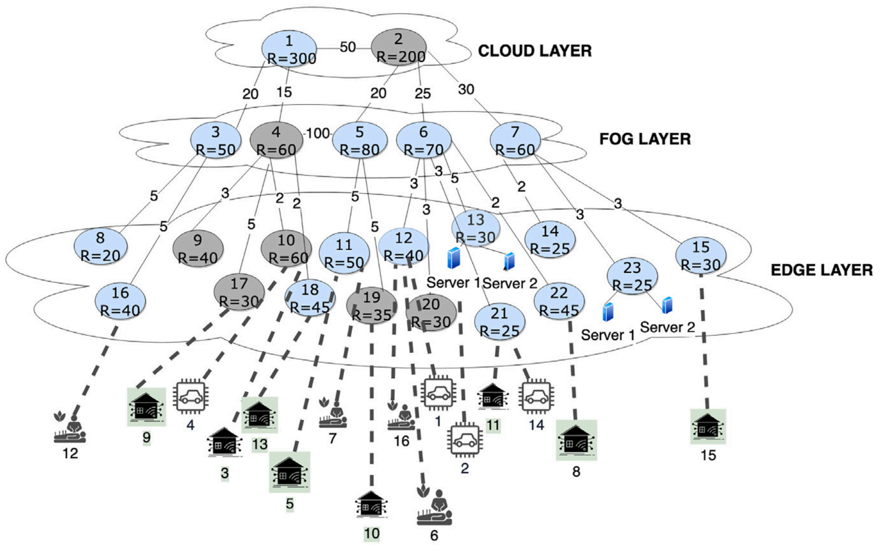

**Fig. 5.** Network Configuration for Evaluation.

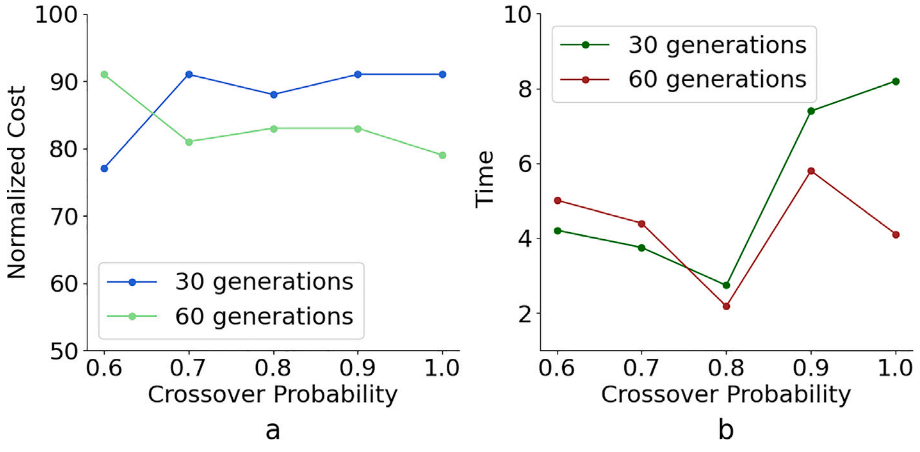

**Fig. 6.** Efficiency vs. Crossover Probability in Genetic-based Mechanism.

**Table 2**
Job requirements and node allocations across mechanisms.

| Job No. | Job Requirement | | | Node Allocated for Respective Mechanism | | | |
|---|---|---|---|---|---|---|---|
| | Resource | Latency | Origin | ILP | Genetic | Modified Genetic | Delay-aware |
| 1 | 40 | 100 | 12 | 7 | 21 | 1 | 18 |
| 2 | 30 | 108 | 13 | 23 | 2 | 12 | 19 |
| 3 | 80 | 100 | 10 | 5 | 14 | 2 | 2 |
| 4 | 30 | 106 | 10 | 22 | 13 | 23 | 14 |
| 5 | 40 | 110 | 11 | 11 | 1 | 5 | 20 |
| 6 | 20 | 106 | 12 | 23 | 3 | 3 | 3 |
| 7 | 30 | 107 | 11 | 16 | 22 | 4 | 5 |
| 8 | 40 | 10 | 22 | 6 | 6 | 21 | 21 |
| 9 | 20 | 20 | 17 | 1 | 18 | 10 | 1 |
| 10 | 30 | 107 | 19 | 15 | 8 | 6 | 8 |
| 11 | 20 | 106 | 21 | 13 | 23 | 23 | 4 |
| 12 | 30 | 10 | 16 | 3 | 16 | 8 | 16 |
| 13 | 30 | 20 | 18 | 18 | 4 | 17 | 9 |
| 14 | 20 | 105 | 21 | 13 | 5 | 22 | 7 |
| 15 | 25 | 30 | 15 | 14 | 23 | 7 | 23 |
| 16 | 40 | 100 | 12 | 12 | 13 | 14 | 23 |

For result analysis, the energy consumption costs are normalised and normalised energy costs are presented throughout the journal. The energy costs of each node and layer include those for operating the switch, router, and server. The normalised energy cost represents 1:1000 of the actual energy consumption in Watts. The operational energy costs for edge and fog layer servers at each node are set at 6 and 3, respectively [56]. The activation energy costs for edge and fog nodes are 8 and 5, respectively. Executing a job at a cloud layer node incurs a fixed energy cost of 9. The time scale used in all the graphical representations is in seconds. Each data point shown in the graphical representations is calculated as an average value derived from 15 independent simulation runs using the same experimental setup. For example, the normalised energy cost at each point is the average of the normalised energy consumption incurred after running the simulations 15 times for the same set of incoming IoT requests for the respective mechanism. Similarly, the average solution time is computed by averaging the solution time incurred for processing the same set of incoming IoT requests over 15 simulation runs for each mechanism. This approach ensures that the results are robust and account for any variability in the simulation environment. The ILP mechanism used for comparison is primarily based on the models discussed in [18,19] without considering the bandwidth requirements of the IoT requests.

## *4.1. Evaluation of genetic-based mechanism*

The genetic-based mechanism was evaluated with an initial genetic loop of 50 generations, maintaining a population size of 10. The top 30 % of the population was classified as elite. The evaluation varied three critical parameters: crossover probability, mutation probability, and the number of generations to assess their impact on energy cost and evaluation time. Crossover probability, tested in the range of 0.6 to 1.0, balances exploration and exploitation in genetic algorithms. Higher probabilities are particularly effective for blending information from parent solutions to produce superior offspring [59].

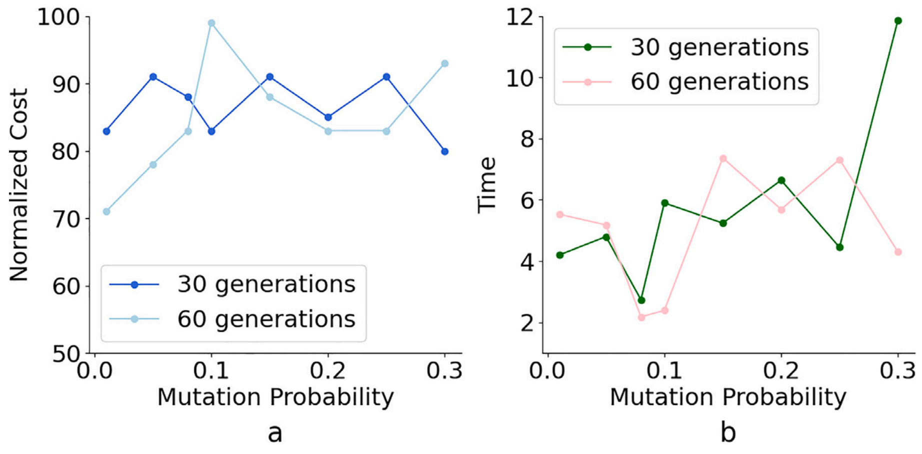

**Fig. 7.** Efficiency vs. Mutation Probability in Genetic-based Mechanism.

Fig. 6 illustrates the impact of varying crossover probability on energy cost and solution time for processing 16 diverse incoming IoT requests using a genetic-based mechanism, with a mutation probability of 0.08 and evaluations conducted over 30 and 60 generations.

Fig. 6.a illustrates the relationship between crossover probability and normalised cost for 30 and 60 generations. At lower crossover probabilities, the normalised cost is higher for 60 generations compared to 30 generations. This is because, with fewer generations, the algorithm tends to converge more quickly, leading to a lower initial energy cost. However, with more generations, the algorithm explores a broader solution space before converging, which may initially result in higher costs as it searches for a more optimal solution. As the crossover probability increases, the algorithm with 60 generations shows a trend of gradually reducing costs as it converges to a more refined solution over time.

Fig. 6.b examines the impact of crossover probability on the time taken to reach a solution for 30 and 60 generations. It can be seen here that with both 30 and 60 generations, the time required generally decreases as the crossover probability increases up to a certain point. For lower crossover probabilities, the algorithm spends more time searching the solution space, especially with 60 generations, due to the increased number of possible combinations it explores. As the crossover probability increases, the algorithm converges faster, reducing the computation time. However, after a certain crossover probability threshold, the time may increase again, as the algorithm might spend additional time ensuring that the solutions are thoroughly explored and refined, particularly evident with 30 generations at a crossover probability of 1.0.

From Fig. 6.a, and Fig. 6.b, it is evident that irrespective of the number of generations, a crossover probability of 0.8 results in the minimum time required to process 16 requests and the overall energy cost incurred in handling these 16 incoming IoT requests is comparatively average compared to other crossover probabilities for both 30 and 60 generations. With a considered balanced trade-off between energy cost and time incurred in processing the diverse incoming IoT requests, 0.8 would be an ideal crossover probability value for the current mechanism evaluation.

The ideal range for mutation probability in a genetic algorithm typically varies, but a standard recommendation is to keep it relatively low to ensure that the algorithm does not diverge too far from potentially optimal solutions by making too many random changes. Often, mutation probabilities are suggested to be between 0.001 and 0.01. This range helps maintain diversity within the population without overriding the role of the crossover in effectively searching through the solution space [60,61]. For the comparison, the mutation probability scale on the x-axis in Fig. 7 varies from 0.01–0.1 in descending order difference of 0.04, 0.03, and 0.02, respectively. Once the x-axis scale crosses the 0.1 mark, then as recommended in [61], the scale is increased with a constant difference of 0.05, which is kept steady later. The increase in scale is to test the impact of higher mutation rates on the diversity of solutions. A mutation probability that is too low may cause the algorithm to converge prematurely to local optima, while a mutation rate that is too high could introduce excessive randomness, reducing convergence efficiency. The x-axis scale is limited to 0.3, equating to a 30 % mutation probability, because there is no need for excessive diversity in the solution search. This mutation probability of 30 % is considered relatively high.

Fig. 7 presents a comparison highlighting the impact of varying mutation probability on both the energy cost and the solution time for processing the diverse incoming IoT requests using the genetic-based approach. This analysis is performed under certain conditions: the crossover probability is maintained at 0.8, there are 16 IoT requests, and the number of generations examined is 30 and 60.

Fig. 7.a, analyses the effect of mutation probability on the normalised energy cost for 30 and 60 generations. As the mutation probability increases, there is a fluctuating trend in normalised costs for both 30 and 60 generations. This fluctuation occurs because mutation introduces randomness into the solution, which can either help the algorithm escape local optima and find better solutions or disrupt the convergence process, leading to higher costs. For 60 generations, the algorithm has more opportunities to refine the solution, resulting in more visible fluctuations. However, the overall trend indicates that lower mutation probabilities tend to keep

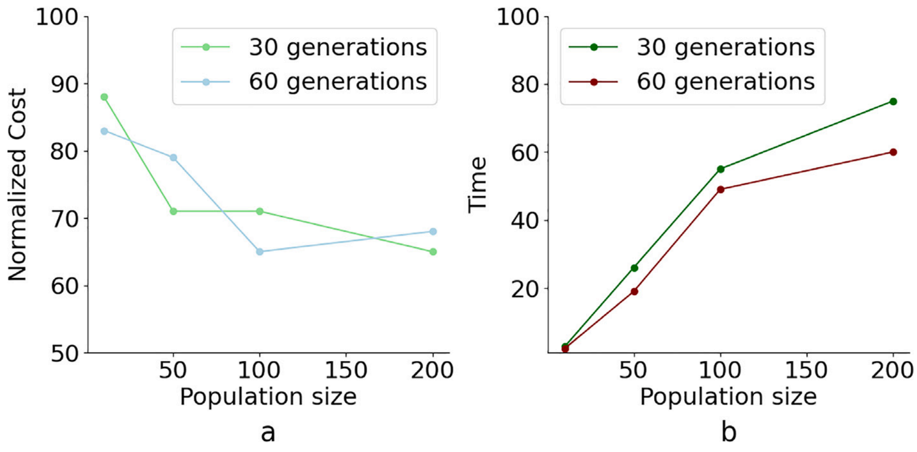

**Fig. 8.** Efficiency vs. Population Size in Genetic-based Mechanism.

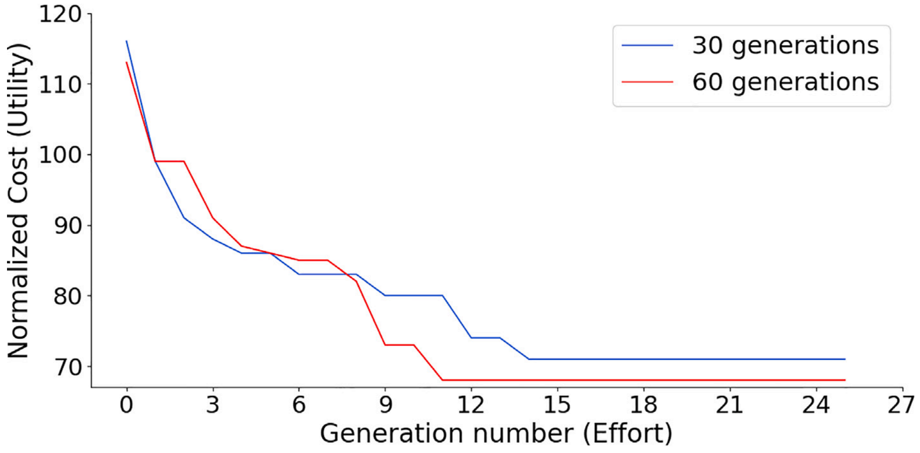

**Fig. 9.** Effort vs. Utility in Genetic-based Mechanism.

the cost more stable, while higher mutation probabilities may cause more variability in the cost, potentially disrupting the solution refinement.

Fig. 7.b, examines the impact of mutation probability on the time taken to reach a solution. Similar to the cost analysis, time exhibits variability with changes in mutation probability. Lower mutation probabilities generally result in a more consistent and faster convergence for both 30 and 60 generations as the algorithm progresses steadily without significant disruptions. However, as mutation probability increases, particularly beyond 0.2, the time required for convergence becomes more unpredictable, especially for 30 generations, where the time spikes significantly at a mutation probability of 0.3. This is likely due to the increased randomness introduced by mutation, causing the algorithm to take longer to converge as it explores a wider range of solutions.

From Fig. 7.a and Fig. 7.b, it can be concluded that across any number of generations, a mutation probability near 0.08 leads to the lowest processing time for the 16 incoming IoT requests and that results in a relatively average energy cost for processing these requests using the genetic-based approach. Thus, considering the balanced trade-off between energy cost and time incurred in processing the diverse incoming IoT requests, 0.08 would be an ideal mutation probability value for the evaluation of the current mechanism.

The analysis of the effect of population variation on the energy cost and solution time is shown in Fig. 8. From Fig. 8.a it is evident that for both generation counts, as the population size grows, the energy cost reduces. The main reason behind the reduction in energy cost is that the mechanism tries to converge towards optimal energy cost. However, as shown in Fig. 8.b, there is an inverse effect on the time required to process the IoT requests, which increases with the larger population size. Though the mechanism tries to converge towards optimal energy cost as the population size increases, it takes an increasingly longer time to achieve this. For example, to keep the time required to process the IoT requests below 100 s, the population size should be restricted to 200.

To understand the convergence of the number of generations required and to get an estimate of the approximate energy cost incurred for a genetic-based mechanism by varying the number of generations, an effort versus utility graph is represented in Fig. 9. Fig. 9 shows the effort as in the number of generations versus fitness value, which is the energy cost incurred in processing diverse

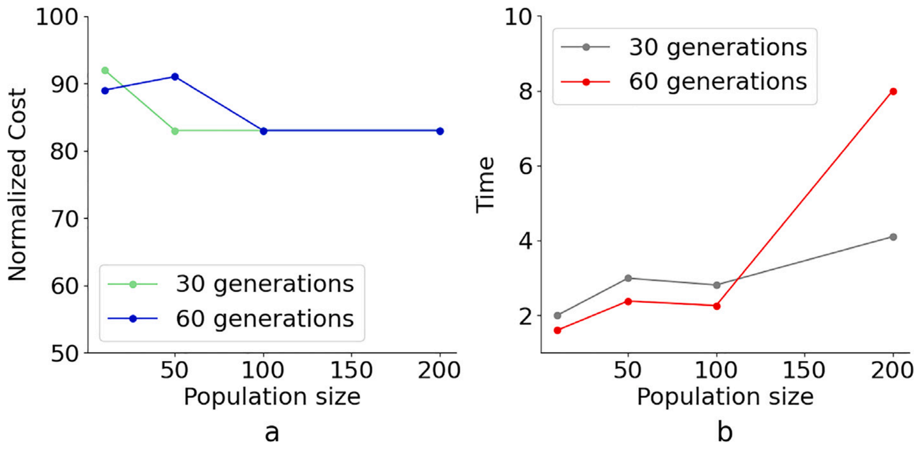

**Fig. 10.** Efficiency vs. Population Size in Modified Genetic-based Mechanism.

**Table 3**
Cost comparison of solutions for 16 IoT requests across mechanisms.

| | Mechanisms | | | |
|---|---|---|---|---|
| | ILP | Genetic | Modified - Genetic | Delay- aware |
| Cost | 78 | 83 | 99 | 110 |
| Time taken (seconds) | 6 | 7 | 1.8 | 0.0018 |

incoming IoT requests when the number of generations is set to 30 and 60. Fig. 9 shows that the genetic-based mechanism requires at least 10 generations to converge towards a minimal energy cost incurred for processing all diverse incoming IoT requests. The total energy cost incurred in processing all the diverse incoming 16 IoT requests is 83 and 77 for 30 and 60 generations, respectively. The time taken to process the IoT request is 7 s and 8 s for 30 and 60 generations, respectively, higher than ILP.

### 4.2. Evaluation of modified genetic-based mechanism

To evaluate the performance of the modified genetic-based mechanism, the same crossover and mutation probabilities of 0.8 and 0.08, respectively, are maintained, as used in the previous analysis. This analysis focuses on how variations in population size, alongside these fixed probabilities, affect the solution time for processing IoT requests and the associated energy cost, with evaluations conducted over 30 and 60 generations. The results are shown in Fig. 10.

Fig. 10 highlights the effect of an increase in population size with different numbers of generations on energy cost and time taken to process the IoT requests. Fig. 10.a clearly shows that with an increase in population size, the cost of energy for processing the diverse IoT requests reduces and nears the energy cost resulting from the ILP-based framework. The increase in population size negatively impacts the solution time, as shown in Fig. 10.b.

The total energy cost incurred in processing 16 diverse incoming IoT requests is 99, which is higher than the ILP and genetic-based mechanism, but the time required is 1.8 s, which is comparatively lower than the ILP and genetic-based mechanism. This explains a trade-off for a balance between energy minimisation and the time taken to process the incoming IoT requests.

### 4.3. Evaluation of delay-aware mechanism

In evaluating the delay-aware mechanism, the processing of 16 incoming IoT requests incurred an energy cost of 110, as detailed in Table 3. The results show that the processing of these requests is remarkably quick, in 0.0018 s. Despite the increase in the cost, this rapid processing speed is crucial as it meets the QoS requirements essential for most IoT use cases.

### 4.4. Comparison of proposed mechanisms

This subsection compares the performance of the proposed mechanisms with each other and against the solution generated by the ILP solver. The focus is on evaluating their energy efficiency and execution time across varying IoT scenarios. The node selections for the considered scenario generated from all the mechanisms are listed in Table 2. The table summarises the QoS requirement of each IoT request and the node to which the IoT request is allocated by different mechanisms.

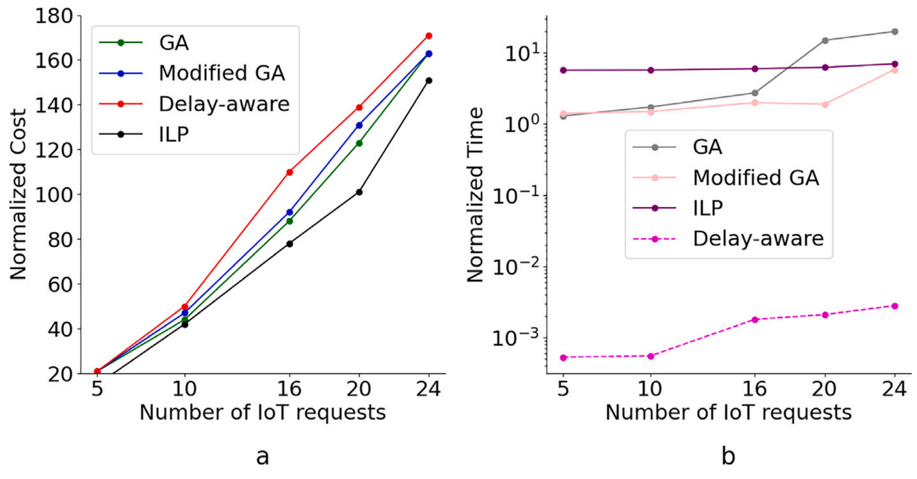

**Fig. 11.** Mechanism Sensitivity vs. Performance.

Table 3 summarises the cost of energy incurred in processing 16 diverse IoT requests and the time taken by each mechanism to execute the diverse IoT requests with the number of generations, population size, crossover probability, and mutation probability set to 30, 10, 0.8 and 0.08, respectively for genetic and modified genetic-based mechanisms.

The evaluation reveals that while the ILP framework offers the lowest energy cost, it requires a considerable amount of time to deliver solutions, particularly as the size of the network increases. On the other hand, it is evident from Table 3 that the delay-aware mechanism has significantly reduced the time to process the diverse incoming 16 IoT requests, but the energy cost incurred in processing the IoT requests has gone up to 110. For comparison, it is important to note that if one considers the scenario where all nodes in the IoT architecture are always active, the total cost would escalate to 177, substantially higher than costs incurred by any of the proposed mechanisms.

A visual representation of the sensitivity of the number of incoming IoT requests processed by the proposed mechanisms, on the energy cost incurred and the time required to execute the diverse IoT requests, is shown in Fig. 11. These comparisons are depicted in Figs. 11.a and Fig. 11.b. Fig. 11.a highlights the performance of energy cost against varying numbers of IoT requests across all proposed mechanisms. It can be observed that the ILP framework consistently achieves the lowest energy cost, and the delay-aware mechanism incurs the highest energy cost. From Fig. 11.b, it is evident that the delay-aware mechanism excels in minimising the time required to execute all incoming diverse IoT requests, which is a high priority for IoT requests incoming from use cases such as smart grid, eHealth, and autonomous vehicles. The results indicate that the delay-aware mechanism prioritises latency at the expense of higher energy consumption, particularly under high-load scenarios. This trade-off highlights the importance of exploring adaptive strategies, such as dynamic resource activation based on priority thresholds or energy-aware delay constraints, to balance energy efficiency with real-time performance.

Based on the results, the modified genetic-based mechanism is effective for applications requiring faster solutions at the expense of higher energy costs. This mechanism is particularly beneficial for structured IoT networks where quick response times are crucial. The delay-aware mechanism further accelerates processing, making it the fastest among all methods, although with a higher energy footprint. Therefore, for real-time decision-making scenarios, the delay-aware mechanism is preferred when speed is critical despite the slight increase in energy consumption. The evaluation demonstrates that the proposed mechanisms process IoT requests with an average latency of less than 1 ms, meeting the standards of real-time applications as outlined in previous research works and required as per standards.

The evaluation highlights the trade-offs between energy efficiency and execution time among the proposed mechanisms. While the ILP framework is optimal for energy minimisation, its practicality in time-sensitive IoT scenarios is limited due to its longer computation times. The genetic and modified genetic-based mechanisms provide a balanced approach, but the delay-aware mechanism stands out for its rapid processing capabilities, making it ideal for IoT applications where processing time is a critical factor.

The performance trade-offs presented in this study are based on an abstracted view of energy and resource availability across nodes. However, actual hardware capabilities, such as processor speed, memory limits, and battery capacity, can significantly impact the real-world applicability of these algorithms. These hardware constraints underline the importance of adaptive algorithmic designs that can scale down computational requirements without compromising energy efficiency.

In addition to improving energy efficiency, the proposed mechanisms contribute to environmental sustainability. By optimising the activation of nodes and servers across the edge–fog–cloud layers, these algorithms reduce power consumption and lower the carbon footprint of large-scale IoT networks. This reduction in energy use translates into fewer greenhouse gas emissions, reduced reliance on non-renewable energy sources, and lower cooling demands in data centres, boosting their positive environmental impact.

## 5. Conclusion

The work presented in this paper proposes genetic, modified genetic, and delay-aware-based mechanisms for node selection that can be used to minimise energy consumption when an edge–fog–cloud IoT architecture is used to support multiple real-time IoT use cases. These mechanisms aim to execute diverse incoming IoT requests with energy efficiency while adhering to specific network and application QoS constraints such as resource capacity, latency and connectivity, server and node availability, and server and node activation. The proposed mechanisms considered the cost of energy associated with executing diverse IoT requests across edge–fog–cloud layers. The proposed mechanisms were evaluated under a range of scenarios and IoT applications to assess their effectiveness.

The evaluation of the proposed mechanisms across various IoT applications and network scenarios revealed significant improvements in processing time compared to traditional ILP solvers. However, this performance gain involves a trade-off in energy consumption, particularly for delay-aware mechanisms prioritising latency-sensitive applications. The analysis also highlighted the impact of network size on both energy costs and solution time, providing valuable insights into the conditions under which each mechanism performs optimally. This work addresses the critical need for energy-efficient and real-time processing in IoT systems while demonstrating its scalability and applicability in real-world scenarios.

Overall, the results offered insights into the mechanisms and scenarios that can attain energy efficiency by using an edge–fog–cloud IoT architecture while meeting the QoS requirements of diverse IoT use cases. The proposed mechanisms not only optimise energy efficiency but also align with the goals of sustainable IoT practices. By enabling smarter resource allocation and reducing power consumption across IoT layers, these mechanisms offer a pathway to lower carbon emissions in large-scale IoT deployments. Future work will assess the potential environmental impact of these algorithms in real-world scenarios to further validate their role in promoting green and sustainable IoT networks. Future research will also focus on incorporating hardware-specific constraints, such as CPU/GPU performance, memory availability, and battery life, to validate the algorithms on real IoT devices. These considerations are critical for extending the practical applicability of the mechanisms in low-power and resource-constrained IoT environments. Additionally, enhancements to the delay-aware mechanism will aim to better balance energy efficiency and latency. Potential adaptations include introducing dynamic thresholds for energy usage based on application priorities and exploring hybrid mechanisms that combine delay awareness with energy-aware scheduling. These refinements will optimise the trade-off between energy consumption and latency in diverse IoT application scenarios.

The next paper will focus on event-driven simulation, which categorises the IoT requests coming from diverse IoT use cases into low, medium, and high priority and test the performance of the mechanism proposed in the current paper to validate its real-time performance in real-world scenarios.

## CRediT authorship contribution statement

**Rolden John Fereira:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Resources, Project administration, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Chathurika Ranaweera:** Validation, Supervision, Project administration, Formal analysis. **Kevin Lee:** Validation, Supervision, Project administration, Formal analysis. **Jean-Guy Schneider:** Validation, Supervision, Project administration.

## Declaration of competing interest

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

## Data availability

I have used simulated data, for network configurations, if required , I can share that.

## References

[1] L.S. Vailshery, Number of IoT connected devices worldwide 2019–2023, with forecasts to 2030, Technol. Telecommun. (2024) URL https://www.statista.com/statistics/1183457/iot-connec-ted-devices-worldwide/.

[2] S. Edirisinghe, O. Galagedarage, I. Dias, C. Ranaweera, Recent development of emerging indoor wireless networks towards 6G, Network 3 (2) (2023) 269–297, http://dx.doi.org/10.3390/network3020014, URL https://www.mdpi.com/2673-8732/3/2/14.

[3] C. Ranaweera, C. Lim, Y. Tao, S. Edirisinghe, T. Song, L. Wosinska, A. Nirmalathas, Design and deployment of optical X-haul for 5G, 6G, and beyond : Progress and challenges [invited], J. Opt. Commun. Netw. 15 (9) (2023) D56–D66, http://dx.doi.org/10.1364/JOCN.492334.

[4] Y. Liu, W. Yi, Z. Ding, X. Liu, O.A. Dobre, N. Al-Dhahir, Developing NOMA to next generation multiple access: Future vision and research opportunities, IEEE Wirel. Commun. 29 (6) (2022) 120–127.

[5] D. Peng, L. Sun, R. Zhou, Y. Wang, Study qos-aware fog computing for disease diagnosis and prognosis, Mob. Netw. Appl. 28 (2) (2023) 452–459.

[6] M. Boban, M. Giordani, M. Zorzi, Predictive quality of service (PQoS): The next frontier for fully autonomous systems, 2021, arXiv preprint arXiv:2109.09376.

[7] M. Erol-Kantarci, A. Caruso, Ultra-reliable and low-latency communications for the smart grid, in: Encyclopedia of Wireless Networks, Springer International Publishing, Cham, 2020, pp. 1427–1431, http://dx.doi.org/10.1007/978-3-319-78262-1_245.

[8] P. Hu, W. Chen, C. He, Y. Li, H. Ning, Software-defined edge computing (SDEC): Principle, open IoT system architecture, applications, and challenges, IEEE Internet Things J. 7 (7) (2020) 5934–5945, http://dx.doi.org/10.1109/JIOT.2019.2954528.

[9] C. Ranaweera, A. Nirmalathas, E. Wong, C. Lim, P. Monti, L.W. Marija Furdek, B. Skubic, C.M. Machuca, Rethinking of optical transport network design for 5G/6G mobile communication, IEEE Futur. Netw. Tech. Focus. 12 (2021).

[10] H. Gupta, A. Vahid Dastjerdi, S.K. Ghosh, R. Buyya, iFogSim: A toolkit for modeling and simulation of resource management techniques in the internet of things, edge and fog computing environments, Softw. - Pract. Exp. 47 (9) (2017) 1275–1296.

[11] R. Buyya, S.N. Srirama, Modeling and simulation of fog and edge computing environments using iFogSim toolkit, in: Fog and Edge Computing: Principles and Paradigms, Wiley Telecom, 2019, pp. 433–465, http://dx.doi.org/10.1002/9781119525080.ch17.

[12] H. Yang, A. Alphones, W.-D. Zhong, C. Chen, X. Xie, Learning-based energy efficient resource management by heterogeneous RF/VLC for ultra-reliable low-latency industrial IoT networks, IEEE Trans. Ind. Inform. 16 (8) (2019) 5565–5576, http://dx.doi.org/10.1109/TII.2019.2933867.

[13] C. Mouradian, F. Ebrahimnezhad, Y. Jebbar, J. Ahluwalia, S. Afrasiabi, R. Glitho, A. Moghe, An IoT platform-as-a-service for NFV-based hybrid cloud/fog systems, IEEE Internet Things J. 7 (7) (2020) 6102–6115, http://dx.doi.org/10.1109/JIOT.2020.2968235.

[14] A. Buzachis, A. Galletta, A. Celesti, M. Fazio, M. Villari, Development of a smart metering microservice based on fast Fourier transform (FFT) for edge/internet of things environments, in: 2019 IEEE 3rd International Conference on Fog and Edge Computing, ICFEC, Larnaca, Cyprus, 2019, pp. 1–6, http://dx.doi.org/10.1109/CFEC.2019.8733148.

[15] C. Ranaweera, P. Monti, B. Skubic, M. Furdek, L. Wosinska, A. Nirmalathas, C. Lim, E. Wong, Optical X-haul options for 5G fixed wireless access: Which one to choose? in: IEEE Conference on Computer Communications (INFOCOM) Workshops, Honolulu, HI, USA, 2018, pp. 1–2, http://dx.doi.org/10.1109/INFCOMW.2018.8406906.

[16] I. Dias, L. Ruan, C. Ranaweera, E. Wong, From 5G to beyond: Passive optical network and multi-access edge computing integration for latency sensitive applications, Opt. Fiber Technol., Mater. Devices Syst. 75 (2023) 103191, http://dx.doi.org/10.1016/j.yofte.2022.103191.

[17] N. Gupta, S. Sharma, P.K. Juneja, U. Garg, SDNFV 5G-IoT: A framework for the next generation 5G enabled IoT, in: 2020 International Conference on Advances in Computing, Communication Materials, ICACCM, Dehradun, India, 2020, pp. 289–294, http://dx.doi.org/10.1109/ICACCM50413.2020.9213047.

[18] R. Fereira, C. Ranaweera, J.-G. Schneider, K. Lee, Optimal node selection in communication and computation converged IoT network, in: 2022 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), Melbourne, Australia, 2022, pp. 539–547, http://dx.doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom57177.2022.00075.

[19] R. Fereira, C. Ranaweera, K. Lee, J.-G. Schneider, Energy efficient node selection in edge-fog-cloud layered IoT architecture, Sensors 23 (13) (2023) http://dx.doi.org/10.3390/s23136039, URL https://www.mdpi.com/1424-8220/23/13/6039.

[20] M. Zerifi, A. Ezzouhairi, A. Boulaalam, Overview on SDN and NFV based architectures for IoT environments: Challenges and solutions, in: 2020 Fourth International Conference on Intelligent Computing in Data Sciences, ICDS, 2020, pp. 1–5, http://dx.doi.org/10.1109/ICDS50568.2020.9268779.

[21] M.A. Bouras, F. Farha, H. Ning, Convergence of computing, communication, and caching in internet of things, Intell. Converg. Netw. 1 (1) (2020) 18–36, http://dx.doi.org/10.23919/ICN.2020.0001.

[22] Q.-D. Ho, Y. Gao, G. Rajalingham, T. Le-Ngoc, Smart grid communications network (SGCN), in: Wireless Communications Networks for the Smart Grid, Springer International Publishing, Cham, 2014, pp. 15–30, http://dx.doi.org/10.1007/978-3-319-10347-1_2.

[23] B. Appasani, D.K. Mohanta, A review on synchrophasor communication system: Communication technologies, standards and applications, Prot. Control. Mod. Power Syst. 3 (4) (2018) 1–17.

[24] A. Khan, A.I. Umar, S.H. Shirazi, W. Ishaq, M. Shah, M. Assam, A. Mohamed, Qos-aware cost minimisation strategy for AMI applications in smart grid using cloud computing, Sensors 22 (13) (2022) http://dx.doi.org/10.3390/s22134969, URL https://www.mdpi.com/1424-8220/22/13/4969.

[25] H. Taleb, A. Nasser, G. Andrieux, N. Charara, E. Motta Cruz, Wireless technologies, medical applications and future challenges in WBAN: A survey, Wirel. Netw. 27 (8) (2021) 5271–5295.

[26] S.A. Abdel Hakeem, A.A. Hady, H. Kim, 5G-V2X: Standardisation, architecture, use cases, network-slicing, and edge-computing, Wirel. Netw. 26 (8) (2020) 6015–6041.

[27] T. Rahman, X. Yao, G. Tao, H. Ning, Z. Zhou, Efficient edge nodes reconfiguration and selection for the internet of things, IEEE Sens. J. 19 (12) (2019) 4672–4679, http://dx.doi.org/10.1109/JSEN.2019.2895119.

[28] K. Lin, Y. Li, Q. Zhang, G. Fortino, AI-driven collaborative resource allocation for task execution in 6G enabled massive IoT, IEEE Internet Things J. (2021) http://dx.doi.org/10.1109/JIOT.2021.3051031, 1–1.

[29] U.M. Malik, M.A. Javed, S. Zeadally, S. u. Islam, Energy efficient fog computing for 6G enabled massive IoT: Recent trends and future opportunities, IEEE Internet Things J. (2021) http://dx.doi.org/10.1109/JIOT.2021.3068056, 1–1.

[30] Z. Han, Y. Yang, H. Ye, A deep reinforcement learning based multiple meta-heuristic methods approach for resource-constrained multi-project scheduling problem, in: 2022 7th International Conference on Intelligent Computing and Signal Processing, ICSP, Xi'an, China, 2022, pp. 26–29, http://dx.doi.org/10.1109/ICSP54964.2022.9778702.

[31] O. Namvar Gharehshiran, V. Krishnamurthy, G. Yin, A regret-based approach to non-stationary discrete stochastic optimisation, in: 2015 54th IEEE Conference on Decision and Control, CDC, Osaka, Japan, 2015, pp. 3959–3964, http://dx.doi.org/10.1109/CDC.2015.7402834.

[32] D.D. Lieira, M.S. Quessada, A.L. Cristiani, R. Immich, R.I. Meneguette, TRIAD: Whale optimisation algorithm for 5G-IoT resource allocation decision in edge computing, in: 2021 16th Iberian Conference on Information Systems and Technologies, CISTI, IEEE, Chaves, Portugal, 2021, pp. 1–6, http://dx.doi.org/10.23919/CISTI52073.2021.9476599.

[33] V. Srinadh, P.V.N. Rao, Implementation of dynamic resource allocation using adaptive fuzzy multi-objective genetic algorithm for IoT-based cloud system, in: 2022 4th International Conference on Smart Systems and Inventive Technology, ICSSIT, IEEE, Tirunelveli, India, 2022, pp. 111–118, http://dx.doi.org/10.1109/ICSSIT53264.2022.9716228.

[34] X. Huang, W. Zhang, J. Yang, L. Yang, C.K. Yeo, Market-based dynamic resource allocation in mobile edge computing systems, in: 2020 7th International Conference on Information Science and Control Engineering, ICISCE, IEEE, Changsha, China, 2020, pp. 829–833, http://dx.doi.org/10.1109/ICISCE50968.2020.00173.

[35] X. Chen, Energy efficient NFV resource allocation in edge computing environment, in: 2020 International Conference on Computing, Networking and Communications, ICNC, IEEE, Big Island, HI, USA, 2020, pp. 477–481, http://dx.doi.org/10.1109/ICNC47757.2020.9049765.

[36] X. Li, Z. Lian, X. Qin, W. Jie, Topology-aware resource allocation for IoT services in clouds, IEEE Access 6 (2018) 77880–77889, http://dx.doi.org/10.1109/ACCESS.2018.2884251.

[37] J. Moraes, N. Matni, A. Riker, H. Oliveira, E. Cerqueira, C. Both, D. Rosário, An efficient heuristic LoRaWAN adaptive resource allocation for IoT applications, in: 2020 IEEE Symposium on Computers and Communications, ISCC, IEEE, Rennes, France, 2020, pp. 1–6, http://dx.doi.org/10.1109/ISCC50000.2020.9219600.

[38] J. Kwak, Y. Kim, J. Lee, S. Chong, DREAM: Dynamic resource and task allocation for energy minimisation in mobile cloud systems, IEEE J. Sel. Areas Commun. 33 (12) (2015) 2510–2523, http://dx.doi.org/10.1109/JSAC.2015.2478718.

[39] J. Zhou, X. Zhang, Fairness-aware task offloading and resource allocation in cooperative mobile-edge computing, IEEE Internet Things J. 9 (5) (2022) 3812–3824, http://dx.doi.org/10.1109/JIOT.2021.3100253.

[40] D.S.L. Wei, K. Xue, R. Bruschi, S. Schmid, Guest editorial leveraging machine learning in SDN/NFV-Based networks, IEEE J. Sel. Areas Commun. 38 (2) (2020) 245–247, http://dx.doi.org/10.1109/JSAC.2019.2959197.

[41] J. Li, W. Liang, W. Xu, Z. Xu, X. Jia, W. Zhou, J. Zhao, Maximising user service satisfaction for delay-sensitive IoT applications in edge computing, IEEE Trans. Parallel Distrib. Syst. 33 (5) (2022) 1199–1212, http://dx.doi.org/10.1109/TPDS.2021.3107137.

[42] Z. Movahedi, B. Defude, A.M. Hosseininia, An efficient population-based multi-objective task scheduling approach in fog computing systems, J. Cloud Comput. 10 (1) (2021) 53, http://dx.doi.org/10.1186/s13677-021-00264-4.

[43] H.K. Apat, K. Bhaisare, B. Sahoo, P. Maiti, Energy efficient resource management in fog computing supported medical cyber-physical system, in: 2020 International Conference on Computer Science, Engineering and Applications, ICCSEA, IEEE, 2020, pp. 1–6, http://dx.doi.org/10.1109/ICCSEA49143.2020.9132855.

[44] P. Singh, R. Singh, Energy-efficient delay-aware task offloading in fog-cloud computing system for IoT sensor applications, J. Netw. Syst. Manage. 30 (1) (2022) 14, http://dx.doi.org/10.1007/s10922-021-09622-8.

[45] F. Hoseiny, S. Azizi, M. Shojafar, F. Ahmadiazar, R. Tafazolli, PGA: A priority-aware genetic algorithm for task scheduling in heterogeneous fog-cloud computing, in: IEEE INFOCOM 2021-IEEE Conference on Computer Communications Workshops, INFOCOM WKSHPS, IEEE, 2021, pp. 1–6, http://dx.doi.org/10.1109/INFOCOMWKSHPS51825.2021.9484436.

[46] M. Jia, J. Zhu, H. Huang, Energy and delay-ware massive task scheduling in fog-cloud computing system, Peer- To- Peer Netw. Appl. 14 (4) (2021) 2139–2155, http://dx.doi.org/10.1007/s12083-021-01118-1.

[47] M. Keshavarznejad, M.H. Rezvani, S. Adabi, Delay-aware optimisation of energy consumption for task offloading in fog environments using metaheuristic algorithms, Cluster Comput. (2021) 1–29, http://dx.doi.org/10.1007/s10586-020-03230-y.

[48] W. Chen, X. Qiu, T. Cai, H.-N. Dai, Z. Zheng, Y. Zhang, Deep reinforcement learning for internet of things: A comprehensive survey, IEEE Commun. Surv. Tutor. 23 (3) (2021) 1659–1692, http://dx.doi.org/10.1109/COMST.2021.3073036.

[49] C. Iwendi, P.K.R. Maddikunta, T.R. Gadekallu, K. Lakshmanna, A.K. Bashir, M.J. Piran, A metaheuristic optimisation approach for energy efficiency in the IoT networks, Softw. - Pract. Exp. 51 (12) (2021) 2558–2571, http://dx.doi.org/10.1002/spe.2797.

[50] B. Natesha, R.M.R. Guddeti, Meta-heuristic based hybrid service placement strategies for two-level fog computing architecture, J. Netw. Syst. Manage. 30 (3) (2022) 47, http://dx.doi.org/10.1007/s10922-022-09660-w.

[51] A. Lakhan, M.A. Mohammed, K.H. Abdulkareem, M.M. Jaber, J. Nedoma, R. Martinek, P. Zmij, Delay optimal schemes for internet of things applications in heterogeneous edge cloud computing networks, Sensors 22 (16) (2022) http://dx.doi.org/10.3390/s22165937, URL https://www.mdpi.com/1424-8220/22/16/5937.

[52] B.L. Nguyen, D.D. Nguyen, H.X. Nguyen, D.T. Ngo, Regret-matching learning-based task assignment in vehicular edge computing, 2022, CoRR.

[53] V. Jafari, M.H. Rezvani, Joint optimisation of energy consumption and time delay in IoT-fog-cloud computing environments using NSGA-II metaheuristic algorithm, J. Ambient. Intell. Humanised Comput. 14 (3) (2023) 1675–1698, http://dx.doi.org/10.1007/s12652-021-03388-2.

[54] A. Hassan, Meta-heuristic algorithms in IoT-based application: A systematic review, in: Proceedings of the Future Technologies Conference, Springer, 2023, pp. 104–116.

[55] Y. Shan, Y. Renping, J. Xin, Whale optimisation algorithm for energy efficient task allocation in the internet of things, Int. J. Adv. Comput. Sci. Appl. 14 (10) (2023) http://dx.doi.org/10.14569/IJACSA.2023.0141026.

[56] I.S.B.M. Isa, T.E.H. El-Gorashi, M.O.I. Musa, J.M.H. Elmirghani, Energy efficient fog-based healthcare monitoring infrastructure, IEEE Access 8 (2020) 197828–197852, http://dx.doi.org/10.1109/ACCESS.2020.3033555.

[57] Y.-p. Chen, D.E. Goldberg, An analysis of a reordering operator with tournament selection on a GA-hard problem, in: E. Cantú-Paz, J.A. Foster, K. Deb, L.D. Davis, R. Roy, U.-M. O'Reilly, H.-G. Beyer, R. Standish, G. Kendall, S. Wilson, M. Harman, J. Wegener, D. Dasgupta, M.A. Potter, A.C. Schultz, K.A. Dowsland, N. Jonoska, J. Miller (Eds.), Genetic and Evolutionary Computation — GECCO 2003, Springer, Berlin, Heidelberg, 2003, pp. 825–836.

[58] M.A. Mohammed, M.K. Abd Ghani, R.I. Hamed, S.A. Mostafa, M.S. Ahmad, D.A. Ibrahim, Solving vehicle routing problem by using improved genetic algorithm for optimal solution, J. Comput. Sci. 21 (2017) 255–262, http://dx.doi.org/10.1016/j.jocs.2017.04.003.

[59] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, V.B.S. Prasath, Choosing mutation and crossover ratios for genetic algorithms — A review with a new dynamic approach, Information 10 (12) (2019) http://dx.doi.org/10.3390/info10120390, URL https://www.mdpi.com/2078-2489/10/12/390.

[60] N.-E. Croitoru, High-probability mutation in basic genetic algorithms, in: 2014 16th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, Timisoara, Romania, 2014, pp. 301–305, http://dx.doi.org/10.1109/SYNASC.2014.48.

[61] N.-E. Croitoru, High probability mutation and error thresholds in genetic algorithms, in: 2015 17th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing, SYNASC, Timisoara, Romania, 2015, pp. 271–276, http://dx.doi.org/10.1109/SYNASC.2015.51.