A Comparative Study of Metaheuristic Optimization Algorithms for Joint Task Offloading and Resource Allocation in Mobile Edge Computing Systems

Le Van Hoa[1][0000-0001-6621-8617], Pham Trung Duc[2][0000-0003-2303-9920], Pham Trong Nguyen[3,4][0009-0007-2461-5957], Vo Viet Minh Nhat[5][0000-0002-0686-5529]

[1] School of Hospitality and Tourism – Hue University, Hue City, Vietnam
[2] Faculty of Technology and Engineering, Dong Thap University, Dong Thap, Vietnam
[3] University of Sciences, Hue University, Hue City, Vietnam
[4] Saigon University, Ho Chi Minh City, Vietnam
[5] Department of Academic and Students' Affairs – Hue University, Hue City, Vietnam

ptnguyen.dhkh25@hueuni.edu.vn

Abstract
Mobile Edge Computing (MEC) is a promising framework to support latency-sensitive and computationally intensive applications for the Internet of Things (IoT) in edge computing systems. The joint task offloading and resource allocation (TORA) policy is important in multi-user, multi-server MEC systems to minimize execution delay, energy consumption, and resource usage. This paper investigates the TORA problem with a comparative study of metaheuristic optimization algorithms. Specifically, the TORA problem is modeled as a constrained mixed-integer nonlinear programming model that minimises the joint consideration of the execution delay, the energy consumption, the connectivity availability and the resource utilisation on an edge server while meeting latency and energy constraints. In this paper, five popular representative metaheuristic algorithms: genetic algorithm (GA), particle swarm optimization (PSO), artificial bee colony (ABC), cuckoo search (CS) and firefly algorithm (FA), are integrated into a unified optimization framework with a shared representation, discretisation scheme, constraint-handling technique and fitness evaluation method. The simulations reveal that GA achieves the lowest cost (1.72×10^{-2}) and the lowest failure rate (6%), with an offloading rate of 65%, while PSO achieves the lowest execution latency, energy consumption and optimisation time due to fast convergence, with a cost of 1.79×10^{-2} and a failure rate of 18%. ABC and CS provide competitive results in between the best and the worst of algorithms whereas FA shows the highest cost (2.26×10^{-2}) and failure rate (27%) among the five evaluated algorithms.

Keywords: Mobile Edge Computing (MEC), Task Offloading and Resource Allocation (TORA), Metaheuristic Optimization, Cost Minimization, Comparative Performance Evaluation.

1 Introduction

As wireless and networking technologies continue to develop, Internet of Things (IoT) application areas are broadening and many IoT devices generate continuously data which is processed by analysis and control at the real-time. Due to a massive amounts of computational data processing for these devices and high availability requirements from application owners, cloud computing technology has emerged and played a critical role to enable massive IoT applications. However, centralized data processing performed on cloud servers may result in issues like network bandwidth saturation due to massive amount of data and the considerable data transmission time, thus cannot address these limitations in real-time application [1]. Mobile Edge Computing (MEC) concept comes into use to solve these challenges, moving computation resources from centralized cloud servers to the network edge servers, which is near end-users [2].

Offloading computation to nearby edge servers reduces service delays, improves system efficiency, and achieves better quality of service (QoS). In multi-user MEC system, users are equipped with several types of mobile devices and perform computational tasks either locally on devices or offloaded to the edge servers. The objective is to jointly optimize task offloading decisions and resource allocation among edge servers to achieve a low execution delay and low energy consumption, as well as efficiently utilize edge resources. Because of its mixed-integer nonlinear characteristics, it falls into NP-hard category, therefore, metaheuristic optimization approach has become the commonly used approach to solve this problem [3-5].

While most of the previous works focus on jointly optimizing the task offloading and resource allocation in MEC environments in order to minimize delay or energy, this work, however, considers and incorporates the status of wireless connectivity and resource utilization of edge servers into the objective function, which results in a comprehensive formulation of the TORA problem. In this paper, we focus on the TORA problem in multi-user and multi-server MEC environments. Five well-known metaheuristic algorithms such as GA, PSO, ABC, CS, and FA, have been adopted to solve the TORA problem in a unified metaheuristic optimization framework to ensure that they run under the same encoding, constraints handling method, and objective function evaluation scheme for comparison of solution quality, convergence, efficiency and robustness [6-8].

The main contributions of this paper are summarized as:

- We proposed a mixed-integer nonlinear constrained optimization model for the TORA problem to jointly minimize execution delay, energy consumption, wireless connectivity penalty and edge resource allocation penalty, while satisfying QoS constraints.
- We developed a unified metaheuristic optimization framework to ensure fair performance evaluation for different candidate metaheuristic algorithms through uniform encoding strategy, discretization of continuous variables, constraint satisfaction, and common objective function calculation, including GA, PSO, ABC, CS, and FA.

- Extensive simulation results show that GA provides the best cost and failure rate with the highest successful offloading rate. Meanwhile, PSO exhibits the fastest convergence time, lowest execution delay, and minimal energy consumption.

This paper is organized as follows. In Section II, we introduce the related work of TORA problem in MEC environment. Section III introduces the system model and formulates the TORA problem. Section IV presents the unified metaheuristic optimization framework. Section V discusses the experimental setup and results, and compares the performance of selected metaheuristic algorithms. Finally, Section VI concludes the paper and points out some possible directions for future research.

2 Related works

As one of the most promising means to achieve a resource-limited MEC system, the TORA strategy is considered. When multiple users exist, and there exist multiple servers for the MEC, the decisions of offloading tasks to a server and resource allocation must jointly optimize the wireless connectivity, the available resources, the processing capacity, the communication latency and energy consumption, respectively. Since servers selection is a discrete process, and communication and computation constraints are usually non-linear, the TORA problem is often formulated as a large-scale NP-hard mixed-integer programming problem [9]. Hence, many metaheuristic optimization algorithms have been proposed to seek the high-quality solutions within an acceptable running time.

For MEC optimization, GA is the most popular metaheuristic algorithm. Many previous studies employed GA and its improved variants to minimize the energy consumption, the response time, and allocate tasks in a cloud-fog system [10], a UAV assisted MEC system [11], a vehicular IoV system [12], and a MEC system based on sensors with tasks dependence [13], respectively. Unfortunately, most of the proposed works focused on individual goals such as latency or energy, and neglected features like mobility, varying networks topology, flying energy or dynamic resources, etc. Cooperative multi-hop MEC with enhanced intelligent optimization was addressed using the HHO-GA algorithm for a Green Smart City MEC system [14], but Cooperative MEC was not well explored.

PSO is also frequently researched, owing to its quick convergence speed and excellent search performance. PSO variants have been used for resource management in IIoT networks [15], optimal selection in edge-cloud interaction systems[16], drone-assisted MEC [17] and the Hybrid GA-PSO approach for resource management in MEC [18]. While most of these algorithms contribute in improved latency, energy cost, convergence speed, these works did not take into account user mobility, multi-hop offloading or sparse MEC system scenarios.

Apart from the most prominent algorithms, several nature-inspired algorithms achieve competitive performance. ABC has been applied for the total cost optimization of a multi-tier MEC system [19], while the ACO algorithm addresses task allocation [20]. In addition, FA algorithms have been developed for the optimal selection in the

edge-fog-cloud environment [21]. These papers did not typically consider joint optimization including communication connectivity.

The existing works also focus on single or dual objective optimization with simplified offloading strategy (often binary offloading), and there is no comprehensive comparative study to identify the performance of the most representative metaheuristic algorithms under the same experimental setup. Hence, there are two significant research gaps: First, the availability of wireless connectivity and resources are usually omitted from the joint optimization with the latency and energy consumption. Second, to make a fair comparison, most works used different system setups, so only evaluate several metaheuristic algorithms individually. To address the limitations, we formulate TORA problem into a constrained mixed-integer non-linear optimization problem and then develop a universal optimization framework to make a comprehensive comparison among GA, PSO, ABC, CS and FA with an identical objective function, constraint-handling method, representation, and environment parameters.

3 System model and problem formulation

3.1 System model

In this work, a Mobile Edge Computing (MEC) system consisting of N User Devices (UDs) and M Edge Servers (ESs) is considered, as depicted in Figure 1.

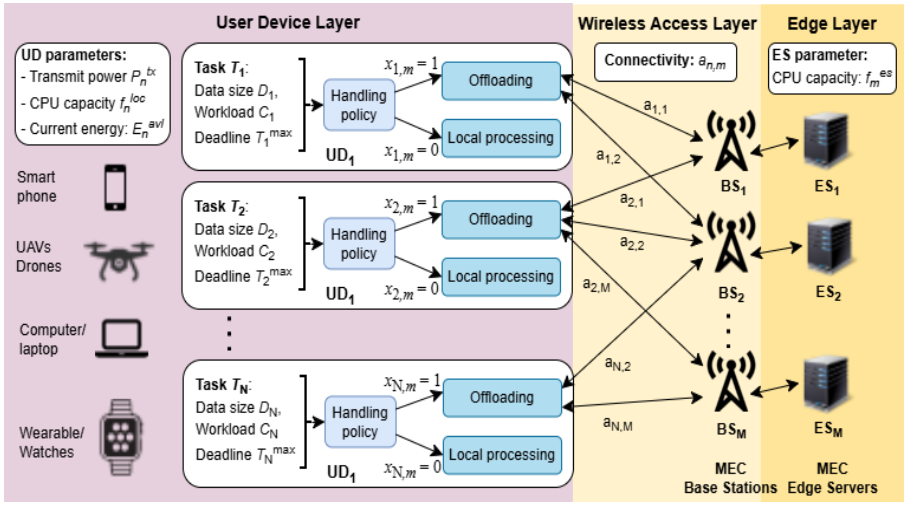


Fig. 1. Joint task offloading and resource allocation in a multi-user multi-server MEC system.

ESs are deployed at the network edge and communicate with UDs through wireless links. ESs are located at the edge of the network and communicate with UDs through wireless links. The wireless link between UDs and ESs is denoted by a matrix $A \in \{0,1\}^{N\times M}$. Each element $a_{n,m}$ in matrix A indicates the condition of the link between

UD n and ES m as follows. $a_{n,m} = 1$ means that UD n is conved by in the communication range of ES m and $a_{n,m} = 0$ means that there is no link between UD n and ES m.

Each UD n can be described by three parameters at local layer: $U = \{f_n^{loc}, E_n^{avl}, P_n^{tx}\}$, where f_n^{loc} is the maximum local computing capacity (in CPU cycles/second), E_n^{avl} is the available energy, and P_n^{tx} is the transmission power in the uplink direction.

For each time slot, we assume that each UD generates one task, and the task index corresponds to the index of its originating UD. Therefore, with (N) UDs, each time slot contains (N) tasks, with one task generated by each UD. A task is characterized by a triplet $T = \{D_n, C_n, T_n^{max}\}$, where D_n is the task data size (in bits), C_n is the total number of CPU cycles needed to accomplish the task and T_n^{max} is the maximum tolerable latency of the task.

The energy consumption considered in the optimization consists of the energy for local computation at UDs and the energy required for uplink transmission when tasks are offloaded to ESs. The energy consumed for task execution at an ES is assumed to be constant and independent of the task offloading and resource allocation decisions; therefore, it is not included as a decision-dependent term in the energy model.

Multiple UDs may attempt to offload tasks concurrently to a single edge server. Each edge server has a computing capacity of f_m^{es} (in Cycles/s) and a limited queue for receiving offloaded tasks. To avoid overload and imbalance, a penalty related to server queue resource allocation is added to the objective function (see Section 3.4).

3.2 Local Computing Model

When the task is computed locally on the UD n, the latency is

$$T_n^{loc} = \frac{C_n}{f_n^{loc}} \tag{1}$$

The energy consumption for the local part under the dynamic voltage and frequency scaling model [22] can be described as

$$E_n^{loc} = \kappa_n C_n (f_n^{loc})^2 t_n^{loc} \tag{2}$$

where κ_n is the switched capacitance coefficient for UD n.

3.3 Offloading Computing Model

When the task is offloaded to ES m, total offloading latency is composed by uplink transmit latency ($T_{n,m}^{up}$) and server execute latency ($T_{n,m}^{edge}$). The downlink feedback latency from server to the UD is ignored because most IoT tasks contain mostly downloading data and its size is far smaller than uploading data, especially compared to most of uplink tasks in the studies [23].

Under the assumption of the predefined bandwidth resource allocation of system, the uplink transmit rate $R_{n,m}$ from UD n to ES m according to the Shannon-Hartley theorem [24] is:

$$R_{n,m} = B_{n,m} log_2 \left(1 + \frac{P_n^{tx} g_{n,m}}{\sigma^2 + I_{n,m}}\right) t_n^{loc} \quad (3)$$

where $B_{n,m}$ is the bandwidth evenly allocated to UD-ES connections, $I_{n,m}$ denotes the co-channel interference, and σ^2 denotes the background noise power.

The uplink transmission latency and corresponding energy consumption are calculated by:

$$T_{n,m}^{tx} = \frac{D_n}{R_{n,m}} t_n^{loc} \quad (4)$$

$$E_{n,m}^{tx} = P_n^{tx} T_{n,m}^{tx} \quad (5)$$

The execute latency of the offloaded task on ES *m* is given by:

$$T_{n,m}^{es} = \frac{C_n}{f_m^{es}} t_n^{loc} \quad (6)$$

3.4 Problem Formulation

The purpose of this problem is the so-called task offloading and resource allocation, to reduce the overall system costs in computation delay, energy consumption, wireless communication and resource usage. Let us introduce the binary decision variable $x_{n,m}$ which take 0 or 1, such that: $x_{n,m} = 1$ if task *n* is assigned on edge server *m*, and $x_{n,m} = 0$ for the other case. With $x_{n,m}$ being selected, the total time spent to compute task *n* and and the energy consumption would be given by the expressions below:

$$T_n = \left(1 - \sum_{m=1}^{M} x_{n,m}\right) T_n^{loc} + \sum_{m=1}^{M} x_{n,m} \left(T_{n,m}^{tx} + T_{n,m}^{es}\right) \quad (7)$$

$$E_n = \left(1 - \sum_{m=1}^{M} x_{n,m}\right) E_n^{loc} + \sum_{m=1}^{M} x_{n,m} E_{n,m}^{tx} \quad (8)$$

Equation (8) accounts for the UD-side energy consumption: local computation energy when a task is executed locally and uplink transmission energy when the task is offloaded. The execution energy at the ES is not included because it is assumed to be constant and independent of the offloading decision and resource allocation in the considered model.

To evaluate the overall optimization objective, the proposed framework employs a composite cost function, denoted by *Cost*, consisting of four components: delay cost, energy cost, connectivity penalty, and edge resource allocation penalty. Cost is defined as:

$$Cost = \sum_{n=1}^{N} \left(\alpha_1 \frac{T_n}{T_n^{max}} + \alpha_2 \frac{E_n}{E_n^{max}}\right) + \alpha_3 P_{disc} + \alpha_4 P_{fail_alloc} \quad (9)$$

where $\alpha_1, \alpha_2, \alpha_3$ and α_4 denote the weights associated with the delay cost, the energy cost, the connectivity penalty, and the edge resource allocation penalty, respectively. The weighting coefficients satisfy $\sum_{i=1}^{4} \alpha_i = 1$. Here, E_n^{max} denotes the maximum energy of user device *n*.

The failure rate is not a direct component of the composite cost function in Equation

(9). Instead, the optimization objective jointly considers execution delay, energy consumption, connectivity penalty, and edge resource allocation penalty, while the failure rate is reported separately as a performance indicator in the simulation results.

The connectivity penalty is measured by the normalized disconnection ratio because the higher disconnection ratio means worse connectivity of network. Then its is defined as the ratio between the number of user devices that become disconnected during task offloading (N_{unoff}) to the total number of user devices (N):

$$P_{disc} = \frac{N_{disc}}{N} \tag{10}$$

The penalty for failure to allocate resources is defined as:

$$P_{fail_alloc} = \frac{1}{M}\sum_{m=1}^{M} \max(\frac{L_m}{\mu_m} - 1, 0) \tag{11}$$

where L_m denotes the total computational workload assigned to edge server m, while μ_m denotes the computational capacity of edge server m. The edge resource allocation penalty is activated only when the allocated workload exceeds the server capacity ($\frac{L_m}{\mu_m} > 1$), thereby preventing server overload and promoting balanced utilization of edge computing resources.

The optimization objective is to determine the task offloading decision matrix $X = \{x_{n,m}\}$ so that minimizes the overall system cost:

$$\min_{X,A} Cost \tag{12}$$

Subject to:

$$\sum_{m=1}^{M} x_{n,m} \leq 1, \forall n \tag{12a}$$

$$T_n \leq T_n^{max}, \forall n \tag{12b}$$

$$E_n \leq E_n^{avl}, \forall n \tag{12c}$$

$$x_{n,m} \leq a_{n,m}, \forall n, m \tag{12d}$$

Constraint (12a) ensures that each task can be assigned to at most one edge server. Constraints (12b) and (12c) guarantee that the execution delay and energy consumption satisfy the QoS requirements. Constraint (12d) ensures that task n can be assigned to edge server m only when a link is available between them.

4 Metaheuristic framework for task offloading optimization

All chosen algorithms apply a common solution representation, a discretization technique, a method for handling constraints and a stopping condition to provide fair comparisons. Each potential solution is modelled as a vector $Y = [y_1, y_2, \ldots, y_N]$ with $y_n \in \{0,1, \ldots, M\}$,, where $y_n = 0$ denotes local execution and $y_n = m$ denotes that task n is offloaded to edge server m. As optimization algorithms operate in continuous search space, the solution is discretized using rounding to the nearest integer and boundary control mechanisms, followed by a repair operator, to ensure that all task offloadings

satisfy energy and connectivity constraints. Finally, the optimization algorithm stops when the global best solution hasn't changed for the last fixed number of iterations.

GA starts with a random population and selection is performed using tournament selection to select the high quality individuals (parents). Crossover operation (arithmetic crossover) produces offspring based on parental information, and Gaussian mutation can add randomness in order to prevent premature convergence by randomly changing task assignments. The elite strategy (top 30% of current population) is preserved into the next generation for the convergence stability [25].

In PSO, particles represent the solutions and fly through the search space. Each particle update velocity and position based on the personal best position found so far and the global best position found. Before discretizing and repairing the position, we apply a constraint in particle velocities with $[-V_{max}, V_{max}]$ for the stability.

CS utilize Lévy flights for updating candidate solutions, thus providing long-distance movement with a certain probability to escape from local optima. A certain percentage of worse nests is abandoned and replaced with new random solutions in random walk paths.

ABC consists of three groups: employed bees, onlooker bees and scout bees. Employed bees modify the food sources locally, onlooker bees choose a food source probabilistically based on the fitness of the current population and scout bees replace lost food sources with new random ones to search globally.

FA models candidate solutions as fireflies which attract each other according to brightness. The attractive force is based on Euclidean distance of two solutions and updating position rules.

5 Simulation and result analysis

The simulation parameters presented in Table 1 were selected based on parameter settings commonly adopted in MEC task-offloading simulation studies [26, 27] and were chosen to represent a typical MEC system configuration. For the objective-function weights, α_1 and α_2 are both set to 0.35, while α_3 and α_4 are set to 0.10 and 0.20, respectively. These values place greater emphasis on execution delay and energy consumption while still accounting for connectivity and edge-resource allocation penalties.

Table 1. Main parameters for the algorithms: GA, PSO, CS, ABC, and FA

Parameter	Description	Value
N	Number of user devices (UD)	100
M	Number of edge servers (ES)	4
D_n	Task data size	[5, 15] MB
C_n	Required CPU cycles for a task	[2, 8] Gcycles
f_n^{loc}	Computing capacity of UD n	[200, 800] MHz
T^{max}	Maximum delay constraint (for all UDs)	0.3 s
E^{avl}	Available energy constraint (for all UDs)	0.05 J
$\alpha_1, \alpha_2, \alpha_3, \alpha_4,$	Cost weighting coefficients	$0.35, 0.35, 0.10, 0.20$
P_n^{tx}	Transmission power of UD n	[0.2, 0.6] W
f_m^{es}	Computing capacity of ES m	[4, 10] GHz

$\omega_n^T,\ \omega_n^E$	The weighting factor for latency vs. energy	0.6, 0.4
σ^2	Background noise power	1×10-7 W
$g_{n,m}$	Wireless channel gain between UD n and ES m.	0.07

We considered five metaheuristic algorithms as GA, PSO, CS, ABC and FA. Each of the metaheuristic algorithms was configured with a population size of 100 candidate solutions and the repeated-run experiments were conducted 10 times to avoid excessive stochastic results influenced by specific runs. The parameters for the metaheuristic algorithms are described in Table 2.

Table 2. Configuration parameters for the algorithms: GA, PSO, CS, ABC, and FA.

Algorithm	Parameter	Description	Value
GA	*mutation_rate*	Mutation probability	0.1
	crossover	Crossover method	Arithmetic
	selection	Selection strategy	Tournament (k = 3)
PSO	W	Inertia weight	0.7
	c_1, c_2	Cognitive and social coefficient	1.5, 1.5
	velocity_range	Velocity bound	[-3, 3]
CS	p_a	Discovery probability of foreign eggs	0.25
	β	Lévy flight exponent	1.5
ABC	*Limit*	Abandonment trial threshold	10
	employed_bees	Number of employed bees	30
	onlooker_bees	Number of onlooker bees	30
FA	β_0	Initial attractiveness	1.0
	γ	Light absorption coefficient	1.0
	θ	Randomization parameter	0.2

5.1 Performance Evaluation of Metaheuristic-based TORA Strategies

Table 3 and Figs. 2–6 display the averaged simulation results of the five metaheuristic algorithms over 10 independent runs. In the considered MEC model, any task falls into either local execution, successful offloading, or failure. Thus, the sums of the local execution rate, offloading rate, and failure rate is100%.

Table 3. Performance metrics comparison (Average results)

Method / Metric	GA	PSO	ABC	CS	FA
Local execution rate (%)	29	56	46	33	23
Offloading rate (%)	65	**26**	36	50	50
Failure rate (%)	**6**	18	18	17	27
Cost	$\mathbf{1.72\times10^{-2}}$	1.79×10^{-2}	1.91×10^{-2}	1.97×10^{-2}	2.26×10^{-2}
Latency (s)	2.91×10^{-2}	$\mathbf{1.30\times10^{-2}}$	2.13×10^{-2}	3.09×10^{-2}	3.35×10^{-2}
Energy consumption (J)	1.01×10^{-2}	$\mathbf{3.79\times10^{-3}}$	8.04×10^{-3}	1.08×10^{-2}	1.10×10^{-2}

Execution time (s)	4.42	**3.57**	7.55	3.67	52

As observed in Table 3 and Figs. 2–6, all the algorithms have their respective trade-offs between the Cost, Latency, Energy consumption, Offloading rate, Failure rate and Execution time. Since the defined objective function optimizes multiple objectives simultaneously, it has more robust evaluation power than a solitary performance measure such as Latency or Energy consumption. GA achieves the lowest cost of 1.72×10^{-2} among the five evaluated algorithms, together with the lowest failure rate of 6% and the highest offloading rate of 65%. Crossover and mutation operations used by GA balance the exploration and exploitation abilities and are efficient in generating feasible task offloading solutions compared to the others. However, GA is not optimal in terms of the lowest latency or energy consumption alone, but it does exhibit the lowest cost among the five evaluated algorithms.

PSO performs better than other algorithms in terms of the lowest execution latency of 1.30×10^{-2} s, lowest energy consumption of 3.79×10^{-3} J and lowest optimization time of 3.57 s due to its faster convergence. However, its lowest offloading rate (26%) and relatively high failure rate (18%) cause its cost higher than GA, despite superior performance in terms of Latency and Energy consumption.

CS and ABC offer a fair performance. Although ABC achieves fairly good optimisation via a cooperative search approach, its failure rate is still 18%. CS, leveraging the Lévy flights for excellent global exploration ability, produces a lower failure rate (17%) than ABC; however, the latency, energy consumption, and cost is still higher than GA.

FA delivers the overall poor performance. FA shows the highest cost (2.26×10^{-2}), latency, energy consumption, failure rate (27%) and execution time (52 s) compared to other algorithms. Pairwise attraction strategy of FA increased the computability and sensitivity, leading to poor convergence and lower optimization efficiency than the other meta heuristic algorithms.

5.2 Convergence Analysis of Metaheuristic Algorithms

Convergence property of the proposed cost function is another performance measure that depicts the ability of each metaheuristic to optimize and explore the search space of TORA. This objective function does not only consider the execution latency and energy consumed by the MEC nodes, but it also combines all the four aspects (delay, energy consumption, penalties of connectivity and edge node resources) to achieve better TORA performance compared to traditional approaches that are often concerned about the execution latency and energy cost of MEC applications. The convergence of cost, latency and energy consumption towards the minimum are presented in Figures 2–6 respectively, which converge over 100 iterations.

PSO presents quick convergence among the algorithms, reaching an almost converged cost value of 1.79×10^{-2} within the initial few (first 10) generations (Fig. 2), due to velocity and position updates which favor quick exploration of the search space, hence very suitable for latency sensitive MEC applications. The GA converges relatively slowly but ultimately achieves the lowest cost of 1.72×10^{-2} after 100 iterations (Fig. 3), whereas ABC, CS and FA converge slowly and maintain higher cost at the end

of the runs (Fig. 4, 5 and 6 respectively).

In terms of execution latency, PSO achieve quickly best latency (1.30×10^{-2} s), whereas for other metaheuristics such as ABC and CS, more number of iterations are required to reach high values. GA exhibits larger variance over the iteration range, since mutation and crossover continue to introduce new solutions, whereas GA and FA converges slowly but consistently.

For energy consumption, PSO attains its lowest value (3.79×10^{-3} J) more quickly than other algorithms. While GA, ABC, CS and FA successfully reduce energy consumption, they are relatively slow in approaching the optimal level and stabilize at values that are slightly higher.

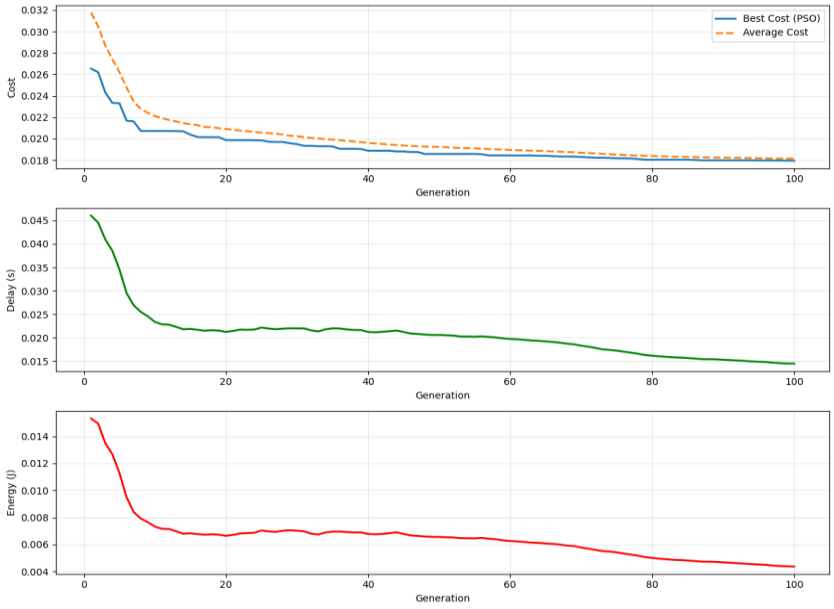


Fig. 2. Convergence of the cost, delay, and energy consumption with PSO.

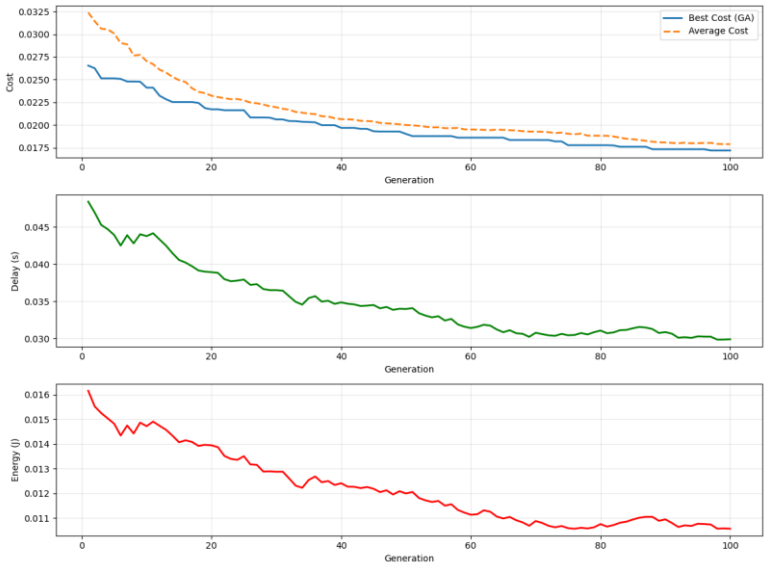


Fig. 3. Convergence of the cost, delay, and energy consumption with GA.

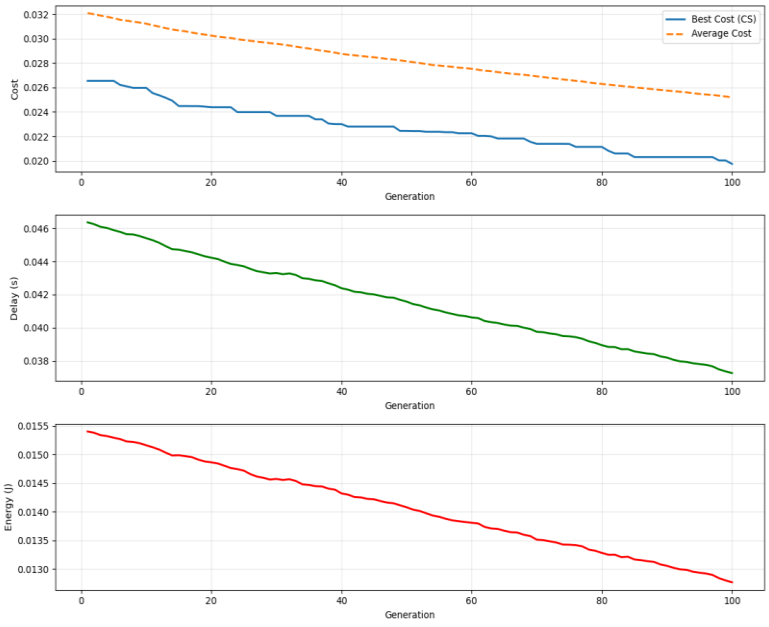


Fig. 4. Convergence of the cost, delay, and energy consumption with CS.

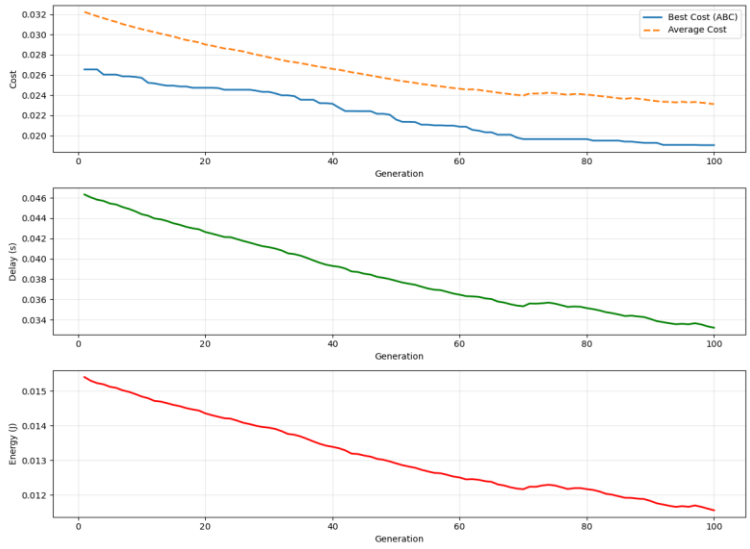


Fig. 5. Convergence of the cost, delay, and energy consumption with ABC.

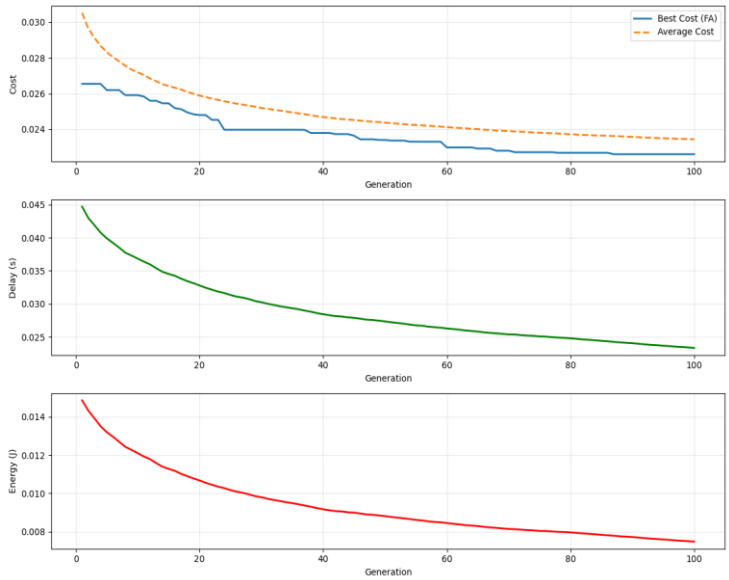


Fig. 6. Convergence of the cost, delay, and energy consumption with FA.

In summary, PSO achieves the lowest execution latency, energy consumption, and optimization time among the five evaluated algorithms, while GA achieves the lowest objective-function value of 1.72×10^{-2} among the five evaluated algorithms, together with the lowest failure rate and highest offloading rate. ABC and CS provide similar

reasonable trade-offs, whereas FA provides the highest cost and failure rate among the evaluated algorithms.

6 Conclusion

This paper studies the joint task offloading and resource allocation (TORA) problem in multi-user multi-server mobile edge computing (MEC) scenario with comparative metaheuristic optimization framework. This problem is modeled as mixed-integer non-linear optimization problem which simultaneously minimize the execution delay, energy consumption, connectivity penalty, and edge resource allocation penalty. In order to tackle this problem, five metaheuristics: Genetic Algorithm (GA), Particle Swarm Optimization (PSO), Artificial Bee Colony (ABC), Cuckoo Search (CS) and Firefly Algorithm (FA) were compared with same optimization framework.

Through extensive simulations, each algorithm provides its different performance tradeoff with cost, delay, energy consumption, success rate, failure rate, and computation time. GA has lowest cost, failure rate with high successful offloading rate. PSO results lowest delay, energy consumption, and optimization time at the expense of high failure rate. ABC and CS provide similar reasonable tradeoff, but FA provides the lowest overall performance among all of them. The comparison shows that each algorithm has their strengths in different performance indicators and it is crucial to choose the right one according to the applications requirement. A novel composite objective function is designed to simultaneously reflect latencies, energy consumptions, connectivities and resource allocations.

In future work, our framework can be extended to study dynamically fluctuating MEC environment with user mobility, time-varying channels, stochastic task arrivals and heterogeneous edge servers. The hybridized metaheuristic approach, such as combined metaheuristics or learning-assisted approach, will also be explored to further enhance the convergence, solution quality and efficiency.

Acknowledgments

This work was partially supported by Hue University under the Core Research Program, Grant No. NCTB.DHH.2025.09.

References

1. F. C. Andriulo, M. Fiore, M. Mongiello, E. Traversa, and V. Zizzo: Edge Computing and Cloud Computing for Internet of Things: A Review. Informatics 11(4) (2024). doi: 10.3390/informatics11040071
2. L. A. Haibeh, M. C. E. Yagoub, and A. Jarray: A Survey on Mobile Edge Computing Infrastructure: Design, Resource Management, and Optimization Approaches. IEEE Access 10, 27591–27610 (2022). doi: 10.1109/ACCESS.2022.3152787
3. W. Dayong, K. Bin Abu Bakar, B. Isyaku, T. Abdalla Elfadil Eisa, and A. Abdelmaboud: A comprehensive review on internet of things task offloading in multi-access edge computing. Heliyon 10(9), e29916 (2024). doi: 10.1016/j.heliyon.2024.e29916

4. K. Sadatdiynov, L. Cui, L. Zhang, J. Z. Huang, S. Salloum, and M. S. Mahmud: A review of optimization methods for computation offloading in edge computing networks. Digit. Commun. Networks 9(2), 450–461 (2023). doi: 10.1016/j.dcan.2022.03.003
5. C. Feng, P. Han, X. Zhang, B. Yang, Y. Liu, and L. Guo: Computation offloading in mobile edge computing networks: A survey. J. Netw. Comput. Appl. 202(April) (2022). doi: 10.1016/j.jnca.2022.103366
6. S. Alali and A. Assalem: Metaheuristics Method for Computation Offloading in Mobile Edge Computing: Survey. J. Adv. Res. Appl. Sci. Eng. Technol. 36(1), 43–73 (2024). doi: 10.37934/araset.36.1.4373
7. R. Latip, J. Aminu, Z. M. Hanafi, S. Kamarudin, and D. Gabi: Metaheuristic task offloading approaches for minimization of energy consumption on edge computing: a systematic review. Discov. Internet Things 4(1) (2024). doi: 10.1007/s43926-024-00089-y
8. Y. Zheng: Survey Of Distributed Task Offloading Optimization Based on Heuristic Algorithms in Edge Computing. Acad. J. Sci. Technol. 17(1), 55–65 (2025). doi: 10.54097/y1vp2j42
9. S. Zhou, W. Jadoon, and I. A. Khan: Computing Offloading Strategy in Mobile Edge Computing Environment: A Comparison between Adopted Frameworks, Challenges, and Future Directions. Electron. 12(11) (2023). doi: 10.3390/electronics12112452
10. M. Keshavarznejad, M. H. Rezvani, and S. Adabi: Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms. Cluster Comput. 24(3), 1825–1853 (2021). doi: 10.1007/s10586-020-03230-y
11. H. Wang: Collaborative Task Offloading Strategy of UAV Cluster Using Improved Genetic Algorithm in Mobile Edge Computing. J. Robot. 2021 (2021). doi: 10.1155/2021/3965689
12. K. Wang, X. Wang, and X. Liu: Sustainable Internet of Vehicles System: A Task Offloading Strategy Based on Improved Genetic Algorithm. Sustain. 15(9) (2023). doi: 10.3390/su15097506
13. S. Chakraborty and K. Mazumdar: Sustainable task offloading decision using genetic algorithm in sensor mobile edge computing. J. King Saud Univ. - Comput. Inf. Sci. 34(4), 1552–1568 (2022). doi: 10.1016/j.jksuci.2022.02.014
14. I. Behera, S. Sobhanayak, and N. Almakayeel: Task offloading in heterogeneous Mobile Edge Computing using meta heuristic approach for sustainable smart city applications. Phys. Commun. 73(October), 102893 (2025). doi: 10.1016/j.phycom.2025.102893
15. Q. You and B. Tang: Efficient task offloading using particle swarm optimization algorithm in edge computing for industrial internet of things. J. Cloud Comput. 10(1) (2021). doi: 10.1186/s13677-021-00256-4
16. T. Gao, Q. Tang, J. Li, Y. Zhang, Y. Li, and J. Zhang: A Particle Swarm Optimization With Lévy Flight for Service Caching and Task Offloading in Edge-Cloud Computing. IEEE Access 10(July), 76636–76647 (2022). doi: 10.1109/ACCESS.2022.3192846
17. M. A. Attalah, S. Zaidi, N. Mellal, and C. T. Calafate: Task-Offloading Optimization Using a Genetic Algorithm in Hybrid Fog Computing for the Internet of Drones. Sensors 25(5), 1–21 (2025). doi: 10.3390/s25051383
18. W. Li, X. Sun, B. Wan, H. Liu, J. Fang, and Z. Wen: A hybrid GA-PSO strategy for computing task offloading towards MES scenarios. PeerJ Comput. Sci. 9, 1–23 (2023). doi: 10.7717/PEERJ-CS.1273
19. 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 2021 (2021). doi: 10.1155/2021/5563531

20. Y. T. Chuang and Y. T. Hung: A real-time and ACO-based offloading algorithm in edge computing. J. Parallel Distrib. Comput. 179, 104703 (2023). doi: 10.1016/j.jpdc.2023.04.004
21. F. A. Saif, R. Latip, Z. Mohd Hanapi, S. Kamarudin, A. V. Senthil Kumar, and A. Salem Bajaher: Multi-Objectives Firefly Algorithm for Task Offloading in the Edge-Fog-Cloud Computing. IEEE Access 12(September), 159561–159578 (2024). doi: 10.1109/ACCESS.2024.3488032
22. E. Cuervo et al.: MAUI: Making Smartphones Last Longer with Code Offload. In: Proceedings of the 8th international conference on Mobile systems, applications, and services, pp. 49–62. ACM, New York, NY, USA (2010). doi: 10.1145/1814433.1814441
23. S. Xiao, C. Liu, K. Li, and K. Li: System delay optimization for Mobile Edge Computing. Futur. Gener. Comput. Syst. 109, 17–28 (2020). doi: 10.1016/j.future.2020.03.028
24. C. E. Shannon: A Mathematical Theory of Communication. Bell Syst. Tech. J. 27(3), 379–423 (1948). doi: 10.1002/j.1538-7305.1948.tb01338.x
25. V. Tomar, M. Bansal, and P. Singh: Metaheuristic Algorithms for Optimization: A Brief Review. In: RAiSE-2023, p. 238. MDPI, Basel Switzerland (2024). doi: 10.3390/engproc2023059238
26. Z. Tong, X. Deng, F. Ye, S. Basodi, X. Xiao, and Y. Pan: Adaptive computation offloading and resource allocation strategy in a mobile edge computing environment. Inf. Sci. (Ny). 537, 116–131 (2020). doi: 10.1016/j.ins.2020.05.057
27. B. Zhang, X. Li, and J. You: Optimal Computation Offloading Decisions Based on System Utility and Cost Balance. IEEE Access 13(April), 68232–68244 (2025). doi: 10.1109/ACCESS.2025.3560291