Efficient Task Offloading Using Ant Colony Optimization and Reptile Search Algorithms in Edge Computing for Things Context

Ting Zhang*, Xiaojie Guo
School of Information Engineering, Jiaozuo University, Jiaozuo 454000, China

***Abstract*—The widespread use of Internet of Things (IoT) technology has triggered unparalleled data creation and processing needs, necessitating effective computation offloading solutions. Conventional edge computing approaches have difficulties in dealing with rising energy usage issues and task allocation delays. This study introduces a novel hybrid metaheuristic algorithm called ACO-RSA, which synergizes two metaheuristic algorithms, Ant Colony Optimization (ACO) and Reptile Search Algorithm (RSA). The proposed approach addresses the energy and latency issues associated with offloading computations in IoT edge computing environments. A comprehensive system design that effectively encapsulates the uplink transmission communication model and a personalized multi-user computing task load model is developed. The system considers various constraints, such as network latency, task complexity, and available computing resources. Based on this, we formulate an optimization objective suitable for computing outsourcing in the IoT ecosystem. Simulations conducted in a real-world IoT scenario demonstrate that ACO-RSA significantly reduces both time delay and energy consumption compared to benchmark algorithms, achieving up to 27.6% energy savings and 25.4% reduction in time delay. ACO-RSA exhibits robustness and scalability when optimizing task offloading in IoT edge computing environments.**

Keywords*—Task offloading; edge computing; ant colony optimization; reptile search algorithm; Internet of Things; energy efficiency

I. Introduction

The exponential growth of Internet of Things (IoT) devices has fundamentally transformed how data is produced, analyzed, and used in numerous sectors, including smart cities [1], healthcare [2], and automated manufacturing [3]. Nevertheless, the substantial increase in data flow places substantial computing requirements on centralized and distributed systems, requiring the investigation of sophisticated task offloading strategies [4]. Conventional cloud computing models encounter delay and data transfer capacity limitations. In this regard, edge computing presents a viable option by bringing computing resources close to data sources [5]. However, improving task offloading in edge computing contexts has distinct difficulties related to energy use and latency [6].

Heuristic optimization techniques have contributed to overcoming these issues by evaluating a variety of task offloading policies. An influential group of these heuristics is Ant Colony Optimization (ACO), a group of algorithms based on the pheromone-based learning of ants foraging [7]. Similar algorithms, such as the Reptile Search Algorithm (RSA), can overcome the exploration versus exploitation challenge by imitating hunter-reptile foraging strategies [8]. The methods have been proven to work on optimization problems across various applications, including image processing, power systems, and edge computing. While the former is difficult to pull out of premature convergence, the latter must be carefully controlled judiciously to avoid falling into unoptimal solutions.

In light of the aforementioned, this paper presents a new metaheuristic method called ACO-RSA, which merges two metaheuristic methods, ACO and RSA, to solve offloading problems in IoT edge computing. Our strategy aims to achieve the benefit of both ACO and RSA by minimizing energy use and time delays. The paper presents the system architecture, with a communication model relying on an uplink transmission and the distribution task model between multiple users. This paper next introduces an objective function for IoT context optimization.

The system design also includes a multi-user task distribution model and an uplink transmission communication model that considers transmission delays, the complexity of the tasks to be solved by standalone devices, and the available computing resources. We design an optimization target specifically for these systems to adapt to IoT task offloading mechanisms and conduct extensive simulations to demonstrate the effectiveness of ACO-RSA. Experimental results show that our solution ACO-RSA can effectively reduce the energy consumption and latency of existing algorithms, which has obvious advantages in practical IoT application scenarios.

The rest of this paper is organized as follows. Section II contains a detailed literature review, including previous work on task offloading and metaheuristic optimization techniques. Section III discusses the system model and problem formulation. Section IV discusses our ACO-RSA algorithm in detail. An experimental setup and simulation results are presented in Section V for evaluating the proposed approach's performance. Lastly, Section VI summarizes the key insights, limitations, and future research possibilities concluding the paper.

II. Background

A. Edge Computing in IoT

In recent years, cloud computing platforms have evolved into the first choice for provisioning services because of their

flexibility and cost-effectiveness [9]. Fundamentally, there has been a shift towards centralized systems for processing and storing data in large data centers [10]. The IoT is advancing rapidly, with major effects on various sectors of society, mainly healthcare, transportation, and manufacturing [11]. They have many IoT devices at the edge producing huge volumes of data, which may require very little or significant resources (e.g., storage and processing) [12]. Service delivery is a big problem in cloud systems. Additional challenges involve the placement of sensors and actuators in space, response time constraints, privacy issues, and data processing. Apart from cloud computing, these barriers are a driving force for innovation in solutions [13].

Edge computing places servers for computation and storage at the edges of the internet, relatively closer to where data is generated. The above approximations are due to the lower latencies, more privacy concerns considering data being transferred, and suboptimal energy overhead [14]. This hybridization extends edge computing capabilities to distribute and gradually interconnect with cloud-based processing components.

Fig. 1 shows the hybrid edge-cloud reference design encompasses a range of computing and storage capabilities. Thing nodes are small and constrained devices with limited computing and storage capabilities, so they can only do basic functions like detecting, acting in response to certain events, and sending and receiving data. In contrast to Thing nodes, local nodes are capable of more processing, storing, and communicating. This allows them to process, analyze, and send data and interface with both the cloud and thing nodes. A local node may also host IoT applications at the network edge.

A local node can be any device such as a gateway, access point, router, switch, local server, cellphone, or linked vehicle. Furthermore, various physical gadgets can fulfill certain functions. A traffic light, for example, can be both a thing node that detects its surroundings and acts accordingly and a local node that collects and analyzes data. Centralized data centers provide extensive processing, storage, and communication capabilities in the cloud. Due to these characteristics, cloud solutions can accommodate many IoT components that require substantial physical resources.

B. Task Offloading Challenges

In edge computing settings, task offloading has distinct issues that must be addressed for efficient and effective task distribution. As shown in Fig. 2, the problems arise from IoT devices' intrinsic attributes, edge computing's decentralized nature, and dynamic and unexpected network circumstances. IoT devices are often situated in geographically scattered locations with diverse network conditions. Latency may impede communication between these devices and edge servers, particularly in cases of network congestion or when the devices are placed far from the edge nodes. In addition, a restricted amount of available bandwidth might result in congestion during data transmission, which causes delay problems.

IoT applications can include a wide range of functions with different levels of complexity, including basic data processing and more demanding processes like real-time video analysis or machine learning inference [15]. Because different workloads may have no shared characteristics and time constraints, the diversity of these jobs adds difficulty for offloading methods. In addition, edge servers could differ widely in the processing power, memory, and storage they support [16].

Energy efficiency is crucial in IoT systems since devices often have limited battery capacity. Task offloading may mitigate the computing burden on IoT devices, thereby conserving their energy [17]. Nevertheless, offloading necessitates energy use, particularly in data transmission to edge servers and the reception of results.

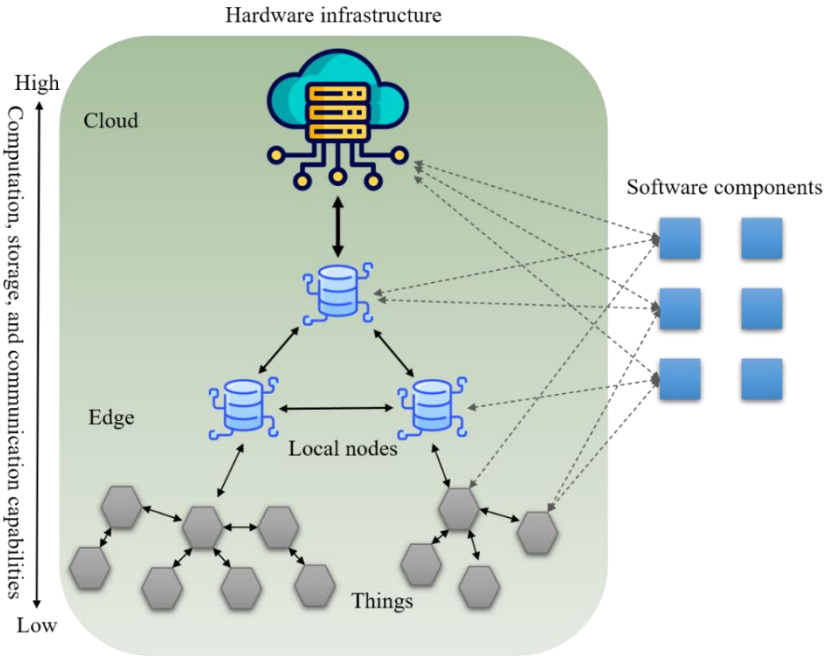

Fig. 1. Edge-cloud reference architecture.

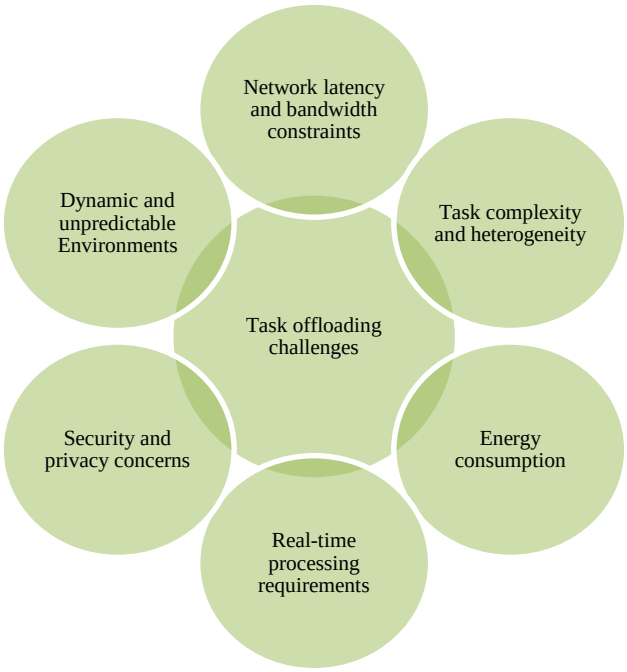

Fig. 2. Task offloading challenges.

IoT applications, including autonomous cars, smart manufacturing, and healthcare monitoring, need immediate processing. During these situations, even a small delay in carrying out a job might result in significant repercussions, highlighting the need to reduce latency and guarantee prompt task fulfillment [18]. The task involves creating offloading algorithms that provide real-time performance while distributing the computing workload across several edge servers.

Transmitting sensitive data via networks might expose it to interception or unwanted access. Moreover, the decentralized structure of edge computing increases attack vulnerability, making it more difficult to protect all nodes from possible risks. The IoT ecosystem is intrinsically characterized by its dynamic nature, including fluctuating network conditions, shifting workloads, and varied resource availability at edge servers. These variables contribute to an uncertain environment that complicates the job offloading process.

C. Edge Computing-Enabled Task Offloading Mechanisms

Xiao, et al. [19] proposed a three-layer IoT structure that addresses different aspects of tasks. Edge computing and blockchain architectures address the problem of sensitivity to task delays and enable service providers to maximize their profits. In addition, due to the features of the edge computing server in the second layer, the proposed algorithm is executed as a smart contract. This smart contract is responsible for managing and distributing edge computing resources. In particular, a complementary approach called stacked cache is proposed to facilitate the fair distribution of resources on edge computing servers.

You and Tang [20] proposed an energy-efficient, low-delay PSO-based task offloading technique for low-resource edge devices. The problem formulated is a multi-objective optimization incorporating latency, task execution cost, and energy usage. The offloading of all tasks to different mobile edge servers represents the particle's fitness function. Simulation experiments compare the PSO offloading strategy with simulated annealing and genetic algorithms. Based on the experimental results, PSO-based task offloading reduces edge server latency, balances energy consumption, and achieves reasonable resource allocation.

Chen, et al. [21] combined two contradictory offloading objectives, namely increasing the job completion rate with an acceptable delay and decreasing the energy consumption of devices. The task offloading issue was established to achieve equilibrium between two challenging aims. Subsequently, they clarified it as a problem of dynamic task offloading based on Markov Decision Processes (MDP). A Deep Reinforcement Learning (DRL)-based dynamic task offloading (DDTO) method was developed to address this issue. The DDTO algorithm adapts to the constantly changing and intricate context and appropriately modifies the way tasks are offloaded. Experiments demonstrate DDTO's rapid convergence. The trial findings confirm the efficiency of the DDTO algorithm in achieving a balance between the finish ratio and power.

Kong, et al. [22] introduced a dependable and effective approach for task offloading using the multi-feedback trust strategy. A reliable and effective framework is built, which may significantly enhance trust computing and job offloading. Furthermore, the broker utilizes dynamic data monitoring to offer a multi-feedback trust management design using interaction frequency and time attenuation. This model aims to establish a trustworthy operating setting. Moreover, a trust-weighted K-means clustering approach is developed using resource qualities to improve service dependability. This algorithm efficiently and correctly groups the resource nodes

needed for the job. A job offloading model uses trust clustering to improve user experience and enhance system performance. In contrast to current task processing models prioritizing task offloading, the suggested approach further incorporates resource pretreatment, trust assessment, and resource clustering before task processing.

Aghapour, et al. [23] presented a solution using deep reinforcement learning to break down offloading and resource allocation into two elementary issues. The Salp Swarm Algorithm (SSA) optimizes resource allocation by updating offloading policy according to environmental data. The proposed method investigates various deep-learning tasks of IoT devices under different cloudlet server capacities. Simulation findings demonstrate that the suggested approach has the lowest latency and power consumption cost. On average, 92%, 17%, and 12% improvements were shown compared to the full local, full offload, joint resource allocation, and computation offloading techniques, respectively.

Bolourian and Shah-Mansouri [24] developed a wireless-powered mobile edge computing system divided into three tiers: IoT devices, edge servers, and cloud. A formulation of a combinatorial optimization problem is presented to minimize wireless energy transmission. Bipartite graph matching is employed to address the issue's complexity, and a harvest-then-offload technique is suggested for IoT devices. The proposed approach utilizes parallel processing to enhance its performance. Empirical tests demonstrate that the recommended technique substantially decreases the energy demand for operating IoT devices compared to other offloading strategies.

Nandi, et al. [25] developed a task offloading strategy to achieve a compromise between device usefulness and execution cost. Utility is determined by job execution delay and energy usage in energy-harvesting IoT devices. The task offloading issue is a problem of selecting a subset that achieves the required trade-off. Social Cognitive Optimization (SCO) addresses the offloading issue and achieves the required polynomial time of execution. The findings confirm the superior effectiveness of the approach regarding job execution speed, energy use, cost efficiency, and task abandonment rate when compared to the most advanced existing methods.

Due to its unique features, the proposed ACO-RSA provides a result-oriented solution to address the issues of energy consumption and latency in task offloading. While ACO has excellent exploration performance thanks to pheromone-assisted learning, it also has premature convergence. As RSA learns adaptively, it can utilize space better and more efficiently; however, exploration capacity may be limited, resulting in suboptimal outcomes. As a result of combining the advantages of both ACO exploration and RSA exploitation, the ACO-RSA hybrid method eliminates these disadvantages. It improves task-shifting efficiency, as illustrated by evaluation results. Using a more balanced and robust approach, we achieve significant reductions in energy consumption and latency, in contrast to existing research that proposes single-objective optimizations or scalability issues. The ACO-RSA can also achieve convergence more quickly and accurately than the current research literature.

III. System Design

A. System Model

Consider a defined geographic region containing a population of N mobile user devices, each denoted by n in the range 1 to N. Each device is assigned a unique computing task with different priorities, such as energy efficiency or latency minimization. This region has a base station equipped with several Mobile Edge Computing (MEC) servers. Mobile devices can communicate with MEC servers via a radio access network to reduce their computing load. The base station is configured with M-channels, each supporting a single connection between the device and the MEC server.

Mobile devices are assigned static channels, and edge servers have limited computing resources. The model also assumes a central cloud server with unlimited capacity. A single-edge server can handle the computing needs of only one device at a time. Tasks that exceed the capacity of an edge server are shifted to the cloud via the backbone network. The mobile edge computing architecture is shown in Fig. 3.

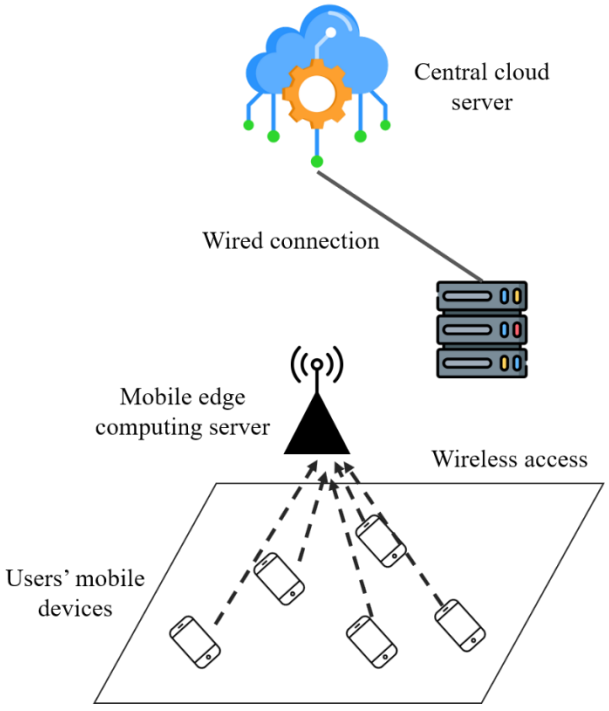

Fig. 3. Network model.

Mobile device offloading involves three steps. First, the device determines whether the task should be executed locally or offloaded to an edge server. During offloading, the system assesses the availability of the edge server resources. Insufficient resources require a decision between queuing at the current edge server or offloading to the cloud.

Orthogonal uplink channels ensure non-interference between devices in the proposed system. According to Eq. 1, the uplink rate R_n is calculated by denoting the transmission power of device n as P_n. Here, S_n refers to the device n's uplink bandwidth, z_n is the channel gain coefficient, and δ_n^2 represents the noise power.

$$R_n = S_n log_2\left(1 + \frac{P_n z_n}{\delta_n^2}\right), \quad \forall n \in N \tag{1}$$

B. Multi-User Computing Task Load Model

This subsection presents a computational task model to simplify the analysis. A computing task is characterized by three main components: (1) volume of data, including program code and input parameters, that needs to be uploaded for outsourced tasks; (2) computing capacity, typically measured in CPU cycles; and (3) results in data to be downloaded for outsourced tasks. Eq. 2 formally defines a computing task Q_v as a function of the result data R_v, the computing capacity C_v, and the data volume D_v.

$$Q_v \triangleq (D_v, C_v, R_v) \tag{2}$$

Using a program called graph analysis, mobile devices can discover these components. A multi-user delay and energy consumption load model is built to meet personalized user needs by analyzing different application types and corresponding data. This model evaluates the impact of local and cloud task execution on performance.

Initially, each mobile device determines whether to execute its task locally or offload it. For locally executed tasks, the computational requirements of device n are denoted by J_{n1} CPU cycles, and the task itself requires j_{n1} CPU cycles. The local execution time, t_{n1}, is calculated according to Eq. 3. Energy consumption, E_{n1}, for local execution is determined by Eq. 4, where λ is a constant energy consumption coefficient related to the device's chip structure, typically set to 10^{-26}. The total local overhead, W_{n1}, is calculated using Eq. 5, incorporating trade-off coefficients α_{n1} and α_{n2} for energy consumption and task execution time, respectively. These coefficients must adhere to the constraints outlined in Eq. 6.

$$t_{n1} = \frac{j_{n1}}{J_{n1}} \tag{3}$$

$$E_{n1} = \lambda j_{n1} (J_{n1})^2 \tag{4}$$

$$W_{n1} = a_{n1} E_{n1} + a_{n2} t_{n1} \tag{5}$$

$$\begin{cases} a_{n1}, a_{n2} \in [0,1] \\ a_{n1} + a_{n2} = 1 \end{cases} \tag{6}$$

When the n^{th} user's task is offloaded to an edge server, the processing delay is determined by Eq. 7. This delay comprises two components: the uplink transmission delay for task input data and the edge server execution time. The energy consumption for transferring the task to the edge server is computed using Eq. 10, where β represents the device's power amplifier efficiency. Eq. 11 formulates the total offloading overhead.

$$t_{n2}(P_n, J_{n2}) = t_{n2(1)}(P_n) + t_{n2(2)}(J_{n2}) \tag{7}$$

$$t_{n2(1)}(P_n) = \frac{s_n}{W_n log_2(1 + \sigma_n P_n)} \tag{8}$$

$$t_{n2(2)}(J_{n2}) = \frac{j_n}{J_{n2}} \tag{9}$$

$$E_{n2} = \frac{P_n}{\beta} t_{n2(1)}(P_n) = \frac{P_n}{\beta} \frac{s_n}{W_n log_2(1 + \sigma_n P_n)} \tag{10}$$

$$W_{n2} = a_{n1} E_{n2} + a_{n2} t_{n2}(P_n, J_{n2}) \tag{11}$$

C. Optimization Objective Function

The total overhead incurred by mobile device n during task offloading is calculated according to Eq. 12. The objective function is formulated to minimize the aggregate overhead across all devices. To achieve this minimum overhead, the optimal offloading strategy (X), uplink power allocation (P), and edge computing resource allocation (M) are determined. Eq. 13 expresses the optimization goal.

$$W_n = (1 - x_n) W_{n1} + x_n W_{n2} \tag{12}$$

$$\min_{X,P,M} W = \sum_{n=1}^{N} (1 - x_n) W_{n1} + x_n W_{n2} \tag{13}$$

Eq. 14 details the constraints. The binary variable x_n indicates the offloading decision for device n (1 for offloading, 0 for local execution). P_{max} represents the maximum transmission power, J_{max} is the maximum computational resource, and N_2 is the set of devices opting for offloading. The first constraint specifies which devices should be offloaded. The maximum power of the uplink device is limited by constraint 2. With constraint 3, edge computing resources are not allocated beyond the server's capacity. Negative resource allocation is prevented by constraint 4. As a final constraint, uplink transmissions are concurrently limited to N.

$$\begin{cases} 1: x_n \in \{0,1\}, \forall n \in N \\ 2: 0 < P_n \leq P_{max}, \forall n \in N_2 \\ 3: \sum_{n \in N_2} J_2 \leq J_{max} \\ 4: J_{n2} > 0, \forall n \in N_2 \\ 5: \sum_{n \in N} x_n S_n \leq B \end{cases} \tag{14}$$

IV. The ACO-RSA Algorithm

A. Ant Colony Optimization

The ACO algorithm is a metaheuristic inspired by the foraging behavior of real ants. Artificial ants iteratively construct solutions to optimization problems by laying down and following virtual pheromone trails. These paths represent solution components whose intensity correlates with the solution quality. Ants will likely select subsequent solution components based on pheromone levels and problem-specific heuristic information. Pheromone levels are dynamically updated to amplify high-quality solutions. ACO is a probabilistic search method that does not guarantee optimal solutions but often provides acceptable approximations within reasonable computing time.

Mathematically, ACO can be described in the following manner. A population of artificial ants is created. Pheromone levels on all potential solution components are initialized to a small positive value. Ants construct solutions by iteratively

selecting components. The probability of selecting component *j* after component *i* is determined by Eq. 15:

$$P_{ij} = \frac{\tau_{ij}^{\alpha}.\eta_{ij}^{\beta}}{\sum(\tau_{ik}^{\alpha}.\eta_{ik}^{\beta})} \quad (15)$$

Where α and β are parameters balancing pheromone and heuristic influence, η_{ij} is the heuristic information of components *i* and *j*, and τ_{ij} is the pheromone level of two components. After selecting a component, an ant updates its pheromone level according to Eq. 16.

$$\tau_{ij} = (1-\rho)\tau_{ij} + \rho\Delta\tau_{ij} \quad (16)$$

Where ρ is the pheromone evaporation rate and $\Delta\tau_{ij}$ is the pheromone the ant deposits. Once all ants have created solutions, pheromone levels are updated globally based on the overall quality of the solution. Combining probabilistic selection, pheromone amplification, and evaporation, this iterative process allows ACO to explore the solution space effectively.

B. Reptile Search Algorithm

RSA is a swarm intelligence metaheuristic derived from crocodile hunting strategies. By modeling competitive and cooperative interactions within a reptile population, RSA determines the global optima for optimization problems. Known for its simplicity, flexibility, and efficiency, RSA has found applications in image processing, power systems, and engineering.

RSA's optimization methodology, outlined in Eq. 17, involves iteratively refining a population of solutions (*X*). Eq. 18 details the random generation of candidate solutions, where $x_{i,k}$ represents the kth position of the i^{th} individual, *N* refers to the population size, *D* specifies the problem dimension, and *rd* indicates a random value within the problem's lower (*LB*) and upper (*UB*) bounds.

$$X = \begin{bmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,D} \\ x_{2,1} & x_{2,2} & \dots & x_{2,D} \\ \vdots & \cdots & \cdots & \vdots \\ x_{N,1} & x_{N,2} & \dots & x_{N,D} \end{bmatrix} \quad (17)$$

$$x_{i,k} = rd \times (UB - LB) + LB, k = 1,2,\dots,n \quad (18)$$

RSA is divided into two stages: exploration (encircling) and exploitation (hunting). During exploration, crocodiles exhibit two distinct movement patterns: high walking and belly walking. Unlike focused hunting, exploratory behaviors allow for a wider range of search areas. RSA emulates these mechanisms during its exploration phase, as defined in Eq. 19.

$$x_{i,k}(t+1) = \begin{cases} Best_k(t) - \eta_{i,k}(t) \times \beta - R_{i,k}(t) \times rd, & t \le \\ Best_k(t) - x_{r1,k}(t) \times ES(t) \times rd, & t \le 2\frac{T_m}{4} \ a \end{cases} \quad (19)$$

$Best_k$ represents the best k^{th} position of the optimal individual, *T* is the current iteration, and T_m is the maximum iteration. Parameter β controls the exploration extent. Random numbers *rd* and *r1* are employed for stochasticity. The hunting operator, $\eta_{i,k}$, calculated in Eq. 20, influences the exploration process.

$$\eta_{i,k} = Best_k(t) \times P_{i,k}(t) \quad (20)$$

To refine the search, a reduced function, Ri,k, is introduced in Eq. 21, where ε is a small constant and *r2* is another random number. This function minimizes the search space. The evolutionary phase, *ES(t)*, calculated in Eq. 22, is a probability ratio that fluctuates between -2 and 2 over iterations, guided by the random number *r3*. $P_{i,k}$, the percentage difference between the $Best k^{th}$ value of the optimal and current solutions, as computed in Eq. 23. The average solution, calculated in Eq. 24, contributes to the overall exploration strategy.

$$R_{i,k} = \frac{Best_k(t) - x_{r2,k}}{Best_k(t) + \varepsilon} \quad (21)$$

$$ES(t) = 2 \times r_3 \times \left(1 - \frac{1}{T_m}\right) \quad (22)$$

$$P_{i,k} = \alpha + \frac{x_{i,k-M_{x_i}}}{Best_k \times (UB_k - LB_k) + \epsilon} \quad (23)$$

$$M_{x_i} = \frac{1}{D}\sum_{k=1}^{D} x_{i,k} \quad (24)$$

Cooperation and coordination are the two main foraging behaviors of crocodiles. These strategies represent different approaches to intensified exploitation. Eq. 25 determines the specific behavior. Hunting coordination is performed when the current iteration (*t*) falls within the range of $2T_m/4$ to $3T_m/4$, where T_m is the maximum iteration. Conversely, hunting cooperation occurs when *t* is between 0 and $T_m/4$ or $3T_m/4$ and T_m.

$$x_{i,k} = \begin{cases} Best_k(t) \times P_{i,k}(t) \times rd, & t \le 3\frac{T_m}{4} \\ Best_k(t) - \eta_{i,k}(t) \times \epsilon - R_{i,k} \times rd, & t \le T_m \ a \end{cases} \quad (25)$$

C. Integration of ACO and RSA

Because ACO relies on pheromone trails, exploration of the solution space is hampered by premature convergence. While strong exploration prevents local optima, excessive exploitation impairs solution quality. RSA effectively balances exploration and exploitation and demonstrates superior performance in various technical areas. We propose a hybrid ACO-RSA strategy based on a high-level relay hybrid (HRH) approach to further improve this balance. ACO and RSA are applied sequentially, homogeneously, or heterogeneously. The proposed method uses a heterogeneous HRH strategy to combine the exploitation of ACO with the exploration of RSA.

Initially, ACO, RSA, and shared parameters are initialized. *N* candidate solutions, each represented by an M-dimensional feature vector, are randomly generated within the range of -1 to 1. These solutions are evaluated using a fitness function to assess their quality relative to previous iterations. Superior solutions are retained, while inferior ones are discarded.

Afterward, the optimal solution is identified and assigned to the initial ACO population. The candidate solutions are then designated as initial ant paths. The ants update candidate solutions based on a subset of features initialized with pheromone values exceeding 0.5. Improvements are based on fitness evaluations, with updates only applied to the fittest solutions (Eq. 26).

$$x_i(g+1) = \begin{cases} x_i^{new}(g), & if\ FF(x_i(g)) > FF(x_i(g+1)) \\ x_i(g), & else \end{cases} \quad (26)$$

ACO or RSA uses the refined candidate solutions as input to explore new promising regions in successive iterations. An algorithm switch occurs when ACO does not improve the solutions, indicating a trap in local optima. RSA is then used to diversify the search space. The iterative process continues until the termination criterion (maximum iterations) is reached.

Initialization is an initial process that randomly generates candidate solutions (using 50 population sizes) to ensure a diverse solution space. We set pheromone levels for ACO to be initialized with small positive values, and we made the pheromone evaporation rate 0.1 (to balance exploration and exploitation). Inspired by the reptile-hunting dielectric technique, the RSA reacts to evolve optimal solutions and seeks convergence of its best solutions. The algorithm stops when 100 iterations are achieved or there hasn't been a significant improvement in loss over ten consecutive iterations. Incorporating ACO exploration with RSA adaptive learning compensates for both suboptimal aspects of the algorithms, avoids premature convergence typical of ACO, and optimizes the exploitation phase of RSA. During task scheduling, each task's priority is assigned based on its scale and complexity (energy-sensitive or latency-sensitive) through the optimization process, as shown in Fig. 2. The system's energy consumption and latency are reduced by balancing the distribution of tasks across available resources, including edge servers or the cloud.

V. Experimental Setup and Results

To validate the effectiveness of the proposed ACO-RSA algorithm, we conducted an extensive series of simulations. The simulation environment is designed to mimic real-world IoT scenarios with varying numbers of mobile user devices and data transfer rates. The environment includes multiple edge servers with different computational capabilities, simulating a typical edge computing context. The primary performance metrics evaluated in the simulations are average time delay and energy consumption. Time delay is the total time required for task completion, while energy consumption is calculated based on the power needed for data transmission and processing.

The simulation parameters were configured according to the 3GPP standard. The simulation considers an area with a 1 km radius, where users transmit data at a maximum power of 25 dBm over a system bandwidth of 15 MHz and user bandwidth of 0.5 MHz. The noise power in the uplink bandwidth is set at −108 dBm. Tasks have input data volumes ranging from 300 to 1500 KB and require CPU cycles between 0.1 and 0.8 GHz. The users' computing power varies from 0.2 to 1.2 GHz, while the MEC server has a computing power of 3 GHz. The simulation also involves a population size of 50 and a maximum of 100 evolutions, with a maximum allowable time delay of 3 to 5 seconds.

To evaluate the convergence of the proposed computation offloading algorithm, a simulation was conducted with 80 users, 15 servers, and 5MB tasks. Fig. 4 illustrates the time delay of the system over different iteration numbers. Fast convergence was observed after 20 iterations, with minimal delay improvements after that, indicating the achievement of a global optimum. This demonstrates the algorithm's strong global optimization and search capabilities and reduces the overall system delay from 0.268 s to 0.194 s, representing a significant performance improvement.

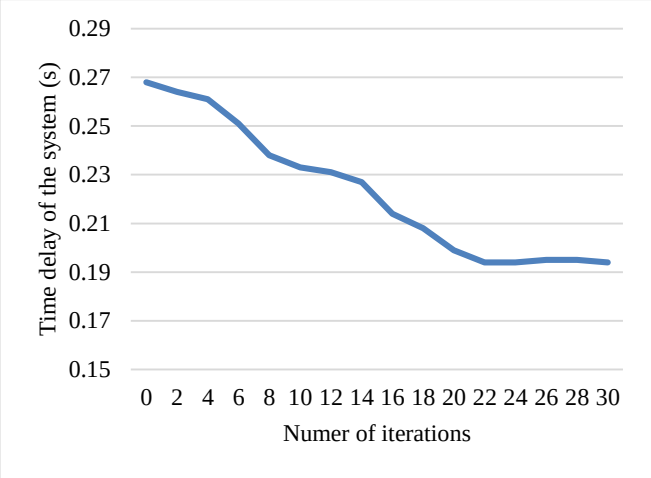

Fig. 4. Stability diagram.

Comparative analyses used algorithms from [26, 27] to assess delay and energy consumption. Fig. 5 shows the average delay for different numbers of users. Energy consumption was evaluated at various transmission rates (Fig. 6). Due to the lack of data transfer, local execution was independent of the transfer rate. All algorithms showed lower energy consumption with increasing transfer rates due to lower offloading overhead. The proposed method showed the largest decrease and lowest overall energy consumption, enabled by more frequent and fine-grained discharge decisions.

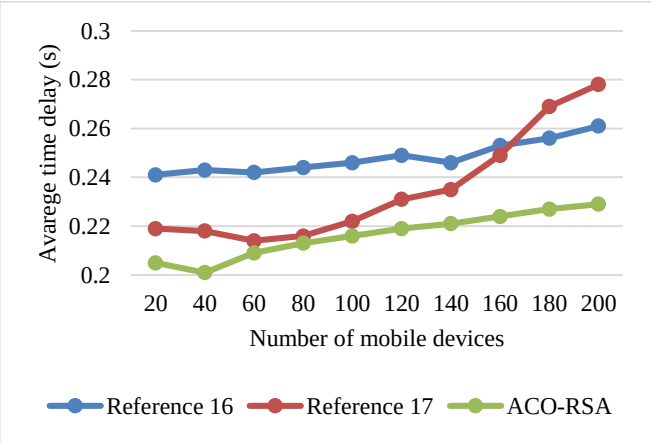

Fig. 5. Delay vs. Number of mobile devices.

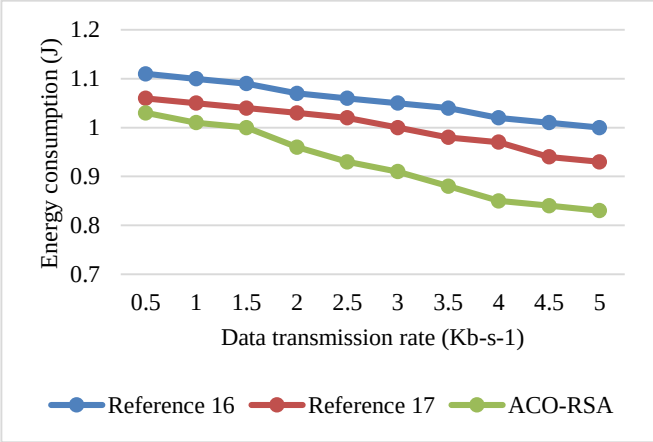

Fig. 6. Energy consumption vs. data transmission rate.

Fig. 7 illustrates the relationship between energy consumption and number of users. The proposed method consistently outperformed others with the lowest energy consumption and growth rate, achieving 0.2 J for 80 users. The local execution caused the highest energy consumption. The proposed strategy's efficiency in task offloading and reduced local execution contributed to overall energy savings.

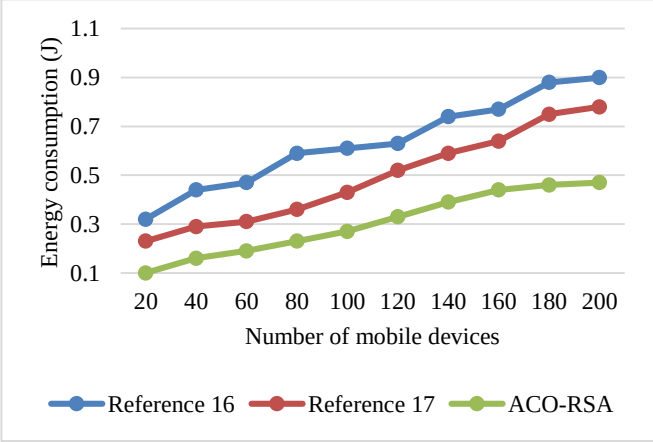

Fig. 7. Energy consumption vs. number of mobile devices.

The results confirm that the ACO-RSA algorithm can effectively solve major task offloading problems in IoT edge computing, such as energy consumption and latency. By combining ACO's exploration and RSA's adaptive learning, the proposed approach distributes tasks efficiently and minimizes system overhead. The proposed ACO-RSA provides an average of 27.6% energy savings and 25.4% latency reduction compared to other existing techniques, resulting in better resource scheduling and task completion.

These results confirm the algorithm's ability to optimize task shifting while balancing energy and performance metrics. Nevertheless, the study emulates the simulation environment of static network states and operational case functions within devices, which may not correspond to the realistic conditions in ideas in IoT environments. Fluctuating network traffic or device roaming can affect the algorithm's performance. Finally, future work should focus on extending ACO-RSA to dynamic edge computing scenarios and incorporating other considerations, such as security and privacy, to improve the robustness and applicability of ACO-RSA in various IoT contexts.

VI. Conclusion

To the best of our knowledge, this is the first study to introduce a new metaheuristic algorithm by combining ACO and RSA into a hybrid version named ACO-RSA, which can overcome the challenges of efficient task offloading in IoT environments powered by edge computing. This feature, therefore, allows the ACO-RSA classifier to make a remarkable compromise between energy consumption and latency, two main components of IoT ecosystems, by utilizing both ACO and RSA within an integrated classification framework. By using ACO, the algorithm takes inspiration from how ants collect food to search for optimal paths; meanwhile, with RSA, it can adaptively explore the IoT environments that are highly dynamic and unpredictable, proposing a strong solution.

ACO-RSA is designed based on an optimization objective function aiming to minimize energy consumption while at the same time reducing task offloading latency. Thus, it is well suited for resource-limited IoT devices that run in edge computing environments. We conducted a series of extensive simulations to verify the feasibility of the proposed algorithm. Results confirmed that the proposed ACO-RSA operates better than the traditional benchmark algorithms. The hybrid algorithm minimizes energy consumption and operates with low latency as the number of mobile users and data rates increase. This result illustrates the promise of ACO-RSA in enhancing resource utilization and prolonging the lifecycle of IoT applications by tuning task offloading policies.

Future work will address the scalability and robustness of ACO-RSA, especially within larger and more complex IoT networks with an assorted range of applications and diverse task requirements. Moreover, investigation into the integration between machine learning and ACO-RSA could enhance adaptivity capabilities even more and aid in better addressing dynamic changes in network conditions and resource availability. In conclusion, the proposed ACO-RSA algorithm is an important step towards an efficient and energy-greedy task offloading strategy in edge computing-enabled IoT environments.

References

[1] A. A. Anvigh, Y. Khavan, and B. Pourghebleh, "Transforming Vehicular Networks: How 6G can Revolutionize Intelligent Transportation?," Science, Engineering and Technology, vol. 4, no. 1, pp. 80-93, 2024.

[2] J. Valizadeh et al., "An operational planning for emergency medical services considering the application of IoT," Operations Management Research, vol. 17, no. 1, pp. 267-290, 2024.

[3] B. Pourghebleh and V. Hayyolalam, "A comprehensive and systematic review of the load balancing mechanisms in the Internet of Things," Cluster Computing, pp. 1-21, 2019.

[4] B. Pourghebleh and N. J. Navimipour, "Data aggregation mechanisms in the Internet of things: A systematic review of the literature and recommendations for future research," Journal of Network and Computer Applications, vol. 97, pp. 23-34, 2017.

[5] N. M. Quy, L. A. Ngoc, N. T. Ban, N. V. Hau, and V. K. Quy, "Edge computing for real-time Internet of Things applications: Future internet revolution," Wireless Personal Communications, vol. 132, no. 2, pp. 1423-1452, 2023.

[6] Q. Wu, S. Wang, H. Ge, P. Fan, Q. Fan, and K. B. Letaief, "Delay-sensitive task offloading in vehicular fog computing-assisted platoons," IEEE Transactions on Network and Service Management, 2023.

[7] S. E. Comert and H. R. Yazgan, "A new approach based on hybrid ant colony optimization-artificial bee colony algorithm for multi-objective electric vehicle routing problems," Engineering Applications of Artificial Intelligence, vol. 123, p. 106375, 2023.

[8] L. Abualigah, M. Abd Elaziz, P. Sumari, Z. W. Geem, and A. H. Gandomi, "Reptile Search Algorithm (RSA): A nature-inspired meta-heuristic optimizer," Expert Systems with Applications, vol. 191, p. 116158, 2022.

[9] R. Chataut, A. Phoummalayvane, and R. Akl, "Unleashing the power of IoT: A comprehensive review of IoT applications and future prospects in healthcare, agriculture, smart homes, smart cities, and industry 4.0," Sensors, vol. 23, no. 16, p. 7194, 2023.

[10] V. Hayyolalam, B. Pourghebleh, M. R. Chehrehzad, and A. A. Pourhaji Kazem, "Single - objective service composition methods in cloud manufacturing systems: Recent techniques, classification, and future trends," Concurrency and Computation: Practice and Experience, vol. 34, no. 5, p. e6698, 2022.

[11] B. Pourghebleh, N. Hekmati, Z. Davoudnia, and M. Sadeghi, "A roadmap towards energy - efficient data fusion methods in the Internet of Things," Concurrency and Computation: Practice and Experience, vol. 34, no. 15, p. e6959, 2022.

[12] F. Kamalov, B. Pourghebleh, M. Gheisari, Y. Liu, and S. Moussa, "Internet of medical things privacy and security: Challenges, solutions, and future trends from a new perspective," Sustainability, vol. 15, no. 4, p. 3317, 2023.

[13] N. A. Angel, D. Ravindran, P. D. R. Vincent, K. Srinivasan, and Y.-C. Hu, "Recent advances in evolving computing paradigms: Cloud, edge, and fog technologies," Sensors, vol. 22, no. 1, p. 196, 2021.

[14] G. Baranwal, D. Kumar, and D. P. Vidyarthi, "Blockchain based resource allocation in cloud and distributed edge computing: A survey," Computer Communications, 2023.

[15] B. Pourghebleh, K. Wakil, and N. J. Navimipour, "A comprehensive study on the trust management techniques in the Internet of Things," IEEE Internet of Things Journal, vol. 6, no. 6, pp. 9326-9337, 2019.

[16] J. Ge, B. Liu, T. Wang, Q. Yang, A. Liu, and A. Li, "Q - learning based flexible task scheduling in a global view for the Internet of Things," Transactions on Emerging Telecommunications Technologies, p. e4111, 2020.

[17] P. V. B. C. d. Silva, C. Taconet, S. Chabridon, D. Conan, E. Cavalcante, and T. Batista, "Energy awareness and energy efficiency in internet of things middleware: a systematic literature review," Annals of Telecommunications, vol. 78, no. 1, pp. 115-131, 2023.

[18] I. Vlachos, R. M. Pascazzi, M. Ntotis, K. Spanaki, S. Despoudi, and P. Repoussis, "Smart and flexible manufacturing systems using Autonomous Guided Vehicles (AGVs) and the Internet of Things (IoT)," International Journal of Production Research, vol. 62, no. 15, pp. 5574-5595, 2024.

[19] K. Xiao, Z. Gao, W. Shi, X. Qiu, Y. Yang, and L. Rui, "EdgeABC: An architecture for task offloading and resource allocation in the Internet of Things," Future generation computer systems, vol. 107, pp. 498-508, 2020.

[20] 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, pp. 1-11, 2021.

[21] Y. Chen, W. Gu, and K. Li, "Dynamic task offloading for internet of things in mobile edge computing via deep reinforcement learning," International Journal of Communication Systems, p. e5154, 2022.

[22] W. Kong, X. Li, L. Hou, J. Yuan, Y. Gao, and S. Yu, "A reliable and efficient task offloading strategy based on multifeedback trust mechanism for IoT edge computing," IEEE Internet of Things Journal, vol. 9, no. 15, pp. 13927-13941, 2022.

[23] Z. Aghapour, S. Sharifian, and H. Taheri, "Task offloading and resource allocation algorithm based on deep reinforcement learning for distributed AI execution tasks in IoT edge computing environments," Computer Networks, vol. 223, p. 109577, 2023.

[24] M. Bolourian and H. Shah-Mansouri, "Energy-efficient task offloading for three-tier wireless-powered mobile-edge computing," IEEE Internet of Things Journal, vol. 10, no. 12, pp. 10400-10412, 2023.

[25] P. K. Nandi, M. R. I. Reaj, S. Sarker, M. A. Razzaque, M. Mamun-or-Rashid, and P. Roy, "Task offloading to edge cloud balancing utility and cost for energy harvesting internet of things," Journal of Network and Computer Applications, vol. 221, p. 103766, 2024.

[26] M. Zhao and K. Zhou, "Selective offloading by exploiting ARIMA-BP for energy optimization in mobile edge computing networks," Algorithms, vol. 12, no. 2, p. 48, 2019.

[27] Y. Shi, Y. Xia, and Y. Gao, "Cross-server computation offloading for multi-task mobile edge computing," Information, vol. 11, no. 2, p. 96, 2020.