A Carbon-Aware Task Offloading Framework for Energy-Efficient Fog Computing

***Abstract*—The increasing energy demands of modern computational tasks necessitate efficient resource management strategies that prioritize sustainability. This paper proposes a carbon-aware task offloading framework that optimizes task allocation across computational nodes based on energy availability, latency, and carbon emissions. Each computational node possesses two energy sources: (1) renewable energy (e.g., solar, wind) with a limited supply and (2) gas-based energy, which is unlimited but results in higher carbon emissions. The proposed model prioritizes nodes utilizing renewable energy whenever possible; however, if renewable energy is insufficient or inefficient due to excessive latency and energy consumption, tasks are allocated to nodes powered by gas-based energy.**

To achieve optimal task allocation, we introduce an optimization-based fitness function that evaluates candidate node assignments based on three factors: energy consumption, latency, and carbon emissions. The decision-making process seeks to (1) minimize overall energy consumption, (2) reduce processing and transmission latency, and (3) lower carbon emissions. The proposed framework incorporates a metaheuristic optimization algorithm, such as a Genetic Algorithm (GA) or Particle Swarm Optimization (PSO), to determine the most efficient allocation strategy dynamically.

***Index Terms*—Task scheduling . Offloading · Fog computing . Cloud computing · Gradient Ascent · Meta-heuristic.**

I. Introduction

As computational workloads increase in complexity, optimizing energy usage has become a critical challenge. Traditional task offloading strategies focus on performance metrics such as execution time and system utilization; however, emerging sustainability concerns necessitate a carbon-aware approach that integrates renewable energy prioritization.

This research introduces a multi-objective task scheduling framework that considers:
Energy efficiency: Minimizing the overall power consumption of computational nodes.
Latency optimization: Reducing transmission and processing delays.
Carbon footprint reduction: Favoring nodes with renewable energy sources before utilizing fossil-fuel-based alternatives.

To implement this strategy, we propose a fitness function that quantifies the trade-offs between energy consumption, latency, and carbon emissions. The system uses an optimization algorithm to dynamically allocate tasks to nodes in a way that minimizes energy wastage and environmental impact.

II. Related work

III. The Proposed Method

A. Problem Definition

We consider a time series of task generation with deadlines that must be executed on fog nodes. Each fog node is equipped with two types of energy resources: renewable and Gas-based Energy. The availability of renewable energy varies across nodes and time. Our objective is to allocate tasks to fog nodes efficiently, prioritizing nodes powered by renewable energy whenever possible. If no suitable renewable energy node is available and we will miss the deadline, tasks will be assigned to nodes utilizing Gas-based energy to ensure execution.

B. Computational Node Energy Model

Each computational node i has two energy resources:

- **Renewable Energy** (R_i): A limited source (e.g., solar, wind) with a finite energy budget per time unit.
- **Gas-based Energy** (G_i): An unlimited source but with high carbon emissions.

Each task j has an energy requirement E_{ij} when processed on node i, where the energy source is either renewable or gas-based.

C. Optimization Constraints

$$\sum_j T_j \leq C_i, \quad \forall i$$

Each node can only process a limited number of tasks based on its computational capacity C_i.

$$P_{ij} = 0 \Rightarrow Q_{ij} = 1$$

If a node has no renewable energy available, it must rely on gas-based resources.

IV. Optimization Model and Fitness Function

We adopt a metaheuristic optimization algorithm to dynamically determine the most efficient task allocation strategy. The chosen algorithm optimizes node selection based on our fitness function. The considered metaheuristic techniques include:

- Energy Consumption: Minimizing the overall energy usage.
- Carbon Emissions: Energy Type and Availability Prioritize renewable-powered nodes if energy is available. Reducing the carbon footprint by prioritizing renewable

energy resources. If renewables are exhausted, select gas-powered nodes.
- Computational Resources Ensure the node has enough CPU/memory to handle the task.
- Task Deadline: Ensuring that tasks meet their deadline constraints to maintain system reliability and performance.

Mathematical Model: Define an objective function: Minimize

$$\sum_{i=1}^{N} C_i P_i + G_i Q_i$$

Ci = Carbon cost per unit task on node i
Pi = Probability of selecting renewable energy on node i
Gi = Carbon cost for gas-powered node i
Qi = Probability of selecting a gas-powered node

Ci + Qi = 1 (A task is either processed on renewable or gas) Pi is higher when renewable energy is available.

My idea is evolving into a multi-objective optimization problem, where you aim to minimize energy consumption, latency, and carbon emissions while maximizing the use of renewable energy. The approach will involve a fitness function that evaluates different task allocation decisions and a framework for implementing the scheduling strategy.

Each computational node has:

- **Renewable Energy Availability** (R_i): Limited (e.g., 10 kJ per node).
- **Gas Energy Availability** (G_i): Unlimited, but emits higher carbon emissions.
- **Latency** (L_{ij}): Time taken for node i to process task j.
- **Energy Consumption** (E_{ij}): Energy used when node i processes task j.
- **Carbon Emission** (C_{ij}): Emissions based on energy source.

To evaluate a task allocation decision, we define a fitness function that balances

- **Energy Efficiency**: Prefer nodes with lower energy consumption.
- **Latency Minimization**: Reduce delay when assigning tasks.
- **Carbon Emission Reduction**: Prioritize renewable energy over gas.

The fitness function is:

$$F(T) = \alpha \sum_{i,j} E_{ij} + \beta \sum_{i,j} D_{ij} + \gamma \sum_{i,j} C_{ij}$$

Where:

- E_{ij} = Energy consumed by node i for task j.
- D_{ij} =denotes the penalty for tasks missing deadlines j on node i.
- C_{ij} = Carbon emission when task j is executed on node i.
- α, β, γ = Weights for trade-offs between energy, latency, and emissions.

V. Optimization Algorithm for Task allocation

To solve the optimization problem, we employ a Genetic Algorithm (GA) due to its robustness in handling multi-objective constraints. The steps include:

1) **Initialization**: Tasks arrive in a time-series manner with specific execution requirements and deadlines.
2) **Resource Assessment:** The availability of renewable and non-renewable energy on fog nodes is monitored to ensure efficient task execution.
3) **Fitness Evaluation**: Compute $F(T)$ for each candidate solution.
4) **Selection**: Choose the best-performing solutions.
5) **Crossover & Mutation**: Generate new candidate solutions through genetic operations.
6) **Convergence Check**: If no further improvement is observed, select the best allocation and execute the tasks accordingly.

VI. Performance Evaluation

To evaluate their performances, we compared the proposed PSO (P-PSO), CSA (P-CSA), PSG, PSG-M [**?**] algorithms.

For experimentation, we take into account makespan and energy like compute, transfer, and memory energy on the fog environment made up of a variety of heterogeneous fog nodes in the underlying network topology. To guarantee an equitable comparison between our proposed methodologies and the comparison models, we consciously attempted to align the parameter configurations employed in our simulation with those comparison models. The number of FNs fluctuates from 30 to 90 for numerous trials, and the number of IoT tasks varies from 100 to 500. The infrastructure graph comprises three types of compute nodes: cloud, fog, and edge devices. All the simulations are coded in the Large Energy-Aware Fog (LEAF) simulator, developed for implementing energy solutions with Python [**?**]. The experiments are conducted on a computer with a dual Intel i5 2.40 GHz processor, 12 gigabytes of RAM, and the Windows 10 operating system. Edge IoT devices connect to nearby fog nodes via Wi-Fi (e.g., IEEE 802.11p), which update data flow mappings and communicate with base stations, forming a mesh network. Furthermore, they can connect to the cloud via WAN. WAN links consume more energy per bit; however, direct Wi-Fi communication between fog nodes is more energy-efficient than communication with edge devices due to the constant direct line of sight and the use of energy-efficient, high-throughput access points [**?**]. To calculate MIPS (Millions of Instructions Per Second) for each CPU, we can use the capacity of each heterogeneous fog node between 2000 to 6000 MIPS at a clock frequency of up to 240 MHz. The amount of compute energy consumption for active mode (dynamic), idle mode (static), transfer energy – WAN and Wi-Fi, and memory energy is considered based on the monitoring application described in [**?**].

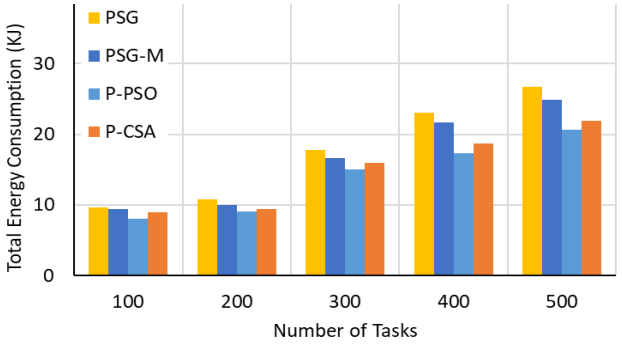
To present the results with high reliability in each value of the mentioned parameters, the algorithm was independently run 30 times for each stated parameter and the mean values are reported. Each run of the proposed algorithm is implemented on a randomly generated simulated workload and node deployment. It should be mentioned that in order to examine the algorithms' performance, the values of the tunable parameters of PSG and PSG-M are taken from [**?**]. Therefore, we employ metrics like makespan, task distribution, iteration time, total energy consumption, and the percentage of optimisation for evaluation.

A. Impact of increasing task count

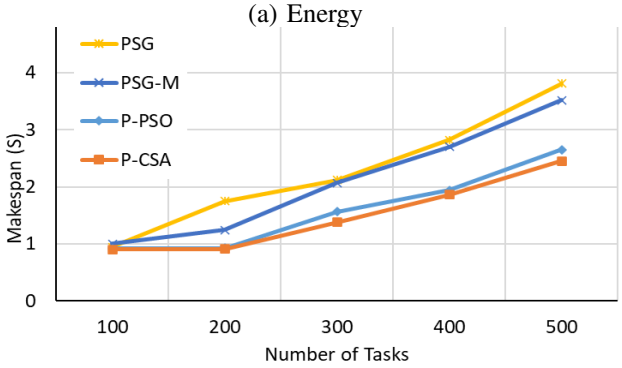
Figure 1 shows how the number of tasks affects algorithm energy consumption. As can be observed in Figure 1, the simulation results demonstrate that an increase in the number of tasks imposes a heavier load on the system in general, leading to increasing energy consumption. Here, we assume the number of FNs to be constantly 60 and the number of tasks was computed from a sparse deployment with 100 tasks to a dense workload consisting of 500 tasks. As Figure 1a shows, with an increment of the number of tasks, the proposed scheme results in more energy consumption. For instance, in P-PSO, energy consumption increases from 8.010 to 20.65 KJ as the number of tasks rises from 100 to 500, requiring more energy to compute all tasks. Similarly, in P-CSA, energy consumption increases from 8.98 to 21.90 KJ. As the Figure indicates, the proposed CSA and PSO method is considerably more energy efficient compared to other approaches, which helps fog nodes to conserve their energy and thus prolongs network lifetime, particularly for dense task deployment in which the problem of lifespan optimisation becomes more complicated than PSG and PSG-M deployments. Moreover, Figure 1b shows the makespan resulting from the proposed method in terms of task density from 100 to 500 tasks with 60 fog nodes. As Figure 1b shows, with an increment of tasks in the workload, the makespan increases to compute all tasks. For instance, with 500 tasks, the makespan for P-CSA and P-PSO are 2.45 and 2.65 seconds, respectively, compared to PSG and PSG-M, which are more than 3.5 seconds.

B. Impact of increasing node count

As Figure 2 indicates, how the number of nodes affects the algorithm's energy consumption. Here, we assume the number of nodes was computed from a sparse deployment with 30 to 90 nodes to a dense workload consisting of 300 tasks. The proposed scheme significantly reduces energy consumption for resource allocation optimisation and thus, prolongs the lifetime of fog nodes. The amount of energy consumption of the P-PSO and P-CSA algorithm for 90 nodes compared with other PSG and PSG-M algorithms is 15.60, 17.90, 21, and 20, respectively. Since each node uses energy in the idle mode, increasing the number of nodes hurts energy consumption. Consequently, the amount of energy consumed for the scenario with more nodes is higher, as you can see in Figure 2a with different numbers of nodes and compared to Figure 1a with different numbers of tasks. Moreover, Figure 2b illustrates the makespan for various numbers of nodes.



(a) Energy



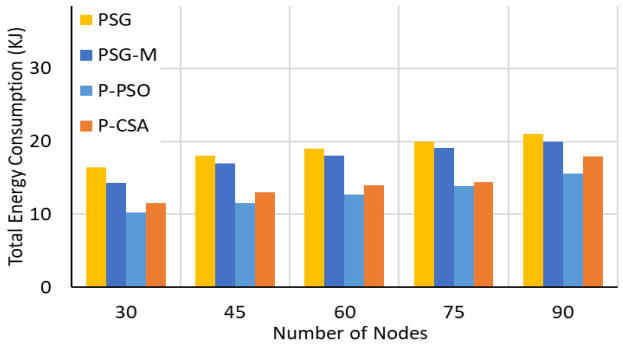
(b) Makespan

Fig. 1. Performance comparison of algorithms with a different number of tasks; simulation results for 60 Fog Nodes; **(a)** total energy consumption, **(b)** total makespan

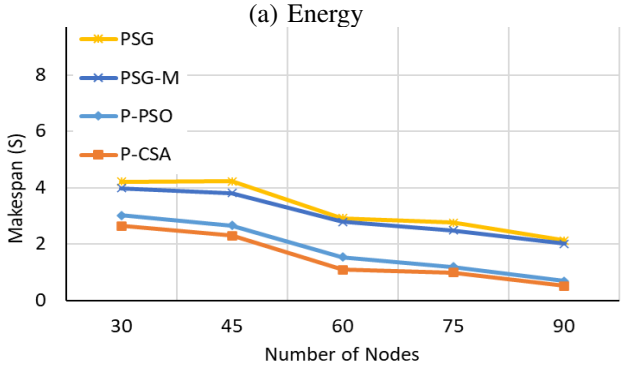
C. Impact of dynamic weight on iteration

In order to evaluate the proposed approach and compare its performance to other comparable approaches, we used 400 tasks and 10 FNs in our experiments. It should be mentioned that in order to examine the algorithms' performance, the values of the tunable parameters of GA are taken from [**?**]. The convergence behaviour of the suggested algorithm is shown in Figure 3 in comparison to the GA method. The result demonstrates the effective operation of the suggested algorithm, which is distinguished by consistent and dependable convergence behaviour towards the optimal solution. Furthermore, when compared to the other method, the suggested algorithm also reaches the best fitness value at a faster rate of convergence. The performance scenarios of the proposed method in various contexts are examined and analysed.

Figure 3 shows the convergence behaviour rate of iterations for the best candidate solution – global best – and then, results from the proposed method in terms of four scenarios. A convergence behavior of an algorithm, it provides insights into the reliability and stability of the algorithm's performance. As Figure 3 shows, with an increment of tasks in the workload, the swarm intelligence algorithms result in more iterations to find the best solution. In terms of convergence speed and fitness value, as Figure 3 shows, P-PSO is superior to those of all other approaches, while P-CSA performs well but marginally worse than P-PSO. GA algorithm, on the other hand, converges



(a) Energy



(b) Makespan

Fig. 2. Performance comparison of algorithms with a different number of fog nodes; simulation results for 300 tasks; **(a)** total energy consumption, **(b)** total makespan

early but plateaus at a greater fitness level. As Figure 3 illustrates, the impact of the proposed scheme on the highly dense workload is decreasing more than that of middle-dense and sparse task deployment.

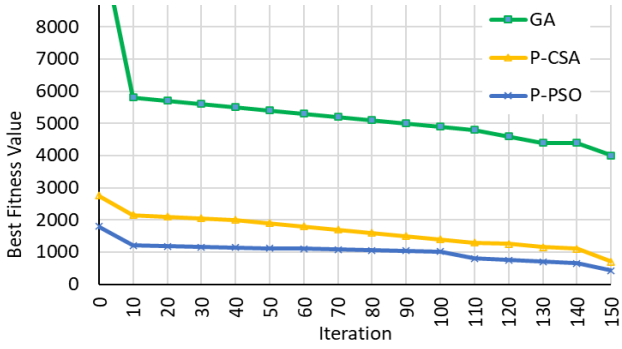


Fig. 3. Iteration-wise Convergence of proposed and other algorithms.

D. Impact of dynamic weight on task distribution

The load of a running FN is calculated based on the number of tasks assigned to it. Therefore, the load of a FN in the time period k is calculated from the number of allocated tasks. As much as the amount of tasks assigned to FNs is close to each other, it can be said that the distribution of task has led to the load balance of FNs [**?**]. It should be mentioned that in order to examine the algorithms' performance, the values of the tunable parameters of HHO are taken from [**?**].

One of the goals of the proposed method is to balance the load – distribute tasks across different nodes simultaneously – in the scheduling process based on the dynamic weight. Figure 4 illustrates the load distribution rate for 1000 tasks and 100 FNs. It is important to mention that the load of each fog node is determined by dividing the number of assigned tasks by the total number of tasks. Also, the load for each FN is provided based on the best solution from P-PSO and P-CSA. As Figure 4 shows, each FN computed a number of tasks with energy and time consumption. It is noticeable that the impact of the proposed scheme on load for each FN is highly valuable. For example, the load for P-PSO algorithm in Figure 4 is between the minimum and maximum task size rate like 0.0096 and 0.0103, while in the HHO algorithm, it is between 0.0094 and 0.0106, respectively. In other words, coefficients used in the proposed approaches like P-PSO and P-CSA can help to have a load balance in task allocation, stopping to allocate tasks to low capacity FN. This reinforces the effectiveness of P-PSO and P-CSA in improving load balancing, with reducing fluctuations in proposed algorithms, leading to distributing tasks more evenly, preventing the overloading of weaker fog nodes with limited computing resources. Therefore, the average rate of task allocation is not considerably increased while the lifespan is enhanced since energy consumption increases at a significantly higher rate than the rise in CPU usage, demonstrating a non-linear relationship between resource use and energy demand.

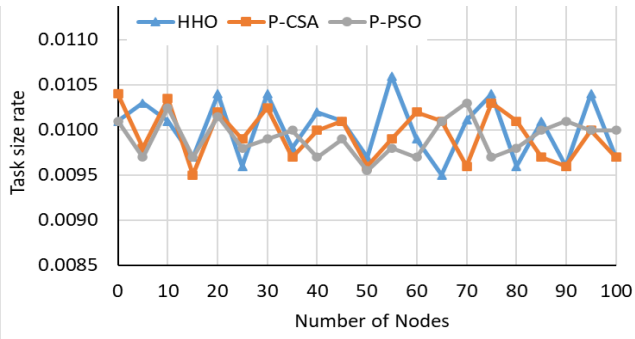


Fig. 4. Task distribution on FNs.

VII. CONCLUSIONS AND FUTURE WORK

In conclusion, this paper addressed the critical challenge of task offloading optimisation in heterogeneous fog computing environments. By introducing a novel energy-centric task offloading algorithm based on meta-heuristic like PSO and CSA and gradient ascent approaches, we effectively distributed computational loads across fog nodes, reducing transfer time and energy consumption. Our algorithm incorporated multiple objectives to optimise resource utilisation and achieve energy efficiency. The use of gradient-based dynamic coefficients ensures flexible and adaptive load distribution, while maintaining CPU usage within an optimal range, thus further minimising energy consumption. In other words, the proposed scheme decreased the number of iterations to obtain an optimal solution for task allocation using dynamic coefficients. Future work will explore distributed communication models to address task scheduling in decentralised fog environments. Our ultimate aim is to develop a distributed multi-objective task framework.

References