TELKOMNIKA Telecommunication Computing Electronics and Control
Vol. 24, No. 2, April 2026, pp. 514~526
ISSN: 1693-6930, DOI: 10.12928/TELKOMNIKA.v24i2.27293

Hybrid PSO-WOA approach for an efficient task offloading in mobile edge computing

Fatima Z. Cherhabil, Sonia-Sabrina Bendib, Maamar Sedrati, Chahrazed Adouane, Sifeddine Benflis
Department of Computer Science, Mustafa Benboulaid University (Batna 2), Batna, Algeria

Article Info

Article history:

Received Jun 16, 2025
Revised Dec 4, 2025
Accepted Jan 30, 2026

Keywords:

Cost
Delay
Energy consumption
Mobile edge computing
Multi-objective optimization
Task offloading
Weighted sum

ABSTRACT

Offering a promising solution for latency-sensitive and resource-constrained internet of things (IoT) applications, mobile edge computing (MEC) extends cloud capabilities to the network edge. However, the decentralized nature of edge resources, coupled with stringent latency requirements and IoT energy constraints, presents significant challenges for efficient task offloading. Integrating IoT with MEC and software-defined networking (SDN) can meet the growing demands for low latency and energy-aware resource management. This paper proposes a hybrid evolutionary algorithm combining whale optimization algorithm (WOA) and particle swarm optimization (PSO) with crossover, mutation, and Lévy flight operators (CML) to balance exploration and exploitation. The algorithm minimizes a weighted sum function (energy 35%, delay 35%, and monetary cost 30%) for joint task offloading and resource allocation in SDN-enabled MEC environments. The proposed approach is evaluated against six well-known metaheuristics, analyzing performance across various metrics including scalability with up to 100 users. Experimental results, validated by non-parametric statistical tests, demonstrate that the proposed algorithm achieves statistically significant improvements in convergence speed, solution quality, and scalability, making it a robust and promising candidate for real-time MEC task scheduling.

Corresponding Author:

Fatima Z. Cherhabil
Department of Computer Science, Mustafa Benboulaid University (Batna 2)
53, Constantine Avenue, Fésdis, 05000, Batna, Algeria
Email: f.cherhabil@univ-batna2.dz

1. INTRODUCTION

With the rapid growth of wearable devices and internet of things (IoT) applications, efficiently processing delay-sensitive and resource-intensive tasks has become a major challenge. These applications impose stringent quality of service requirements due to factors such as mobility, interactive environments, and the need for real-time responsiveness. Due to limited resources and energy, IoT devices cannot meet these stringent performance demands. Offloading the whole or part of a task to another processor or server in proximity can be used to accelerate resource-intensive or latency-sensitive applications and reduce energy consumption when compared to cloud computing [1], [2].

Mobile edge computing (MEC) has emerged as a promising approach that brings computation closer to IoT devices, typically within the radio access network. It can achieve a better trade-off between delay-sensitive and computation-intensive tasks [3]. When integrated with software-defined networking (SDN), a recently proposed technology that separates the control plane from the data plane, MEC gains additional flexibility in resource management through centralized control and dynamic network configuration [4], [5].

Despite these advantages, the dynamic and heterogeneous nature of IoT environments makes the joint task offloading and resource allocation problem highly complex and non-deterministic polynomial-time hard (NP-hard) [6], especially when the number of users increases. As the number of users and tasks increases, finding optimal offloading decisions becomes increasingly challenging due to interdependencies between energy consumption, execution delay, and monetary cost. Traditional deterministic methods struggle to adapt to such dynamic conditions.

Evolutionary and swarm-based algorithms, metaheuristics known for their global search capability and adaptability, have shown great potential in addressing such problems [7]. They are emerging as a dominant approach due to their ability to handle NP-hard problems. For instance, genetic algorithms (GA) have been used to optimize transmission power and execution frequency [8] and to minimize task overhead in internet of vehicles (IoV) systems [9]. Zhu and Wen [10], an improved version of GA (IGA) using knowledge-based crossover was introduced. The paper focused on comparing the proposed algorithm with the most commonly used benchmarks, which are all local and all offloaded execution strategies. However, GAs often suffer from high computational complexity [11].

A binary version of the cuckoo search algorithm (CS) was proposed in [12] for offloading decision-making, focusing on minimizing time, energy, and cost. The study highlighted how multiple parameters, such as the number of mobile devices and tasks, influence the effectiveness of the offloading process. Meanwhile, Abbas *et al.* [13] compared grey wolves optimization (GWO), ant colony optimization (ACO) and whale optimization algorithm (WOA) to find an optimal selection of offloading tasks. Simulation results showed that the performance of GWO is relatively much better than ACO and WOA. However, both works focused their experimental tests only on an environment of a single edge node and a number of end-devices. Artificial bee colony (ABC) algorithm [14] have demonstrated effectiveness in balancing latency and energy in a proposed three-tier edge-cloud integration framework. Particle swarm optimization (PSO) was used in [15] to address task dependencies in job-divided computation offloading with multiple users and MEC servers and multi-population cooperative elite algorithm (MCE-PSO). A binary version of PSO was also applied in [16] for resource allocation and offloading strategy optimization in multi-tier multi-MEC-server architectures within 5G heterogeneous networks.

WOA, known for its exploitation capabilities, has been integrated with other evolutionary techniques to enhance performance. Li *et al.* [17], WOA was integrated with differential evolution (DE) and immune system to improve the searching strategy of the whale and enhance the efficiency of dependent task offloading in MEC environments, while in [18] authors decomposed the problem of computation offloading in non-orthogonal multiple access (NOMA) based MEC system into sub-problems. They solved them using convex optimization and the use of a gradient-free swarm intelligence approach of WOA. However, the simulations were conducted with a single server and a very small number of users (5-25). Zhang and Tuo [19], authors had merged WOA with Lévy flight and GWO to improve the population initialization and alpha-wolf selection steps of the GWO algorithm. The system used a multi-server and multi-user vehicular task offloading with a selective offloading, which is, in their case, the ability to execute the task locally, on edge server, or an idle vehicle.

Other evolutionary and swarm intelligence algorithms have been explored. Gorilla troops optimization (GTO) was proposed and improved in [20] to solve the dependent task-offloading problem in a multi-server MEC environment with the same three objectives, namely, the completion time, energy consumption, and monetary cost. Furthermore, biogeography-based optimization (BBO) was employed in [21] to solve the task offloading issues for edge servers and considering the central cloud.

It is worth noting that the architectural setups differ, making direct numerical comparisons challenging even for experimental evaluation. As summarized in Table 1, the majority of prior works optimized only energy and delay objectives and they did not consider the payment cost objective. They used a binary offloading, which makes it harder to compare with the proposed algorithm that uses a partial offloading. This approach enables optimization of all the objectives by exploiting the benefits of both sides, edge servers and end-devices. Moreover, while effective, the above studies often struggle with exploration-exploitation balance or premature convergence, especially in multi-objective contexts. Thus, there is still a need for an algorithm that effectively balances exploration and exploitation while simultaneously optimizing all three performance metrics.

The current work distinguishes itself by proposing a novel hybrid algorithm that combines the strengths of PSO and WOA, enhanced with crossover, mutation, and Lévy flight (CML) operators to provide a better balance between global exploration and local exploitation. The hybrid PSO-WOA (CML) algorithm jointly optimizes energy consumption, execution delay, and monetary cost in an SDN-enabled MEC framework, achieving superior convergence behavior and scalability across various system sizes. Our contributions include:

- An SDN-enabled MEC framework that simplifies management and addresses IoT heterogeneity and mobility through centralized control.

- A novel hybrid PSO-WOA (CML) algorithm integrating WOA's spiral update, PSO's velocity-based learning, and CML operators for enhanced exploration and convergence.
- Comprehensive benchmarking against six metaheuristics under identical conditions, supported by statistical validation (Friedman and Wilcoxon tests).
- Scalability analysis showing the algorithm's robust performance and stable execution time (ET) as the number of users increases from 20 to 100.

The rest of this paper is structured as follows: section 2 presents the method, detailing the system model, problem formulation, and the proposed hybrid algorithm. Section 3 provides the results and discussion, where we present the experimental setup and conduct a comparative analysis of performance, scalability, and statistical significance. Finally, section 4 offers the conclusion and outlines future research directions.

Table 1. Comparison with related works

Reference	Algorithm	Energy	Delay	Cost	Offloading degree	Multi-server	Multi-user
[8]	GA	✓	✓	✗	Selective	✓	✓
[9]	GA	✓	✓	✗	Binary	✓	✓
[10]	GA	✓	✓	✗	Binary	✗	✓
[12]	CSA	✓	✓	✓	Binary	✗	✓
[13]	GWO	✓	✓	✗	Binary	✓	✓
[14]	ABC	✓	✓	✗	Selective	✓	✓
[15]	PSO	✓	✓	✗	Binary	✓	✓
[16]	PSO	✓	✓	✗	Binary	✓	✓
[17]	WOA + DE	✓	✓	✓	Binary	✓	✓
[18]	WOA + Convex optimization	✓	✓	✓	Binary	✗	✓
[19]	WOA + GWO	✓	✓	✗	Selective	✓	✓
[20]	GTO	✓	✓	✓	Binary	✓	✗
[21]	BBO	✓	✓	✓	Selective	✓	✓
Proposed algorithm	PSO + WOA	✓	✓	✓	Partial	✓	✓

2. METHOD

2.1. System model

We consider a multi-user, multi-server SDN-enabled MEC environment, where a set of IoT devices $N = \{U_1, \dots, U_N\}$ randomly distributed within a 100×100 m^2 simulation coverage area. These devices communicate wirelessly with a macro base station (BS) equipped with an SDN controller, which orchestrates task offloading to a set of edge servers $M = \{S_1, \dots, S_M\}$, connected via high-speed wired links.

The SDN controller, with its network programming capabilities, stands out as a natural candidate for orchestrating the network, services, and devices by hiding the complexities of the heterogeneous environment from end-devices [22]. IoT devices periodically send task metadata and network status to the controller, which determines the offloading strategy using a hybrid evolutionary optimization algorithm.

Each task is processed either locally, offloaded to an edge server, or partitioned between both. The offloading strategy aims to minimize energy consumption, delay, and monetary costs, with decisions encoded in two vectors:

- $\alpha = (\alpha_1, \dots, \alpha_N)$: Continuous values representing the offloading ratio for each task;
- $\beta = (\beta_1, \dots, \beta_N)$: Discrete values indicating the server assigned to each task.

If a task is offloaded ($\alpha_i > 0$), then $\beta_i = k, i \in \{1, \dots, M\}$ designates the chosen server. For local execution, $\alpha_i = 0$ and $\beta_i = 0$.

2.2. Communication model

Wireless transmission from IoT devices to the BS occurs over a shared radio spectrum with total bandwidth B [16], [23]. The uplink data rate for device U_i is modeled as:

$$R_i = B\ log_2\left(1 + \frac{P_i H_i}{\sigma^2 + \sum_{j=1, j\neq i}^{N} P_j H_j}\right) \quad (1)$$

where P_i is the transmission power, H_i is the channel gain, σ^2 is the channel noise, and the term $\sum_{j=1, j\neq i}^{N} P_j H_j$ represents the effects from other IoT devices.

2.3. Computation model

Each task is defined by the triplet $Q_i = (D_i, C_i, T_i^{max})$, where D_i defines the input data size, C_i is the required CPU cycles, and T^{max} is deadline. The delay and energy consumption in the local execution are typically [10] given as:

$$T_i^{local} = \frac{C_i}{f_i} \quad (2)$$

$$E_i^{local} = \kappa^{user} (f_i)^2 C_i \quad (3)$$

where κ^{user} a constant based on device architecture and f_i is the computing capability of device U_i.

For offloaded tasks, the transmission time and energy consumption between device U_i and the BS are defined as:

$$T_i^{Trans} = \frac{D_i}{R_i} \quad (4)$$

$$E_i^{Trans} = \frac{P_i D_i}{R_i} \quad (5)$$

The ET and cost on an edge server S_{β_i} are given by:

$$T_i^{Edge} = \frac{C_i}{f^{serv_{\beta_i}}} \quad (6)$$

$$C_i^{Edge} = Cost(\beta_i) * C_i \quad (7)$$

here $f^{serv_{\beta_i}}$ represents the MEC server (S_{β_i}) computational load allocated to device U_i, and $Cost(\beta_i)$ is the unit cost per cycle charged by the server β_i.

The final stage involves downloading the results from the edge server to the device. This time is ignored due to the small output size compared to input data, which is a common simplification in the literature [8]. The total response time, energy consumption, and monetary cost are given by:

$$T^{total} = \sum_{i=1}^{N} \left[(1 - \alpha_i) T_i^{local} + \alpha_i (T_i^{Trans} + T_i^{Edge}) \right] \quad (8)$$

$$E^{total} = \sum_{i=1}^{N} \left[(1 - \alpha_i) E_i^{local} + \alpha_i (E_i^{Trans}) \right] \quad (9)$$

$$C^{total} = \sum_{i=1}^{N} \left[\alpha_i (C_i^{Edge}) \right] \quad (10)$$

2.4. Problem formulation

The goal is to find the optimal vectors α and β that minimize the weighted sum of energy, delay, and monetary cost with $\mu_1 = 0.35$, $\mu_2 = 0.35$, and $\mu_3 = 0.3$ predefined weights based on the preferences of the decision-maker.

$$P: \underset{(\alpha,\beta)}{MIN} (\mu_1 E^{total} + \mu_2 T^{total} + \mu_3 C^{total}) \quad (11)$$

Subject to:

C1: $\alpha_i \in [0, 1], \forall i \in (1, \dots, N)$
C2: $\beta_i \in \{0, 1, \dots, M\}, \forall i \in (1, \dots, N)$
C3: $Load_j = \sum_{i=1}^{N} \left[\delta_{j,\beta_i} \alpha_i C_i \right] \leq f^{serv_{max}}, \forall j \in (1, \dots, M)$
C4: $f_i > 0, \forall i \in (1, \dots, N)$
C5: $\left[(1 - \alpha_i) T_i^{local} + \alpha_i (T_i^{Trans} + T_i^{Edge}) \right] \leq T_i^{max}, \forall i \in (1, \dots, N)$

where: δ_{j,β_i} is the kronecker delta function, which is equal to 1 if $j = \beta_i$ (indicating task Q_i is assigned to server j), and 0 otherwise

The constraint C1 ensures partial or full offloading; C2 enforces valid server selection, C3 ensures that the cumulative workload assigned to each server does not exceed its maximum computational capacity ($f^{serv_{max}}$), preventing overloading. In practice, if this condition is violated, a penalty proportional

to the overload ($\lambda_E \cdot (Load_j - f^{serv_{max}})$) is added to the remote execution energy of all tasks mapped to that server, discouraging infeasible allocations. C4 ensures the constraint of device local capacity (f_i). Finally, for constraint C5, which ensures task deadlines, if the local or remote response time exceeds the task deadline (T_i^{max}), the value is scaled by a penalty factor ($\lambda_T > 1$), degrading the overall fitness as follows:

$$T_i^{penalized} = T_i + \lambda_T \cdot (T_i - T_i^{max}), T_i > T_i^{max} \quad (12)$$

This approach differentiates between small and large violations, providing a smoother search landscape and improving convergence. Given the NP-hard nature of the formulated problem [6], exact methods become impractical for large-scale systems. Thus, metaheuristic and evolutionary algorithms are adopted to obtain near-optimal solutions within reasonable computation time.

2.5. Proposed task offloading and resource allocation algorithm

Evolutionary approaches, in practice, converge faster to the neighborhood of an optimal solution and can be very effective if the domain knowledge is exploited [24]. Among them WOA, which is inspired by the bubble-net hunting strategy of humpback whales, alternates between encircling prey, spiral movement, and global search phases for robust solution exploration and fine-tuned exploitation [25]. On the other hand, PSO emulates the collective and intelligent behavior of bird flocking or fish schooling, where particles (solutions) adjust their positions based on personal and group bests, facilitating efficient searches [26]. The hybrid PSO-WOA (CML) algorithm combines PSO's social learning and WOA's spiral/encircling to balance exploration and exploitation. It maintains diversity via mutation/crossover and escapes local optima using CS Lévy flight jamps, while dynamically adapts parameters for convergence speed. Each particle is updated using both PSO and WOA operators, followed by CML evolutionary operators (mutation, crossover, Lévy flight) probabilistically. The algorithm workflow is summarized as follows.

2.5.1. Initialization and solution representation

The process begins by initializing a population of $nPop$ particles and velocities. Each particle X_i represents a potential solution and is composed of two distinct components, reflecting the mixed-variable nature of the problem:

- Offloading decisions (alpha): a continuous vector $X_i.\alpha = (\alpha_1, \ldots, \alpha_N)$, where $\alpha_i \in [0, 1]$ represents the portion of task i to be offloaded.
- Server assignments (beta): a discrete vector $X_i.\beta = (\beta_1, \ldots, \beta_N)$, where $\beta_i \in \{1, \ldots, M\}$ is the integer identifier of the MEC server assigned to task i.

This explicit separation allows for the correct application of continuous and discrete optimization operators to the respective parts of the solution.

2.5.2. Hybridization strategy

Each particle's velocity is adjusted by considering PSO three components [26]: the influence of its previous velocity (V_i), its personal best position ($pBest_i$), and the global best position ($gBest$). This is expressed as:

$$V_i(t+1) = w.V_i(t) + c_1.r_1.(pBest_i - X_i(t)) + c_2.r_2.(gBest - X_i(t)) \quad (13)$$

where: w is the inertia weight, c_1 cognitive factor, c_2 social factor, and both (r_1, r_2) are random numbers in $[0, 1]$ to introduce stochastic behavior.

The particle's position is then updated using WOA-inspired mechanisms [25], selected based on a random value $r \in [0,1]$:

- Shrinking encircling mechanism (if $r < 0.5$): update the position using the formula:

$$X_i(t) = gBest - A \cdot D \quad (14)$$

where $A = 2a \cdot r - a$ is a control parameter, $D = |C \cdot gBest - X_i(t)|, C = 2r$ and a decreases from 2 to 0 over iterations.

- Spiral updating Mechanism (if $r \geq 0.5$): use the spiral equation to update the position:

$$X_i(t) = D'.e^{bl}.cos(2\pi l) + gBest \quad (15)$$

where $D' = |gBest - X_i(t)|$, b is a constant, and l is a random number in [−1, 1].

Next, the updated velocity is applied to further refine the particle's position:

$$X_i(t+1) = X_i(t) + V_i(t+1) \quad (16)$$

2.5.3. CML enhancement operators

To enhance the search space, the algorithm further improves solution space exploration by applying one of the three CML operators based on predefined probabilities (p_Cross, p_Mut, and p_Levy):

- Crossover: combines the positions of two particles to generate new candidate solutions using blend crossover for the continuous α vectors and a uniform crossover for discrete β vectors.
- Mutation: slightly modifies a particle's position to explore local regions by using Gaussian mutation to α vector and random-reset mutation for β vector.
- Lévy flight: enabling occasional long-distance jumps in the search space to help the algorithm escaping from local optima. Thus, a Lévy step is calculated and added to the α vector.

2.5.4. Dynamic parameter tuning

To avoid static behavior across generations, key PSO parameters are adapted dynamically over the iterations:

- The inertia weight w is linearly decreased using a damping value d to shift the search from exploration to exploitation.
- The cognitive $c1$ is gradually decreased while the social coefficient $c2$ is increased to shift the search focus from individual experiences to global learning.

2.5.5. Fitness evaluation and optimal solution update

During every iteration, boundary constraints for α_i are enforced by clipping values to the range [0, 1], while β_i values are rounded to the nearest integer and clipped to the range [0, M]. Then, the fitness of each new particle is evaluated based on the three objectives (energy, delay, and monetary cost) with weights set respectively as 0.35, 0.35, and 0.3. Then, the $pBest$ and $gBest$ values are updated until a maximum number of iterations is reached.

2.5.6. Pseudocode of the proposed algorithm

The complete structure of PSO-WOA (CML) algorithm is summarized in the pseudocode below:

- Inputs: task parameters (D_i, C_i, T^{max}), device/server capacities (f_i, $f^{serv_{\beta_i}}$), and network conditions (p_i, h_i, σ^2).
- Outputs: optimal task offloading decision (vector $\alpha*$) and server assignments (vector $\beta*$) minimizing energy, delay, and monetary cost.

Algorithm 1. PSO-WOA (CML) for MEC task offloading

```
// --- Initialization ---
Initialize population (P) of nPop particles with random α and β vectors
Initialize empty velocity vectors for each particle
FOR each particle Xi in P DO
    Calculate fitness F(Xi) = (0.35*Total_Energy + 0.35*Total_Delay + 0.3* Total_Cost)
    Initialize personal best pBest_i with Xi and F(Xi)
END FOR
Find global best (gBest) particle from P // the minimum value of F(Xi)
// --- Main Loop ---
FOR t = 1 TO MaxIter DO
  FOR each particle Xi in P DO
    // --- Core hybrid update ---
    // -- Calculate PSO velocity vector --
     Vi(t) = calculate_pso_velocity(Xi, pBest_i, gBest, w, c1, c2) with eq. 13
    // -- WOA strategic repositioning --
      r = rand()
      IF r < 0.5 THEN
          Xi_new = shrinking_encircling (Xi, gBest, a) using eq. 14
      ELSE
          Xi_new = spiral_updating mechanism(Xi, gBest, a) using eq. 15
```

```
            ENDIF
           // -- Final position update with PSO velocity (Refinement) --
            Xi = Xi_new + Vi(t)
           // --- CML enhancement operators ---
          r = rand()
          IF r < p_Mut THEN
              Apply mutation to Xi (Gaussian for α, random-reset for β)
          ELSE IF r < p_Mut + p_Cross THEN
              Select two parents P1, P2 from P
              Apply crossover to Xi, P1, P2 (blend for α, uniform for β)
          ELSE
              Apply Lévy flight to the α vector of Xi
          END IF
          // --- Evaluation and updates ---
          Enforce boundary constraints on Xi.α and Xi.β
          Evaluate fitness F(Xi)
          IF F(Xi) < F(pBest_i) THEN pBest_i = Xi
          IF F(Xi) < F(gBest) THEN gBest = Xi
      END FOR
       // --- Dynamic parameter tuning ---
           Decrease w, c1, and increase c2
  END FOR
  RETURN gBest // The optimal solution
```

2.5.7. Complexity analysis

Let N be number of users, M number of servers, $nPop$ population size, and I number of iterations.

a. Fitness evaluation (per particle): a naive calculation of the fitness function in (11) is dominated by two components:
   - Calculating all N transmission rates (1), which requires computing the interference sum $\sum_{i=1}^{N} P_i H_i$ for each user, resulting in an $O\ (N^2)$ complexity.
   - Verifying the server load constraint (C3), which, if implemented by checking each of the M servers, requires summing N user loads, resulting in an $O(N \times M)$ complexity.

   In our implementation, we reduce this cost significantly with two optimizations. First, the $O\ (N^2)$ interference term is reduced to $O(N)$ by pre-calculating the global interference sum $\sum_{i=1}^{N} P_i H_i$ once and then finding each user's specific interference via $O(1)$ subtraction. Second, the $O(N \times M)$ load check is reduced by initializing an M-sized array for server loads ($O(M)$), then updating it in a single pass over all N users ($O(N)$). Consequently, the optimized fitness evaluation cost (including all objectives and constraint checks) for a single particle is $O(N + (M + N)) = O(M + N)$. Given that $M \ll N$ in our scenarios, this cost simplifies to $O(N)$.
b. PSO/WOA updates: updating velocities and positions of all particles takes $O\ (nPop \cdot N)$.
c. CML operators*:* mutation, crossover, and Lévy flight modify particle positions and they require $O\ (nPop \cdot N)$.
d. Updating best positions: $O(nPop)$
e. Total complexity: considering all iterations, the practical computational complexity of the algorithm is $O(I \cdot nPop \cdot N)$. Furthermore, scalability to very large-scale systems can be improved through parallelized evaluation, as fitness computations for particles are independent. Parameter tuning also can reduce constants.
f. Memory complexity: the algorithm must store the position, velocity, and $pBest$ for all $nPop$ particles. Since each particle's representation (α and β vectors) is of size N, the total memory complexity is $O\ (nPop \cdot N)$, which remains modest for typical MEC settings.

3. RESULTS AND DISCUSSION

The performance of six state-of-the-art metaheuristic algorithms (GAs [8], [9], [10], CS [12], ABC [14], GWO [13], PSO [15], and WOA [25]) is evaluated and compared against the proposed hybrid algorithm. The objective is to minimize a composite cost (C-Cost) function that incorporates energy consumption, execution delay, and monetary cost.

3.1. Experimental setup

Simulations were conducted in a network with 20 IoT users and 3 MEC servers managed by an SDN controller. Each algorithm was executed over 200 iterations and 10 independent runs to ensure statistical reliability. Table 2 summarizes the system parameters, which are based on common values found in the literature. All algorithms were implemented in MATLAB 2013a on an Intel Core i7, 2.11 GHz processor with 20 GB RAM to ensure fair comparison.

Table 2. System parameters

Description	Symbol	Value
Input data size of a task to be executed (Mb)	D_i	[0.1, 0.5]
CPU cycles required to complete each task (Gcycles)	C_i	[0.2, 1.2]
Maximum tolerable delay by each task (sec)	T_i^{max}	[1, 5]
Local computing capacity of each user (GHz)	fi	[0.2, 1.0]
Effective capacitance constant for end-device architecture	κ^{user}	5×10^{-27}
Bandwidth of the wireless channel (MHz)	B	50
Uplink transmit power of each user to MEC server (mW)	P_i	[5, 30]
Channel gain between device U_i and BS (dBm)	H_i	[10, 25]
Gaussian channel noise	σ^2	10^{-10}
Portion of MEC server computational load per user (GHz)	$f^{serv_{\beta_i}}$	[5, 8]
Maximum server computing capacity (GHz)	$f^{serv_{max}}$	35
Unit cost per cycle charged by server β_i	$Cost(\beta_i)$	[0.5, 1.5]
Initial inertia coefficient	w	1
Inertia coefficient damping factor	d	0.99
Personal acceleration coefficient	$c1$	2.5
Social acceleration coefficient	$c2$	1.5
Probability of applying crossover operator	$p.Cross$	0.3
Probability of applying mutation operator	$p.Mut$	0.4
Probability of applying Lévy flight operator	$p.Levy$	0.3
Population size	$nPop$	100

3.2. Performance evaluation

To provide a comprehensive analysis, we evaluated the algorithms based on convergence behavior, final solution quality, and statistical significance. Figure 1 illustrates the convergence trends and Table 3 summarizes the average results. The evaluation considers the mean ± standard deviation of the final C-Cost, the approximate number of iterations required to converge (Conv_Iter), the ET of a full run, and the effective execution time (EET), which reflects the actual runtime required to find the final best solution and reach convergence. The proposed algorithm is further compared with the baseline strategies; namely "All local mode" where all the tasks are performed in end-devices and the "All offloaded mode" where all the tasks are performed in edge servers [10].

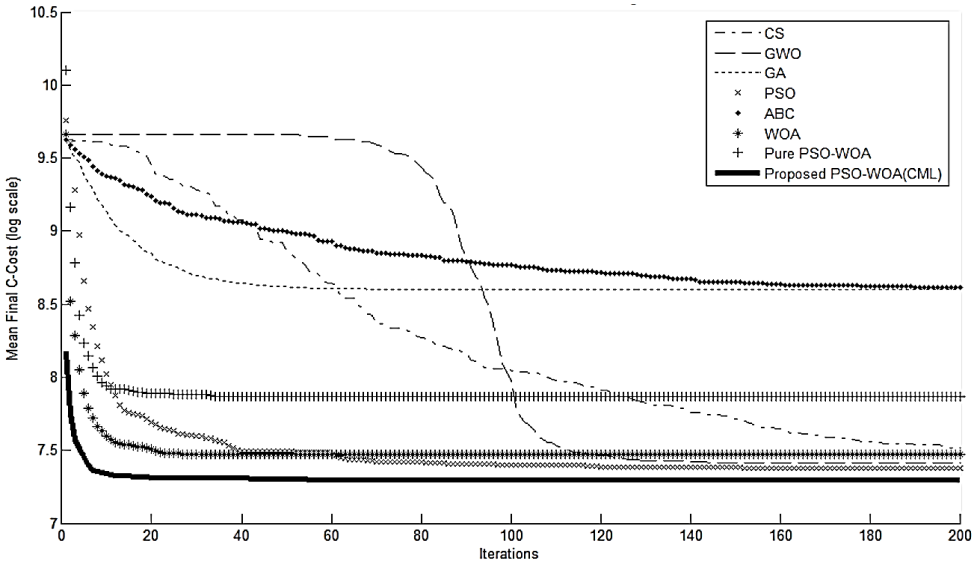

Figure 1. Convergence behavior of proposed hybrid algorithm vs. standard metaheuristics

Table 3. Performance overview

Algorithm	Mean ± std (final C-cost)	ET (sec)	Conv_Iter	EET (sec)
CS	7.517 ± 0.675	2.094	~186	1.964
GWO	7.415 ± 0.334	2.362	~137	1.624
GA	8.600 ± 0.510	3.045	~89	1.382
PSO	7.375 ± 1.088	3.326	~107	1.794
ABC	8.615 ± 1.224	3.429	~135	2.294
WOA	7.474 ± 0.087	2.578	~19	0.247
Pure PSO-WOA	7.871 ± 0.804	4.411	~40	0.723
PSO-WOA CML	7.296 ± 0.739	3.763	~54	1.014
All local	9.670 ± 1.700	-	-	-
All offloaded	13.675 ± 1.216	-	-	-

In terms of solution quality, the proposed PSO-WOA CML algorithm achieved the lowest average final C-Cost (7.296), outperforming all competing algorithms, including GWO (7.415), PSO (7.375), and WOA (7.474). It also outperformed the pure PSO-WOA (7.871); indicating that naive hybridization may not always yield benefits and can degrade balance between exploration and exploitation, whereas the integration of CML operators significantly improved optimization efficiency. Relative to the baseline strategies, PSO-WOA CML reduced the final solution by 24.6% compared to all local execution and by more than 46% compared to full offloading, confirming its strong multi-objective optimization capability.

Regarding convergence behavior, WOA demonstrated the fastest convergence (~19 iterations) but suffered from premature stagnation, yielding suboptimal solutions. CS converged extremely slowly (~186 iterations) with minimal gains. In contrast, the PSO-WOA CML algorithm achieved a balanced behavior, converging within ~54 iterations, which is faster than GA (~89) and PSO (~108) while still providing the best overall solution quality. This reflects the hybrid algorithm's ability to maintain exploration in early stages and shift towards exploitation during later iterations.

The ET analysis shows that CS and WOA were the fastest algorithms in raw runtime (~2.1 sec and ~2.6 sec, respectively), while Pure PSO-WOA was the slowest (~4.4 sec). The proposed PSO-WOA CML required ~3.8 sec per run, which is moderate considering the significant quality improvements achieved. When considering the EET, which better reflects practical convergence speed, PSO-WOA CML reaches stable solutions in ~1sec, faster than GA, PSO, and ABC, highlighting its suitability for real-time edge scenarios.

3.3. Statistical validation

Since metaheuristic results are typically non-normally distributed, we adopted non-parametric tests following the methodology outlined in [27]. The Friedman test was applied to rank the algorithms across runs, and the Wilcoxon signed-rank test was used for pairwise comparisons with the proposed PSO-WOA CML. Table 4 summarizes the Friedman ranks and Wilcoxon outcomes. The proposed PSO-WOA CML algorithm achieved the lowest average rank (best performance), followed by WOA and PSO, while GA and ABC consistently ranked worst. The Friedman test rejected the null hypothesis of equal medians ($p < 0.01$), confirming statistically significant differences among algorithms. For Wilcoxon signed-rank test, pairwise comparisons between PSO-WOA CML and each competitor confirmed that the improvements are statistically significant in most cases. For example, PSO-WOA CML vs. GA ($p < 0.001$), vs. ABC ($p < 0.001$), and vs. CS ($p < 0.01$) showed strong significance. Differences with PSO and WOA were less pronounced but still statistically meaningful ($p < 0.05$). These findings statistically substantiate the effectiveness of the proposed hybridization and its diversity-enhancing operators in avoiding premature convergence.

Table 4. Friedman ranks and Wilcoxon test results

Algorithm	Mean final C-Cost	Friedman rank	Wilcoxon vs. CML (p-value)
PSO-WOA (CML)	7.296	1.22	–
WOA	7.474	2.10	0.043 (< 0.05)
PSO	7.375	2.45	0.038 (< 0.05)
GWO	7.415	3.05	0.018 (< 0.01)
CS	7.517	3.48	0.007 (< 0.01)
GA	8.600	5.10	<0.001
ABC	8.615	5.60	<0.001

3.4. Scalability analysis

To assess scalability, the algorithms were tested under an increasing number of users (20–100) with three MEC servers. Figure 2 illustrates the evolution of mean final C-Cost and EET averaged over 10 runs. As the number of users increases from 20 to 100, all algorithms show a natural rise in the mean C-Cost due to the heavier computational demand. However, the proposed PSO-WOA CML algorithm consistently delivers the lowest cost across all scales, starting at 7.296 (20 users) and rising smoothly to 54.508 (100 users). In contrast, other methods like GA and ABC exhibit much steeper growth, exceeding 129 and 131, respectively, at 100 users. This highlights the superior scalability of the hybrid design, which maintains cost efficiency under increasing system load. Notably, PSO-WOA CML also outperforms the pure hybrid PSO-WOA, demonstrating the significant benefits of incorporating crossover, mutation, and Lévy flight operators.

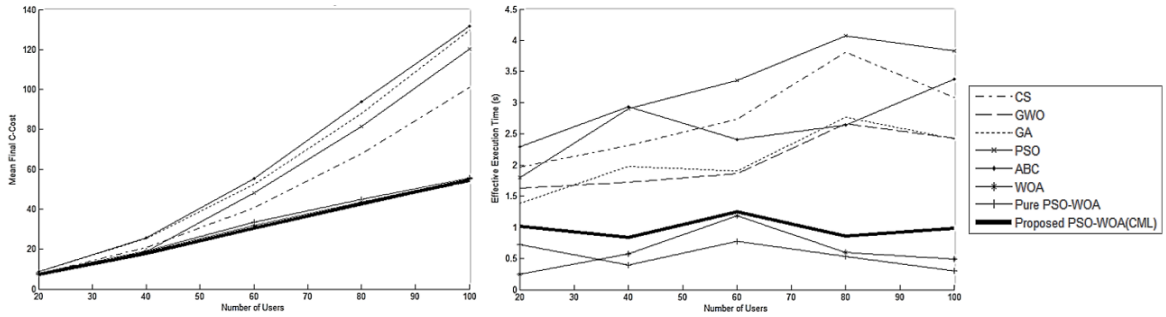

Figure 2. Scalability behavior of proposed hybrid algorithm vs. standard metaheuristics

In terms of EET, the proposed PSO-WOA CML algorithm shows nearly constant runtime (≈1 sec) as user numbers grow compared to other algorithms such as ABC (4.074 sec at 80 users) or CS (3.811 sec at 80 users). This suggests that while the problem size increases, the enhanced search mechanism of the CML operators allows the algorithm to find high-quality solutions even faster, which implies good scalability in time complexity. Interestingly, WOA and pure PSO-WOA report lower EET values at larger scales, but their performance is unstable and paired with higher costs, reflecting premature convergence. Overall, PSO-WOA CML maintains a near-linear scalability trend, achieving the best balance between computational efficiency and optimization quality, which is essential for large-scale MEC task offloading environments.

3.5. Discussion

The PSO-WOA CML clearly demonstrates superior robustness, rapid convergence, and high-quality solutions across diverse scenarios. Its hybrid structure integrates complementary and diverse search strategies and mechanisms. PSO's social learning mechanism drives the population toward promising regions, while WOA's dynamic spiral update provides powerful local exploitation. Crucially, the inclusion of Lévy flights allows the algorithm to make occasional large jumps, a proven strategy for escaping local optima where simpler algorithms might stagnate. Likewise, the genetic operators of crossover and mutation maintain population diversity, preventing premature convergence and ensuring a robust exploration of the solution space. Finally, the dynamic parameter tuning gradually moves the search from exploration to exploitation, accelerating convergence.

The superiority of the hybrid approach observed in our results aligns with findings reported in recent literature. For instance, [17] demonstrated that integrating WOA with DE significantly improves task offloading efficiency compared to standard algorithms. Similarly, our results confirm that hybrid mechanisms, specifically the inclusion of CML operators, prevent the premature convergence often seen in standalone WOA implementations. Furthermore, while [20] utilized GTO to minimize energy, delay, and cost, our proposed PSO-WOA CML achieves comparable stability in convergence but offers improved scalability for larger user sets (up to 100 users). This suggests that hybrid evolutionary strategies are increasingly essential for handling the high-dimensional search spaces typical of dense SDN-MEC environments, a conclusion also supported by the multi-user multi-server analysis in [28], [29].

4. CONCLUSION

This paper addressed the joint task offloading and resource allocation problem in SDN-enabled MEC environments for latency-sensitive IoT applications. A hybrid PSO-WOA algorithm enhanced with

crossover, mutation, and Lévy flight CML operators was proposed to minimize a weighted sum of energy consumption, execution delay, and monetary cost.

Comparative evaluation against six benchmark algorithms (ABC, CS, GA, GWO, PSO, and WOA) demonstrated that the proposed algorithm delivers statistically significant improvements in solution quality, convergence speed, and scalability. Notably, the algorithm maintained its performance advantage as the number of users scaled from 20 to 100, exhibiting a stable and EET that is critical for real-time systems. The success of the hybridization is attributed to its effective integration of PSO's explorative social learning and WOA's exploitation capabilities, further enhanced by CML evolutionary operators that promote diversity and escape from local optima.

Future work will focus on validating this approach on real-world testbeds or through large-scale simulations using platforms like EdgeCloudSim. We also plan to extend the model to dynamic scenarios with random task arrivals and explore adaptive mechanisms for real-time parameter tuning and workload prediction to further enhance its applicability in production MEC environments.

FUNDING INFORMATION

Authors state no funding involved.

AUTHOR CONTRIBUTIONS STATEMENT

The first author prepared the manuscript; the second and third authors provided supervision and critical guidance, while the rest helped in writing reviews and editing.

Name of Author	C	M	So	Va	Fo	I	R	D	O	E	Vi	Su	P	Fu
Faatima Z. Cherhabil	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓			
Sonia-Sabrina Bendib	✓	✓			✓	✓	✓			✓	✓	✓	✓	
Maamar Sedrati	✓	✓			✓	✓				✓	✓	✓	✓	✓
Chahrazed Adouane	✓	✓								✓	✓			
Sifeddine Benflis	✓	✓								✓	✓			

C : **C**onceptualization
M : **M**ethodology
So : **So**ftware
Va : **Va**lidation
Fo : **Fo**rmal analysis
I : **I**nvestigation
R : **R**esources
D : **D**ata Curation
O : Writing - **O**riginal Draft
E : Writing - Review & **E**diting
Vi : **Vi**sualization
Su : **Su**pervision
P : **P**roject administration
Fu : **Fu**nding acquisition

CONFLICT OF INTEREST STATEMENT

Authors state no conflict of interest.

DATA AVAILABILITY

Data availability is not applicable to this paper.

REFERENCES

[1] F. Saeik *et al.*, "Task offloading in edge and cloud computing: a survey on mathematical, artificial intelligence and control theory solutions," *Computer Networks*, vol. 195, p. 108177, Aug. 2021, doi: 10.1016/j.comnet.2021.108177.
[2] N. Abbas, Y. Zhang, A. Taherkordi, and T. Skeie, "Mobile edge computing: a survey," *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450–465, Feb. 2018, doi: 10.1109/JIOT.2017.2750180.
[3] M. Chen and Y. Hao, "Task offloading for mobile edge computing in software defined ultra-dense network," *IEEE Journal on Selected Areas in Communications*, vol. 36, no. 3, pp. 587–597, Mar. 2018, doi: 10.1109/JSAC.2018.2815360.
[4] A. C. Baktir, A. Ozgovde, and C. Ersoy, "How can edge computing benefit from software-defined networking: a survey, use cases, and future directions," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2359–2391, 2017, doi: 10.1109/COMST.2017.2717482.
[5] F. Z. Cherhabil, M. Sedrati, and S.-S. Bendib, "The integration of software defined network in mobile edge computing: state of the art," in *20th International Conference on remote Engineering and Virtual Instrumentation*, Thessaloniki, 2023.
[6] K. Li, X. Wang, Q. Ni, and M. Huang, "Entropy-based reinforcement learning for computation offloading service in software-defined multi-access edge computing," *Future Generation Computer Systems*, vol. 136, pp. 241–251, Nov. 2022, doi: 10.1016/j.future.2022.06.002.
[7] F. Z. Cherhabil, M. Sedrati, and S.-S. Bendib, "AI-based approaches for task offloading, resource allocation and service placement of IoT applications: state of the art," *Information and Communication Engineering*, vol. 17, no. 2, pp. 137–143, 2023.

[8] B. Tang, S. Zheng, and Q. Yang, "Joint optimization of energy and delay for computation offloading in vehicular edge computing," *Peer-to-Peer Networking and Applications*, vol. 16, no. 6, pp. 2681–2695, Nov. 2023, doi: 10.1007/s12083-023-01568-9.
[9] K. Wang, X. Wang, and X. Liu, "Sustainable internet of vehicles system: a task offloading strategy based on improved genetic algorithm," *Sustainability*, vol. 15, no. 9, p. 7506, May 2023, doi: 10.3390/su15097506.
[10] A. Zhu and Y. Wen, "Computing offloading strategy using improved genetic algorithm in mobile edge computing system," *Journal of Grid Computing*, vol. 19, no. 3, p. 38, Sep. 2021, doi: 10.1007/s10723-021-09578-8.
[11] L. Kuang, T. Gong, S. OuYang, H. Gao, and S. Deng, "Offloading decision methods for multiple users with structured tasks in edge computing for smart cities," *Future Generation Computer Systems*, vol. 105, pp. 717–729, Apr. 2020, doi: 10.1016/j.future.2019.12.039.
[12] M. Alqarni, A. Cherif, and E. Alkayyal, "ODM-BCSA: an offloading decision-making framework based on binary cuckoo search algorithm for mobile edge computing," *Computer Networks*, vol. 226, p. 109647, May 2023, doi: 10.1016/j.comnet.2023.109647.
[13] A. Abbas, A. Raza, F. Aadil, and M. Maqsood, "Meta-heuristic-based offloading task optimization in mobile edge computing," *International Journal of Distributed Sensor Networks*, vol. 17, no. 6, p. 155014772110230, Jun. 2021, doi: 10.1177/15501477211023021.
[14] M. Babar, M. S. Khan, A. Din, F. Ali, U. Habib, and K. S. Kwak, "Intelligent computation offloading for IoT applications in scalable edge computing using artificial bee colony optimization," *Complexity*, vol. 2021, no. 1, Jan. 2021, doi: 10.1155/2021/5563531.
[15] B. Cheng, "Multi-population cooperative elite algorithm for efficient computation offloading in mobile edge computing," *Journal of Grid Computing*, vol. 21, no. 4, p. 54, Dec. 2023, doi: 10.1007/s10723-023-09688-5.
[16] L. N. T. Huynh, Q.-V. Pham, X.-Q. Pham, T. D. T. Nguyen, M. D. Hossain, and E.-N. Huh, "Efficient computation offloading in multi-tier multi-access edge computing systems: a particle swarm optimization approach," *Applied Sciences*, vol. 10, no. 1, p. 203, Dec. 2019, doi: 10.3390/app10010203.
[17] J. Li, Q. Wang, S. Hu, and L. Li, "Hybrid immune whale differential evolution optimization (HIWDEO) based computation offloading in MEC for IoT," *Journal of Grid Computing*, vol. 21, no. 4, p. 70, Dec. 2023, doi: 10.1007/s10723-023-09705-7.
[18] H.-G. T. Pham, Q.-V. PHAM, A. T. Pham, and C. T. Nguyen, "Joint task offloading and resource management in NOMA-based MEC systems: a swarm intelligence approach," *IEEE Access*, vol. 8, pp. 190463–190474, 2020, doi: 10.1109/ACCESS.2020.3031614.
[19] W. Zhang and K. Tuo, "Research on offloading strategy for mobile edge computing based on improved grey wolf optimization algorithm," *Electronics*, vol. 12, no. 11, p. 2533, Jun. 2023, doi: 10.3390/electronics12112533.
[20] K. M. Hosny, A. I. Awad, M. M. Khashaba, and E. R. Mohamed, "New improved multi-objective Gorilla Troops algorithm for dependent tasks offloading problem in multi-access edge computing," *Journal of Grid Computing*, vol. 21, no. 2, p. 21, Jun. 2023, doi: 10.1007/s10723-023-09656-z.
[21] H. Li, P. Zheng, T. Wang, J. Wang, and T. Liu, "A multi-objective task offloading based on BBO algorithm under deadline constrain in mobile edge computing," *Cluster Computing*, vol. 26, no. 6, pp. 4051–4067, Dec. 2023, doi: 10.1007/s10586-022-03809-7.
[22] Imran, Z. Ghaffar, A. Alshahrani, M. Fayaz, A. M. Alghamdi, and J. Gwak, "A topical review on machine learning, software defined networking, internet of things applications: research limitations and challenges," *Electronics*, vol. 10, no. 8, p. 880, Apr. 2021, doi: 10.3390/electronics10080880.
[23] N. Kiran, C. Pan, S. Wang, and C. Yin, "Joint resource allocation and computation offloading in mobile edge computing for SDN based wireless networks," *Journal of Communications and Networks*, vol. 22, no. 1, pp. 1–11, Feb. 2020, doi: 10.1109/JCN.2019.000046.
[24] I. Giagkiozis and P. J. Fleming, "Methods for multi-objective optimization: an analysis," *Information Sciences*, vol. 293, pp. 338–350, Feb. 2015, doi: 10.1016/j.ins.2014.08.071.
[25] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, May 2016, doi: 10.1016/j.advengsoft.2016.01.008.
[26] P. Kumar and K. Silambarasan, "Enhancing the performance of healthcare service in IoT and cloud using optimized techniques," *IETE Journal of Research*, vol. 68, no. 2, pp. 1475–1484, Mar. 2022, doi: 10.1080/03772063.2019.1654934.
[27] J. Derrac, S. García, D. Molina, and F. Herrera, "A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 3–18, Mar. 2011, doi: 10.1016/j.swevo.2011.02.002.
[28] Q. You and B. Tang, "Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things," *Journal of Cloud Computing*, vol. 10, no. 1, p. 41, Dec. 2021, doi: 10.1186/s13677-021-00256-4.
[29] M. Liang and B. Wang, "A hybrid heuristic task offloading algorithm in mobile edge computing for joint optimization of delay and energy consumption," *Journal of Network and Systems Management*, vol. 33, no. 3, p. 65, Jul. 2025, doi: 10.1007/s10922-025-09945-w.

BIOGRAPHIES OF AUTHORS

Fatima Z. Cherhabil received her Computer Science Master's degree from the University of Mustafa Benboulaid (Batna2), Fesdis, Batna, Algeria. She is currently pursuing a Ph.D. degree at the same university. She is also a Middle School Computer Science teacher. Her research interests include artificial intelligence, multi-objective optimization, and computer communications, particularly in IoT networks. She can be contacted at email: f.cherhabil@univ-batna2.dz.

Sonia-Sabrina Bendib received her Ph.D. in Computer Science from the University of Batna2, Algeria. She is currently an associate professor at the same university. She teaches courses on cloud computing, software engineering, knowledge engineering, and decision support systems. Her main research interests include optimization techniques for cloud-fog-edge and IoT environments, as well as real-time systems. She can be contacted at email: ss.bendib@univ-batna2.dz.

Maamar Sedrati received his Engineering degree in 1985 from UMC Constantine, Algeria, and his Ph.D. degree in 2011 from UHL Batna University, Algeria. He is currently serving as an assistant professor and a member of the LaSTIC Laboratory at the Computer Science Department, University of Batna2, Algeria. His research interests include computer networks, internet technologies, mobile computing, artificial intelligence security, and quality of service (QoS) for multimedia applications in wireless and mobile networks. He also works on various aspects of the internet of things (IoT). He serves on technical program committees for numerous national and International Algerian conferences, including ICACIS, CN2TI, ICCSA, MISC, IAM, and IC3IT. He can be contacted at email: m.sedrati@univ-batna2.dz.

Chahrazad Adouane received her Engineering degree in Computer Science from the University of Batna 1. She is currently pursuing a Ph.D. degree in Computer Science at University of Batna 2, where her research focuses on optimizing security in internet of things (IoT) systems using blockchain technologies. Her research interests include blockchain security, IoT architecture, distributed consensus, and smart contract development. She can be contacted at email: adouane.c@gmail.com.

Sifeddine Benflis received the Master's degree in Networks and Distributed Systems and is currently pursuing the PhD. degree in Computer Science, specializing in Information Systems, at the University of Batna 2, where he is works in the field of cloud computing. His current research interests include cloud computing, energy efficiency, resource utilization efficiency, and virtual machines. He can be contacted at email: sif.benflis@univ-batna2.dz.