Sensors and Materials, Vol. 38, No. 4 (2026) 2287–2299
MYU Tokyo

S & M 4437

Edge Computing Resource-balanced Scheduling: Solving Efficiency and Load Issues via Hybrid Genetic–Ant Colony Optimization Algorithm

Liu Chunxiao,[1*] Zhang Yan,[1] Wang Yanfeng,[2] and Li Long[3]

[1]Changzhou College of Information Technology,
No. 22, Mingxin Middle Road, Science and Education Town, Changzhou City 213164, China
[2]Suqian University, No. 399, Huanghe South Road, Suqian City 223800, China
[3]Jiujiang Polytechnic University of Science and Technology,
No. 8, Huiyuan Road, Export Processing Zone, Jiujiang, Jiangxi 332000, China

(Received February 10, 2026; accepted April 21, 2026)

Keywords: edge computing, resource scheduling, ant colony algorithm, genetic algorithm, efficiency and load issues

To resolve the contradictions in edge computing for technology services, such as a long resource scheduling time, an uneven load distribution, and the conflict between the limited resources of edge nodes and the requirements of low latency and high reliability for tasks, in this paper, we propose an edge computing resource-balanced scheduling algorithm for technology services that integrate a genetic–ant colony hybrid algorithm. First, an edge computing scenario is constructed on the basis of cloud distance, and an edge computing network architecture is established. The constraints on the cloud distance of nodes and data transmission rates are clarified, while a task model incorporating task priority classification and an edge computing node model are developed. Second, a multi-objective resource-balanced scheduling model is built by integrating task parameters and scheduling time. A hybrid strategy combining the genetic and ant colony algorithms is adopted to solve the model: the global search capability of the genetic algorithm is used to quickly locate the high-quality solution space, and then the local optimization advantage of the ant colony algorithm is employed to accurately optimize the scheduling scheme, achieving the dual goals of reducing the task execution time and realizing a balanced load distribution across the cluster. Finally, the performance of the algorithm is verified through simulation experiments. The results show that the proposed algorithm can effectively solve the problems of long resource scheduling time and uneven load distribution in edge computing for technology services, significantly reduce system energy consumption, improve system resource utilization, and fully meet the core requirements of low latency and high reliability for edge computing tasks. The algorithm proposed in this paper can directly provide low-latency and highly reliable computing offloading support for Internet of Things sensor terminals, optimize the real-time processing and transmission efficiency of sensor data, and enhance the deployment and application capabilities of sensing systems in technical service scenarios.

*Corresponding author: e-mail: xiaoxiao198525@163.com
https://doi.org/10.18494/SAM6286

ISSN 0914-4935

1. Introduction

The rapid popularization of Internet of Things and sensor technology has generated massive amounts of sensing data in technical service scenarios such as smart homes, industrial IoT, smart cities, and telemedicine, creating an urgent demand for low-latency, highly reliable real-time processing.[1–3] With the rapid advancement of the IoT, 5th-Generation Mobile Networks communication, and AI technologies, the technology service industry is undergoing a profound digital transformation. Emerging application scenarios, including smart homes, industrial IoT, smart cities, and telemedicine, generate massive volumes of data.[4–6] These scenarios also impose stringent requirements on data processing, such as low latency, high bandwidth, robust security, and privacy preservation. The traditional centralized cloud computing paradigm, which transmits all data to remote cloud data centers for processing, is increasingly inadequate in meeting these demands—especially regarding network bandwidth constraints and data transmission latency.[7–9] As an emerging computing paradigm, edge computing effectively addresses the limitations of cloud computing by decentralizing computing, storage, and network resources to the network edge. This deployment places resources closer to data sources and end-users, enabling local or near-edge data processing. This approach significantly reduces network transmission latency, alleviates bandwidth pressure on the core network, and enhances data privacy and security. Consequently, edge computing is recognized as a critical infrastructure to underpin the future development of the technology service industry.

Nevertheless, while edge computing offers substantial advantages, its inherent characteristics introduce new challenges. Unlike cloud data centers with nearly unlimited resources, edge computing nodes (e.g., edge servers, gateways, and intelligent devices) typically suffer from resource constraints, distributed heterogeneity, and dynamic changes in network environments. In the application scenarios of the technology service industry, tasks often exhibit varying priorities and Quality of Service requirements. How to efficiently and rationally schedule the influx of massive heterogeneous tasks to appropriate edge nodes for execution in such a complex and constrained environment has become an urgent core issue—namely, the edge computing resource scheduling problem.

As the data source, sensor terminals generate tasks characterized by strong real-time performance, high concurrency, and small data volume but high frequency, which poses higher requirements for resource allocation and the load balancing of edge nodes. Aiming at technical service scenarios with a large number of sensor terminals, we studied the balanced scheduling method of edge computing resources, aiming to improve the efficiency of sensor data processing and enhance the support capability of edge computing for the IoT perception layer.

The structure of this study is outlined as follows: A comprehensive review and critical analysis of the current state of relevant research both domestically and internationally is provided in Sect. 2. In Sect. 3, we elaborate in detail on the design of the hybrid algorithm proposed in this study, encompassing the system model, problem formulation, and algorithm workflow. In Sect. 4, we validate and perform a comparative analysis of the performance of the proposed algorithm through simulation experiments. In Sect. 5, we summarize the entire study and provide prospects for future research directions.

2. Current Status of Domestic and International Research

As a typical Non-deterministic Polynomial-hard problem, edge computing resource scheduling research mainly focuses on how to efficiently and evenly allocate limited resources to meet the low-latency and high-reliability requirements of tasks. Studies based on traditional heuristic algorithms,[10] such as First-Come-First-Served (FCFS) and Shortest Job First (SJF), feature simple computation and fast response. However, they generally ignore multi-objective optimization, which easily leads to unbalanced node loads and degraded system performance. For instance, the Shortest Completion Time First (SCTF) strategy can optimize average latency but risks overloading high-performance nodes. Studies based on meta-heuristic algorithms[11–13] provide approaches to obtain better solutions. Genetic Algorithms (GAs) are widely adopted owing to their strong global search capability, but they suffer from weak local search ability and a tendency for premature convergence. Ant Colony Optimization (ACO) performs excellently in path optimization through positive feedback mechanisms; however, in edge computing, its lack of initial pheromones results in slow convergence. Particle Swarm Optimization (PSO) and Simulated Annealing (SA) are also commonly applied, but they face issues such as being prone to local optima (for PSO) and high parameter sensitivity with slow convergence (for SA). Multi-objective optimization research[14] has attracted increasing attention as application scenarios become more complex. Researchers have begun to consider multiple objectives simultaneously, including execution time, energy consumption, cost, and load balancing. Linear weighting methods or the concept of Pareto optimality are usually adopted. Hybrid algorithm research[15] has become a research hotspot to overcome the limitations of single algorithms. Domestic and international scholars have attempted to integrate different algorithms, for example, using GAs to generate initial populations and then applying PSO for fine-grained search, or introducing SA mechanisms into ACO to enhance global exploration capabilities. In particular, hybrid strategies combining GA and ACO have shown potential in combinatorial optimization problems. Nevertheless, the systematic application of such hybrid strategies to edge computing scenarios in the technology service industry is still insufficient. Existing studies mostly focus on minimizing latency or energy consumption, pay insufficient attention to the collaborative optimization of "load balancing" (a key objective), and lack comprehensive verification in heterogeneous and dynamic edge environments. To summarize, although significant progress has been made in current edge computing resource scheduling research, the following major shortcomings remain: Most algorithms struggle to balance scheduling efficiency and solution quality. The load balancing objective is often treated as a secondary factor and fails to deeply collaborate with core performance indicators. Customized models and algorithms for the high heterogeneity and multi-priority characteristics of the technology service industry need further development. Moreover, existing studies rarely integrate scheduling algorithms with sensor terminals and IoT perception scenarios, and fail to fully consider the transmission and processing characteristics of sensor data. In this study, we aim to systematically address the above challenges by constructing a multi-objective balanced scheduling model that comprehensively considers task priority, node–cloud distance, and real-time load status, and designing a phased GA-ACO hybrid algorithm.

3. System Model and Problem Formulation

In this paper, an edge computing network architecture is constructed for technical service scenarios with a large number of sensor terminals. The terminal device layer consists of various IoT sensors, cameras, intelligent acquisition devices, and so forth, which are responsible for continuously generating perception data and computing tasks. We constructed a three-tier edge computing architecture, consisting of a cloud center, an edge layer, and a terminal device layer, as shown in Fig. 1.

- Terminal Device Layer: Composed of various IoT devices, it is responsible for generating computing tasks.
- Edge Layer: This consists of multiple geographically distributed edge nodes (ENs). Each EN is a small server or gateway with certain computing and storage capabilities, and nodes are interconnected via local area networks (LANs) or high-speed networks.
- Cloud Center: Equipped with nearly unlimited computing and storage resources, it handles non-real-time, computationally intensive tasks and performs global management.

To quantify network transmission performance, in this study, we introduce the concept of "cloud distance". It is not a pure geographical distance but a logical distance that integrates

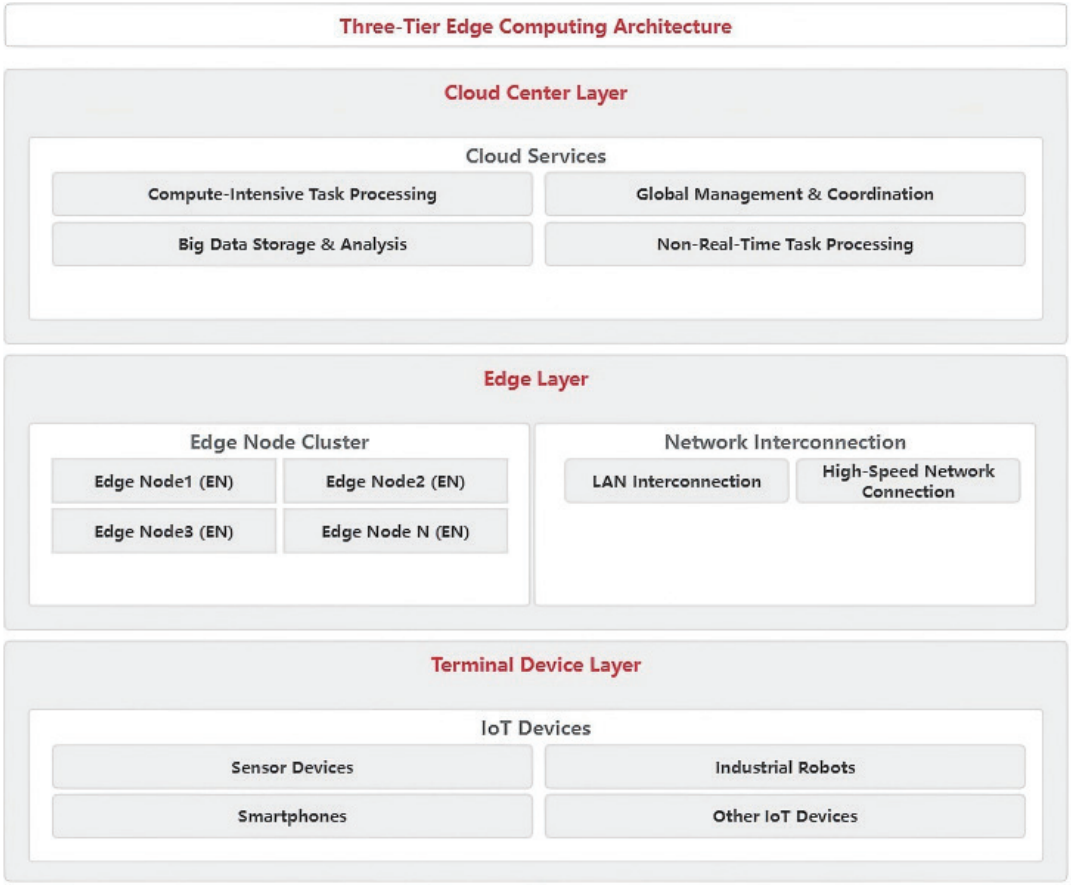


Fig. 1. (Color online) Three-tier edge computing architecture.

network hops, bandwidth, and latency. Tasks are usually scheduled in the edge layer first; offloading to the cloud center is only considered when edge layer resources cannot meet the requirements.

(1) Task Model

Assume there is a set of independent tasks $T = \{T_1, T_2, ..., T_n\}$ to be scheduled. Each task T_i can be described by a 4-tuple:$T_i = (D_i, C_i, P_i, DL_i)$, where

D_i: input data volume of the task (in MB);

C_i: total computing resources required to complete the task (in CPU cycles);

P_i: task priority, divided into three levels (high, medium, and low, which can be represented by 3, 2, 1, for example). High-priority tasks (e.g., emergency alerts) need to be scheduled and executed first;

DL_i: maximum tolerable delay of the task (in seconds).

(2) EN Model

Assume there is a set of ENs $EN = \{EN_1, EN_2, ..., EN_m\}$. Each edge node EN_j can be described by a 4-tuple: $EN_j = (F_j, M_j, B_j, L_j^{current})$, where

F_j: computing capability of the node (in CPU cycles per second);

M_j: available memory of the node (in MB);

B_j: data transmission rate between the node and the task request source (in MB per second), which is related to "cloud distance";

$L_j^{current}$: current load status of the node, which can be measured using the current CPU utilization or the length of the task queue being executed.

(3) Multi-objective Resource-balanced Scheduling Model

In this paper, resource scheduling refers to the process of rationally allocating computing tasks to ENs. The technology service industry refers to the service industry centered on data processing, intelligent perception, and real-time services. ENs refer to edge computing devices close to data sources. We define the edge computing resource-balanced scheduling problem as a combinatorial optimization problem of assigning tasks to optimal ENs, with objectives of task execution time and load balancing degree, under the constraints of task delay and node resources.

Our goal is to find an optimal task-to-node mapping $S: T \rightarrow EN$, while optimizing the following two objectives:

Objective 1: Minimize Total Task Execution Time

The execution time ET_{ij} of task T_i on node EN_j includes the transmission time $Trans_{ij}$ and the computing time $Comp_{ij}$:

$$Trans_{ij} = D_i/B_j, \tag{1}$$

$$Comp_{ij} = C_i/F_j, \tag{2}$$

$$ET_{ij} = Trans_{ij} + Comp_{ij}. \tag{3}$$

The total execution time objective function F_1 is defined as the weighted sum of the execution times of all tasks, with the weight being the task priority P_i. This ensures that high-priority tasks contribute more to the total time, thereby prompting their faster completion:

$$F_1 = \min \sum_{i=1}^{n} \sum_{j=1}^{m} x_{ij} \cdot P_i \cdot ET_{ij} \tag{4}$$

Objective 2: Achieve System Load Balancing

The variance of the load rates of all ENs is used to measure the system's load balancing degree. The load rate $Load_j$ of node EN_j is the ratio of its total computing demand to its total computing capability:

$$Load_j = \frac{\sum_{i=1}^{n} x_{ij} \cdot C_i}{F_j} + L_j^{current}. \tag{5}$$

The load balancing objective function F_2 is defined as

$$F_2 = \min \sqrt{\frac{1}{m} \sum_{j=1}^{m} (Load_j - \overline{Load})^2}, \tag{6}$$

where $\overline{Load}$ is the average load rate of all nodes.

Finally, this problem is modeled as a bi-objective optimization problem: Minimize(F_1, F_2). For ease of solution, we used the linear weighting method to convert it into a single-objective problem and defined the comprehensive objective function as

$$F = \alpha \cdot \frac{F_1}{F_1^{norm}} + \beta \cdot \frac{F_2}{F_2^{norm}}, \tag{7}$$

where α and β are weight coefficients $\alpha + \beta = 1$, and F_1^{norm} and F_2^{norm} are normalization factors used to eliminate the effect of dimensions.

4. Resource-balanced Scheduling Algorithm Based on Hybrid Genetic–Ant Colony Optimization Algorithm

The steps of the proposed resource-balanced scheduling algorithm based on the genetic–ant colony hybrid algorithm are as follows.

Step 1: Initialize all parameters required for the operation of the genetic and ant colony optimization algorithms.

Step 2: The genetic algorithm performs global search through selection, crossover, and mutation operations to generate a high-quality initial solution set.

Step 3: Convert the high-quality solution set obtained by the genetic algorithm into pheromones to initialize the pheromone matrix of the ant colony optimization algorithm.
Step 4: On the basis of the initialized pheromones, the ant colony optimization algorithm conducts local fine-grained optimization through ant path construction and pheromone positive feedback mechanisms.
Step 5: Output the globally optimal task scheduling scheme obtained after optimization by the ant colony optimization algorithm.
Step 6: The algorithm terminates and the scheduling is completed.

4.1 Global exploration of genetic algorithm

In the global exploration phase of the genetic algorithm, integer encoding is adopted, where the length of each chromosome equals the number of tasks, n, and a value j at gene position i indicates that task T_i is assigned to EN EN_j. The initial population is generated randomly, with each chromosome strictly satisfying constraints such as resource limitations. The fitness function is defined as $Fitness = 1/F$, where F represents the comprehensive objective function defined above; a smaller F corresponds to a higher fitness value. Selection is performed through roulette wheel selection, ensuring that individuals with higher fitness have a greater probability of being selected for the next generation. Crossover operation employs two-point crossover, where segments of two parent chromosomes are exchanged at a certain probability to produce new offspring. Mutation operation involves randomly altering the value of a gene in the chromosome (i.e., reassigning a task to a different node) with a low probability. The GA phase terminates after reaching a preset number of iterations and outputs a set of individuals with the highest fitness in the current generation (elite solution set).

The pseudocode of the algorithm is shown in Algorithm 1.

4.2 Pheromone matrix initialization

This is a key step in the hybrid algorithm. Unlike traditional ACO, which starts with a uniform pheromone matrix, this algorithm leverages the exploration results of GA: it converts the elite solution set obtained by GA into paths, and then initializes the pheromone matrix τ_{ij}

Algorithm 1

Input: Number of tasks n, node list nodes, population size popSize, maximum number of iterations maxIter, crossover probability crossP, mutation probability mutateP, number of elites eliteSize
Output: Elite solution set (the eliteSize allocation schemes with the highest fitness)

```
function GA(n, nodes, popSize, maxIter, crossP, mutateP, eliteSize):
    population = InitPopulation(n, nodes, popSize)
    for i = 1 to maxIter:
        fitness = CalcFitness(population)
        selected = RouletteSelection(population, fitness, popSize)
        offspring = TwoPointCrossover(selected, crossP)
        offspring = Mutate(offspring, nodes, mutateP)
        population = offspring
    return SelectElites(population, fitness, eliteSize)
```

based on the quality of these elite paths. Here, τ_{ij} represents the degree of expectation for assigning task T_i to EN EN_j.

4.3 Pheromone matrix initialization

4.3.1 Solution construction

Each ant constructs a scheduling scheme for a sequence of tasks. The probability that ant k selects node EN_j for task T_i is determined using the following formula:

$$p_{ij}^{k} = \frac{[\tau_{ij}]^{\alpha} \cdot [\eta_{ij}]^{\beta}}{\sum_{l \in allowed_k} [\tau_{il}]^{\alpha} \cdot [\eta_{il}]^{\beta}} \text{ if } j \in allowed_k, \tag{8}$$

where τ_{ij} is the pheromone concentration on edge (i, j); η_{ij} is the heuristic information, defined here as $1/(P_i \cdot ET_{ij})$ (indicating priority for nodes that can process high-priority tasks faster); α and β are the parameters controlling the relative importance of pheromone and heuristic information; and $allowed_k$ is the set of nodes that ant k can currently select and that satisfy resource constraints.

4.3.2 Pheromone Update

Local Update:

After an ant assigns a task, local pheromone evaporation is performed on the corresponding edge to prevent all ants from converging to the same path too quickly:

$$\tau_{ij} = (1 - \rho) \cdot \tau_{ij}. \tag{9}$$

4.3.3 Global Update

After all ants have constructed complete paths, global pheromone enhancement is only performed on the optimal path of the current iteration (or the paths of elite ants), where $\tau_{ij} = \tau_{ij} + \Delta_{ij}$, Δ_{ij} is proportional to the fitness value of the optimal path.

The ACO phase terminates after reaching the preset number of iterations and outputs the globally optimal scheduling scheme found by the entire hybrid algorithm.

The pseudocode of the algorithm is shown in Algorithm 2.

5. Experimental Simulation Results and Analysis

To verify the effectiveness and superiority of the proposed hybrid GA-ACO algorithm, simulation experiments were conducted, comparing it with the standard GA,[16] standard ACO,[17] and the Minimum Completion Time-first algorithm (Min-CT).[18]

Algorithm 2

```
Input: Number of tasks n, nodes nodes, number of ants antNum, number of iterations maxIter, parameters α,β,ρ, number of elites eliteNum
Output: Globally optimal scheduling scheme
function ACO(n, nodes, antNum, maxIter, α, β, ρ, eliteNum):
    τ = InitPheromone(n, nodes)
    globalBest = null; globalBestFitness = ∞
    for iter = 1 to maxIter:
        solutions = []
        for ant = 1 to antNum:
            solution = []; allowed = InitAllowed(nodes)
            for i = 1 to n:
                probs = CalcProb(i, allowed, τ, α, β, nodes)
                j = SelectNode(probs)
                solution.append((i,j))
                allowed = UpdateAllowed(allowed, j, i)
                τ = LocalUpdate(τ, i, j, ρ)
            solutions.append(solution)
        fitness = CalcFitness(solutions)
        currentBest = SelectBest(solutions, fitness)
        if currentBest.fitness < globalBestFitness:
            globalBest = currentBest
        τ = GlobalUpdate(τ, SelectElites(solutions, fitness, eliteNum))
    return globalBest
```

5.1 Experimental environment and parameter settings

The edge computing simulation environment was built using Python with the SimPy discrete-event simulation library. A system containing 1 cloud center and 10 heterogeneous ENs was simulated. The computing capability F_j of nodes was randomly generated within the range [2.0, 5.0] GHz and the memory M_j within [8, 32] GB. The data transmission rate B_j was negatively correlated with cloud distance, varying in the range [10, 100] Mbps. Tasks arrived randomly, with the number of tasks ranging from 100 to 1000 to test algorithm performance under different scales. The task data volume $D_i \sim$ U[10, 500]MB, the computational load $C_i \sim$ U[1000, 20000]MI, and priorities were assigned randomly.

Algorithm parameters:

GA: Population size *popSize* = 50, maximum iterations *maxIter* = 100, crossover probability *crossP* = 0.8, mutation probability *mutateP* = 0.1.

ACO: Number of ants *antNum* = 30, maximum iterations *maxIter* = 100, $\alpha = 1$, $\beta = 2$, $\rho = 0.5$.

GA-ACO: 50 iterations in the GA phase and 50 iterations in the ACO phase; weight coefficients $\alpha = 0.7$, $1 - \alpha = 0.3$.

5.2 Results and analysis

5.2.1 Comparison of comprehensive performance

In a typical scenario with 500 tasks, the performance comparison of the four algorithms is shown in Table 1.

Table 1
Performance comparison of the four algorithms.

Algorithm	Comprehensive objective (F)	Average execution time (s)	Load balancing standard deviation	System energy consumption (kJ)
Min-CT	1.25	45.6	0.38	58.7
Standard GA	0.95	38.2	0.25	52.1
Standard ACO	0.89	36.8	0.21	50.5
Proposed GA-ACO	0.76	32.5	0.15	47.8

The proposed GA-ACO hybrid algorithm outperforms the comparison algorithms in all performance metrics. Compared with Min-CT, GA-ACO reduces the average execution time by approximately 28.7%, improves the load balancing degree by about 60.5%, and decreases energy consumption by around 18.6%. This demonstrates the necessity of optimized scheduling for enhancing the overall system performance.

Compared with the single GA and ACO, the comprehensive objective value of GA-ACO is reduced by 20 and 14.6%, respectively. This verifies the effectiveness of the hybrid strategy: the global exploration of GA provides a high-quality starting point for ACO, avoiding the blind search of ACO in the initial stage, while the local optimization of ACO refines the rough results of GA, leading to better solutions.

The algorithm proposed in this paper is only verified through simulation and has not been deployed on real edge computing hardware or sensor nodes. In practical systems, the performance improvement may decrease slightly owing to factors such as hardware performance, network fluctuations, and sensor sampling frequency, but the overall optimization trend remains consistent. The proposed framework is feasible for engineering implementation, can be deployed on general-purpose edge servers and IoT gateways, and supports the real-time offloading and processing of sensor data.

5.2.2 Comparison of comprehensive performance

A curve depicting the variation inthe comprehensive objective value F with the number of iterations during the solution process of the three meta-heuristic algorithms is plotted. The F-value is the comprehensive evaluation metric designed in this paper, which is used to uniformly measure the overall performance of task execution time, load balancing degree, and energy consumption. A smaller value indicates a higher scheduling performance. The vertical axis “Index value” in Fig. 2 refers to the value of this comprehensive metric.

The simulation results in Fig. 2 show that the GA curve declines rapidly in the initial stage, demonstrating strong global exploration capability, but tends to flatten in the middle and later stages, falling into local optima with slow improvement. The ACO curve declines slowly in the initial stage owing to the time required for pheromone accumulation, while its convergence accelerates in the middle and later stages. The GA-ACO curve exhibits a two-stage efficient convergence: it drops rapidly in the first 0–50 generations (GA phase); after 50 generations (ACO phase), it continues to decrease rapidly from the high starting point provided by GA, and finally converges to a value that is better than both GA and ACO. This intuitively illustrates the perfect connection and complementary advantages of the two phases.

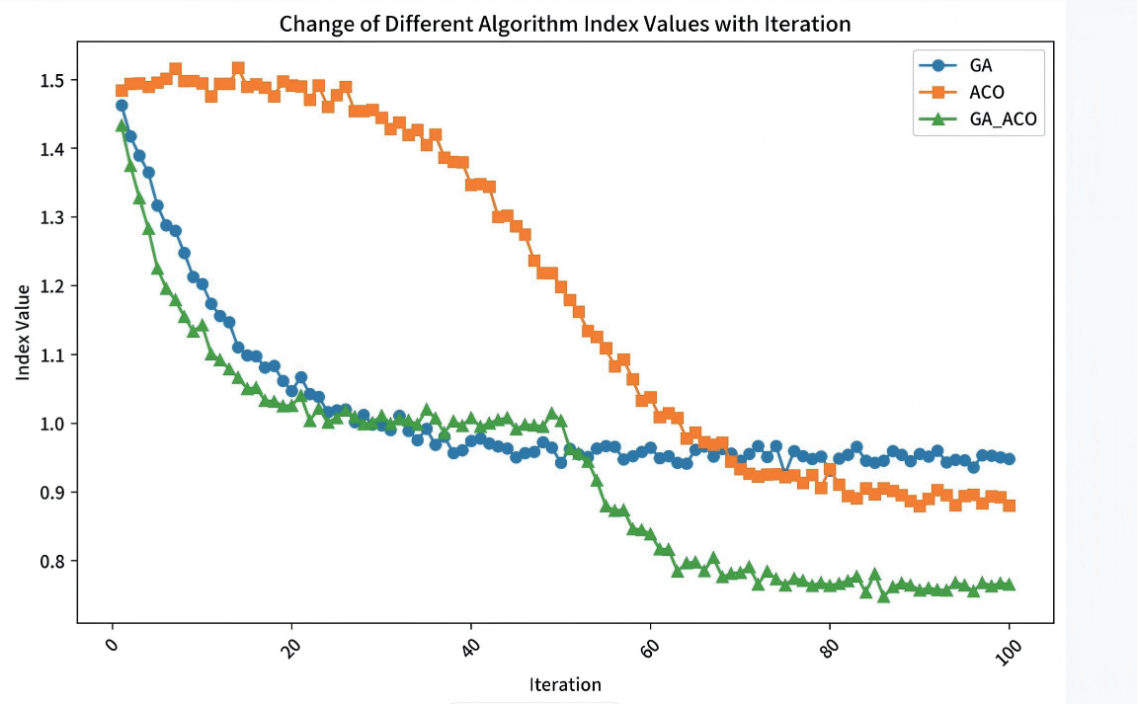


Fig. 2. (Color online) Comprehensive objective value F with the number of iterations.

The simulation results fully indicate that the proposed genetic–ant colony hybrid algorithm can significantly improve the quality of scheduling schemes, achieving the efficient convergence and the highest performance in multiple key indicators, such as task execution time, load balancing, and system energy consumption. Through the two-stage hybrid strategy, it overcomes the defects of single algorithms and realizes fast and high-quality convergence.

6. Conclusions

In this paper, we focused on the technical service IoT scenarios involving sensor terminals and investigated the balanced scheduling method for edge computing resources. The proposed algorithm can optimize the delay of sensor data processing and improve the load balancing level of ENs, which is of practical significance for enhancing the real-time performance and reliability of sensing systems. We conducted an in-depth study and exploration on the resource scheduling challenges existing in the edge computing environment of the technical service industry, such as long scheduling time, unbalanced load, and resource constraints. The main work completed is as follows: A practical system model is constructed: an edge computing system model incorporating cloud distance constraints, task priority, and node heterogeneity is established, making the problem description more in line with the application scenarios of the technical service industry. A multi-objective optimization model is built: by taking the weighted task execution time and load balancing degree as dual optimization objectives, the problem of edge computing resource-balanced scheduling is defined in a formal way. A genetic–ant colony hybrid algorithm is

designed and implemented: the global exploration capability of the genetic algorithm is innovatively combined with the local optimization advantages of the ant colony algorithm. By using the search results of GA to initialize the pheromone matrix of ACO, a high starting point search space is provided for ACO, thus accelerating convergence and finding a higher-quality scheduling scheme. Comprehensive experimental verification is carried out: through simulation experiments and comparison with a variety of classic algorithms, the results show that the proposed algorithm has significant advantages in reducing task delay, achieving load balancing and lowering system energy consumption, which verifies its effectiveness and superiority. This paper has the following limitations: validation is only conducted in a simulation environment without actual testing on real edge hardware and sensor platforms; dynamic node failure and large-scale sudden task scenarios are not considered; and the algorithm complexity increases rapidly with the scale of tasks. In the future, we will carry out experimental research in real IoT sensing scenarios to optimize the real-time performance and robustness of the algorithm. The research results provide an efficient and reliable solution for solving the problem of edge computing resource scheduling in the technical service industry and have positive reference value for promoting the practical application of edge computing in the technical service industry with high requirements for real-time performance and reliability.

Acknowledgments

This work was supported by the Industrial Big Data and Industrial Intelligence Engineering Technology Research and Development Center, Qinglan Project of Colleges and Universities in Jiangsu Province, and the 2026 National Higher Vocational College General Artificial Intelligence Course Teaching Research Project: Construction and Empirical Research on Intelligent Training Platform for General Artificial Intelligence Courses (KT2508108).

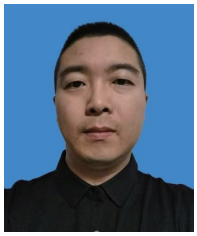
References

1 X. Li, T. Chen, D. Yuan, J. Xu, and X. Liu: IEEE Trans. Serv. Comput. **16** (2023) 845. https://doi.org/10.48550/arXiv.2102.12236
2 D. Y. Zhang, M. T. Rashid, X. Li, N. Vance, and D. Wang: ACM/IEEE IoT DI 2019. https://doi.org/10.1145/3302505.3310067
3 D. Y. Zhang, Y. Ma, C. Zheng, Y. Zhang, X. S. Hu, and D. Wang: 2018 IEEE/ACM Symposium on Edge Computing (SEC, IEEE) pp. 243–259. https://doi.org/10.1109/SEC.2018.00025
4 D. Y. Zhang and D. Wang: INFOCOM (2019) 766–774. https://doi.org/10.1109/INFOCOM.2019.8737409
5 L. Gu, J. Cai, D. Zeng, Y. Zhang, H. Jin, and W. Da: Future Gener. Comput. Syst. **95** (2019) 89. https://doi.org/10.1016/j.future.2018.12.062
6 L. Long, Z. Liu, J. Shen, and Y Jiang: Future Gener. Comput. Syst. **166** (2025) 107627. https://doi.org/10.1016/j.future.2024.107627
7 B. Zhang and D. X. Chen: Comput. Commun. **159** (2020) 299. https://doi.org/10.1016/j.comcom.2020.04.051
8 J. Liu, T. Yang, J. Bai, and B. Sun: Future Gener. Comput. Syst. **121** (2021) 48. https://doi.org/10.1016/j.future.2021.02.018
9 C. Yang, F. Liao, S. Lan, L. Wang, W. Shen, and G. Q. Huang: Eng. **22** (2023) 60. https://www.sciencedirect.com/science/article/pii/S2095809921004537
10 V. Arulkumar, K. K. Raju, R. Pemula, and M. S. Vigil: Expert Syst. Appl. **293** (2025) 128594. https://doi.org/10.1016/j.eswa.2025.128594
11 Y. Liu, B. Yan, and J. Yu: IET Cyber-Physical Syst.: Theory Appl. **9** (2024) 63. https://doi.org/10.1049/cps2.12067
12 X. Yin and L. Chen: J. Inf. Process. Syst. **19** (2023) 450. https://doi.org/10.3745/JIPS.01.0095

13 H. Yuan, Z. Zheng, J. Bi, J. Zhang, and M. C. Zhou: 2024 IEEE Int. Conf. Systems, Man, and Cybernetics (SMC) 647–652. https://doi.org/10.1109/smc54092.2024.10831521
14 H. D. NThanh, P. N. N. Thien, and B. B. Cong: Lecture Notes in Networks and Systems (2024) 355–364. https://doi.org/10.1007/978-981-97-3299-9_29
15 S. E. V. S. Pillai, S. Paramasivam, K. Polimetla, S. Dhanasekaran, K. K. Agrawal, S. P. Yadav, J. Logeshwaran, and G. Gatto: Syst. Soft Comput. **7** (2025) 200323. https://doi.org/10.1016/j.sasc.2025.200323
16 Z. Nan, L. Wenjing, and Y. N. Nahar: Comput. Mater. Continua **71** (2021) 843. https://doi.org/10.32604/cmc.2022.017504
17 F. Al-Turjman and M.Abujubbeh: Secur. Commun. Netw. **1** (2021) https://doi.org/10.1155/2022/ 4868618
18 Y. Xu, T. Zhang, L. Xiao, and X. Chen: IEEE Trans. Veh. Technol. **70** (2021) 10324. https://doi.org/10.1109/TVT.2021.3112853

About the Authors

Liu Chunxiao received her B.S. degree from Northeast Electric Power University, China, in 2007 and her M.S. and Ph.D. degrees from Northeastern University, China, in 2009 and 2013, respectively. From 2013 to 2021, she was a lecturer at Bohai University, China. Since 2021, she has been a lecturer at Changzhou College of Information Technology. Her research interests include data analysis, edge computing, and routing scheduling. (xiaoxiao198525@163.com)



Zhang Yan received his B.S. degree from Northeast Electric Power University, China, in 2007. From 2013 to 2021, he was a lecturer at Bohai University, China. Since 2021, he has been a lecturer at Changzhou College of Information Technology. His research interests are in edge computing and automatic control. (neduzy@163.com)

Wang Yanfeng received his M.S. and Ph.D. degrees from Northeastern University, China, in 2009 and 2013, respectively. From 2013 to 2021, he was an assistant professor at Huzhou University, China. Since 2021, he has been a professor at Suqian University. His research interests are in networked control system and automatic control. (neu2009wyf@163.com)

Li Long graduated from Dhurakij Pundit University, Thailand, in September 2024 and is currently pursuing his Ph.D. degree at Bangkokthonburi University. Since August 2020, he has been working at Jiujiang Vocational and Technical University as a full-time teacher. His main research interests include mathematics, computer science, and related fields. (lixiaolong2008com@163.com)