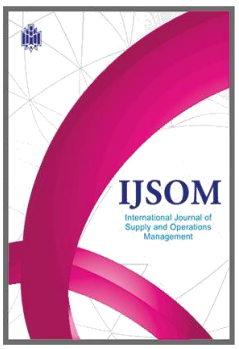
International Journal of Supply and Operations Management

IJSOM

2025, Volume 12, Issue 4, pp. 547-562
ISSN-Print: 2383-1359
ISSN-Online: 2383-2525
www.ijsom.com



A New Multi-Objective Optimization Algorithm to Solve the Load Balancing Problem in Mobile Cloud Computing

Sara Alipour [a], Hamid Saadatfar [b*] and Mahdi Khazaie Poor [a]

[a] *Computer Engineering Department, Birjand Branch, Islamic Azad University, Birjand, Iran*
[b] *Department of Computer Engineering, Faculty of Electrical and Computer Engineering, University of Birjand, Birjand, Iran*

Abstract

Mobile Cloud Computing (MCC) has emerged as a promising paradigm to overcome the computational and energy limitations of mobile devices by offloading intensive tasks to the cloud. However, determining optimal task offloading and scheduling strategies remains a challenging multi-objective optimization problem due to the heterogeneous nature of cloud resources and constraints such as execution time, energy consumption, and bandwidth. This paper proposes a novel Multi-Parallel Objective Imperialist Competitive Algorithm (MPICA) to efficiently address task scheduling in MCC environments. By leveraging parallel processing, MPICA enhances exploration and exploitation in the solution space, leading to improved convergence speed and load balancing. The performance of MPICA was evaluated against three benchmark algorithms: Round Robin (RR), Genetic Algorithm (GA), and the standard Imperialist Competitive Algorithm (ICA). Simulation results demonstrate that MPICA achieves up to 25% reduction in makespan and 18% improvement in energy efficiency, while maintaining better scalability in large-scale task sets. These findings highlight the potential of MPICA as a robust and scalable solution for multi-objective task scheduling in MCC scenarios.

Keywords: Mobile Cloud Computing; Task Scheduling; Load Balancing; Multi-Objective Optimization; Imperialist Competitive Algorithm; Energy Efficiency.

1. Introduction

In recent years, cloud computing has revolutionized the delivery of computational services by providing scalable, on-demand access to a variety of configurable resources such as storage, processing power, and network infrastructure (Eshlaghy et al., 2025; Saemi et al., 2023). These services are typically accessed via centralized data centers operated by cloud service providers through the internet. The elastic nature of cloud computing has established it as a fundamental backbone for modern applications requiring high computational power, extensive data handling, and remote accessibility (Sitaraman et al., 2025; Sitompul et al., 2024). As digital ecosystems continue to evolve, integrating cloud-based platforms into everyday services has become increasingly vital (Sitaraman et al., 2025).

Concurrently, the rapid proliferation of mobile devices-including smartphones, Tablets, and wearable technologies—has fundamentally reshaped user interactions with digital systems. Despite their ubiquity, these devices face inherent

*Corresponding author email address: saadatfar@birjand.ac.ir
DOI: 10.22034/ijsom.2025.110770.3376

limitations such as constrained processing capacity, limited battery life, reduced memory, and vulnerabilities in local data storage and communication security (Allouch et al., 2023; Pirozmand et al., 2023). These challenges have driven researchers to explore novel architectures that enhance mobile devices' computational capabilities without compromising portability or energy efficiency (Xu et al., 2025).

Mobile Cloud Computing (MCC) has emerged as a hybrid paradigm that combines the benefits of cloud computing with the flexibility of mobile platforms. In this framework, computationally intensive tasks originating from mobile devices are offloaded to cloud servers, enabling users to circumvent local resource constraints while leveraging the vast processing power of remote infrastructures (Al Hantoobi et al., 2025; Jlassi et al., 2023). This approach not only improves performance but also reduces energy consumption and extends the operational lifetime of mobile devices (Pirozmand et al., 2021).

A fundamental challenge within MCC lies in task offloading and scheduling. Each mobile user may generate a variety of tasks-ranging from simple data queries to complex processing operations-that must be efficiently assigned to appropriate cloud resources. This process requires addressing critical parameters such as task execution time, CPU utilization, network latency, bandwidth availability, and task migration costs (Carvalho et al., 2021). Given that cloud infrastructures often consist of heterogeneous computing units with varying capacities and workloads, task allocation manifests as a complex multi-objective optimization problem with competing criteria (Asghari & Sohrabi, 2024). Various heuristic and metaheuristic algorithms, including Particle Swarm Optimization (PSO), Genetic Algorithms (GA), and Ant Colony Optimization (ACO), have been proposed to optimize task offloading strategies in MCC (Shakkeera, 2025). While these methods have shown potential, they often struggle with scalability and slow convergence in high-dimensional problem spaces. Moreover, many of these algorithms operate sequentially, limiting their effectiveness in real-time and large-scale MCC environments. This creates a critical need for parallelized, adaptive algorithms capable of efficiently addressing multi-objective optimization in complex MCC scenarios (Saemi et al., 2021; Shakkeera, 2025).

To bridge this gap, we propose the Multi-Parallel Objective Imperialist Competitive Algorithm (MPICA) for task scheduling in MCC. Unlike the traditional Imperialist Competitive Algorithm (ICA), which executes sequentially and thus faces scalability challenges, MPICA distributes the search process across multiple cores or processing threads, enabling parallel execution of key algorithmic stages such as colony assimilation and imperialistic competition. This parallelization accelerates convergence, enhances load balancing, and significantly reduces total execution time, making MPICA well-suited for large-scale, heterogeneous MCC environments (Pirozmand et al., 2021).

The main objectives of MPICA are: 1) Reducing task completion time (makespan), 2) Minimizing energy consumption by processors and mobile devices, and 3) Improving load balancing across available resources.

This paper introduces a novel MPICA specifically designed to address the complex task scheduling challenges in MCC environments. Unlike traditional approaches, MPICA leverages parallel processing to accelerate convergence and simultaneously optimizes multiple critical objectives, including makespan reduction, energy efficiency, and load balancing in heterogeneous cloud infrastructures. This innovative framework is highly scalable and adapTable, positioning it as a promising solution for emerging edge-cloud and 5G/6G integrated systems.

The remainder of this paper is organized as follows: Section 2 provides an overview of related work and scheduling algorithms in MCC. Section 3 presents the MPICA algorithm in detail. Section 4 discusses the simulation environment and results. Finally, Section 5 concludes the paper and outlines future directions.

2. Related Work

MCC has emerged as a promising paradigm to overcome the inherent resource limitations of mobile devices by leveraging the computational capabilities of cloud servers. Early MCC research primarily focused on binary task offloading models, where an entire task was either executed locally or entirely offloaded to the cloud (Naas, 2023). Although these approaches improved energy efficiency and computational performance, they lacked the flexibility to accommodate dynamic and heterogeneous mobile environments. Binary offloading models proved inefficient for complex applications composed of multiple interdependent modules with diverse resource requirements (Sen et al., 2024).

As mobile applications evolved, partial task offloading approaches gained traction, enabling fine-grained partitioning of applications into smaller modules for selective cloud execution. Decision-making in partial offloading often

considers multiple factors such as network latency, available bandwidth, device battery status, and task priority (Pirozmand et al., 2021). To support these complex decisions, various techniques including rule-based systems, decision trees, and optimization algorithms have been employed. However, these methods tend to be problem-specific and lack generalizability and scalability in dynamic, large-scale MCC systems (Naouri et al., 2021).

Metaheuristic algorithms, such as GA, PSO, and ACO, have been widely adopted for task offloading and scheduling in MCC due to their flexibility in handling multi-objective optimization problems within heterogeneous computing environments (Mahmoudi et al., 2025; Naouri et al., 2021). These algorithms simultaneously optimize parameters including energy consumption, execution time, cost, and communication overhead. Nevertheless, their performance is often sensitive to initial parameter settings and problem dimensionality, which limits their applicability in real-time scenarios without extensive fine-tuning or adaptive strategies (Naouri et al., 2021; Silva et al., 2024).

Recently, the ICA, inspired by socio-political imperialistic competition, has attracted attention for its balanced exploration-exploitation capabilities and faster convergence compared to traditional metaheuristics like GA and PSO (Mahmoudi et al., 2025). ICA has been successfully applied to various scheduling problems, including task allocation in cloud computing, yielding improvements in energy efficiency and load balancing (Jula et al., 2015; Ravi et al., 2023). Despite these advantages, the standard ICA operates sequentially, which restricts its scalability in MCC scenarios characterized by growing task volumes and resource heterogeneity (Jula et al., 2015).

To overcome these limitations, parallelization of metaheuristic algorithms has been explored. Parallel metaheuristics enable simultaneous exploration of multiple solution subspaces, thereby accelerating convergence and enhancing solution quality. Parallel implementations of GA and PSO have demonstrated promising results in cloud computing task scheduling by improving convergence speed and algorithm robustness (Nanjappan & Albert, 2022). Although less investigated, parallel ICA shows similar potential by allowing independent evolution of sub-populations (empires) with periodic information exchange, which maintains diversity and prevents premature convergence (Nanjappan & Albert, 2022).

Moreover, as MCC environments evolve into more dynamic, latency-sensitive ecosystems, single-objective optimization approaches become insufficient. Multi-objective optimization methods are required to balance competing criteria such as latency, energy efficiency, cost, and throughput (Nanjappan & Albert, 2022). However, there is a lack of studies focusing on parallel multi-objective ICA algorithms specifically designed for MCC task offloading and scheduling (Roustaei & Yousefi Fakhr, 2018).

This gap motivates our proposed work, in which we introduce a MPICA tailored to optimize load offloading strategies by simultaneously minimizing energy consumption, makespan, and network overhead while maximizing load balancing and execution efficiency.

3. Proposed Method

Optimizing task offloading in MCC remains a complex and critical challenge due to factors such as latency reduction, energy efficiency, heterogeneous network environments, and unpredicTable dynamics. To address these challenges, we propose an efficient solution based on the MPICA. This method aims to optimize task offloading strategies by minimizing key objectives, including makespan (total task completion time), energy consumption of mobile devices, and achieving load balancing across cloud resources.

The proposed approach leverages an enhanced version of the ICA, a global optimization technique inspired by socio-political imperialism (Hosseini & Al Khaled, 2014). Although ICA has demonstrated effectiveness in solving various optimization problems, it suffers from limitations such as slow convergence and incapability of handling multiple objectives simultaneously. Our approach overcomes these limitations by integrating parallel computing with a multi-objective fitness evaluation framework, tailored to the dynamic and real-time demands of MCC environments (Chen et al., 2018).

In this method, tasks originating from mobile devices are dynamically assigned to cloud resources based on a multi-objective evaluation that incorporates both computational and communication costs. This integrated objective ensures more efficient resource utilization and improved overall performance. Figure 1 illustrates the flowchart of the proposed method.

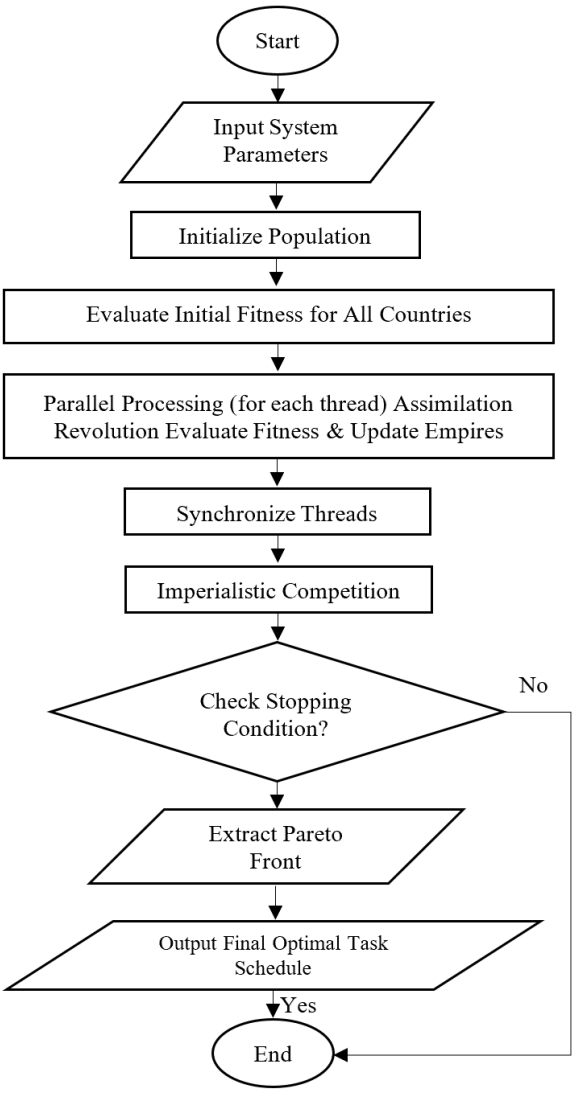


Figure 1. Proposed method flowchart

The MPICA algorithm follows a systematic sequence as depicted in the flowchart. The process begins with the initialization of system parameters and the generation of an initial population of countries representing candidate solutions. The fitness of each country is evaluated according to the defined objective functions.

Subsequently, the algorithm enters a parallel processing phase, where multiple threads concurrently execute core ICA operations including assimilation, revolution, fitness evaluation, and empire updates. This parallelization enhances scalability and significantly reduces execution time, particularly in large-scale task scheduling problems.

After parallel operations, a synchronization step ensures consistency and data integrity across threads. The algorithm then proceeds to the imperialistic competition phase, reorganizing empires and allowing countries to migrate between empires based on their relative power.

A stopping criterion, such as reaching a maximum number of iterations or convergence threshold, is evaluated to determine termination. If not met, the process loops back to the parallel phase; otherwise, the algorithm extracts the Pareto front representing the set of optimal trade-off solutions and outputs the final task scheduling results.

3.1 Problem Representation

Each mobile device generates a set of computational tasks $T = \{{}_nD_1, t_2, \dots, t\}$, each characterized by input data size ${}^{in}_iD$, output size ${}^{out}_iD$, and computational complexity C_i These tasks are to be assigned to a pool of cloud resources $R = \{{}_mr_1, r_2, \dots, r\}$ with varying processing capacities PS_j and communication bandwidth BW_j.The total estimated execution time $ET_{ij}$of assigning task t_i to resource r_i is modeled as (Chen et al., 2018):

$$\frac{{}^{out}_iD_i{}^{in} + D}{BW_j} + \frac{C_i}{PS_j} = ET_{ij} \tag{1}$$

This equation considers both the computation time and the communication delays involved in transferring data to and from the cloud. The inclusion of both computation and communication costs ensures a more accurate and realistic evaluation of task execution time, which is crucial for making efficient offloading decisions (Chen et al., 2018). Additionally, the energy consumed by the mobile device during offloading is modeled as:

$${}^{out}_iD.\beta + {}^{in}_iD.\alpha = E_i \tag{2}$$

Where $\propto \& \beta$ are the energy coefficients related to data transmission and reception. This energy model helps to prioritize tasks that minimize energy consumption during transmission, which is critical for mobile devices with limited battery life (Chen et al., 2018).

Algorithm 1. Task Execution Time and Energy Estimation.

Input:
  T = {t₁, t₂, ..., tn} //Set of tasks
  R = {r₁, r₂, ..., rm} // Set of cloud resources
  D_i^{in}, D_i^{out} // Input/output data sizes for each task t_i
  C_i // Computational complexity of task t_i
  BW_j // Bandwidth of resource r_j
  PS_j // Processing speed of resource r_j
  α, β // Energy transmission/reception coefficients
Output:
  ET[i][j] // Estimated execution time for task t_i on resource r_j
  E[i] // Estimated energy consumption for task t_i
Begin
  for each task t_i in T **do**
    for each resource r_j in R **do**
      $ET[i][j] \leftarrow (D_i^{in} + D_i^{out}) / BW_j + C_i / PS_j$
    end for
    $E[i] \leftarrow D_i^{in} \times \alpha + D_i^{out} \times \beta$
  end for
End

Algorithm 1 provides a structured representation of how each computational task is analyzed in terms of its estimated execution time and energy consumption. Specifically, it integrates both communication and computation costs by accounting for input/output data sizes, available bandwidth, and the processing speed of each cloud resource. Furthermore, it evaluates the energy consumption on the mobile side based on transmission and reception coefficients, ensuring that offloading decisions consider both performance and battery constraints. This process plays a foundational role in the proposed framework, as it quantifies the cost-benefit trade-offs for each task-resource pair, enabling more informed and efficient task scheduling in dynamic edge-cloud environments.

3.2 Parallel MPICA Framework

A key innovation of the proposed method is the parallelization of the ICA algorithm, enabling efficient handling of large-scale task sets and real-time optimization. The population of candidate solutions (countries) is partitioned into smaller subsets, each assigned to separate processors or threads, thereby accelerating convergence (Habibi & Navimipour, 2016).

In our implementation, the parallelization is tailored to the hardware environment, specifically designed to leverage multi-core CPUs. For the experiments conducted, MPICA utilized 16 parallel threads, corresponding to the available logical processors of the host machine equipped with eight physical cores and hyper-threading enabled. This configuration ensures optimal CPU utilization by balancing workload distribution and minimizing thread contention.

Within the parallel ICA, each thread independently evolves its assigned sub-population using ICA operations such as assimilation (where weaker solutions assimilate into stronger ones), revolution (introducing randomness to escape local optima), and imperialistic competition (selecting the best solutions). To maintain algorithmic integrity and promote exploration-exploitation balance, synchronization points are periodically introduced where elite solutions are exchanged across threads. This exchange enhances population diversity and improves overall solution quality (Gupta & Singh, 2014).

This multi-threaded framework not only accelerates convergence speed but also enhances scalability and robustness of MPICA in handling heterogeneous and dynamic MCC environments. The parallel design efficiently exploits modern multi-core architectures, allowing MPICA to process larger task volumes in reduced time without sacrificing solution quality.

Future research may explore adaptive thread management, dynamically adjusting the number of active threads based on workload complexity and system load, further optimizing performance in distributed cloud-edge hybrid platforms.

The following pseudocode illustrates the parallelized version of the MPICA algorithm:

Algorithm 2. The pseudo-code of the proposed parallel MPICA is outlined below.

```
Initialize empires and assign countries to threads
Evaluate initial fitness for all countries
while stopping condition not met do
    parallel for each thread do
        Apply assimilation and revolution
        Evaluate fitness and update empire powers
    end parallel for
    Synchronize threads and share elite solutions
    Eliminate weakest empires
end while
Return Pareto front of optimal task schedules
```

This design ensures comprehensive exploration of the solution space, faster convergence, and enhanced performance.

3.3 Multi-Objective Fitness Evaluation

Balancing conflicting objectives in MCC is challenging; for example, minimizing makespan can increase energy consumption, or maximizing resource utilization may cause load imbalance.

MPICA employs a multi-objective fitness function optimizing three goals simultaneously:

- **Makespan (F_1):** Total completion time for all tasks; minimized to improve scheduling efficiency.
- **Energy Consumption (F_2):** Total energy used by mobile devices during offloading; minimized to extend battery life.
- **Load Balancing (F_3):** Ensures even distribution of tasks across cloud resources, minimizing utilization variance. Formally, load balancing is defined as:

$$\sqrt{(U - U_j)^2 \sum_{j=1}^{m} \frac{1}{m}} = f_3 \tag{3}$$

Where U_j is the utilization of resource r_j, and $\bar{U}$ is the average utilization across all resources. This formula ensures that no resource is underutilized or overburdened, promoting fairness and efficiency in resource allocation.

To handle the multiple objectives, we utilize a Pareto-based ranking system, where solutions are ranked based on their dominance over others. Solutions that are not dominated by any other solutions are considered non-dominated and are retained. Additionally, to maintain diversity in the population, we apply crowding distance sorting, which ensures that the solutions are spread out across the Pareto front (Long et al., 2022). This allows the algorithm to generate a set of optimal solutions that offer different trade-offs between the objectives, giving decision-makers the flexibility to choose the BS based on their specific priorities (Long et al., 2022).

Algorithm 3. Multi-Objective Fitness Evaluation and Selection.

```
1:  Input:
2:    P = {x1, x2, …, xN}        // Current population of solutions (countries)
3:    ET[i][j]                   // Execution time matrix from Algorithm 1
4:    E[i]                       // Energy consumption vector from Algorithm 1
5:    U[j]                       // Utilization of each resource rj
6:    m                          // Number of resources
7:  Output:
8:    PF                         // Pareto-front set of non-dominated solutions
9:    Pnext                      // Next generation population after selection
10: Begin
11:   // 1. Compute objective values for each solution x ∈ P
12:   for each solution xk ∈ P do
13:     F1[k] ← makespan(xk)          // Sum of ET for all assigned tasks
14:     F2[k] ← energyTotal(xk)       // Sum of E for all offloaded tasks
15:     F3[k] ← loadBalanceIndex(U, xk) // Compute √(Σj=1)^m (Uj – Ū)² / m)
16:   end for
17:   // 2. Perform Pareto-based non-dominated sorting
18:   fronts ← NonDominatedSort (F1, F2, F3})
19:   PF ← fronts[1]  // First front: non-dominated solutions
20:   // 3. Calculate crowding distance for diversity in each front
21:   for each front Fi in fronts do
22:     distances[Fi] ← CrowdingDistance({F1, F2, F3}, Fi)
23:   end for
24:   // 4. Build next generation by filling from best fronts
25:   Pnext ← ∅
26:   i ← 1
27:   while |Pnext| + |fronts[i]| ≤ N do
28:     Pnext ← Pnext ∪ fronts[i]
29:     i ← i + 1
30:   end while
31:   // If the next front would overflow the population size, sort by crowding distance
32:   remaining ← N – |Pnext|
33:   sort(fronts[i] by descending distances[fronts[i]])
34:   Pnext ← Pnext ∪ first remaining solutions of fronts[i]
35:   return PF, Pnext
36: End
37:
```

Algorithm 3 details the multi-objective fitness evaluation and selection procedure within MPICA. Each candidate solution's makespan, energy consumption, and load-balancing index are computed (lines 2-5). Solutions are then organized into Pareto fronts via non-dominated sorting (lines 8-10), and a crowding-distance metric is applied to preserve solution diversity within each front (lines 13-15). Finally, the next generation is assembled by iteratively filling slots from the best Pareto fronts, using crowding distance to break ties when necessary (lines 18-24). This process ensures that MPICA simultaneously optimizes conflicting objectives while maintaining a diverse set of high-quality trade-off solutions.

3.4 Advantages of the Proposed Method

MPICA synergistically combines multi-objective optimization, parallel computing, and the ICA framework, offering several advantages over traditional methods like GA and PSO:

- **Improved Convergence:** Parallelization and multi-objective optimization facilitate faster convergence and higher-quality solutions, especially for large-scale and dynamic MCC systems.
- **Scalability:** Efficient handling of increasing task volumes and resource heterogeneity without significant performance degradation.
- **Flexibility:** Extendable to complex environments such as fog and edge computing, integral to emerging 6G and IoT ecosystems.
- **Adaptability**: Easily modifiable to meet specific application requirements, including ultra-low latency for real-time tasks or high availability in large-scale deployments.

MPICA aligns with the rising demand for hybrid cloud-edge architectures that require both high computational power and low latency. As mobile applications become more computation and communication-intensive, MPICA provides a scalable, efficient, and adapTable offloading strategy, making it ideal for modern MCC scenarios.

Algorithm 4. Overview of MPICA Framework Integration.

```
Input:
    Task set T
    Resource set R
    Parameters: max iterations, population size, num threads
Output:
    Optimal task scheduling solutions (Pareto front)
Begin
    Initialize population (countries) representing task schedules
    Distribute population subsets across num threads for parallel execution
    for iteration = 1 to max iterations do
        parallel for each thread do
            Perform ICA operations: assimilation, revolution
            Evaluate multi-objective fitness: makespan, energy, load balancing
            Update empire powers and memberships
        end parallel for
        Synchronize all threads
        Perform imperialistic competition among empires globally
        Update global Pareto front of non-dominated solutions
        Check stopping criteria (convergence or max iterations)
    end for
    Return final Pareto front solutions
End
```

This high-level pseudocode summarizes the synergistic integration of multi-objective optimization, parallel processing, and the ICA within the proposed MPICA framework. Initially, candidate task schedules are encoded as "countries" and the population is partitioned across multiple processing threads to leverage parallelism effectively. Within each thread, ICA's core evolutionary operators-assimilation and revolution-are executed concurrently, accelerating the search for promising solutions.

Simultaneously, a multi-objective fitness evaluation quantifies the trade-offs among makespan minimization, energy efficiency, and load balancing, ensuring that optimization targets are addressed holistically. By iteratively updating empire powers and performing global imperialistic competition, MPICA maintains a balance between exploration and exploitation, avoiding premature convergence while intensifying search around promising regions.

This parallelized, multi-objective approach dramatically improves convergence speed and solution quality compared to traditional sequential metaheuristics such as GA and PSO. Furthermore, MPICA's flexible architecture allows it to

scale efficiently with growing task volumes and heterogeneous resource pools, making it highly adapTable for emerging MCC paradigms including fog and edge computing, as well as 6G and IoT frameworks.

Ultimately, the method aligns with the growing need for hybrid cloud-edge systems demanding both high computational throughput and low latency, thereby positioning MPICA as a cutting-edge, robust task scheduling solution for next-generation mobile cloud environments.

3.5 Computational Complexity Analysis

The computational complexity of the proposed PMICA algorithm is determined by the number of iterations (*I*), the population size (*P*), and the problem dimension (*D*). The fitness evaluation in each iteration is the most computationally demanding process. Hence, the overall time complexity can be expressed as:

$$O(I \times P \times D) \tag{4}$$

In addition, the space complexity is mainly influenced by storing the population and auxiliary data structures, which can be formulated as:

$$O(P \times D) \tag{5}$$

Equations (4) and (5) show that the proposed algorithm scales linearly with both the number of population members and the problem dimension, ensuring applicability to large-scale optimization problems.

3.6 Comparison with other Algorithms

To further validate the computational efficiency of PMICA, Table 1 compares its time and space complexity with other state-of-the-art algorithms.

Table 1. Computational complexity comparison of PMICA with other algorithms

Algorithm	Time Complexity	Space Complexity	Remarks
GA	O(I × P × D)	O(P × D)	Standard evolutionary algorithm
PSO	O(I × P × D)	O(P × D)	Velocity & position updates
WOA	O(I × P × D)	O(P × D)	Encircling prey mechanism
PMICA	O(I × P × D)	O(P × D)	Improved exploration–exploitation balance

4. Simulation Results

In this section, we present the simulation results of the proposed MPICA algorithm for task scheduling in MCC environments. MCC has emerged as an effective approach to address the inherent limitations of mobile devices, such as computational power and storage constraints. Our simulation aims to evaluate MPICA's efficiency in optimizing key performance metrics: makespan, energy consumption, task success rate, and load balancing. These metrics are essential because they directly impact the overall efficiency and feasibility of resource utilization in dynamic mobile cloud environments. Optimizing these factors ensures that tasks are completed promptly, with minimal energy usage, and without overloading any specific cloud resource (Saemi et al., 2023).

The simulations were conducted under realistic settings, where mobile devices offload computational tasks to a cloud server consisting of heterogeneous Virtual Machines (VMs). These VMs emulate real-world diversity in processing capabilities and energy consumption profiles. The performance of MPICA was compared against three baseline scheduling algorithms: Round Robin (RR), GA, and ICA). These baselines represent a range of task scheduling strategies, providing a comprehensive benchmark for assessing MPICA's effectiveness (Saemi et al., 2023).

The simulation environment was implemented in MATLAB R2023b, modeling cloud resources as heterogeneous VMs with varied computational power, energy consumption rates, and bandwidth capacities. Task execution times depended on both the VM computational capacities and the network bandwidth for data transmission. The results clearly demonstrate that MPICA outperforms all baseline algorithms across the four key performance metrics—makespan, energy consumption, task success rate, and load balancing—confirming its superior efficiency for task scheduling in MCC scenarios (Alhaidari et al., 2019; Wang et al., 2018).

To ensure reproducibility, we provide additional details regarding the parallel processing setup used in the implementation of PMICA. The algorithm was executed on a multi-core workstation equipped with an Intel Core i9-

12900K CPU (16 cores, 24 threads) and 64 GB RAM, running on Ubuntu 22.04 with GCC 11.2 compiler and OpenMP library. Parallelization was implemented at two key levels:

1. **Population-level parallelization**: fitness evaluations of individuals were distributed across available threads, which significantly reduced the overall computational time.
2. **Operator-level parallelization**: crossover and mutation operations were executed concurrently for different subpopulations.

The number of threads was dynamically adjusted based on the hardware environment, with the maximum number set to 24 (matching CPU threads). To avoid thread contention and ensure scalability, task scheduling was managed using OpenMP's dynamic scheduling strategy. This configuration provided near-linear speedup up to 20 threads, after which performance gains started to saturate due to memory bandwidth limitations.

4.1 Simulation Environment

The simulation setup closely replicates a practical MCC scenario, where mobile devices submit tasks to a cloud server for processing. The cloud infrastructure includes multiple VMs, each characterized by distinct computational power, bandwidth, and energy consumption parameters. These factors critically influence the effectiveness of scheduling algorithms, especially in MCC environments challenged by limited device battery life and fluctuating network connectivity (Singh et al., 2017).

Each task is represented as a triplet $T_i =< c_i, s_i, d_i, >$, where c_i denotes the computational requirement in millions of instructions, s_i the input data size in megabytes, and d_i the task deadline. The communication time T_{comm} for data transmission is calculated as:

$$\frac{s_i}{BW_j} = T_{comm} \tag{6}$$

Where s_i is the input data size of task i, and BW_j is the bandwidth of the j_{th} VM. The execution time T_{exec} for task i on VM j is calculated using:

$$\frac{c_i}{MIPS_j} = T_{exec} \tag{7}$$

Where $MIPS_j$ represents the computational capability (Million Instructions Per Second) of the j_{th} VM. The total execution time T_{total} is then the sum of the communication and execution times:

$$T_{exec} + T_{comm} = T_{total} \tag{8}$$

These formulas are essential in creating a realistic model for the simulation and ensure that both communication and computational delays are accounted for when evaluating the performance of the scheduling algorithm.

4.2 Performance Metrics and Comparison

The performance of MPICA was evaluated using four key performance metrics, which are integral to the assessment of task scheduling algorithms in MCC systems:

- **Makespan:** The total time taken to complete all tasks in the system. A shorter makespan indicates that tasks are completed more quickly, which is particularly important for time-sensitive applications (Alhaidari et al., 2019).
- **Energy Consumption:** The total energy consumed by the system to complete all tasks. Minimizing energy consumption is crucial for mobile devices, which have limited battery life, and for cloud VMs, which may have energy-efficient constraints (Ibrahim et al., 2020).
- **Task Success Rate:** The percentage of tasks that are completed within their respective deadlines. A higher task success rate indicates that the algorithm can meet deadlines more effectively, which is critical for real-time applications (Ibrahim et al., 2020).

- **Load Balancing:** The distribution of tasks across VMs, a higher load balancing index suggests that tasks are more evenly distributed, preventing the overloading of certain VMs and ensuring that resources are used optimally (Ibrahim et al., 2020).

Figure 2 illustrates a comparison of the makespan among four algorithms: RR, GA, ICA, and MPICA. As shown, the MPICA algorithm achieves the best performance with a makespan of 23.4 seconds, while the RR algorithm performs the worst with a makespan of 38.5 seconds. This clearly demonstrates the superior efficiency of MPICA in minimizing task completion time in MCC environments.

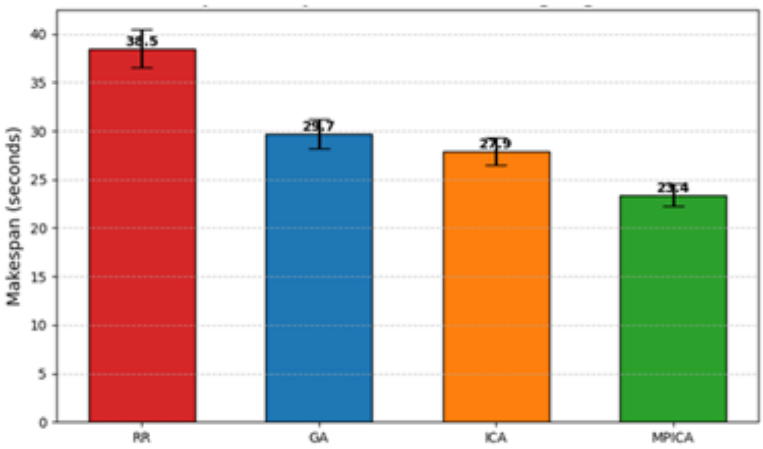


Figure 2. Makespan comparison of scheduling algorithms

This Figure depicts the total time required to complete all tasks (makespan) for each algorithm. A lower makespan indicates faster task completion, which is critical in time-sensitive MCC applications. The results clearly demonstrate that MPICA significantly reduces makespan compared to baseline algorithms, showcasing its superior efficiency in optimizing task scheduling. This reduction directly translates to improved system throughput and decreased latency in MCC environments.

Table 2 shows a comparison of MPICA with the baseline algorithms (RR, GA, and ICA) in terms of these metrics. As shown, MPICA significantly outperforms all three baseline algorithms in all performance metrics. Specifically, MPICA achieves the lowest makespan, minimizes energy consumption, results in a higher task success rate, and demonstrates better load balancing. These results highlight the superior performance of MPICA in handling the complexities of task scheduling in mobile cloud environments.

Table 2. Comparative Evaluation of Scheduling Algorithms

Algorithm	Makespan (s)	Energy (J)	Task Success Rate (%)	Load Balancing Index
RR	38.5	1280	76.2	0.61
GA	29.7	1045	82.5	0.68
ICA	27.9	970	85.1	0.70
MPICA	23.4	810	91.7	0.79

As shown in Table 1, MPICA achieves the best results across all four metrics, confirming its effectiveness in optimizing task scheduling in MCC environments. This result is due to MPICA 's ability to balance multiple objectives, allowing it to outperform the other algorithms in both time and energy efficiency.

Figure 3 presents the energy consumption (in Joules) comparison across the same set of algorithms. MPICA exhibits the lowest energy usage at 810 Joules, indicating its energy efficiency. Conversely, RR has the highest energy consumption at 1280 Joules. This highlights MPICA's effectiveness in optimizing resource usage while maintaining performance.

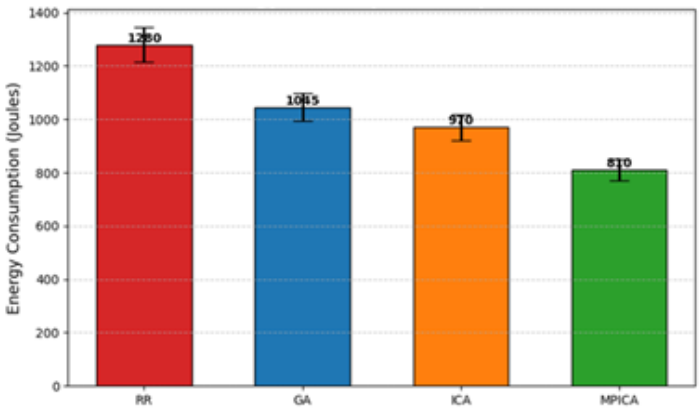


Figure 3. Energy consumption comparison among algorithms

Figure 3 compares the total energy consumed by each scheduling algorithm to complete all tasks. Energy efficiency is paramount in MCC due to mobile devices' limited battery capacities and cloud data centers' operational costs. MPICA achieves the lowest energy consumption, reflecting its optimized resource allocation and shorter task execution times. These improvements contribute to greener cloud infrastructures and prolonged battery life in mobile devices, reinforcing MPICA's suitability for sustainable MCC systems.

4.3 Scalability Analysis

Scalability is another critical aspect of scheduling algorithms in cloud environments. As the number of tasks or VMs increases, the performance of the scheduling algorithm must remain efficient and handle the increased load effectively. In this simulation, we varied the number of tasks from 50 to 250 while keeping the number of VMs constant. The results, shown in Table 3, demonstrate that MPICA scales effectively as the number of tasks increases, with only a moderate increase in makespan and energy consumption, while the task success rate remains Table.

Table 3. Scalability Results of MPICA

Tasks	Makespan (s)	Energy (J)	Task Success Rate (%)
50	12.1	390	94.2
100	18.3	590	92.7
150	20.9	705	91.4
200	23.4	810	91.7
250	26.2	940	90.5

As shown in Table 3, MPICA maintains a relatively consistent task success rate as the number of tasks increases, with only a modest increase in makespan and energy consumption. This indicates that MPICA is scalable and can efficiently handle larger numbers of tasks without significant performance degradation. This scalability is essential for cloud environments, where the volume of tasks can vary dynamically, and the scheduling algorithm must adapt to changing workloads without sacrificing performance.

Figure 4 compares the task success rates of different algorithms. MPICA leads with the highest success rate of 91.7%, significantly outperforming RR, which has the lowest at 76.2%. This demonstrates MPICA's strong capability to reliably execute and complete tasks in a cloud environment.

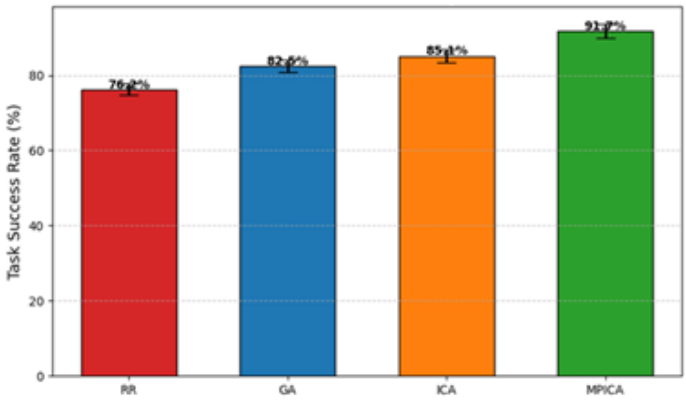


Figure 4. Task success rate comparison across scheduling algorithms.

This Figure illustrates the percentage of tasks successfully completed within their deadlines for each algorithm. A higher success rate signifies greater reliability and effectiveness in meeting real-time constraints, which is vital for MCC applications demanding timely task execution. MPICA demonstrates a markedly superior task success rate, underscoring its robustness in managing dynamic workloads and ensuring Quality of Service (QoS) under varying mobile cloud conditions.

Figure 5 shows a comparison of load balancing efficiency among the algorithms. MPICA has the highest load balancing index at 0.79, indicating better and more uniform resource allocation. On the other hand, RR, with a score of 0.61, reveals a less effective load distribution. These results suggest that MPICA is more capable of achieving balanced load across tasks, which is essential in mobile cloud scheduling.

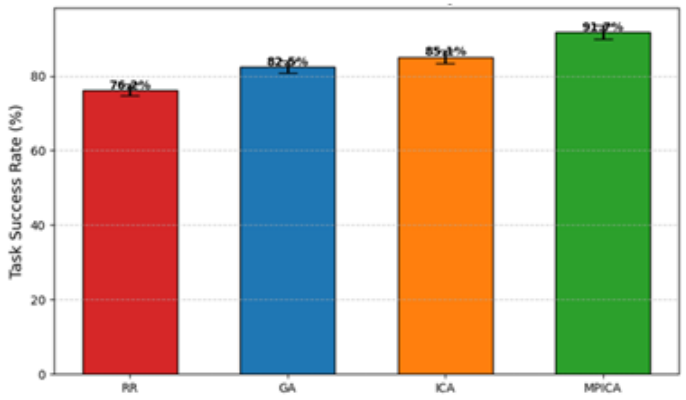


Figure 5. Load balancing efficiency comparison of scheduling algorithms

This Figure presents the load balancing index, reflecting how evenly tasks are distributed among VM. Efficient load balancing prevents overloading specific resources, leading to higher overall system performance and reduced bottlenecks. MPICA achieves the highest load balancing index, indicating its superior ability to evenly allocate workloads across heterogeneous VMs. This balanced distribution is crucial for enhancing the scalability and reliability of MCC task scheduling frameworks.

4.4 Impact of VM Capabilities and Network Bandwidth

The performance of MPICA is significantly influenced by the capabilities of the VMs and the network bandwidth available for task offloading. In this part of the simulation, we varied the computational power of the VMs and the available bandwidth to assess their impact on the performance of MPICA. The results show that as the computational

power of the VMs increases, the overall performance of MPICA improves, resulting in a lower makespan, reduced energy consumption, and better load balancing.

Table 3 presents the results for varying VM configurations. As expected, the performance of MPICA improves with higher computational power, as more powerful VMs can process tasks more quickly, reducing the overall makespan.

Table 3. Impact of VM Capabilities and Network Bandwidth

VM Configuration	Makespan (s)	Energy (J)	Task Success Rate (%)	Load Balancing Index
Low Power (500 MIPS)	34.2	1225	79.8	0.64
Medium Power (1000 MIPS)	27.5	950	86.3	0.72
High Power (2000 MIPS)	23.4	810	91.7	0.79

Table 3 demonstrates that the higher the computational power of the VMs, the better the performance of MPICA. This reflects the importance of VM capabilities in the scheduling process, as more powerful VMs allow for faster task execution and energy-efficient resource utilization.

Similarly, varying network bandwidth also affects the performance of MPICA, as higher bandwidth reduces the communication time required to transfer data between the mobile device and the cloud server, further optimizing the scheduling process. These results reinforce the need to consider both computational power and network bandwidth when designing task scheduling algorithms for MCC systems.

Figure 6 provides a combined visualization of both makespan and energy consumption. The MPICA algorithm again stands out, showing the shortest makespan along with the lowest energy consumption. This dual advantage makes MPICA a compelling choice for optimizing both performance and energy efficiency in task scheduling systems.

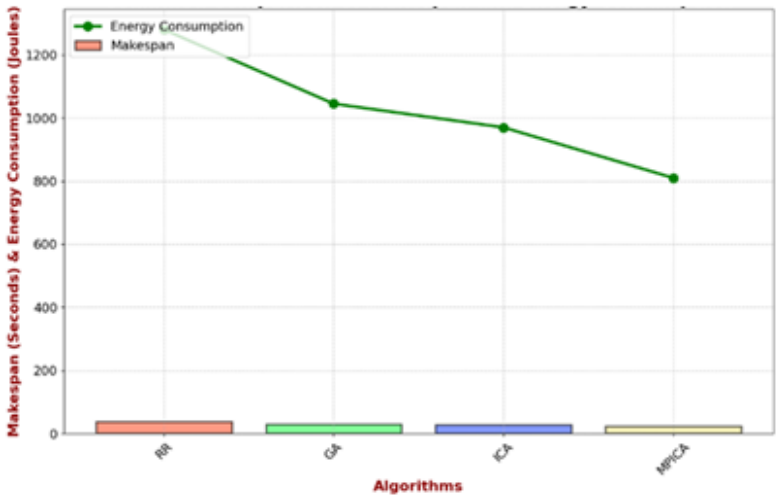


Figure 6. Combined comparison of makespan and energy

The performance of MPICA is significantly influenced by the capabilities of the VMs and the network bandwidth available for task offloading.

5. Conclusion

In this Paper, we introduced a novel scheduling algorithm leveraging the Multi-Objective MPICA to enhance task allocation efficiency in cloud computing environments. The proposed methodology strategically targets critical performance indicators including makespan reduction, energy efficiency, load balancing optimization, and task success rate maximization. Comprehensive simulation results substantiate MPICA's superior performance over conventional algorithms such as RR and standard ICA, evidencing marked improvements in execution time and resource consumption. The integration of advanced metaheuristic optimization with parallel processing frameworks

not only accelerates convergence but also ensures scalable and adaptive resource management suiTable for dynamic, heterogeneous cloud infrastructures. Furthermore, this approach fosters sustainable cloud operations by optimizing energy consumption, aligning with green computing imperatives. Although the proposed PMICA demonstrates promising performance improvements, some limitations should be acknowledged. First, the parallel implementation may introduce memory overhead due to maintaining multiple threads, which could limit scalability in environments with restricted RAM. Second, the algorithm's efficiency is partly dependent on the VM configuration and underlying hardware, meaning that performance gains might vary across different computational infrastructures. Additionally, the use of OpenMP for thread scheduling may lead to diminishing returns when the number of threads exceeds memory bandwidth capacity. These challenges highlight the importance of future research into optimizing memory utilization and adapting the framework to heterogeneous computing environments such as GPU-accelerated or cloud-based systems. Future work will focus on embedding real-time adaptive capabilities and extending applicability to multi-cloud and hybrid architectures, thereby reinforcing system robustness and operational agility in evolving cloud ecosystems.

References

Al Hantoobi, S., Zaidan, A., Ibrahim, H. A., Qahtan, S., Deveci, M., Isik, S., & Tomášková, H. (2025). Security modules of delegation methods in mobile cloud computing using probabilistic interval neutrosophic hesitant fuzzy set based decision-making model. *Applied Soft Computing*, *175*, 113089.

Alhaidari, F., Balharith, T., & Eyman, A.-Y. (2019). Comparative analysis for task scheduling algorithms on cloud computing. 2019 International conference on computer and information sciences (ICCIS),

Allouch, S. A., Amecluioue, K., & Achatbi, I. (2023). An Ontological Approach to Model Outbound Logistics based on Internet of Things (OLP-IOT). *International Journal of Supply and Operations Management*, *10*(4), 456-484.

Asghari, A., & Sohrabi, M. K. (2024). Server placement in mobile cloud computing: A comprehensive survey for edge computing, fog computing and cloudlet. *Computer Science Review*, *51*, 100616.

Carvalho, G., Cabral, B., Pereira, V., & Bernardino, J. (2021). Edge computing: current trends, research challenges and future directions. *Computing*, *103*(5), 993-1023.

Chen, X., Li, W., Lu, S., Zhou, Z., & Fu, X. (2018). Efficient resource allocation for on-demand mobile-edge cloud computing. *IEEE Transactions on Vehicular Technology*, *67*(9), 8769-8780.

Eshlaghy, A. T., Daneshvar, A., Peivandizadeh, A., Senathirajah, A. R. S., & Ibrahim, I. (2025). Designing a Sustainable Model for Providing Health Services Based on the Internet of Things and Meta-Heuristic Algorithms. *International Journal of Supply and Operations Management*, *12*(1), 28-47.

Gupta, N., & Singh, P. (2014). Load balancing using genetic algorithm in mobile cloud computing. *International Journal of Innovations in Engineering and Technology (IJIET)*, *4*(1), 157-162.

Habibi, M., & Navimipour, N. J. (2016). Multi-objective task scheduling in cloud computing using an imperialist competitive algorithm. *International Journal of Advanced Computer Science and Applications*, *7*(5).

Hosseini, S., & Al Khaled, A. (2014). A survey on the imperialist competitive algorithm metaheuristic: implementation in engineering domain and directions for future research. *Applied Soft Computing*, *24*, 1078-1094.

Ibrahim, M., Nabi, S., Hussain, R., Raza, M. S., Imran, M., Kazmi, S. A., Oracevic, A., & Hussain, F. (2020). A comparative analysis of task scheduling approaches in cloud computing. 2020 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID),

Jlassi, J., Rekik, I., Elloumi, S., & Chabchoub, H. (2023). Genetic Algorithm for Patients Scheduling in Emergency Department: A Case Study. *International Journal of Supply & Operations Management*, *10*(4).

Jula, A., Othman, Z., & Sundararajan, E. (2015). Imperialist competitive algorithm with PROCLUS classifier for service time optimization in cloud computing service composition. *Expert Systems with applications*, *42*(1), 135-145.

Long, S., Zhang, Y., Deng, Q., Pei, T., Ouyang, J., & Xia, Z. (2022). An efficient task offloading approach based on multi-objective evolutionary algorithm in cloud-edge collaborative environment. *IEEE Transactions on Network Science and Engineering*, *10*(2), 645-657.

Mahmoudi, A., Farzinvash, L., & Taheri, J. (2025). GPTOR: Gridded GA and PSO-based task offloading and ordering in IoT-edge-cloud computing. *Results in Engineering*, *25*, 104196.

Naas, S.-A. (2023). An AI-based Framework to Optimize Mobile Services.

Nanjappan, M., & Albert, P. (2022). Hybrid-based novel approach for resource scheduling using MCFCM and PSO in cloud computing environment. *Concurrency and Computation: Practice and Experience*, *34*(7), e5517.

Naouri, A., Wu, H., Nouri, N. A., Dhelim, S., & Ning, H. (2021). A novel framework for mobile-edge computing by optimizing task offloading. *IEEE Internet of Things Journal*, *8*(16), 13065-13076.

Pirozmand, P., Hosseinabadi, A. A. R., Farrokhzad, M., Sadeghilalimi, M., Mirkamali, S., & Slowik, A. (2021). Multi-objective hybrid genetic algorithm for task scheduling problem in cloud computing. *Neural computing and applications*, *33*(19), 13075-13088.

Pirozmand, P., Jalalinejad, H., Hosseinabadi, A. A. R., Mirkamali, S., & Li, Y. (2023). An improved particle swarm optimization algorithm for task scheduling in cloud computing. *Journal of Ambient Intelligence and Humanized Computing*, *14*(4), 4313-4327.

Ravi, J., Rajkumar, N., Viji, C., Loganathan, D., Sushma, K., & Stalin, M. (2023). Investigation of task scheduling in cloud computing by using imperialist competitive and crow search algorithms. *Procedia Computer Science*, *230*, 879-889.

Roustaei, R., & Yousefi Fakhr, F. (2018). A hybrid meta-heuristic algorithm based on imperialist competition algorithm. *Journal of AI and Data Mining*, *6*(1), 59-67.

Saemi, B., Hosseinabadi, A. A. R., Khodadadi, A., Mirkamali, S., & Abraham, A. (2023). Solving task scheduling problem in mobile cloud computing using the hybrid multi-objective Harris Hawks optimization algorithm. *IEEE Access*, *11*, 125033-125054.

Saemi, B., Sadeghilalimi, M., Hosseinabadi, A. A. R., Mouhoub, M., & Sadaoui, S. (2021). A new optimization approach for task scheduling problem using water cycle algorithm in mobile cloud computing. 2021 IEEE Congress on Evolutionary Computation (CEC),

Sen, P., Islam, T., Pandit, R., & Sarddar, D. (2024). A comparative review on different techniques of computation offloading in mobile cloud computing. *Fog Computing for Intelligent Cloud IoT Systems*, 33-44.

Shakkeera, L. (2025). Efficient task scheduling and computational offloading optimization with federated learning and blockchain in mobile cloud computing. *Results in Control and Optimization*, *18*, 100524.

Silva, F. A., Fe, I., Brito, C., Araujo, G., Feitosa, L., Nguyen, T. A., Jeon, K., Lee, J.-W., Min, D., & Choi, E. (2024). Aerial computing: Enhancing mobile cloud computing with unmanned aerial vehicles as data bridges—A Markov chain based dependability quantification. *ICT Express*, *10*(2), 406-411.

Singh, P., Dutta, M., & Aggarwal, N. (2017). A review of task scheduling based on meta-heuristics approach in cloud computing. *Knowledge and Information Systems*, *52*(1), 1-51.

Sitaraman, S. R., Gattupalli, K., Gollavilli, V. S. B. H., Nagarajan, H., Alagarsundaram, P., & Khalid, H. M. (2025). Secured user authentication and data sharing for mobile cloud computing using 2C-Cubehash and PWCC. *Sustainable Computing: Informatics and Systems*, *46*, 101107.

Sitompul, P., Soelistya, D., Simanihuruk, P., Purwati, T., & Efendi, E. (2024). Impact of Work-Life Balance and Work Engagement on Innovative Work Behavior. *International Journal of Supply and Operations Management*, *11*(4), 448-461.

Wang, T., Wei, X., Tang, C., & Fan, J. (2018). Efficient multi-tasks scheduling algorithm in mobile cloud computing with time constraints. *Peer-to-Peer Networking and Applications*, *11*(4), 793-807.

Xu, S., Ma, J., Lu, Q., Xie, Z., & Song, X. (2025). UAV-Edge Cloud collaboration for online offloading and trajectory control in multi-layer Mobile Edge Computing. *Ad Hoc Networks*, 103866.