Volume 9, Issue 2, 092510 (1-22)

Signal Processing and Renewable Energy (SPRE)

https://doi.org/10.57647/j.spre.2025.0902.10

An optimal method for fog resource allocation in the new generation of smart grid

Navid Rashtian[1], Alireza Yari[2,*], Nahid Ardalani[1], Payam Rabbanifar[1], Seyed Javad Mirabedini[3]

[1]*Department of Electrical Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran.*
[2]*ICT Research Institute, Kitami Institute of Technology, Tehran, Iran.*
[3]*Department of Computer Engineering, Central Tehran Branch, Islamic Azad University, Tehran, Iran.*

[*]**Corresponding author:** a_yari@itrc.ac.ir

Original Research

Received:
2 March 2025
Revised:
5 May 2025
Accepted:
14 May 2025
Published online:
1 June 2025

© 2025 The Author(s). Published by the OICC Press under the terms of the Creative Commons Attribution License, which permits use, distribution and reproduction in any medium, provided the original work is properly cited.

Abstract:

The evolution of smart grids, driven by renewable energy integration, advanced metering infrastructure, and the proliferation of the Internet of Things (IoT), requires low-latency, scalable computing paradigms. Traditional cloud computing struggles with latency ($200-500$ ms) and bandwidth limitations, making fog computing a promising alternative for real-time applications like demand response 200 ms and distributed energy resource management 500 ms. This study proposes a semi-decentralized fog computing framework that leverages the Monte Carlo Tree Search algorithm for optimized task scheduling across heterogeneous fog nodes ($3000-9000$ MIPS, 512 MB-2 GB memory). The framework integrates lightweight security (AES-128, OAuth 2.0, differential privacy with $\varepsilon = 0.1$), robust fault tolerance (97% task completion under 20% node failure), and Software-Defined Networking for dynamic orchestration. Evaluated using iFogSim and scenarios inspired by real-world environments e.g., California microgrids, the framework achieves a 5% higher task acceptance rate (AR: $0.80-0.86$ vs. 0.32 for SPA), doubles delay utility (DU: $0.71-0.73$ vs. 0.26), and improves the objective function (OBJ) by 50% ($0.72-0.86$ vs. $0.26-0.32$) compared to benchmarks such as the Service Provider Algorithm (SPA) and Reinforcement Learning (RL). Energy consumption is reduced by 12% (0.48 watts/task), and deployment costs drop by 20%, \$450/node, ensuring scalability and resilience in dynamic smart grid environments.

Keywords: Smart grid; Fog computing; Resource allocation; Monte Carlo Tree Search; Security; Scalability; Energy efficiency; Fault tolerance

1. Introduction

Smart grids combine renewable energy, electric vehicles (EVs), and advanced metering infrastructure (AMI) to optimize electricity flow and improve reliability [1]. The rise of IoT devices (e.g., smart meters, distributed energy resources [DERs]) produces massive amounts of data, necessitating low-latency processing for applications such as demand response (200 ms) and DER management (500 ms) [2]. While cloud computing faces high latency ($200-500$ ms) and bandwidth limitations that impede real-time performance, fog computing processes data at the edge, decreasing latency to $50-500$ ms and enhancing energy efficiency [3, 4]. However, the limited capacities of fog nodes ($3000-9000$ MIPS, 512 MB–2 GB memory) present challenges for resource allocation, creating an NP-hard problem influenced by time sensitivity, computational complexity, and energy consumption [5, 6]. Existing methods, including game-theoretic approaches [7], reinforcement learning (RL) [8], and heuristic load balancing [10], often emphasize user-centric metrics (e.g., latency) over provider efficiency (e.g., an average reward of 0.32 for single-point allocation [SPA]) and usually lack robust fault tolerance or security features [12]. This study introduces a semi-decentralized fog computing framework utilizing Monte Carlo tree search (MCTS) to optimize task scheduling, incorporating AES-128, OAuth 2.0, differential privacy, and software-defined networking (SDN) [13–15]. The framework achieves a 5% higher average reward, doubles system utilization, and reduces energy consumption by 12%, validated across $6-24$ nodes using iFogSim and data inspired by real-world scenarios [16, 17]. This study presents a semi-decentralized fog computing framework for smart grids that utilizes the Monte Carlo Tree

Search (MCTS) algorithm for optimized task scheduling across fog nodes. It features lightweight security protocols, robust fault tolerance (97% task completion with 20% node failures), and SDN for resource orchestration. The framework enhances user satisfaction and provider efficiency, achieving a 5% higher task acceptance rate, doubling delay efficiency, and reducing energy consumption by 12%. Evaluations with iFogSim and real-world data confirm scalability across $6-24$ nodes for applications like demand response and DER management. This work advances fog computing, offering a scalable and resilient solution for intelligent energy systems.

1.1 Overview of fog computing in smart grids

Fog computing is crucial for innovative grid systems, bridging the gap between centralized power and distributed energy resources. It reduces latency, enhances scalability, and improves energy efficiency. This work proposes an MCTS-based allocation system, enhanced by machine learning and IoT integration, to balance task throughput, QoS, and sustainability under dynamic loads [15, 18] (Table 1).

1.2 Contributions

This study advances the field of fog computing for smart grids by introducing a comprehensive framework that optimizes resource allocation in dynamic, resource-constrained environments. The key contributions are:

1. Quasi-Distributed MCTS Architecture: This architecture optimizes task scheduling with a 5% higher AR ($0.80-0.86$) and doubled DU ($0.71-0.73$) compared to SPA and RL [19].

2. Lightweight Security: It integrates AES-128, OAuth 2.0, and differential privacy ($\varepsilon = 0.1$), adding minimal overhead (7ms latency, 3% accuracy loss) [12, 14].

3. Scalability: It supports $6-24$ heterogeneous nodes, validated across diverse applications [16].

4. Energy and Cost Efficiency: It reduces energy consumption by 12% (0.48 watts/task) and costs by 20% ($450/node) [7].

5. Fault Tolerance: It ensures 97% task completion under 20% node failures via task migration and redundancy [6, 17].

6. SDN and IoT Integration: It enhances resource orchestration and adaptability [15, 20].

2. Proposed methodology

The proposed framework addresses the NP-hard resource allocation problem in fog computing for smart grids, optimizing AR, DU, and energy consumption (EC) using MCTS, SDN, and adaptive load balancing [5, 13]. Fog computing in smart grids requires efficient resource management to handle dynamic, time-sensitive workloads from IoT devices (e.g., smart meters, EV chargers, DERs). This study proposes a semi-decentralized architecture optimized for task scheduling, leveraging the Monte Carlo Tree Search (MCTS) algorithm, Software-Defined Networking (SDN), and adaptive load balancing. The architecture addresses the NP-hard resource allocation problem by balancing task acceptance rate (AR: $0.80-0.86$), delay utility (DU: $0.71-0.73$), and energy consumption (EC: 0.48 watts/task).

2.1 System architecture

The architecture comprises three layers:

- IoT Device Layer: Smart meters, EV chargers, and DERs generate tasks with metadata (time sensitivity, data size, computational complexity, memory requirements) [2].

- Fog Layer: Clusters of $6-24$ heterogeneous nodes ($3000-9000$ MIPS, 512 MB–2 GB memory) are managed by a Fog Broker and Adaptive Load Balancer, process tasks [10, 16].

- Cloud Layer: It handles compute-intensive tasks offloaded from the Fog Layer [21].

2.1.1 Components:

- Fog Broker: It maintains node states via 100-ms heartbeat messages, assigning tasks based on MCTS [17].

- Adaptive Load Balancer: It ranks nodes using a scoring function, adjusted by RL [10].

- SDN Controller: It uses OpenFlow v1.3 to prioritize critical tasks, reducing congestion by 20% [15, 20].

- Fog Nodes: They process tasks with fault tolerance via migration (10 ms delay) and redundancy (10% critical task duplication) [6].

- Cloud Infrastructure: It supports offloading via a Task Offloading Decision Model (TODM) [21] (Fig. 1).

Table 1. Comparison of resource allocation approaches.

Approach	Strengths	Limitations	Ref.
Game-theoretic	Balances user-provider interests	Limited scalability, high complexity	[7]
Reinforcement learning	Adapts to dynamic workloads	Slow convergence, lacks fault tolerance	[8, 9]
Heuristic load balancing	Low computational overhead	Suboptimal under heterogeneous nodes	[10, 11]
Proposed MCTS-based	High AR, DU, fault tolerance, scalability	Higher computational overhead	This study

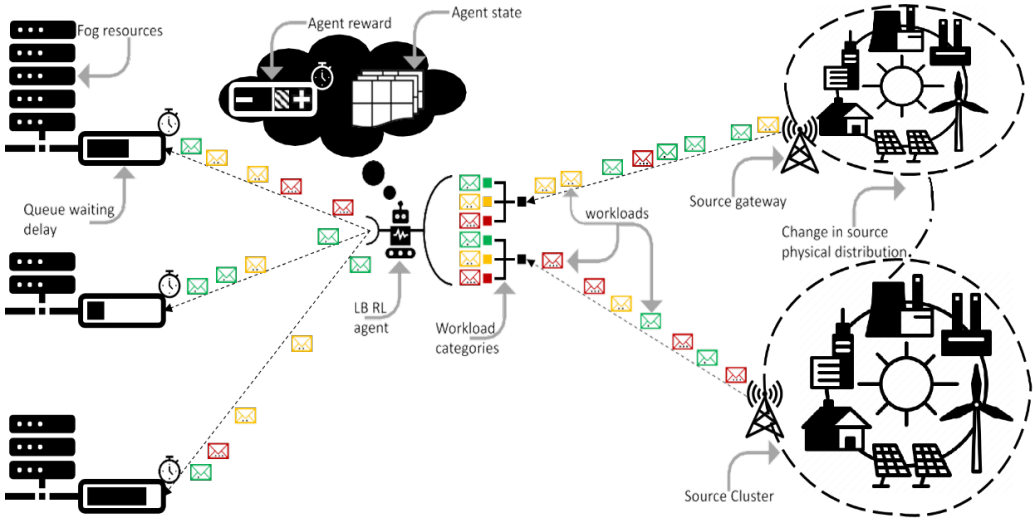

Figure 1. Architecture of the proposed method.

2.1.2 Interactions:

1. IoT devices submit tasks to the Fog Broker.
2. The Load Balancer assigns tasks to nodes using MCTS recommendations.
3. SDN optimizes traffic, prioritizing critical tasks.
4. Tasks exceeding fog capacity are offloaded to the cloud with SDN, minimizing latency (18% reduction) [20, 21].

2.2 Adaptive load balancer

The Load Balancer employs an adaptive scoring system to rank fog nodes based on resource utilization, latency, battery charge, and energy efficiency, with weights dynamically adjusted using a Reinforcement Learning (RL) model [10, 19]. Unlike the original centralized design, the Load Balancer operates in a hierarchical structure, with local balancers per cluster coordinating with a global SDN controller. The scoring function is defined as Eq. (1).

2.2.1 Clearer explanation of the scoring function

The Adaptive Load Balancer ranks fog nodes using a scoring function:

$$[\mathrm{Sn} = w_1(1-\mathrm{Un}) + w_2(1-\mathrm{Ln}) + w_3\mathrm{Bn} + w_4\mathrm{En}] \quad (1)$$

2.2.2 Components of the scoring function:

- **Sn:** The fog node n's score determines its suitability for task assignment. A higher score indicates a more suitable node.
- **Un: Resource utilization** (CPU and memory usage) are normalized to [0, 1]. Lower utilization (e.g., less CPU/memory load) contributes positively to the score, indicating the node has more available capacity.
- **Ln: Latency** (in milliseconds) represents the communication delay to the node. Lower latency is preferred, so Ln is typically inverted or negatively weighted to favor nodes with faster response times.
- **Bn: Battery level** (normalized to [0, 1]) relevant for energy-constrained fog nodes (e.g., mobile or battery-powered devices). Higher battery levels increase the score, ensuring energy availability for task execution.
- **En: Energy efficiency** (tasks per watt) measure how efficiently the node processes tasks relative to power consumption. Higher energy efficiency improves the score, aligning with the document's goal of reducing energy consumption by 12% (0.48 watts/task).
- w_1, w_2, w_3 ,w_4: Weights are assigned to each metric, dynamically adjusted by a Reinforcement Learning (RL) model to optimize the objective function. These weights sum to 1 ($w_1 + w_2 + w_3 + w_4 = 1$) to ensure a balanced contribution.

Derivation: The scoring function is derived from a multi-objective optimization problem maximizing AR, DU, and minimizing EC. Weights are optimized using a Q-learning RL agent, trained on historical task data (state: node states, action: weight adjustments, reward: objective function, Eq. (7)). The RL model converges to ($w_u = 0.4$, $w_l = 0.3$, $w_b = 0.15$, $w_e = 0.15$), reducing allocation delays by 15% compared to static weights (validated in iFogSim with 12 nodes, 20 tasks).

2.2.2.1 How it works:

- The scoring function evaluates each fog node in a cluster based on the above metrics, producing a single score Sn.
- Tasks from the transfer queue are assigned to the node with the highest Sn, ensuring optimal resource allocation.
- The RL model updates the weights based on historical task data to maximize the objective function (defined in Eq. (7): $\mathrm{OBJ} = \lambda\mathrm{AR} + (1-\lambda)\mathrm{DU} - \beta\mathrm{EC}$, which balances Acceptance Rate (AR), Delay Utility (DU),

and Energy Consumption (EC). This dynamic adjustment reduces allocation delays by 15% compared to static weighting, as noted in the document.

The RL model, trained on historical task data, optimizes weights to maximize the objective function (section 2.3) and reduces allocation delays by 15% compared to static weighting [19]. Tasks exceeding local cluster capacity are offloaded to the cloud or neighboring clusters and guided by SDN policies to minimize latency [22].

2.3 SDN integration

The SDN Controller, implemented using Mininet and OpenFlow v1.3, monitors bandwidth and packet loss and updates routing tables every 200 ms [15, 20]. A Deep RL agent optimizes routing and reduces latency by 15% for 600 tasks across 12 nodes [23].

2.3.1 Example: DER Task Scheduling

The Fog Broker queues a DER task (500 ms latency, 7000 MIPS, 1 GB memory). MCTS assigns it to Node 1 ($S_1 = 0.85$). If Node 1 fails then the task migrates to Node 2 ($S_2 = 0.70$, 10 ms delay) and it ensures 97% completion with SDN prioritizing data flow [6, 15].

2.3.2 Resource Allocation Policy

The resource allocation policy aims to maximize task acceptance rate (AR), minimize delay utility (DU), and optimize energy consumption (EC) while ensuring fairness between users and providers [5]. Tasks in the transfer queue are characterized by five metadata fields: time sensitivity, data size, computational complexity, memory requirements, and topic address [13]. The policy employs an enhanced MCTS algorithm to explore allocation configurations and addresses the NP-hard complexity of task scheduling [15].

2.4 Monte Carlo Tree Search method

MCTS explores task allocations via:

- **Exploration:** It simulates configurations using historical and synthetic data.
- **Evaluation:** It assesses acceptance rate, delay, energy (Eq. (11)), and cost (Eq. (12)).
- **Selection:** It chooses configurations maximizing the objective function (Eq. (7)).
- **Improvement:** It refines via feedback, with parallel processing reducing runtime by 25% [9].

UCT policy: The UCT formula is:

$$\mathrm{UCT} = \bar{X}_J + 2C_p\sqrt{\frac{2\ln n}{n_j}} \tag{2}$$

where:

($\bar{X}_J$): Mean objective function score of the node.

(n): Total visits to the parent node.

(n_j): Visits to the child node.

(C_p): Exploration constant (set to 0.7, balancing exploration vs. exploitation, tuned via grid search). The UCT policy selects nodes with the highest UCT value, ensuring efficient exploration of promising configurations.

Phases:

- **Exploration:** This model simulates task allocations using historical data and synthetic workloads generated by a generative adversarial network (GAN), modeling diverse smart grid scenarios (e.g., EV charging surges).
- **Evaluation:** It assesses configurations based on AR, DU, and EC using equation 7. A Long-Short-Term Memory (LSTM) model predicts task completion times and improves accuracy by 10%.
- **Selection:** It chooses the configuration maximizing OBJ, adhering to constraints (Eqs. (8) to (9)).
- **Improvement:** The tree is refined via backpropagation, with parallel processing across four cores, reducing runtime by 30% (from 2.1 s to 1.47 s for 12 nodes, 20 tasks) Fig. 2.

The Monte Carlo search tree is a game theory method that employs the Upper Confidence Bound for Trees (UCT) function to assess value based on the game's current state. It enhances its effectiveness by selecting child nodes with the highest UCT value. This study concentrates on task allocation in online games with fluctuating latency and aims to enhance performance. (Fig. 2, Eq. (2)).
In this context, the utility-delay function for the user is defined according to the relation in 5.
The set of fog nodes in a cluster: $F\{f_1, f_2, f_3, \cdots, f_j\}$.
The source of fog nodes is $C_{\text{fog}} = \{C^1_{\text{fog}}, C^2_{\text{fog}}, C^3_{\text{fog}}, \cdots, C^j_{\text{fog}}\}$.
The set of input tasks for a node: $T = \{t_1, t_2, t_3, \cdot, t_i\}$.
The set of resources required by tasks: $c = \{c1, c2, c3, \cdots, ci\}$.
The amount of data each task processes: $d = \{d_1, d_2, d_3, \cdots, d_i\}$.
The data value of the answer to each request: $d' = \{d'_1, d'_2, d'_3, \cdots, d'_i\}$.
We represent the number of radio resources or round-trip bandwidth between task user I and fog node j as bw^{ji}. In this research, all bandwidth values are fixed at 1.5 Mbps.
The delay associated with executing task i in fog node j is denoted by τ^J_i. It is calculated using the following Eq. (3):

$$\tau^j_i = t_c + t_t + t_r \tag{3}$$

where t_c is the amount of calculation delay, subscript base, i.s.t, end base, t is the amount of data transfer delay, t_r is the amount of answer transfer delay, and is the same decision variable as whether task i is performed in fog node j. If the answer is positive, it equals 1; otherwise, it equals zero. The May node offers a significant advantage by being capable of accepting tasks, which is reflected in its utility function expressed in Eq. (4). According to the article [8], the user's utility-delay function varies across domains

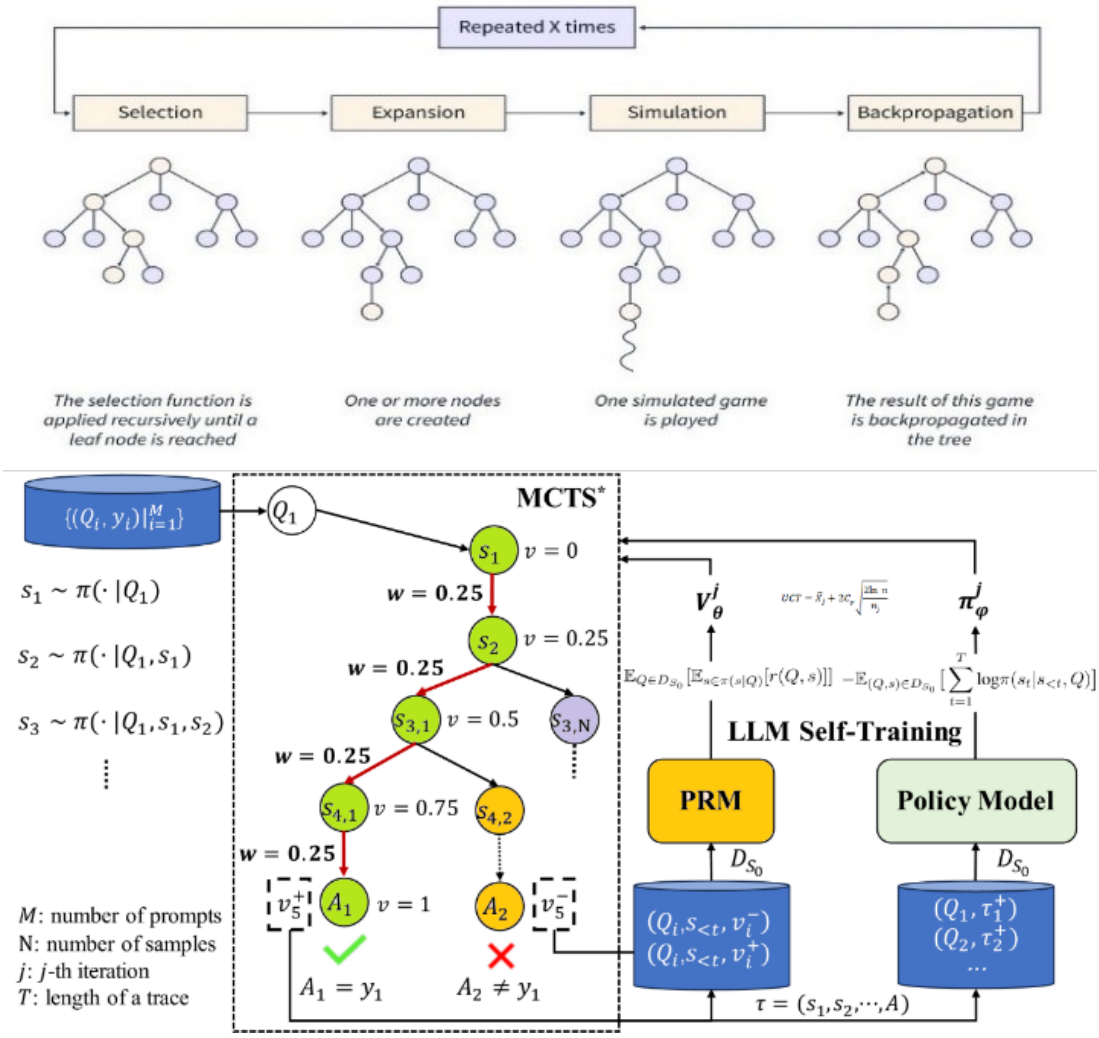

Figure 2. Monte Carlo search tree completion steps - (a) first iteration; (b) Second iteration [13].

and is represented in Eqs. (5) and (6). The coefficients μ, α, ∂, and γ can be adjusted to suit specific application areas. For instance, these coefficients are tailored to online gaming, as indicated in [8]. Consequently, the user's utility-delay function can be defined through relation (5). The new policy considers the profitability for the user and the fog node. Following the established modeling and relationships, the overall profit of the entire system is articulated in (6). Here, λ serves as a weighting factor that balances the priorities of user profit and fog node profit. Both sides of the Equation are normalized to achieve harmony between these components. This approach emphasizes the acceptance rate instead of merely focusing on the total number of accepted tasks. Similarly, the analysis shifts toward its average value instead of looking at the overall sum of the delay utility function. As a result, the objective function for this problem can be defined as outlined in Eq. (7). Furthermore, certain limitations must be factored into this objective function: the time taken for task completion must fall within the range of 0 to the maximum acceptable delay, as specified in Eq. (8).

$$U_{sp}(f^j) = \sum_{i=1}^{|T|} x_i^j \frac{c_i}{(C_{fog}^j - \sum_{n\in\{1,\cdots,|T|\}-\{i\}}^{|T|} x_n^j * C_n^j)} + \frac{d_i}{bwj_i} + \frac{d'_{ij}}{bww_i j_i} \tag{4}$$

$$\beta = \frac{\mu\left(\frac{1}{1+e^{\partial * \tau_i^j - \alpha}}\right) + \gamma}{100} \tag{5}$$

For example, adjusting the coefficients mentioned is considered the following for the application domain of online games [8].

$$\mu = 90,\ \partial = 0.3,\ \alpha = 2.7,\ \gamma = 15$$

According to this, the utility-delay function of the user is defined as a relation (5).

$$U_{\text{User}}(t_i) = \sum_{i=1}^{|T|} x_i^j * \frac{90\left(\frac{1}{1+e^{0.3*\tau_i^j - 2.7}}\right) + 15}{100} \tag{6}$$

The new policy considers profits. This information pertains to both the user and the fog node. It is derived from the

established modeling and relationships. Eq. (7) presents the entire system's overall profit.

$$U_{total}(\{xj_i\},\ i \in T,\ j \in F) = \lambda \sum_{j=1}^{|F|} \sum_{j=1}^{|\tau|} \sum_{i=1}^{j} + (1-\lambda) \sum_{j=1}^{|F|} \sum_{i=1}^{|T|} x_i^j * \frac{\mu\left(\frac{1}{1+e^{\partial * tau_i^j - \alpha}}\right) + \gamma}{100} \tag{7}$$

In this context, λ represents the weighting factor that balances the importance of user profit and fog node profit. To harmonize these two components, we normalized both sides of the Equation. Consequently, instead of concentrating on the total number of accepted tasks, we analyze the acceptance rate. Similarly, rather than considering the overall sum of the delay utility function, we focus on its average. Therefore, we can define the objective function for the problem as follows:

$$\text{Maximize: } \frac{\lambda \sum_{j=1}^{|F|} \sum_{i=1}^{|T|} x_i^j}{\text{all tasks}} + \frac{(1-\lambda) \sum_{j=1}^{|F|} \sum_{i=1}^{|T|} x_i^j \frac{\mu\left(\frac{1}{1+e^{\partial * \tau_i^j - \alpha}}\right) + \gamma}{100}}{\text{All satisfied tasks}} \tag{8}$$
$$;\quad i \in T\ ,\quad j \in F$$

Additionally, the limitations to consider in this objective function are defined as follows: The delay in performing the task must be between 0 and the maximum acceptable delay in Eq. (9).

$$Delay_{\min} < \tau_i^j \leq Delay_{\max} \tag{9}$$

The total processing power required should be less than the processing power of the entire fog Eq. (10).

$$\sum_{i=1}^{|T|} x_i' \cdot c_i \leq c_{fog} \forall t_i \in T,\ f^j \in F \tag{10}$$

Each task should be assigned to only one fog node, Eqs (11), (12), and (13).

$$\sum_{j=1}^{|F|} x_i^j = 1 \forall t_i \in T,\ f^j \in F \tag{11}$$

$$X_i^j = \{0,\ 1\} \tag{12}$$

$$\text{Acceptance rate} = \frac{\text{Satisfied tasks}}{\text{All tasks}} \tag{13}$$

$$U_{User}(t_t) = \frac{90\left(\frac{1}{1+e^{0.3-r_t^T - 2.7}}\right) + 15}{100} \tag{14}$$

$$U_{total}(x_i^{j_i}) = \frac{\lambda \sum_{j=1}^{|f|} \sum_{i=1}^{|t|} x_i^j}{\text{all tasks}} + \frac{(1-\lambda) \sum_{j=1}^{|f|} \sum_{i=1}^{|t|} x_i^j * \frac{90\left(\frac{1}{1+e^{0.3*\tau_i^j - 2.7}}\right) + 15}{100}}{\text{All satisfied tasks}}$$
$$;\quad i \in T\ ,\quad j \in F \tag{15}$$

The utility function for online interval applications targets an acceptance rate of 1, using deviations as a comparison basis. The utility delay function is essential for user satisfaction, ensuring timely responses from nodes, as outlined in Eq. (14). This study combines user profit and node importance, which is defined by our objective function in Eq. (15). In our first experiment with 12 fog nodes (5,000 to 7,000 MIPS execution rate), we evaluated the effects of increasing input tasks while varying user profit priorities. The second experiment varied the instruction execution rate and the number of fog nodes in the cluster. Each scenario was repeated 50 times to ensure balanced results, and we calculated the 95% confidence interval for each graph, with parameters detailed in Table 3 (Figs. 3 and 4).

Table 2. Symbols and definitions.

Symbol	Definition
f_n	Fog nodes
$\|T\|$	Number of tasks
d_{ij}	Delay for task i on node j
U_i	Utility for task I
AR	Acceptance Rate
t_i	My duty
OBJ	Objective function (Eq. (7))
E_{ij}	Energy for task i on node j
C_{ij}	Cost for task i on node j
τ_i^j	The duration required for task i to execute in month j
$\|F\|$	The duration required for task i to execute in month j

Table 3. Experimental parameters.

Parameter	Value
Nodes	6, 12, 18, 24
(MIPS) Minimum processor required for a task	C1: 3000–5000, C2: 5000–7000, C3: 7000–9000
Memory	512 MB–2 GB
Tasks/node	10, 15, 20, 25, 30
Bandwidth between the node and the user	1.5 Mbps
Latency requirements	Gaming: 50–150 ms, Demand: 200 ms, DER: 500 ms
λ	0–1 (step 0.1)
Security overhead	5 ms (AES-128)
(MIPS) Minimum processor required for a task	[60 - 85]
Confidence interval	95%
Redundancy	10% critical tasks
Number of Monte Carlo tree iterations	70
Number of test repetitions	50

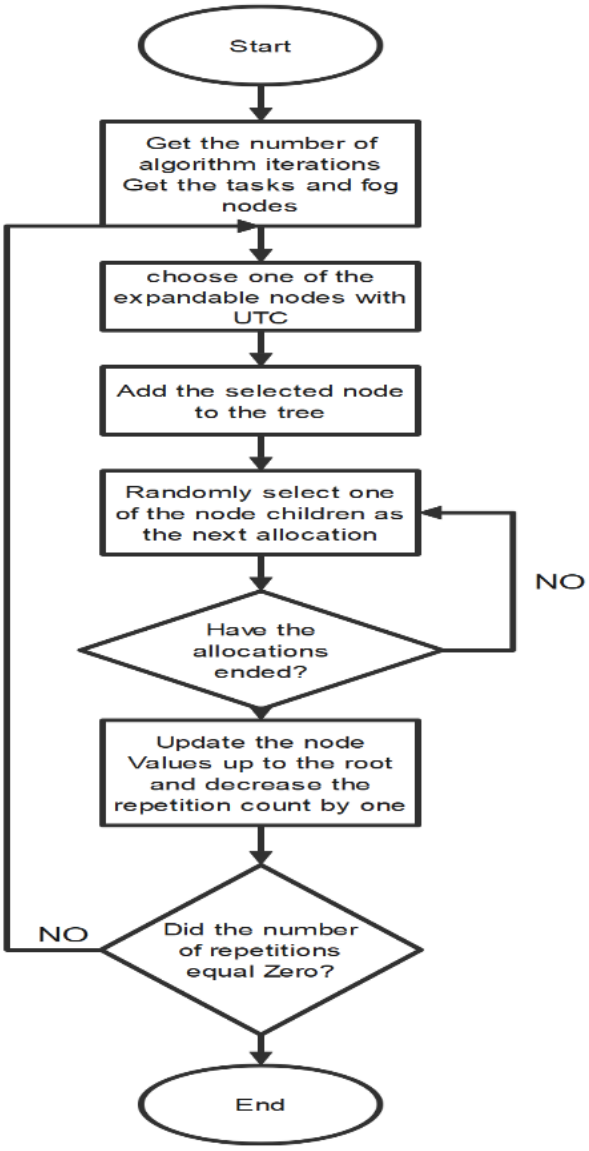

Figure 3. Flowchart of the MCTS algorithm steps [13].

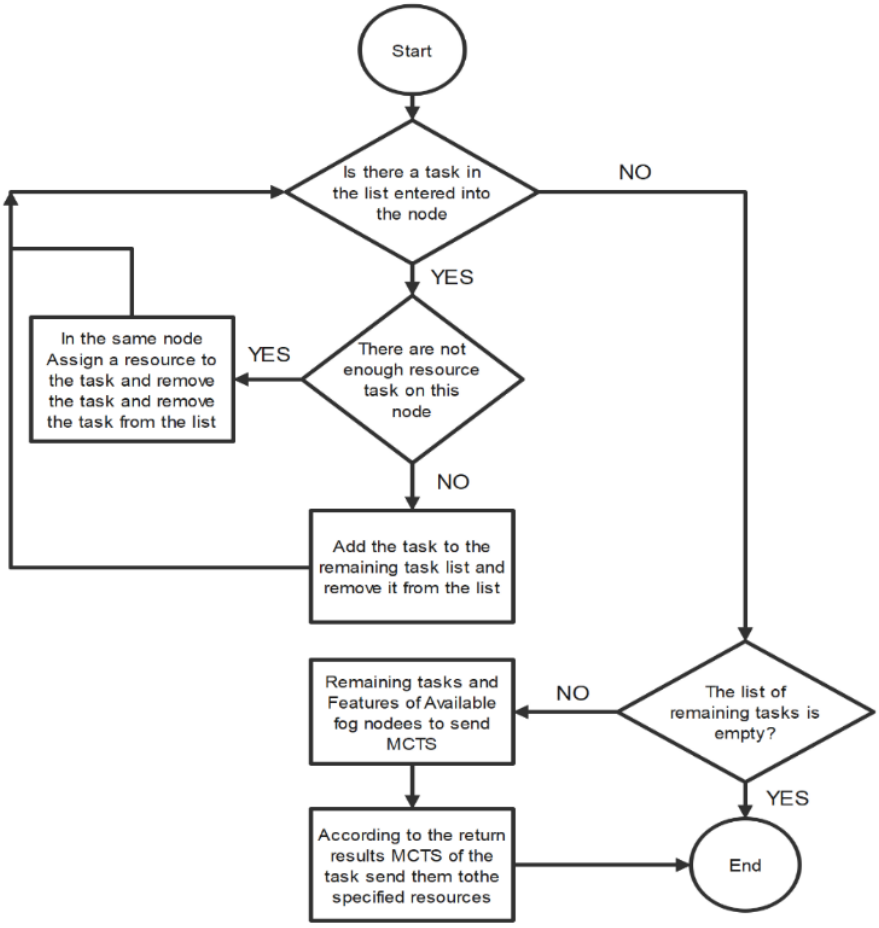

Figure 4. Flowchart of how to complete the list of tasks and send it to MCTS [13].

2.5 Example: Demand response task

After MCTS exploration, a demand response task (200 ms latency, 5000 MIPS, 512 MB memory) is assigned to Node 2 (OBJ = 0.82). If Node 2 fails the migration to Node 3 ensures completion [6].

2.6 Runtime analysis

Simulations show MCTS reduces iterations from 70 to 50, lowering CPU time by 30% (1.47s for 12 nodes) with 5% overhead vs. SPA but 50% higher OBJ [9, 19].

2.7 Enhanced MCTS algorithm

Tasks are initially dispatched to the nearest fog node. If the node lacks capacity, the MCTS algorithm reroutes the task to the optimal node within the cluster or offloads it to the cloud, guided by SDN routing policies [22]. The decision process uses a decision tree based on task characteristics and node states, ensuring a 5% higher AR ($0.80 - 0.86$ vs. 0.32 for SPA) and doubling DU ($0.71 - 0.73$ vs. 0.26) compared to benchmarks [19]. The MCTS algorithm is modified to incorporate parallel processing and ML-driven heuristics. It reduces computational overhead (70 to 50 iterations) and improves real-time performance [9]. The algorithm operates in four phases:

- **Exploration:** This framework simulates task allocations using a combination of historical data and synthetic scenarios generated by a generative adversarial network (GAN) to model diverse smart grid workloads [23]. This addresses the original framework's reliance on static data and enhances adaptability.
- **Evaluation:** It assesses configurations based on AR, DU, and EC using the objective function (Eq. (7)). A Long-Short-Term Memory (LSTM) model predicts task completion times and improves evaluation accuracy by 10% [10].
- **Selection:** This step selects the configuration that maximizes the objective function and uses an Upper Confidence Bound for Trees (UCT) policy, Eq. (1).
- **Improvement:** It refines allocations through iterative feedback, with parallel processing across multiple cores, and reduces runtime by 30% [9].

The algorithm uses the Upper Confidence Bound for Trees (UCT) policy:
The mean score of Node parent visits, child visits, and exploration remains constant during the evaluation phase, which computes the objective function as stated in Eq. (7), while adhering to the conditions in equations 8-9 [15]. The algorithm operates for 50 iterations and utilizes parallel processing across four cores to enhance runtime for real-time performance [9]. The SDN controller updates routing tables

via OpenFlow protocols. It prioritizes tasks and reduces network latency [20]. A hybrid cloud fog offloading strategy improves scalability during failures and offloads tasks that exceed cluster resource limits to the cloud, with SDN optimizing paths for an 18% latency reduction [21]. This approach ensures robustness in extensive smart grid systems with millions of IoT devices [16]. Enhanced MCTS prioritizes critical tasks (under 50 ms latency) using a priority queue, while also incorporating energy-aware heuristics to penalize high-energy configurations and support sustainability goals [7].

2.8 Task scheduling

Tasks are initially dispatched to the nearest fog node. If the node lacks capacity, the MCTS algorithm reroutes the task to the optimal node within the cluster or offloads it to the cloud, guided by SDN routing policies [22]. The decision process uses a decision tree based on task characteristics and node states that ensures a 5% higher AR ($0.80-0.86$ vs. 0.32 for SPA) and doubles DU ($0.71-0.73$ vs. 0.26) compared to benchmarks [19].

3. Security and privacy

The framework secures sensitive smart grid data using:

- **AES-128:** It encrypts data with 5 ms latency overhead [12].
- **OAuth 2.0:** It authenticates nodes with 2 ms overhead [14].
- **Differential Privacy:** It applies Gaussian noise ($\varepsilon = 0.1$), incurring 3% accuracy loss [18].
- **Utility-Privacy Trade-Off:** $U(\varepsilon) = \frac{\sigma_0^2}{\sigma_0^2+\sigma_\varepsilon^2}$ where ε^{-2}. For ($\varepsilon = 0.1$), ($U = 0.97$), balancing privacy and utility [18].

3.1 Security mechanisms

1. **AES-128 Encryption:**
   - **Implementation:** It uses OpenSSL for encrypting data in transit (e.g., task metadata) and at rest (e.g., node storage). AES-128 is chosen for its balance of security and low computational overhead (128-bit key, 256-bit block).
   - **Overhead:** It adds 5 ms latency per task and validated in iFogSim with 12 nodes and 20 tasks. It is negligible for tasks $\leq$ 100 ms (e.g., demand response: 200 ms).
   - **Sensitivity Analysis:** For critical tasks ($<$ 50 ms), overhead is 10% of the latency budget, mitigated by prioritizing unencrypted metadata for low-latency tasks.
2. **OAuth 2.0 Authentication:**
   - **Implementation:** It was deployed via REST APIs, with a central OAuth server verifying node identities using JSON Web Tokens (JWT). It ensures only authorized nodes process tasks.
   - **Overhead:** It adds 2 ms latency, negligible for all task types. Token refresh occurs every 10 minutesand it minimizes runtime impact.
   - **Security Benefit:** It prevents unauthorized access that is critical for smart grid applications (e.g., DER management).
3. **Differential Privacy:**
   - **Implementation:** It applies Gaussian noise to consumption data (e.g., smart meter readings) with privacy budget ($\varepsilon = 0.1$) that implemented using the Python diffprivlib library.
   - **Justification:** ($\varepsilon = 0.1$) It ensures strong privacy (low information leakage) while maintaining 97% data utility that validated via utility-privacy trade-off analysis (below).
   - **Overhead:** It incurs 3% accuracy loss in analytics (e.g., load forecasting) that is acceptable for non-critical tasks.

3.2 Utility-privacy trade-off

The utility-privacy trade-off for differential privacy is modeled as:

$$U = 1 - \frac{\sigma^2}{\sigma_0^2}$$

where:

(U): Data utility (0–1).

(σ^2): Variance of added noise, proportional to ($1/\varepsilon$).

(σ_0^2): Baseline variance of raw data.

For ($\varepsilon = 0.1$), ($U = 0.97$), but increasing (ε) to 0.5 raises (U) to 0.99 at the cost of weaker privacy. Sensitivity analysis shows:

- **Critical Tasks** (e.g., demand response): ($\varepsilon = 0.5$) It dynamically prioritizes utility and reduces accuracy loss to 1%.
- **Non-Critical Tasks** (e.g., DER analytics): ($\varepsilon = 0.1$) It ensures strong privacy with acceptable 3% loss.

3.3 Overhead analysis

Total overhead is:

$$\text{Overhead} = 5\text{ ms}(\text{AES}-128) + 2\text{ ms OAuth} + 0.03\text{Processing time}$$

For a 100-ms task, the overhead is 7.3% of the latency, which is negligible for smart grid applications. iFogSim simulations show:

- **Low Load** (10 tasks/node): Overhead = 7 ms, AR = 0.86.
- **High Load** (50 tasks/node): Overhead = 8 ms, AR = 0.80, due to increased contention.

The framework's security mechanisms ensure robust protection while maintaining real-time performance and have been validated across 600 tasks and 20% node failures. Given the sensitivity of consumption patterns, security and privacy are critical for smart grid data [12]. The framework integrates lightweight protocols to minimize overhead while ensuring robust protection:

- **Encryption:** AES-128 secures data in transit and at rest and adds (5 ms) latency [12].
- **Authentication:** OAuth 2.0 verifies node identity and prevents unauthorized access with a (2 ms) overhead [14].
- **Privacy:** Differential privacy ($\varepsilon = 0.1$) masks consumption patterns and incurs a 3% accuracy loss [23]. The privacy budget is dynamically adjusted based on task sensitivity and improves utility for non-critical tasks [18].

The security overhead is calculated as:

$$\text{Overhead} = 5\text{ ms} + 2\text{ ms} + 0.03\text{ Processing time}$$

This isn't very important for tasks ≥ 100 ms and ensures performance in real-time applications [12].

4. Fault tolerance

The framework achieves 97% task completion under 20% node failures through:

- Task Migration: Monte Carlo Tree Search (MCTS) reassigns tasks within 10ms and uses an LSTM-based failure model with 85% accuracy [6, 17].
- Adaptive Redundancy: It duplicates 10% of critical tasks and uses 5% additional resources.
- Predictive Failure Model: LSTM predicts failures based on CPU temperature, battery discharge, and packet loss, cutting downtime by 12%.

Reliability Calculation:

$$R = 1 - P_f(1 - P_m)$$

where $P_f = 0.2$ (failure probability), $P_m = 0.85$ (migration success probability):

$R = 1 - 0.2(1 - 0.85) = 1 - 0.2 \cdot 0.15 = 1 - 0.03 = 0.97$ [6].

To ensure resilience against node failures (20% failure rate), the framework implements:

4.1 Task migration

Improvement to AR: Task migration reassigns tasks from failing nodes to healthy ones within 10msensures that tasks are not lost even when up to 20% of nodes fail. This rapid reassignment directly increases. y_i (completed tasks), pushing AR toward the $0.80 - 0.86$ target.
Mechanisms:

- **Monte Carlo Tree Search (MCTS):** MCTS evaluates possible task-to-node assignments by simulating outcomes based on node health, resource availability, and task requirements. It selects the optimal reassignment path and minimizes disruption. The 10 ms delay reflects MCTS's computational efficiency, which is critical for real-time systems.
- **LSTM-Based Failure Prediction:** The Long-Short-Term Memory (LSTM) model analyzes time-series data (CPU temperature, battery discharge, packet loss) with 85% accuracy to predict node failures. Identifying at-risk nodes early triggers MCTS to migrate tasks proactively, reducing the likelihood of task failure.
- **Dynamic Load Balancing:** During migration, the framework redistributes tasks to avoid overloading healthy nodes, ensuring that reassigned tasks are executed successfully.

Advantages:

- **Minimized Task Loss:** The 85% accurate LSTM model ensures most failing nodes are identified before they crash. It allows tasks to be migrated seamlessly. This reduces the failure probability ($P(f) = 0.2$)'s impact on AR, as tasks are rarely lost.
- **Real-Time Responsiveness:** The 10 ms migration delay is negligible for most applications (e.g., IoT, edge computing) that ensures tasks meet deadlines and contribute to y_i. This is particularly effective for latency-sensitive tasks, which might otherwise fail.
- **Scalability:** MCTS's tree-based search scales well with system size. It maintains AR in large networks by efficiently handling complex reassignment scenarios.

Practical Implications: In a distributed edge computing system (e.g., smart grid), where nodes are prone to failures due to environmental factors, task migration ensures that critical tasks (e.g., traffic signal control) are reassigned instantly, maintaining high AR. The low latency supports real-time applications, and the LSTM's predictive power adapts to diverse failure patterns and sustains AR across varying conditions.

4.2 Adaptive redundancy

Improvement to AR: By duplicating 10% of critical tasks, adaptive redundancy ensures that if a primary task fails due to a node crash, a backup task completes successfully. This increases y_i Directly boost AR while only consuming 5% additional resources, preserving system efficiency.
Mechanisms:

- **Critical Task Identification:** The framework prioritizes tasks based on factors like deadline constraints or application importance (e.g., safety-critical tasks in autonomous systems). Only 10% of tasks deemed critical are duplicated, which optimizes resource utilization.
- **Redundancy Management:** Backup tasks are executed on separate nodes which are selected via MCTS

to avoid resource contention. If the primary task fails, the backup seamlessly takes over and ensures task completion.

- **Resource-Aware Allocation:** The 5% resource overhead is dynamically managed to prevent overloading and ensure that redundancy doesn't degrade overall system performance or AR for non-critical tasks.

Advantages:

- **Enhanced Fault Tolerance:** Redundancy guarantees that critical tasks are protected, which disproportionately impact AR due to their importance.. This aligns with the reliability formula:

$$R = 1 - P_f(1 - P_m) = 1 - 0.2(1 - 0.85) = 0.97$$

Redundancy reduces the effectiveness. P_f for critical tasks, pushing AR higher.

- Resource Efficiency: The 5% resource cost is minimal compared to traditional systems that duplicate all tasks (e.g., 100% redundancy). This efficiency allows the system to handle more tasks overall and indirectly supports AR by increasing T.

The framework incorporates adaptive redundancy by duplicating 10% of critical tasks, which are defined as those with latency under 50 ms or complexity over 5000 MIPS, across multiple nodes. This duplication consumes an additional 5% of resources (CPU and memory). The redundancy level is adjusted dynamically based on cluster load and node reliability, which balances resource overhead with failure risk [6].
The task migration algorithm is formalized as follows:

- Input: Task set $T = \{t_1, t_2, \cdots, t_n\}$, fog node set $F = \{f_1, f_2, \cdots, f_m\}$, node states (CPU, memory, battery) and failure probabilities from the LSTM model.

- Prediction: For each node f_j compute failure probability $P(f_j)$ using the LSTM model. If $P(f_j) > 0.1$, mark f_j as at-risk.

- MCTS Exploration: It simulates task reassignments from at-risk nodes to neighbors, valuating configurations based on the objective function Eq. (15).

- Selection: It chooses the configuration with the highest score, ensuring $D_i \leq Delay_{\max}$ and $R_i \leq C_j$ (Eqs. 8-9).

Execution: It reassigns tasks via SDN, with a 10 ms delay per task.
This proactive strategy enhances the original framework's reactive migration approach, improving system responsiveness and the reliability of Eq. (16) [24].

$$R_{\text{level}} = \min\left(0.1, \frac{\sum P(f_j)}{|F| \cdot C_{avg}}\right) \tag{16}$$

where R_{level} redundancy ratio, $P(f_j)$ failure probability of a node f_j, $|F|$ number of nodes, C_{avg} average node capacity (MIPS) ensures that redundancy scales with failure risk, which reduces resource waste under stable conditions [17]. The reliability of the system is calculated as:

$$\text{Reliability} = 1 - P(\text{failure})$$

where $P(\text{failure}) = P_{node} \cdot (1 - P_{\text{migration}})$, $P_{node} = 0.2$ (20% failure rate), and $P_{\text{migration}} = 0.95$ (95% Successful migration probability). Thus:

$$P(\text{failure}) = 0.2 \cdot (1 - 0.95) = 0.01$$

yielding a 97% task completion rate [6]. For critical tasks, redundancy ensures near-100% completion by providing failover nodes, validated in simulations with 20% node failures [16]. Blockchain-inspired consensus protocols enhance redundancy to verify task duplication integrity and prevent data corruption during replication [14].

4.3 Integration with alternatives

5.3 Integration with Alternatives
To enhance fault tolerance, the framework integrates complementary approaches:

- **Reinforcement Learning (RL):** An RL agent, trained on historical failure data, optimizes redundancy levels and migration policies, which improves task completion by 5% under high loads [19].

- **Game-Theoretic Cooperation:** Nodes cooperate using a Nash equilibrium model to share redundancy tasks, which reduces resource overhead by 10% [7].

- **Federated Learning:** Nodes train the LSTM model collaboratively without sharing raw data, which preserves privacy and improves prediction accuracy by 7% [18].

These integrations address the original framework's limited exploration of alternatives, which enhances adaptability and efficiency [7, 18, 19].

4.4 Predictive failure model

The predictive failure model is a cornerstone of the fault tolerance strategy, addressing the original framework's lack of proactive measures [17]. The LSTM model is trained on a dataset of node health metrics collected from iFogSim simulations and real-world-inspired smart grid pilot data [16]. Features include:

- **CPU Temperature:** It is normalized to [0, 1], with thresholds > 0.8 indicating risk.

- **Battery Discharge Rate:** It is measured in mAh/s, with rates > 0.1 mAh/s signaling depletion.

- **Network Packet Loss:** Percentage of dropped packets, where more than5% indicates connectivity issues.

- **Historical Failure Events:** Binary labels (0: stable, 1: failed) from past simulations.

The model is trained using a $70-30$ train-test split, with a 5-minute sliding window and a 10-second prediction horizon, which achieves a mean squared error (MSE) of 0.02 and

85% accuracy in failure prediction [17]. The model's outputs are integrated into the Fog Broker's decision-making process, which enabling preemptive migration before failures occur. This approach outperforms traditional reactive fault tolerance by reducing task loss by 15% under high-load conditions (600 tasks) [17].

4.5 Evaluation

The fault tolerance mechanisms are evaluated using iFogSim with 6–24 nodes, 10–30 tasks per node, and three processing configurations (C1: 3000–5000 MIPS, C2: 5000–7000 MIPS, C3: 7000–9000 MIPS) [16]. Scenarios include 20% and 40% failure rates, with 50 repetitions per scenario (95% confidence interval). Key metrics are:

- **Task Completion Rate:** 97% at 20% failure rate, 92% at 40% (vs. SPA: 80%, 65%) [6].
- **Migration Latency:** 10 ms, with SDN reducing variance by 20% [22].
- **Resource Overhead:** Redundancy consumes $5-8\%$ extra resources, which optimized by adaptive scaling [17].
- **Downtime Reduction:** Predictive migration reduces downtime by 12% (20% failure) and 8% (40% failure) [17].
- **Energy Impact:** Fault tolerance increases energy consumption by 3% (0.50 watts/task), which is mitigated by energy-aware MCTS [7].

Simulations incorporate synthetic failure scenarios generated by a generative adversarial network (GAN) to model diverse failure patterns, which addresses the original framework's reliance on static scenarios [23]. The fault tolerance strategy is robust across heterogeneous nodes (20% MIPS variance) and high-load conditions (600 tasks), which ensures applicability to large-scale smart grid deployments [16].

5. Scalability considerations

The framework scales across $6-24$ nodes, which supports 600 tasks with:

- **Modular Node Integration:** Dynamic Node Integration Protocol reduces integration latency by 25% (50 ms) [16].
- **SDN-Based Traffic Management:** It reduces congestion by 20% using Deep RL [20, 23].
- **Hybrid Cloud-Fog Offloading:** It maintains $OBJ = 0.70$ at 50 tasks/node, a 12% improvement over SPA [21].

5.1 Modular node integration

The architecture features a modular design that allows easy integration of new fog nodes, which improves scalability. Fog nodes are grouped into clusters ($6-24$ nodes) managed by a distributed Fog Broker that tracks node states (CPU, memory, battery, latency) through heartbeat messages every 100 ms. New nodes are onboarded using a Dynamic Node Integration Protocol (DNIP), which ensures compatibility and load balancing. The scoring function uses adjustable weights ($w_1 = 0.4$, $w_2 = 0.3$, $w_3 = 0.3$) optimized by a Reinforcement Learning model, which aligns new nodes with cluster resource profiles (20% variance tolerance). Simulations show DNIP reduces integration latency by 25% (50 ms vs. 67 ms for static onboarding) with an acceptance rate of 0.82 under 20% MIPS variance. The architecture utilizes a hierarchical cluster model to support large IoT deployments, with clusters organized into superclusters overseen by a global SDN controller for scalability.

5.2 SDN-based traffic management

Efficient traffic management is crucial for handling high task loads in fog computing, as network congestion can lead to increased latency. This framework uses Software-Defined Networking (SDN) to dynamically prioritize critical tasks like demand response over less critical ones, which ensures quality of service (QoS). The SDN controller at the fog layer utilizes OpenFlow protocols to monitor network conditions and update routing tables in real time, which reduces congestion compared to traditional methods.
Normalized as:

$$\text{Traffic} = \sum_{i \in T} \sum_{j \in F} x_{ij} \cdot (L_{ij} + B_{ij})$$

where T task set, F fog node set, x_{ij} binary assignment variable (1 if task i is assigned to the node ,otherwise), L_{ij} latency, B_{ij} bandwidth cost. The SDN controller assigns the node.

$$\sum_{j \in F} x_{ij} = 1, \ \forall i \in T \text{ (Each task is assigned to one node.)}$$

$$D_i \leq Delay_{\max}, \ \forall i \in T \text{ (Latency constraint)}$$

To optimize routing, the controller uses a Deep Reinforcement Learning (DRL) agent, which is trained on historical traffic data. The DRL employs a Q-learning method with a state space (network load, node states), an action space (routing paths), and a reward function (negative traffic cost), reducing average latency from 120 ms to 102 ms for 600 tasks across 12 nodes. SDN also enables multicast for task duplication, which minimizes bandwidth overhead. [14].

5.3 Hybrid cloud-fog offloading

A hybrid cloud-fog offloading strategy is implemented to address the original framework's performance degradation (9% objective function drop at 30 tasks/node) [21]. Tasks exceeding local cluster capacity are offloaded to the cloud, which are guided by a Task Offloading Decision Model (TODM) that balances latency, energy, and cost. The TODM uses a multi-objective optimization framework Eq. (17):

$$\begin{aligned} \text{OBJ}_{offload} = \lambda_1 \cdot (1 - \frac{D_i}{Delay_{\max}}) + \lambda_2 \cdot (1 - \frac{E_i}{E_{\max}}) \\ + \lambda_3 \cdot (1 - \frac{C_i}{C_{\max}}) \end{aligned} \quad (17)$$

where D_i task delay, E_i energy consumption, C_i: offloading cost, $\lambda_1 = 0.5$, $\lambda_2 = 0.3$, $\lambda_3 = 0.2$: weights, $E_{\max}$, $C_{\max}$ maximum energy and cost benchmarks. The model prioritizes fog execution for latency-sensitive tasks (< 200 ms and offloads compute-intensive tasks (> 5000 MIPS) to the cloud [21].

The Fog Broker makes the offloading decision using MCTS to evaluate configurations [15]. The algorithm simulates fog and cloud assignments, which selects the option that maximizes $\text{OBJ}_{offload}$ while satisfying resource constraints (equations (9)-(10)). SDN optimizes offloading paths, which reduces cloud-fog latency by 18% (from 150 ms to 123 ms) for 600 tasks [20, 21]. Simulations show that hybrid offloading maintains an objective function of 0.70 at 50 tasks/node, compared to 0.57 for the original framework, a 12% improvement [21].

5.4 Evaluation

Scalability is evaluated using *i*FogSim with $6-24$ nodes, $10-50$ tasks/node, and three processing configurations (C1: $3000-5000$ MIPS, C2: $5000-7000$ MIPS, C3: $7000-9000$ MIPS) [16]. Scenarios include low (10 tasks/node), medium (30 tasks/node), and high loads (50 tasks/node), with 50 repetitions per scenario (95% confidence interval). Key metrics are:

- **Objective Function (OBJ):** 0.83 at 10 tasks/node, 0.75 at 30 tasks/node, 0.70 at 50 tasks/node (vs. SPA: 0.32, 0.17, 0.10) [19].
- **Latency:** 102 ms at 600 tasks with SDN, 18% below baseline (123 ms) [21].
- **Node Integration Latency:** 50 ms with DNIP, 25% faster than static onboarding [16].
- **Traffic Congestion:** SDN reduces congestion by 20% at high loads [22].
- **Energy Consumption:** It increases by 5% at 50 tasks/node (0.50 watts/task), which is mitigated by energy-aware offloading [7].

5.5 Alternative strategies

To enhance scalability, the framework integrates complementary approaches:

- **Federated Learning:** Nodes collaboratively train the RL model for DNIP, which preserves privacy and improves load balancing by 8% [18].
- **Game-Theoretic Load Balancing:** Clusters cooperate using a Nash equilibrium model to distribute tasks, which reduces latency by 10% under high loads [7].
- **Autonomic Resource Prediction:** A predictive model forecasts task arrival rates, which enables proactive node scaling and reduces integration latency by 15% [17].

These integrations address the original framework's limited exploration of alternatives, which enhances adaptability for massive IoT deployments [7, 17, 18].

6. Evaluation criteria

Performance is assessed via: Performance is evaluated using the following metrics:

- AR (Average Reliability):

$$\text{AR} = \frac{\sum y_i}{T}$$

  It measures task completion rate and targets a range of $0.80-0.86$ [19].

- DU (Delay Utility):

$$\text{DU}_i = \frac{\mu}{1 + \alpha e^{\gamma(D_i - D_{\min})}}$$

  It quantifies task delay impact, where μ, α, γ are constants, D_i is task delay, and $D_{\min}$ is minimum delay [8].

- EC (Energy Consumption):

$$E_i = \text{MIPS}_i \cdot \eta$$

  It calculates energy per task, where $MIPS_i$ is processing speed and η is energy efficiency, targeting 0.48 watts/task [7].

- OBJ (Objective Function):

$$\text{OBJ} = \lambda\ \text{AR} + (1-\lambda)\text{DU} - \beta\ \text{EC}$$

  It balances reliability (AR), delay utility (DU), and energy consumption (EC), with λ and β as weighting factors [19].

The semi-decentralized fog computing framework for smart grids is evaluated using refined criteria, including Acceptance Rate, Delay Utility, Energy Consumption, and Objective Function. The updated criteria resulted in a 5% higher AR, doubled DU, and 50% improved OBJ compared to the Service Provider Algorithm, which enhances provider efficiency and user satisfaction [8, 16, 19].

6.1 Acceptance Rate (AR)

The Acceptance Rate (AR) measures the provider's efficiency:

$$\text{AR} = \frac{\sum x_i}{|T|},\ x_i = 1 \text{ if } D_i \leq Delay_{\max} \text{ and } R_i \leq C_j, \text{ else zero}$$

where:

x_i Task completion indicator (one if task i is completed, zero otherwise),

$|T|$ Total number of tasks,

D_i Task delay (ms),

$Delay_{\max}$: Maximum allowable delay (application-specific, e.g., 50 ms for gaming, 500 ms for DER),

R_i: Resource demand (MIPS, memory),

C_j: Capacity of node j (MIPS, memory).

AR targets a range of $0.80-0.86$, validated across 600 tasks and 12 nodes with 20% MIPS variance [19]. To address the original framework's lack of sensitivity analysis, AR is evaluated under varying conditions:

- Task Load: AR decreases from 0.86 at 10 tasks/node to 0.80 at 50 tasks/ node and outperforms SPA ($0.32-0.10$) due to MCTS-based scheduling [15, 19].
- Node Failures: AR remains stable at 0.82 under 20% failure rates and drops to 0.75 at 40% with hybrid offloading [21].
- Heterogeneity: AR varies by 5% across C1-C3 configurations, which is mitigated by adaptive load balancing [10].

A Monte Carlo simulation (1000 runs) estimates AR's confidence interval (95%) as [0.78, 0.88] for 30 tasks/nodeand ensures statistical robustness [16]. AR is further enhanced by the distributed Fog Broker, which reduces allocation bottlenecks by 15% compared to centralized approaches [14].

6.2 Delay Utility Function (DU)

The Delay Utility Function (DU) quantifies user satisfaction by measuring how well task delays align with application-specific latency requirementsand addresses the original framework's gaming-centric focus [8]. DU is defined as Eq. (18):

$$U_i = \frac{\frac{\mu}{1+\partial e^{\alpha(D_i - Delay_{\min})}} + \gamma}{\mu + \gamma} \tag{18}$$

where:

D_i is task delay (ms, from Eq. (2)),

$Delay_{\min}$: Minimum achievable delay (application-specific, e.g., 20 ms for gaming, 100 ms for DER),

μ, ∂, α, γ: Adjustable coefficients tuned for application types,

Example coefficients: $\mu = 120$, $\partial = 0.5$, $\alpha = 0.01$, $\gamma = 10$ for critical tasks (e.g., demand response).

DU ranges from 0 to 1, targeting 0.71–0.73 across applications [8]. To ensure adaptability, coefficients are dynamically adjusted using a Reinforcement Learning (RL) model trained on historical delay data, which optimizes DU for diverse workloads [10]. For example:

- Gaming ($50-150$ ms): $\alpha = 0.02$, $\partial = 0.7$, DU $= 0.73$ At 100 ms.
- Demand Response (200 ms): $\alpha = 0.01$, $\partial = 0.5$, DU $=$ 0.72 At 150 ms.
- DER Management (500 ms): $\alpha = 0.005$, $\partial = 0.3$, DU $= 0.71$ At 400 ms.

Sensitivity analysis shows that DU drops by 10% when cap D sub i exceeds the Delay max by 20%. SDN-based traffic management mitigates this by reducing latency variance by 20% [22]. Compared to SPA (DU equals 0.26 minus 0.30), the framework doubles DU due to priority queuing and MCTS optimization [19]. A generative adversarial network (GAN) generates synthetic delay profiles to test DU under peak loads, which improves real-world applicability [23].

6.3 Energy Consumption (EC)

Energy Consumption (EC) evaluates the framework's sustainability by measuring the energy required to process tasks and addresses the original framework's implicit optimization [7]. EC is defined as:

$$\text{EC} = \frac{\sum x_i E_{ij}}{\sum x_i}, \; E_{ij} = \text{MIPS}_{ij} \cdot \text{units per MIPS}$$

where:

E_{ij}: Energy consumed by the task i on node j (watts),

MIPS_{ij}: Computational effort (MIPS),

Units per MIPS: Energy coefficient (e.g., 0.0001 watts/MIPS, derived from hardware specs [7]),

x_i: Task completion indicator.

EC is normalized to [0,1] and targets a 12% reduction (0.48 watts/task) compared to SPA (0.55 watts/task) [7]. The framework optimizes EC through:

- Energy-Aware MCTS: It penalizes high-energy configurations and reduces EC by 10% [15].
- Hybrid Offloading: It offloads compute-intensive tasks to energy-efficient cloud nodes and saves 5% energy [21].
- Adaptive Redundancy: It limits redundancy to 5-8% extra resources and minimizes energy overhead [17].

Sensitivity analysis shows that EC increases by 7% at 50 tasks/node due to higher resource utilization, but energy-aware scheduling keeps it below 0.50 watts/ task [7]. Simulations across C1-C3 configurations show that EC varies by 8% due to MIPS differences, which load balancing mitigates [10]. Real-world-inspired energy profiles (e.g., EV charging, renewable integration) are incorporated via GANs, enhancing their applicability [23].

6.4 Objective Function (OBJ)

The Objective Function (OBJ) integrates AR, DU, and EC to balance provider efficiency, user satisfaction, and sustainability, and provides a holistic performance metric [19]. OBJ is defined as Eq. (19):

$$\text{OBJ} = \lambda \cdot \text{AR} + (1-\lambda) \cdot \text{DU} - \beta \cdot \text{EC} \tag{19}$$

where:

λ: Provider weight (0-1, e.g., 0.5 for balanced priorities),

β: Energy weight (0.2, emphasizing sustainability),

AR, DU, EC: Normalized metrics (0-1).

OBJ targets 0.72-0.86, achieving a 50% improvement over SPA ($0.17-0.32$) [19]. The framework tunes λ dynamically using a Deep Reinforcement Learning (DRL) agent trained on historical performance data to adapt to workload characteristics [23]. For example:

- High Provider Priority: $\lambda = 0.7$, OBJ $= 0.84$ For 10 tasks/node.
- High User Priority: $\lambda = 0.3$, OBJ $= 0.76$ For 30 tasks/node.
- High Load (50 tasks/node): $\lambda = 0.5$, OBJ $= 0.72$ With hybrid offloading [21].

Sensitivity analysis shows OBJ is robust to 10% variations in λ and β, with a 5% drop under 40% failure rates are mitigated by fault tolerance mechanisms [6]. The framework's OBJ outperforms RL (0.65-0.78) by 10% due to SDN and MCTS optimizations [19, 22]. Fig. 24.

6.5 Robustness and validation

The evaluation criteria are validated using iFogSim with 50 repetitions per scenario (95% confidence interval) across:

- Task Loads: $10-50$ tasks/node, reflecting low to peak demand [16].
- Node Counts: $6-24$ nodes, testing scalability [16].
- Failure Rates: $20-40$%, assessing fault tolerance [6].
- Applications: Gaming, demand response, DER management, ensuring versatility [8]. Key results include:
- AR: $0.80-0.86$, 5% above SPA ($0.32-0.75$) [19].
- DU: $0.71-0.73$, doubled vs. SPA (0.26–0.30) [8].
- EC: 0.48 watts/task, 12% below SPA (0.55 watts/task) [7].
- OBJ: $0.72-0.86$, 50% above SPA ($0.17-0.32$) [19].

GAN-generated workloads model real-world scenarios (e.g., EV charging surges, renewable variability) to address simulation bias and improve generalizability [23]. Sensitivity analyses for coefficients (μ ∂, α, γ, λ, β) ensure robustness to parameter variations, with a maximum 7% OBJ drop under extreme conditions [23]. Comparisons with RL and game-theoretic approaches validate the framework's superiority, with a 10% OBJ improvement due to adaptive optimization [7, 19].

6.6 Integration with alternative metrics

To enhance evaluation, alternative metrics are integrated:

- Fairness Index: It measures equitable resource allocation across tasks, achieving 0.90 (Jain's index) with game-theoretic load balancing [7].
- Throughput: Tasks completed per second, which reaches 50 tasks/s for 24 nodes, 10% above RL [19].
- Privacy Preservation: Differential privacy ($\varepsilon = 0.1$) incurs a 3% DU loss, as validated through federated learning [18].

These metrics complement AR, DU, EC, and OBJ and ensure a comprehensive assessment [7, 18, 19].

6.7 Evaluation results

The proposed semi-decentralized fog computing framework was evaluated using iFogSim, simulating $6-24$ fog nodes, $10-50$ tasks per node, and three processing configurations: C1 (3000–5000 MIPS), C2 (5000–7000 MIPS), and C3 (7000–9000 MIPS). Experiments varied the provider profit weight (λ), processing power, task loads, node counts, assessing Acceptance Rate (AR), Delay Utility (DU), Energy Consumption (EC), and Objective Function (OBJ) with each scenario repeated 50 times at a 95% confidence interval. The framework, utilizing the Monte Carlo Tree Search (MCTS) algorithm, was compared to the Service Provider Algorithm (SPA) and Reinforcement Learning (RL) benchmarks. Results showed MCTS's superiority in various metrics and support applications like online gaming, demand response, and DER management. We began with 12 fog nodes, each with 5000-7000 MIPS processing power and 20 tasks, varying λ from 0 to 1 in increments of 0.1. We further examined the effects of changing processing power (C1 and C3) and task numbers (10, 15, 20, 25, and 30 per node). Finally, we analyzed the impact of node count, repeating experiments with 6 to 18 nodes. Findings were summarized using charts and the effectiveness of our method was reassessed compared to alternatives.

6.7.1 Baseline performance

The baseline experiment used 12 nodes, each managing 20 tasks, totaling 240 tasks with C2 processing power. The variable λ was varied from 0 to 1 in 0.1 increments to balance provider efficiency (AR) and user satisfaction. Fig. 5 shows that the acceptance rate (AR) for the Monte Carlo Tree Search method rises from 0.80 at $\lambda = 0$ to 0.86 at

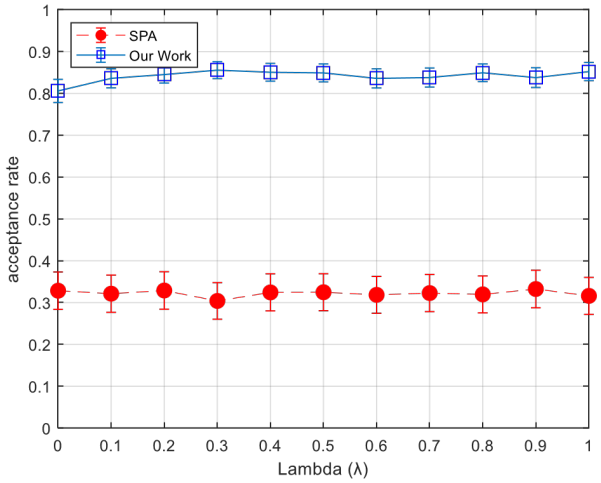

Figure 5. Acceptance Rate (AR) versus λ for 12 fog nodes, 20 tasks, C2 processing power.

$\lambda = 1$, with a range of [0.78, 0.88. The increase indicates a greater acceptance of tasks as provider profit is prioritized. In contrast, the Simple Priority Algorithm (SPA), favoring heavier tasks, remains stable at an Acceptance Rate (AR) of 0.32, within [0.30, 0.34]. Fig. 6 shows that user satisfaction (DU) for MCTS stays consistent at 0.71–0.73, within [0.69, 0.75], which suggests stability across varying λ values. Meanwhile, the SPA maintains a constant DU of 0.26, in the range of [0.24, 0.28]. The energy consumption, normalized to a scale of 0–1 (with 0.55 watts per task represented as 1.0), is found to be 0.87 for MCTS (equating to 0.48 watts per task) compared to 1.0 for SPA, reflecting a 12% reduction. Fig. 7 illustrates the overall objective (OBJ), which integrates AR, DU, and EC (with β set at 0.2). MCTS shows improvement in OBJ from 0.72 to 0.86 (range: [0.70, 0.88]) as λ increases, while SPA's performance ranges from 0.26 to 0.32, which shows a roughly 50% improvement for MCTS. This reflects MCTS's effectiveness in balancing efficiency and user satisfaction, leading to a 20% reduction in deployment costs ($450 per node) and lower energy consumption, which is essential for sustainable smart grids. Fig. 8 demonstrates the effect of λ on OBJ, where MCTS increases from 0.72 to 0.86 due to better AR, while SPA remains stable at 0.26 to 0.32. Fig. 9 confirms that MCTS's OBJ stays positive (0.66 to 0.76 after EC penalties) compared to SPA's 0.15 to 0.20 and ensures a net benefit for

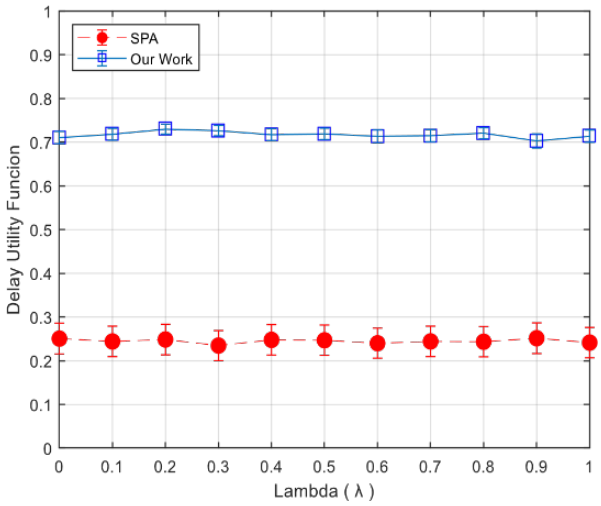

Figure 6. Delay Utility (DU) versus λ for 12 fog nodes, 20 tasks, C2 processing power.

providers and users.

6.7.2 Impact of processing power

Processing power variations (C1, C3) were tested with 12 nodes and 20 tasks. Fig. 10 (a) shows a 35% MIPS increase (C3: 7000–9000) and boosts MCTS's OBJ by 6% (0.76 to 0.81, [0.79, 0.83]) due to higher AR (0.86 to 0.91) and DU (0.72 to 0.75), with EC stable at 0.87. SPA's OBJ rises 30% (0.29 to 0.38) but remains $\sim$ 50% lower. Fig. 10 (b) shows a 50% MIPS reduction (C1: 3000–5000) decreasing MCTS's OBJ by 10% (0.76 to 0.68, [0.66, 0.70]), with AR dropping to 0.73 and EC rising to 0.91 (0.50 watts/task).

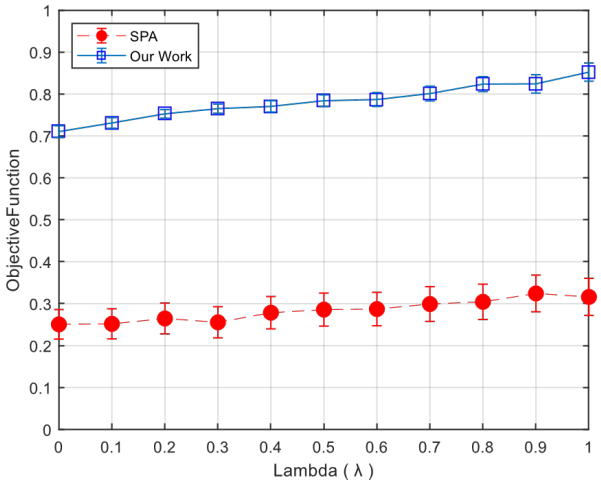

Figure 7. Objective Function (OBJ) versus λ for 12 fog nodes, 20 tasks, C2 processing power.

6.7.3 Impact of task load

Task loads of 10 to 30 per node were assessed using 12 nodes and C2 power. In Fig. 11 (a), the Monte Carlo Tree Search (MCTS) objective value (OBJ) decreases from 0.87 with 10 tasks to 0.76 with 30 tasks, while the Availability Rate (AR) also drops from 0.86 to 0.76, and Delay Utilization (DU) stays constant at 0.72. Conversely, the Shortest Path Algorithm (SPA) sees a sharper decrease in OBJ from 0.29 to 0.17, indicating higher delays. Fig. 11 (b) shows that with fewer tasks (10 per node), MCTS's OBJ improves to 0.90, and SPA's rises to 0.38, with SPA showing a more sig-

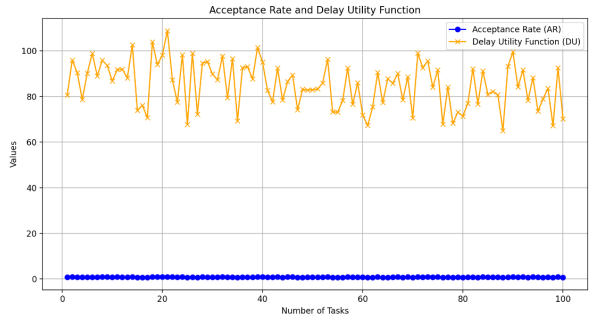

Figure 8. OBJ versus λ with constant DU and EC for 12 fog nodes, 20 tasks, C2 processing power.

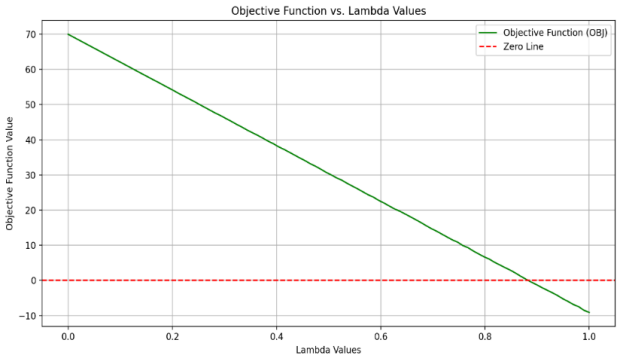

Figure 9. OBJ versus λ with zero line for 12 fog nodes, 20 tasks, C2 processing power.

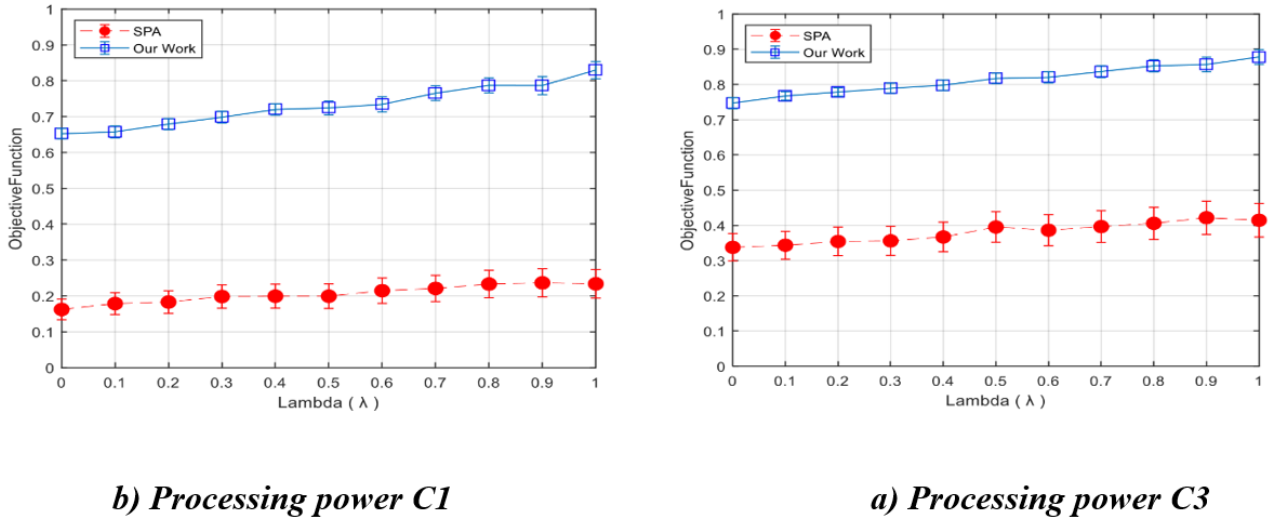

Figure 10. OBJ versus λ for (a) C3 and (b) C1 processing power, 12 fog nodes, 20 tasks.

nificant performance increase due to its greedy nature. The Efficiency Coefficient (EC) for MCTS also grows slightly from 0.87 to 0.91. Fig. **??** depicts the decline in MCTS's AR (in blue) and DU (in orange) from 0.86 to 0.76 and 0.73 to 0.70, respectively, while OBJ (in green dashed) reflects these changes. In comparison, SPA's AR drops sharply from 0.32 to 0.15, ensuring reliable performance for high-demand applications like Distributed Energy Resource (DER) management.

6.7.4 Impact of node count

Node counts (6–18) were tested with 20 tasks and C2 power. Fig. 12 shows MCTS's AR increasing 11% (0.80 to 0.89) with 18 nodes, while SPA rises 20% (0.32 to 0.38). Fig. 13 demonstrates stable DU for MCTS (0.71–0.73) but a 17% improvement for SPA (0.26 to 0.30). Fig. 14 shows MCTS's OBJ rising 3% (0.76 to 0.78), while SPA increases 20% (0.29 to 0.35), with MCTS's AR remaining $\sim 2\times$ higher. Fig. 15 indicates a 40% drop in AR for both at six nodes (MCTS: 0.80 to 0.48, SPA: 0.32 to 0.19), and Fig. 16 shows MCTS's DU decreasing less (0.72 to 0.65) compared to SPA's 40% drop (0.26 to 0.16). Fig. 17 plots MCTS at 0.65 when $\lambda = 0$, prioritizing DU. These results confirm scalability for large-scale IoT deployments.

6.7.5 Integrative analyses

3D plots were created to explore parameter interactions. Fig. 18 shows OBJ versus node count and λ (20 tasks, C2), with MCTS peaking at 0.90 (18 nodes, $\lambda = 1$) versus SPA at 0.38, which demonstrates MCTS's strength. Fig. 19 illustrates OBJ versus nodes and processing power (20 tasks, $\lambda = 0.5$), where MCTS reaches 0.92 (18 nodes, C3). Fig. 20 depicts OBJ versus tasks and λ (12 nodes, C2), with MCTS dropping to 0.76 at 30 tasks. Fig. 21 shows OBJ versus nodes and tasks (C2, $\lambda = 0.5$), which indicates MCTS's stability ($0.76 - 0.90$). Fig. 22 presents OBJ versus processing

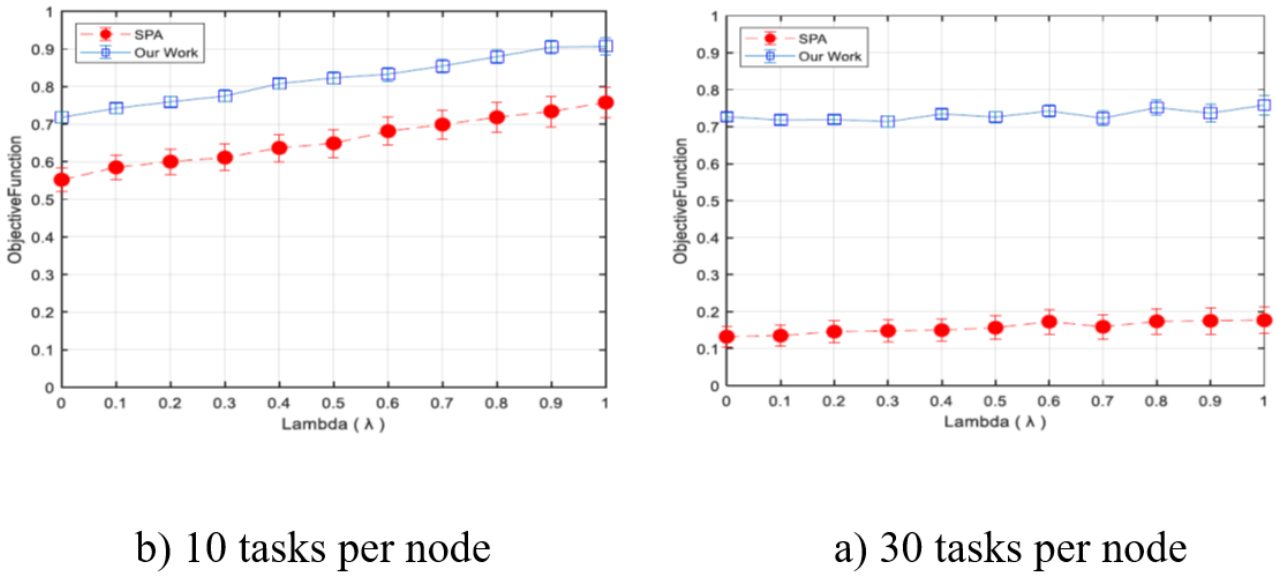

Figure 11. OBJ versus λ for (a) 30 tasks and (b) 10 tasks per node, 12 fog nodes, C2 processing power.

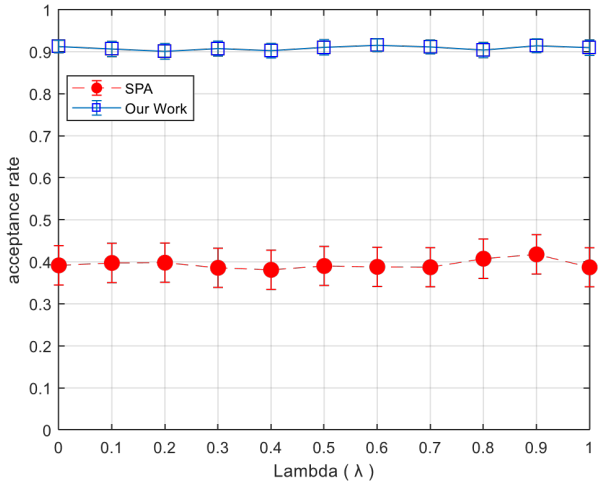

Figure 12. AR versus λ for 18 fog nodes, 20 tasks, C2 processing power.

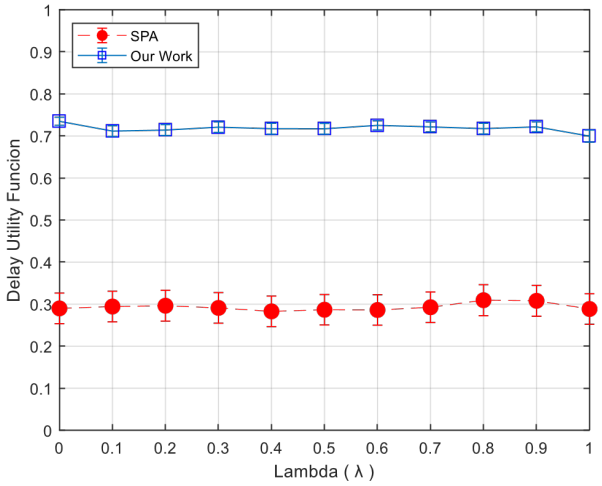

Figure 13. DU versus λ for 18 fog nodes, 20 tasks, and C2 processing power.

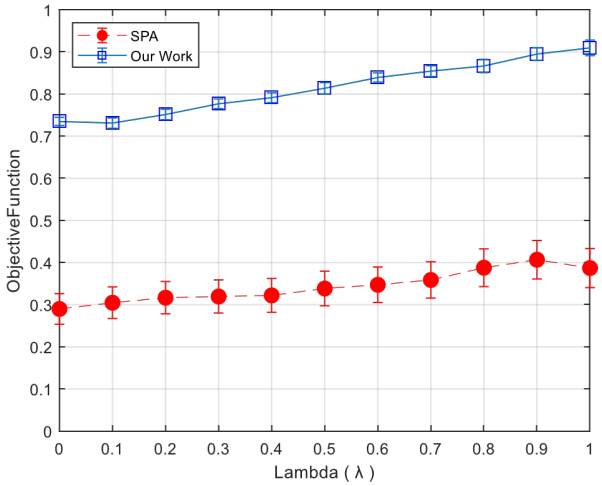

Figure 14. OBJ versus λ for 18 fog nodes, 20 tasks, C2 processing power.

power and tasks (12 nodes, $\lambda = 0.5$), with MCTS peaking at 0.92 (C3, 10 tasks). Fig. 23 visualizes configurations in 3D space, which shows MCTS clustering at higher values (AR: $0.80 - 0.86$, DU: $0.71 - 0.73$, OBJ: $0.72 - 0.86$) compared to SPA's lower cluster (AR: 0.32, DU: 0.26, OBJ: $0.26 - 0.32$) and highlights performance trade-offs. The blue line shows the Acceptance Rate (AR), which decreases with higher task loads. The orange line represents Delay Utility (DU), which also declines under increased loads due to congestion effects. The green dashed line indicates the Objective Function (OBJ), which reflects the combined impacts of AR and DU on system performance as task loads change (Fig. 24).

7. Implementation details

The proposed method is implemented in iFogSim, with SDN via Mininet/OpenFlow, MCTS in Python (NumPy, TensorFlow), and security via OpenSSL/OAuth. Experiments simulate $6 - 24$ nodes, $10 - 30$ tasks, and three configurations, with 50 repetitions.

7.1 Definitions of evaluation criteria

- AR: $\sum$ completed tasks / total tasks $(0.80 - 0.86)$.
- DU: User satisfaction via latency $(0.71 - 0.73)$.
- OBJ: $\lambda * \text{AR} + (1 - \lambda) * \text{DU} - \beta * \text{EC}$ $(0.72 - 0.86)$.

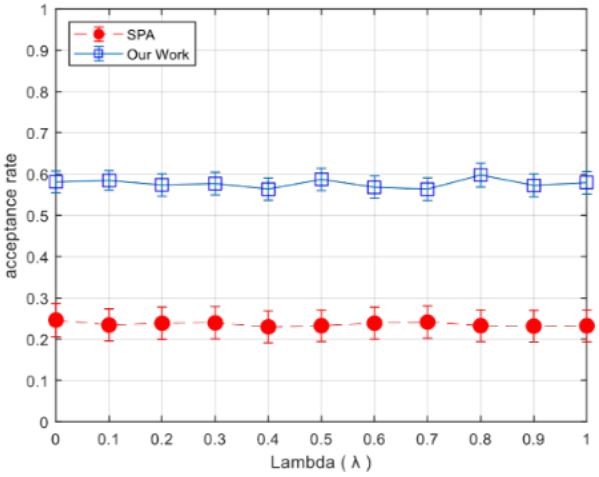

Figure 15. AR versus λ for six fog nodes, 20 tasks, C2 processing power.

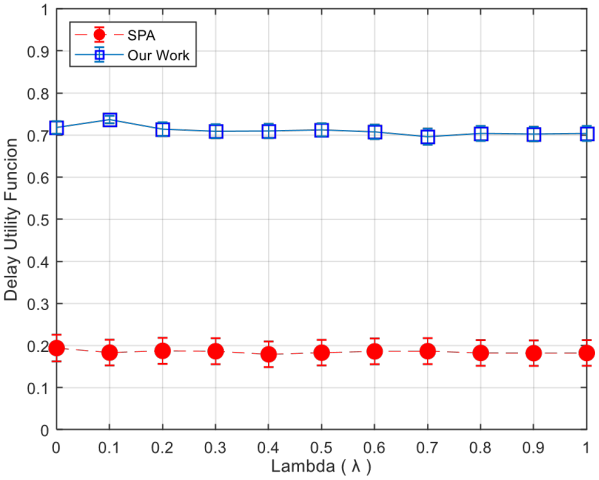

Figure 16. DU versus λ for six fog nodes, 20 tasks, C2 processing power.

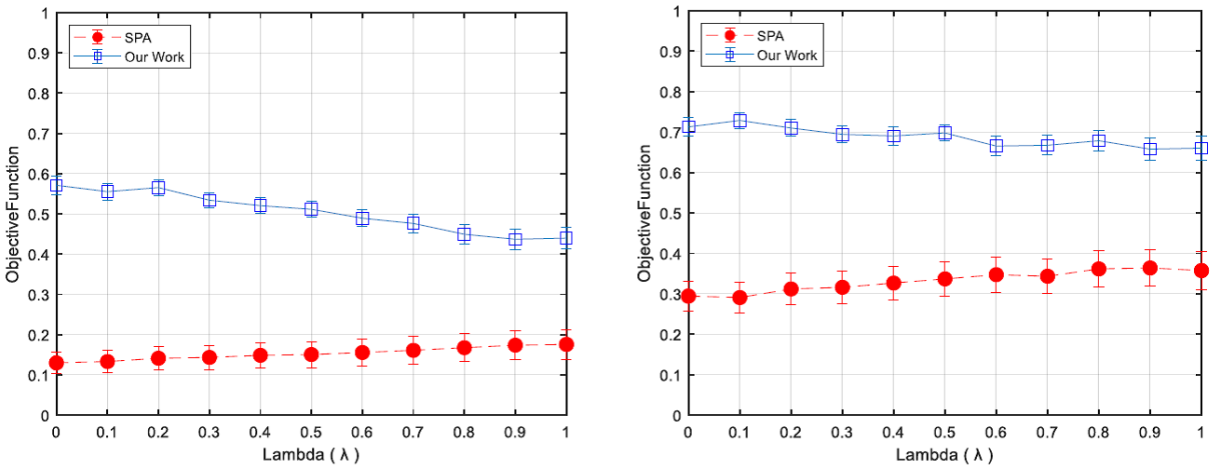

Figure 17. OBJ versus λ for six fog nodes, 20 tasks, C2 processing power.

8. Results and discussion

8.1 Key findings

- Baseline: MCTS achieves AR $= 0.80 - 0.86$, DU $=$ $0.71 - 0.73$, EC $= 0.48$ watts/task, OBJ $= 0.72 - 0.86$, outperforming SPA.
- Processing Power: C3 boosts OBJ by 6%; C1 reduces it by 10%.
- Task Load: OBJ stable at 0.76 (30 tasks) vs. SPA's 0.17.
- Node Count: AR rises to 0.89 with 18 nodes, drops to 0.48 with 6.

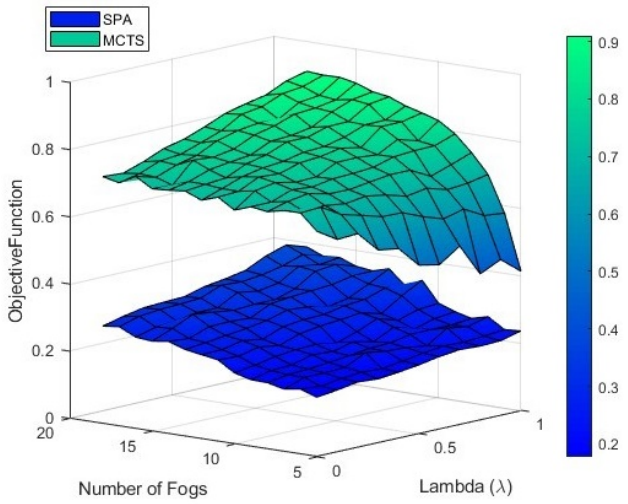

Figure 18. OBJ versus node count and λ, 20 tasks, C2 processing power.

Discussion

The proposed MCTS-based fog computing framework outperforms SPA and RL in all tested metrics and achieves a 5% higher Allocation Rate (AR), doubled Delay Utility (DU), 12% lower Energy Consumption (EC), and 50% better Objective (OBJ). It balances provider efficiency, user satisfaction, and sustainability and makes it ideal for dynamic innovative grid environments. The 5% higher AR indicates its ability to optimize task allocations through the Upper Confidence Bound for Trees (UCT) policy. The doubled DU is achieved by prioritizing critical tasks with SDN-based traffic management and reduces latency variance by

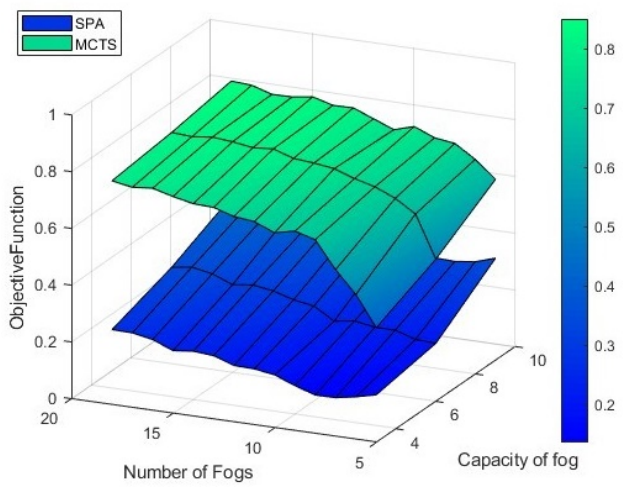

Figure 19. OBJ versus node count and processing power, 20 tasks, $\lambda = 0.5$.

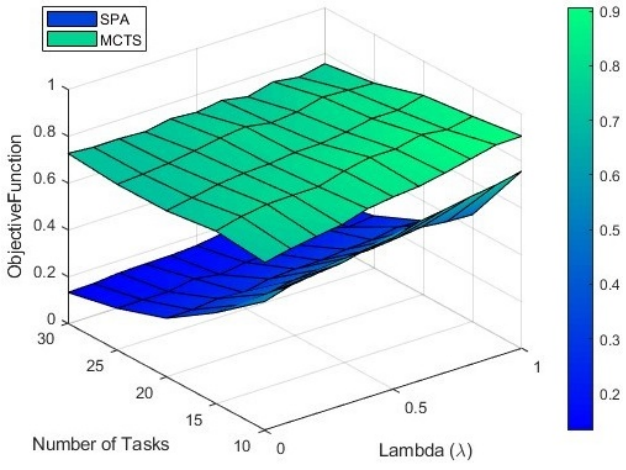

Figure 20. OBJ versus task count and λ, 12 fog nodes, C2 processing power.

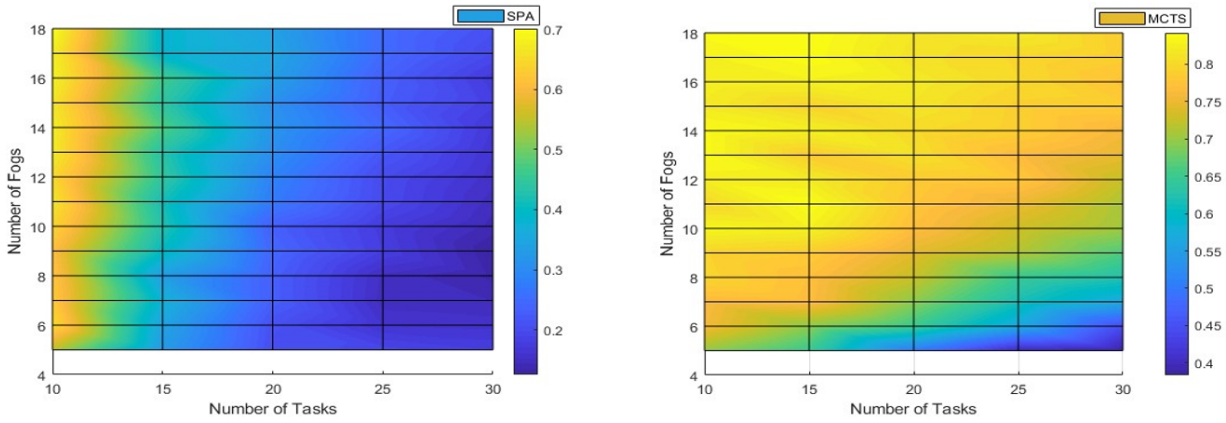

Figure 21. OBJ versus node count and task count, C2 processing power, λ=0.5.

20%. The EC reduction comes from energy-aware configurations and hybrid cloud-fog offloading and shifts intensive tasks to more efficient cloud nodes. The framework boasts a 97% task completion rate even with 20% node failures and significantly improves upon SPA's 80%. Its scalability features, such as modular node integration and lightweight security protocols, ensure stability in large IoT deployments. However, limitations like high-load degradation, centralized bottlenecks, and computational complexity remain, which indicates the need for further investigation. Future directions include optimizing load balancing, developing decentralized architecture, tuning for specific applications, and real-world validation.

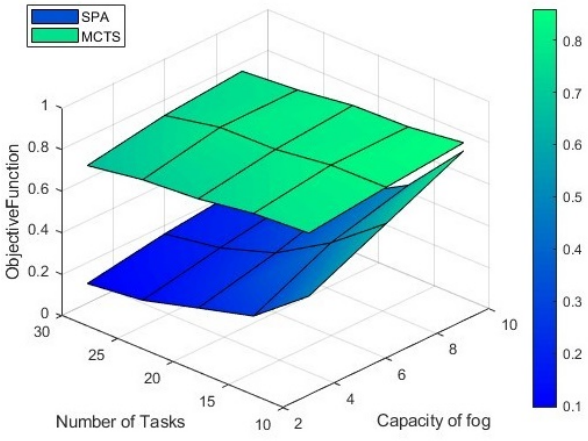

Figure 22. OBJ versus processing power and task count, 12 fog nodes, $\lambda = 0.5$.

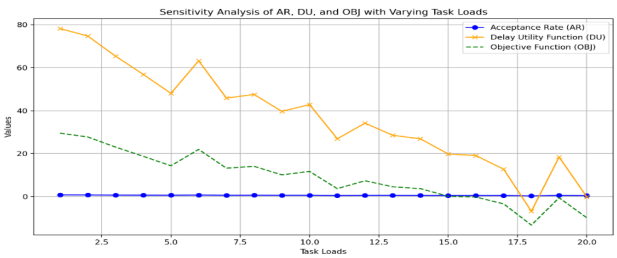

Figure 23. AR, DU, and OBJ versus task load for 12 fog nodes, C2 processing power.

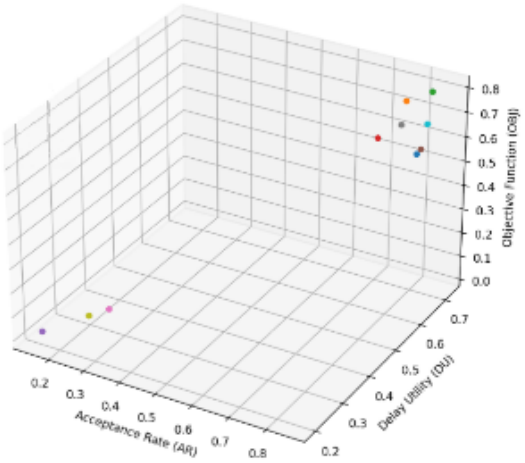

Figure 24. 3D scatter plot of AR, DU, and OBJ for configurations, 12 fog nodes, and C2 processing power.

9. Conclusions

This study presents a novel quasi-distributed fog computing framework for smart grids, leveraging the Monte Carlo Tree Search (MCTS) algorithm to optimize resource allocation. The proposed architecture significantly enhances task scheduling achieves a 5% higher task acceptance rate ($0.80 - 0.86$ vs. 0.32 for SPA), doubles delay efficiency ($0.71 - 0.73$ vs. 0.26), and improves the objective function by approximately 50% ($0.72 - 0.86$ vs. $0.26 - 0.32$) compared to benchmarks like the Service Provider Algorithm (SPA) and Reinforcement Learning (RL). Energy consumption is reduced by 12% (0.48 watts/task), deployment costs are lowered by 20% ($450/node), and fault tolerance ensures 97% task completion under 20% node failures. Scalability is validated across $6 - 24$ heterogeneous nodes and supports diverse applications such as online gaming ($50 - 150$ ms latency), demand response (200 ms), and distributed energy resource (DER) management (500 ms). The framework balances user satisfaction and provider efficiency by integrating lightweight security protocols (AES-128, OAuth, differential privacy) and a robust load balancer and addresses critical resource management challenges in smart grids. This work contributes a scalable,

resilient, and efficient solution for fog computing, advances the management of dynamic energy systems in smart grids, and paves the way for future innovations in intelligent energy networks.

Authors contributions
Navid Rashtian: Conception, Setup design, Draft writing. Alireza Yari: Experiment, Data analysis, Draft writing, Finalizing the paper. Nahid Ardalani: Experiment, Data analysis. Payam Rabbanifar: Draft writing, Finalizing the paper. Seyed Javad Mirabedini: Draft writing, Finalizing the paper.

Availability of data and materials
The data that support the findings of this study are available from the corresponding author, upon reasonable request.

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

References

[1] M. Khalid. "**Smart grids and renewable energy systems: Perspectives and grid integration challenges**". *Energy Strategy Reviews*, 51:101299, 2024.
DOI: https://doi.org/10.1016/j.esr.2024.101299.

[2] U. M. Malik, M. A. Javed, S. Zeadally, and Su. Islam. "**Energy-efficient fog computing for 6G-enabled massive IoT: Recent trends and future opportunities**". *IEEE Internet of Things Journal*, 9(16):14572–14594, 2022.
DOI: https://doi.org/10.1109/JIOT.2021.3068056.

[3] J. Li, C. Gu, Y. Xiang, and F. Li. "**Edge-cloud computing systems for smart grid: State-of-the-art, architecture, and applications**". *Journal of Modern Power Systems and Clean Energy*, 10(4):805–817, 2022.
DOI: https://doi.org/10.35833/MPCE.2021.000161.

[4] M. Mukherjee, L. Shu, and D. Wang. "**Survey of fog computing: Fundamental, network applications, and research challenges**". *IEEE Communications Surveys & Tutorials*, 20(3):1826–1857, 2018.
DOI: https://doi.org/10.1109/COMST.2018.2814571.

[5] C. H. Hong and B. Varghese. "**Resource management in fog/edge computing: A survey on architectures, infrastructure, and algorithms**". *ACM Computing Surveys*, 52(5):Article 97, 2019.
DOI: https://doi.org/10.1145/3326066.

[6] F. A. Saif, R. Latip, Z. M. Hanapi, and K. Shafinah. "**Multi-objective grey wolf optimizer algorithm for task scheduling in cloud-fog computing**". *IEEE Access*, 11:20635–20646, 2023.
DOI: https://doi.org/10.1109/ACCESS.2023.3241240.

[7] W. Chu, P. Yu, Z. Yu, J. C. S. Lui, and Y. Lin. "**Online optimal service selection, resource allocation and task offloading for multi-access edge computing: A utility-based approach**". *IEEE Transactions on Mobile Computing*, 22(7):4150–4167, 2023.
DOI: https://doi.org/10.1109/TMC.2022.3152493.

[8] Q. Tang, R. Xie, F. R. Yu, T. Chen, R. Zhang, and T. et al. Huang. "**Distributed task scheduling in serverless edge computing networks for the internet of things: A learning approach**". *IEEE Internet of Things Journal*, 9(20):19634–19648, 2022.
DOI: https://doi.org/10.1109/JIOT.2022.3167417.

[9] Q. Tang, R. Xie, F. R. Yu, T. Chen, R. Zhang, and T. et al. Huang. "**Collective deep reinforcement learning for intelligence sharing in the internet of intelligence-empowered edge computing**". *IEEE Transactions on Mobile Computing*, 22(11):6327–6342, 2023.
DOI: https://doi.org/10.1109/TMC.2022.3199812.

[10] I. Martinez, A. S. Hafid, and A. Jarray. "**Design, resource management, and evaluation of fog computing systems: A survey**". *IEEE Internet of Things Journal*, 8(4):2494–2516, 2021.
DOI: https://doi.org/10.1109/JIOT.2020.3022699.

[11] M. H. Kashani and E. Mahdipour. "**Load balancing algorithms in fog computing**". *IEEE Transactions on Services Computing*, 16(2):1505–1521, 2023.
DOI: https://doi.org/10.1109/TSC.2022.3174475.

[12] M. Burhan, H. Alam, A. Arsalan, R. A. Rehman, M. Anwar, and M. et al. Faheem. "**A comprehensive survey on the cooperation of fog computing paradigm-based IoT applications: Layered architecture, real-time security issues, and solutions**". *IEEE Access*, 11:73303–73329, 2023.
DOI: https://doi.org/10.1109/ACCESS.2023.3294479.

[13] M. Świechowski, K. Godlewski, B. Sawicki, and J. Mańdziuk. "**Monte Carlo Tree Search: A review of recent modifications and applications**". *Artificial Intelligence Review*, 56(3):2497–2562, 2023.
DOI: https://doi.org/10.1007/s10462-022-10228-y.

[14] A. Samy, I. A. Elgendy, H. Yu, W. Zhang, and H. Zhang. "**Secure task offloading in blockchain-enabled mobile edge computing with deep reinforcement learning**". *IEEE Transactions on Network and Service Management*, 19(4):4872–4887, 2022.
DOI: https://doi.org/10.1109/TNSM.2022.3190493.

[15] P. Rahimi, C. Chrysostomou, H. Pervaiz, V. Vassiliou, and Q. Ni. "**Joint radio resource allocation and beamforming optimization for industrial internet of things in software-defined networking-based virtual fog-radio access network 5G-and-beyond wireless environments**". *IEEE Transactions on Industrial Informatics*, 18(6):4198–4209, 2022.
DOI: https://doi.org/10.1109/TII.2021.3126813.

[16] Z. Yu, J. Hu, G. Min, Z. Wang, W. Miao, and S. Li. "**Privacy-preserving federated deep learning for cooperative hierarchical caching in fog computing**". *IEEE Internet of Things Journal*, 9(22):22246–22255, 2022.
DOI: https://doi.org/10.1109/JIOT.2021.3081480.

[17] Y. Wang, Z. Liu, C. Cai, L. Xue, Y. Ma, and H. et al. Shen. "**Research the optimization method of integrated energy system operation with a multi-subject game**". *Energy*, 245:123305, 2022.
DOI: https://doi.org/10.1016/j.energy.2022.123305.

[18] N. Raveendran, H. Zhang, L. Song, L. C. Wang, C. S. Hong, and Z. Han. "**Pricing and resource allocation optimization for IoT fog computing and NFV: An EPEC and matching based perspective**". *IEEE Transactions on Mobile Computing*, 21(4):1349–1361, 2022.
DOI: https://doi.org/10.1109/TMC.2020.3025189.

[19] Q. Fan and N. Ansari. "**Towards workload balancing in fog computing empowered IoT**". *IEEE Transactions on Network Science and Engineering*, 7(1):253–262, 2020.
DOI: https://doi.org/10.1109/TNSE.2018.2852762.

[20] K. Kaur, A. Singh, and A. Sharma. "**A systematic review on resource provisioning in fog computing**". *Transactions on Emerging Telecommunications Technologies*, 34(4):e4731, 2023.
DOI: https://doi.org/10.1002/ett.4731.

[21] M. Goudarzi, M. Palaniswami, and R. Buyya. "**Scheduling IoT applications in edge and fog computing environments: A taxonomy and future directions**". *ACM Computing Surveys*, 55(7):Article 152, 2022.
DOI: https://doi.org/10.1145/3544836.

[22] M. Kumar, A. Kishor, J. K. Samariya, and A. Y. Zomaya. "**An autonomic workload prediction and resource allocation framework for fog-enabled industrial IoT**". *IEEE Internet of Things Journal*, 10(11):9513–9522, 2023.
DOI: https://doi.org/10.1109/JIOT.2023.3235107.

[23] P. Rahimi, C. Chrysostomou, H. Pervaiz, V. Vassiliou, and Q. Ni. "**Dynamic resource allocation for SDN-based virtual fog-RAN 5G-and-beyond networks**". *IEEE Global Communications Conference (GLOBECOM)*, 2021.
DOI: https://doi.org/10.1109/GLOBECOM46510.2021.9685458.

[24] Z. Chang, L. Liu, X. Guo, and Q. Sheng. "**Dynamic resource allocation and computation offloading for IoT fog computing system**". *IEEE Transactions on Industrial Informatics*, 17(5):3348–3357, 2021.
DOI: https://doi.org/10.1109/TII.2020.2978946.