applied sciences

Article

Global Resource Scheduling for Distributed Edge Computing

Aiping Tan [1], Yunuo Li [2], Yan Wang [1,*] and Yujie Yang [1]

[1] College of Information, Liaoning University, Shenyang 110036, China; aipingtan@lnu.edu.cn (A.T.); yyjfighting@163.com (Y.Y.)
[2] Ming Yang Institute, Shenyang 110163, China; liyn@mingyangtech.com.cn
* Correspondence: wang_yan@lnu.edu.cn

Abstract: Recently, there has been a surge in interest surrounding the field of distributed edge computing resource scheduling. Notably, applications like intelligent traffic systems and Internet of Things (IoT) intelligent monitoring necessitate the effective scheduling and migration of distributed resources. In addressing this challenge, distributed resource scheduling must weigh the costs associated with resource scheduling, aiming to identify an optimal strategy amid various feasible solutions. Different application scenarios introduce diverse optimization objectives, including considerations such as cost, transmission delay, and energy consumption. While current research predominantly focuses on the optimization problem of local resource scheduling, there is a recognized need for increased attention to global resource scheduling. This paper contributes to the field by defining a global resource scheduling problem for distributed edge computing, demonstrating its NP-Hard nature through proof. To tackle this complex problem, the paper proposes a heuristic solution strategy based on the ant colony algorithm (ACO), with optimization of ACO parameters achieved through the use of particle swarm optimization (PSO). To assess the effectiveness of the proposed algorithm, an experimental comparative analysis is conducted. The results showcase the algorithm's notable accuracy and efficient iteration cost performance, highlighting its potential applicability and benefits in the realm of distributed edge computing resource scheduling.

Keywords: resource scheduling; distributed systems; edge computing

Citation: Tan, A.; Li, Y.; Wang, Y.; Yang, Y. Global Resource Scheduling for Distributed Edge Computing. *Appl. Sci.* **2023**, *13*, 12490. https://doi.org/10.3390/app132212490

Academic Editor: Sven Gotovac

Received: 10 October 2023
Revised: 10 November 2023
Accepted: 14 November 2023
Published: 19 November 2023

1. Introduction

The application scope of global resource scheduling in distributed edge computing is extensive. Existing research predominantly emphasizes optimization objectives such as maximizing resource utilization, load balancing, minimizing latency, and reducing energy consumption. However, there is a pressing need for more effective definitions of resource quality, particularly concerning tasks and data, within the constraints imposed. For instance, Figure 1a illustrates a scenario within an intelligent traffic system. In such a system, diverse edge computing nodes are distributed across different regions, monitoring locally owned traffic resources such as traffic police, rescue vehicles, and ambulances. These nodes submit real-time resource scheduling requests based on the prevailing traffic situation. In Figure 1a, both Node B and Node C require Resource 3, and there exist multiple feasible options for allocating Resource 3 among these nodes. Figure 1a exemplifies a scenario in the field of Internet of Things (IoT) edge computing. Base Station A receives data from Sensors 1, Sensor 2, and Sensor 3, and must determine the target base station to which the data from each sensor should be forwarded. For instance, the decision of where Base Station A should forward the data from Sensor 1 is crucial. From a global perspective, minimizing latency becomes the optimization objective, considering that all data are ultimately transferred. In these scenarios, the consideration of resource quality, akin to resource price, becomes imperative. The existence of different resource prices further complicates the challenge posed by the global resource scheduling problem.

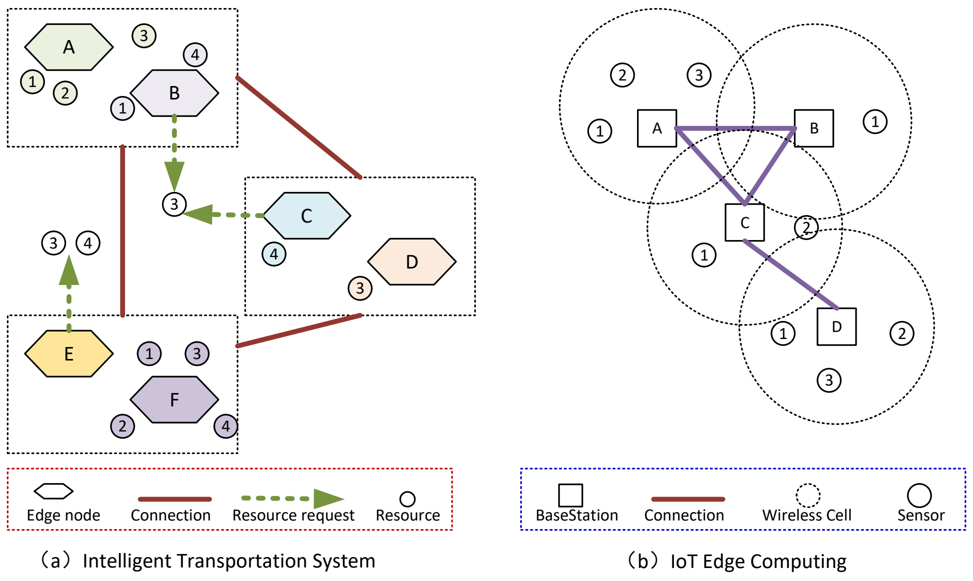

Figure 1. The case of large-scale distributed resource migration and scheduling.

Several scholars have undertaken extensive research on large-scale distributed resource migration and dispatching, each based on different application scenarios [1–3]. For instance, Jia et al. [4] proposed a two-layer distributed collaborative evolution (DCE) architecture with adaptive computing resources for large-scale optimization. Experimental studies demonstrated that the proposed distributed architecture exhibits high scalability and efficiency. Askarizade et al. [5] utilized virtualization technology to apply a hybrid approach to resource management. This technology employs the K average cluster to the mapper and a micro-genetic algorithm to enhance the ramming method. The research indicated that KMGA technology effectively reduces energy consumption in data centers and maintains continuous service quality, providing a favorable compromise. Additionally, compared with particle swarm optimization (PSO) and similar hybrid technologies based on genetic algorithms, it minimizes the number of virtual machine migrations and generation time. Yuan et al. [6] established an intelligent manufacturing workshop resource dispatch model. They improved non-dominating sorting genetic algorithms (NSGA-II), selected evaluation functions based on sorting levels and congestion, and introduced a competitive mechanism. The generation of new groups was based on process and machine-based random mutation strategies and cross-methods. An improved elite retention strategy, a variable proportional method to determine probability, and a layered analysis method to decide the optimal solution were incorporated. The efficacy and superiority of the improved algorithm were validated through benchmark case tests and real-world production and processing challenges. However, existing research faces certain challenges, mainly characterized by the following: firstly, a relatively limited focus on global optimization; secondly, a tendency towards single optimization goals, typically centering on specific factors like delay and energy consumption; and thirdly, a lack of consideration for dynamic change parameters during the optimization process.

Current research predominantly centers on proposing optimization scheduling solutions tailored to specific application scenarios, highlighting a need for more extensive exploration of a unified global scheduling model. This gap is particularly evident in the following aspects: Firstly, existing optimization models lack generality and a standardized mathematical definition for the global resource scheduling problem in edge computing. Secondly, current optimization scheduling methods need to adequately consider resource diversity and account for constraints on scheduling different qualities of resources. Thirdly, most research falls under static scheduling and needs to consider dynamic parameter changes adequately.

To address these challenges, this paper introduces a global resource scheduling problem for distributed edge computing and presents a solution algorithm. The primary motivations for this paper include:

(1) Definition of the global resource scheduling problem: In contrast to existing research, this paper incorporates two factors into the optimization objective: resource prices and migration costs. Depending on different application scenarios, migration costs may include latency and load balancing. Innovative constraints, such as the price budget constraint, are introduced within the context of resource scheduling. The paper conducts a computational complexity analysis and proves that the problem is NP-Hard.

(2) Ant colony optimization (ACO)-based adaptive dynamic scheduling algorithm: Recognizing the difficulty of solving NP-Hard problems in polynomial time, this paper proposes an ACO-based adaptive dynamic scheduling algorithm. ACO converges quickly and is less likely to get stuck in local optima. However, the method's performance is constrained by parameter settings, and ant colony optimization involves numerous parameters that could be challenging to set manually. Therefore, the paper leverages a particle swarm optimization (PSO) approach to optimize parameters during the ACO process, enhancing the reasonableness of parameter settings for each iteration.

(3) Comparative analysis with general methods: The paper conducts a comparative analysis of the proposed algorithm with standard methods, including basic ACO algorithms, genetic algorithms (GA), and ACO-GA algorithms. The algorithms are compared in terms of accuracy and iteration cost.

The remainder of this paper is organized as follows: Section 2 presents an overview of the relevant research work. Section 3 defines the global resource scheduling problem and analyzes its computational complexity. Section 4 introduces an adaptive dynamic scheduling algorithm based on the ant colony algorithm and utilizes particle swarm optimization for parameter optimization. Section 5 establishes an experimental platform and conducts a comparative analysis of algorithm performance. Finally, Section 6 provides a summary of the work presented in this paper.

2. Related Work

Currently, the research focus in distributed resource scheduling is broad and encompasses various domains. These include network data flow scheduling [7], cloud computing [8], language processing [9], energy scheduling [10], blockchain [11], vehicular edge computing [12], and other fields [13]. The classification of related research work is summarized in Table 1.

Table 1. Summary of related work.

Number	Category	Description	Literature
1	Resource scheduling	Resource scheduling is an important research area in edge computing, involving how to effectively manage and allocate computational resources in distributed systems to achieve performance optimization, load balancing, and maximized resource utilization.	[14–23]
2	Routing scheduling	Research in the context of edge computing with a mesh topology focuses on determining the optimal path for migrating edge computing tasks within a distributed resource environment. Studies in this area primarily emphasize performance metrics such as low latency and high reliability.	[24–27]
3	Network scheduling	Network scheduling in edge computing is a critical research area aimed at optimizing resource management and task scheduling within the edge computing environment to meet the growing computational demands and enhance user experience.	[28–31]

(1) Resource scheduling

Shen et al. [14] transformed the multi-virtual machine scheduling problem into a structured combinatorial optimization problem, approaching it as a reinforcement learning problem. They introduced a reinforcement learning algorithm with an incremental reward scheme and a scenario-guided sampling strategy. Goudarzi et al. [15] proposed an optimization model for computing resource allocation that utilizes cooperative evolutionary computation. This model addresses the joint optimization problem of queue-based computation offloading and adaptive computing resource allocation. The method ensures task computation latency for all mobile nodes (MNs) within a time block, optimizes the total accessibility rates of MNs, and reduces energy consumption for both unmanned aerial vehicles (UAVs) and MNs, all while meeting the constraints of task computation. In [16], regional optimization of the irrigation plan is achieved by combining a multiobjective algorithm with the exponential efficiency coefficient method. Ref. [17] proposes several optimization attributes based on high-end equipment development characteristics, and designs an effective heuristic algorithm. Given that the resource-constrained project scheduling problem is NP-Hard, three problem neighborhoods are constructed, and a variable neighborhood search algorithm is developed for problem resolution. Wu et al. [18] addressed issues in the employee evaluation mechanism of enterprises. They improved the current evaluation model based on neural networks and analytic hierarchy process, scientifically evaluating the matching degree of employees' abilities and job requirements. This is achieved by learning the radial basis function (RBF) neural network model for optimal employee-job matching and migration. Yuan et al. [19] studied the scheduling method of dynamic service resources in a cloud manufacturing environment. This method aims at time, cost, quality, and capacity and establishes an optimal scheduling model for emotional service resources. ACO is improved, and GA functions are used to optimize the objective function. A GA-based optimization fusion algorithm is proposed to solve the model. Wang et al. [20] proposed a distributed particle swarm optimization (PSO) method extended to large-scale cloud workflow scheduling. This method divides the entire population into multiple groups, employing the master–slave multigroup distributed model to co-evolve these groups into a distributed PSO (DPSO) to enhance algorithm diversity. The DPSO adopts a dynamic group learning (DGL) strategy to balance diversity and convergence. Jiang et al. [21] proposed a distributed intelligent resource scheduling (DIRS) framework to minimize the sum of task delay and energy consumption. The scheduling problem is formulated as a mixed-integer nonlinear programming. Maab et al. [22] proposed an intelligent metaheuristic optimization model to address the problem of low-quality service and achieved the optimization of time and precision in task unloading through PSO on GPU. Zhang et al. [23] introduced a coevolutionary algorithm with function-independent decomposition (FID). For large-scale problems, the binary code of the original model is converted to integer code to reduce dimensionality, and a new FID is designed to decompose the problem effectively.

(2) Routing scheduling

Routing scheduling research finds wide-ranging applications in engineering, such as urban vehicle scheduling and robot path planning in intelligent factories [24]. Jahic et al. [25] introduced a scheduling model for urban bus charging, considering a heterogeneous fleet of electric buses with different battery capacities. The optimized scheduling method aims to minimize the number of buses charged simultaneously, thereby reducing the peak load. Li et al. [26] proposed a time scheduling method for fixed line-oriented urban bus departures. They proved the NP-hardness of the scheduling problem, simplified it as a variant of a k-clustering problem, and provided a practical application of dynamic programming. Mousa et al. [27] proposed parallelizing particle swarm optimization based on a graphics processing unit (GPU) to balance cluster sizes and identify the shortest path.

(3) Network scheduling

Farahbakhsh et al. [28] introduced a context-aware offloading problem for multiple users based on Bayesian learning automata (BLA). BLA is employed to learn all states and actions within the network, subsequently enhancing the offloading algorithm. Contextual information is gathered through autonomous management in a monitoring–analysis–planning–execution loop throughout all offloading processes. Agiwal et al. [29] studied current 4G–5G interworking solutions, analyzed the practical application feasibility and challenges of various interworking solutions, discussed the possibility of spectrum sharing between 4G and 5G wireless networks, and explored the migration path to 5G independent networks. Son et al. [30] proposed a dynamic resource allocation algorithm for virtual network functions (VNFs). The algorithm considers the latency requirements of different applications in the service function chain, allowing delay-sensitive applications to reduce end-to-end network latency through edge resources. Zhao et al. [31] proposed an adaptive distributed scheduling algorithm on resource-constrained edge clusters of the Internet of Things. The algorithm utilizes a convolutional neural network (CNN) to optimize the distributed resource scheduling problem, using minimum memory occupation as the objective function. This method adopts a new work scheduling process to improve data reuse and reduce overall execution delay.

In summary, current research spans various fields, encompassing intelligent grids [32], smart mining [33], smart factories [34], and more. However, a predominant trend in current research is the concentration on a singular optimization goal, such as minimizing delay or finding the shortest path. Different engineering application scenarios present distinct challenges, and their solution methods vary accordingly. Therefore, the design of optimization methods should align with specific requirements dictated by the nature of each application.

3. System Formulation

3.1. Problem Definition

The topology of the system is illustrated in Figure 2. Each resource node in the figure can represent any entity with resources, such as an intelligent edge node or an IoT base station as shown in Figure 1. Each resource node possesses multiple resources of the same type, and each resource is characterized by two key attributes $\langle C, P \rangle$. Here, C denotes the quantity of resources the entity owns. If it is a positive number, the entity possesses C resources; if it is zero or negative, the entity must request at least C resources. The attribute P represents the resource price. If $C > 0$, then P signifies the unit price of the resource; if $C \leq 0$, P represents the total resource budget requested. The weight value of the edge connecting any two nodes represents the contribution value of resource migration between them. This value primarily reflects factors such as the distance between nodes, migration history, and other relevant information. The value is automatically updated with each node resource migration. In Figure 2, nodes A and B are part of the same cell, denoted by dotted lines, indicating that the distance cost of resource migration (or communication cost in IoT edge computing applications) between these two nodes is relatively small. As an illustrative example, consider that node A requires at least three No. 0 resources, with a total budget of 23. Nodes B, C, and D each possess No. 0 resources, where the unit price of node D is 9, and node B is 8. If three resources are obtained from either D or B, we have $3 \times 9 > 23$ and $3 \times 8 > 23$, which do not meet the budget requirements. Therefore, only one part can be selected from D and B. In this example, there are multiple options, such as migrating 2 No. 0 resources from B and 1 No. 0 resource from C, resulting in a total cost of $2 \times 8 + 1 \times 7 = 23$, meeting the budget requirements. Alternatively, we can migrate 1 No. 0 resource from B, C, and D, respectively, resulting in a total cost of $8 + 7 + 9 = 24$. As the number of nodes and resource types increases, when multiple nodes are involved in resource migration, ensuring the minimum scheduling cost for each node becomes a challenging global optimization problem.

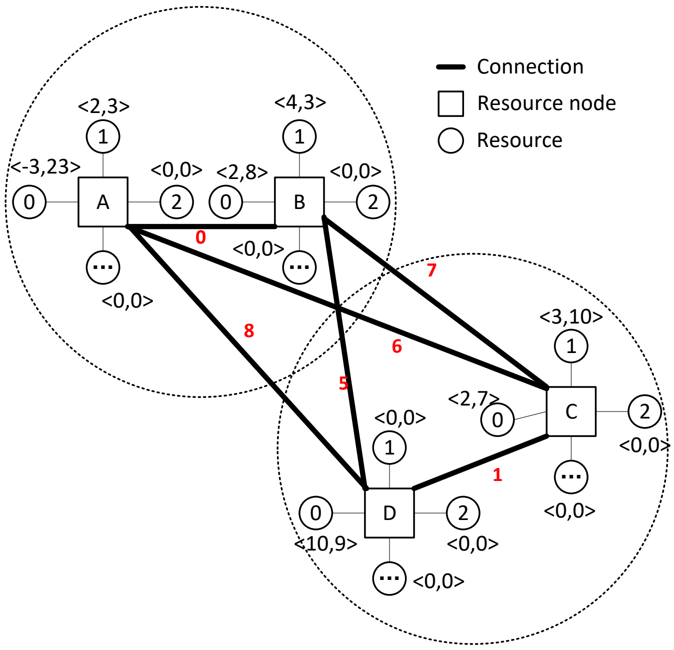

Figure 2. Topology of the problem.

Suppose there are I nodes in total, and each node has the same J resource type. $\forall i \in \{0, 1, \ldots, I-1\}, j \in \{0, 1, \ldots, J-1\}$, $C_{i,j}$ represents the quantity of the j-th resource of the i-th node; $P_{i,j}$ represents the unit price of the j-th resource at the i-th node. $\forall i, i' \in \{0, 1, \ldots, I-1\} \land i \neq i'$, $L_{i,i'}$ represents the migration cost between node i and node i'. Suppose maxP and maxL are used to represent the upper limit of resource unit price and migration cost; then the quality of service (QoS) function of node i requesting i' migration resource j is shown as follows:

$$Q_{i,i;}(j) = \frac{1}{\frac{X_{i,i'}(j) \times P_{i',j}}{max\mathsf{P}} \times \alpha + \frac{L_{i,i'}}{max\mathsf{L}} \times (1-\alpha)} \tag{1}$$

where α is a balance factor used to set the importance of price and migration cost according to user needs, where $X_{i,j}(j)$ represents the number of resource j migrated from node j to node i. $L_{i,i'}$ is not fixed. It includes the migration distance between nodes i and i', historical migration records, etc. After each migration, we can modify the value of $L_{i,i'}$ according to the results of the previous migration. The modification principles can be determined according to the actual situation in specific engineering applications. For example, for the edge computing scenario of intelligent transportation, if the last scheduling effect between two edge nodes is good, $L_{i,i'}$ can be increased; otherwise, it can be reduced.

Definition 1. *For given I, J, C, P, L, we can define a global resource scheduling problem of distributed edge computing as follows:*

$$max \sum_{i=0}^{I-1} \sum_{i'=0}^{I-1} \sum_{j=0}^{J-1} Q_{i,i'}(j)$$

$s.t.\ \forall C_{i,j} \leq 0$

$$\sum_{i'=0}^{I-1} X_{i,i'}(j) \geq -C_{i,j} \tag{2}$$

$$P_{i,j} \geq \sum_{i'=0}^{I-1} X_{i,i'}(j) \times P_{i',j} \tag{3}$$

3.2. Computational Complexity Analysis

For given I, J, C, P, L, we use $S(I, J, C, P, L)$ to represent the solution function of the global resource scheduling problem of distributed edge computing. We have the following theorem:

Theorem 1. *Problem $S(I, J, C, P, L)$ is NP-Hard.*

Let us prove theorem 1. According to definition 1, we define subproblem S' of problem S:

Definition 2. *Let $J = 1$, and in Formula (1), let $\alpha = 0, max_L$. Therefore, the price P has no meaning for the objective function Q, so the subproblem $S'(I, 1, C, P, L)$ of problem S can be described as:*

$$max \sum_{i=0}^{I-1} \sum_{i'=0}^{I-1} Q_{i,i'}(0)$$

$s.t.\ \forall C_{i,0} \leq 0$

$$\sum_{i'=0}^{I-1} X_{i,i'}(0) \geq -C_{i,0} \tag{4}$$

$$P_{i,0} \geq \sum_{i'=0}^{I-1} X_{i,i'}(0) \times P_{i',0} \tag{5}$$

where $Q_{i,i'}(j) = \begin{cases} \frac{1}{L_{i,i'}} & L_{i,i'} \neq 0 \\ 0 & L_{i,j} = 0 \end{cases}$

We have the following theorem:

Theorem 2. *Problem $S'(I, 1, C, P, L)$ is NP-Hard.*

Proof. To prove theorem 2, we need to find a known NP-C problem and then prove that the problem can be reduced to problem S'. A 0–1 knapsack problem is proposed and proved to be an NP-C problem [35]. □

Definition 3. *There are N items and a backpack with a capacity of K. The weight of the i-th item is $w[i]$, and the value is $v[i]$. The function $x[i] = \{1, 0\}$ represents whether the item is packed. Then, the 0-1 knapsack problem is expressed as:*

$$max \sum_{i=0}^{N-1} (v[i] \times x[i])$$

$$s.t. \sum_{i=0}^{N-1} (w[i] \times x[i]) \leq K \tag{6}$$

For given N items and knapsack capacity X, we use the function $G(N, K, w, v)$ to represent the 0–1 knapsack problem. For problem G, any input N, K, w, v, we will convert it to the input of S'; the principle is as follows:

For problem S', let $I = N + 1$, $C_{0,0} = 0$, $P_{0,0} = K$. $\forall i \in \{1, 2, \ldots, I-1\}$, let $C_{i,0} = 1$, $P_{i,0} = w_{i-1}$, $L_{0,i} = \frac{1}{v_{i-1}}$. (★)

We need to prove that the same output can be obtained for questions G and S':

1 Suppose that the problem S' obtains an output result X, satisfying Definition 2. Since only $C_{0,0} \leq 0$, we only consider $X_{0,i'}(0)$. Therefore, $\forall i' \in \{1, \ldots, I-1\}$, let $x[i'-1] = X_{0,i'}(0)$.

According to Formula (5) of Definition 2, we have $P_{0,0} \geq \sum_{i'=0}^{I-1} X_{0,i'}(0) \times P_{i',0}$. In the input conversion principle(★), we have $K \geq \sum_{i'=1}^{I-1} x[i'-1] \times w[i'-1]$. Since $I = N+1$, we have $K \geq \sum_{i'=0}^{N-1} x[i'] \times w[i']$, which satisfies the Formula (6) of Definition 3. Therefore, the output of question S' can be transformed into the output of question G. (■)

2 Suppose that the problem G obtains an output result x, satisfying Definition 3. $\forall i \in \{0, 1, \ldots, N-1\}$, let $X_{0,i+1}(0) = x[i]$. According to Formula (6), we have $\sum_{i=0}^{N-1} x[i] \times w[i] \leq K$. In the input conversion principle (★), we have $\sum_{i=0}^{N-1} X_{0,i+1}(0) \times P_{i+1,0} \leq P_{0,0}$. Due to $I = N+1$, we have $\sum_{i=1}^{I-1} X_{0,i}(0) \times P_{i,0} \leq P_{0,0}$. Because only $C_{0,0} = 0$, we have $\sum_{i'=0}^{I-1} X_{0,i'}(0) \geq 0 = -C_{0,0}$, satisfying Formula (4), Definition 2. Therefore, the output of question G can be transformed into the output of question S'. (■■)

In conclusion, (■) and (■■) indicate that problem of G and S' having the same output. Because the input conversion process of (★) is completed in polynomial time, Theorem 2 holds!

Since it has been proven that S' is an NP-hard problem, the more complex problem S must be an NP-hard problem. Theorem 1 holds.

4. Algorithm Design

4.1. Design of Heuristic Algorithm Based on ACO

Since problem S has been proven to be NP-Hard, we choose a heuristic algorithm to solve this problem. In the heuristic algorithm, this paper chooses the ACO to design.

The ACO is an intelligent heuristic algorithm. It solves some optimization problems by simulating the process of ants' foraging in nature. In the whole search process, the ants will leave pheromones on the path every time they pass through a path. The ants tend to find the path with high pheromone concentration in the subsequent process. The higher the concentration of pheromones, the higher the probability that the path will be selected. The ACO was first proposed to solve the travelling salesman problem (TSP). The process of implementing TSP with the ACO is as follows: select m ants and n cities, and first give the initial pheromone concentration on the path as τ_0, and then let m ants move from the random initial city, passing through all cities once and only once, and then update the pheromone on the path. In the next iteration, the ants will choose a path with a higher concentration. The probability $p_{ij}^k(t)$ of the k-th and going from the i-th city to the j-th city at time t can be expressed as follows:

$$p_{ij}^k(t) = \begin{cases} \frac{[\tau_{ij}(t)]^\alpha \cdot [\eta]^\beta}{\sum_{s \in N_i^k} [\tau_{is}(t)]^\alpha \cdot [\eta]^\beta} & j \in N_i^k \\ 0 & j \notin N_i^k \end{cases} \tag{7}$$

where N_i^K represents the next hop node-set, which node i can select in the k-th iteration.

The heuristic information η_{ij} is generally set as $1/d_{ij}$. Moreover, α is the relative importance of the pheromones left in the previous iteration, β is used to measure the importance of heuristic information, and N_i^K is the set of possible target cities, that is, the cities that have not been visited. We use a tour list to preserve the cities that ants have passed through, that is, cities that can no longer be selected during the selection process. The pheromone is updated according to the following formulas:

$$\tau_{ij}(t+n) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij} \tag{8}$$

$$\Delta\tau_{ij} = \sum_{k=0}^{m} \Delta\tau_{ij}^k$$

$$\Delta\tau_{ij}^k = \begin{cases} \frac{Q}{L_k} & \text{if edge}(i,j) \text{ its tour} \\ 0 & \text{otherwise} \end{cases}$$

where ρ indicates the volatilization rate of the pheromone, then $(1-\rho)$ refers to the extent to which the pheromone inherits the pheromone at the last time. It is also called the residual factor. $\Delta\tau_{ij}$ represents the increased concentration of pheromones on edges(i,j). Q is a constant, and L_k represents the length of the path that the k-th ant walked in this iteration.

The traditional ACO is prone to slow convergence and may only obtain the optimal local solution but not the optimal global solution. Moreover, for the parameters in the traditional ACO α, β, the parameters obtained through experiments in advance are not necessarily the optimal parameters.

In this paper, the pheromone concentration on the path composed of every two nodes at the initial time is the same, which is a constant τ_0. Each edge has the same attraction to ants at the initial time, so the probability of selecting the next node depends on the distance d_{ij} and the price of resources. From a global perspective, for the path with low scheduling cost, the local shortest path length does not mean it is the optimal result. In Figure 3, assume that $l_{ij_1} < l_{ij}$ and $l_{jk} < l_{j_1k_1}$, but $l_{ik} < l_{ik_1}$, it can be concluded that the scheduling cost of each subpath is not necessarily the minimum on the path with the lowest scheduling cost.

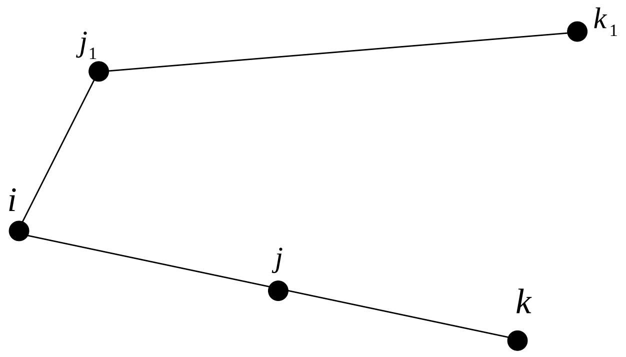

Figure 3. Example of path selection for global optimization and local optimization.

To address the challenge of the traditional ACO potentially converging to a suboptimal solution in the large-scale distributed resource migration problem, this paper introduces an enhanced approach. In the first iteration, the standard ACO may quickly converge to a layout migration result with the lowest cost, potentially overlooking a set of migration nodes that collectively offer the lowest total scheduling cost. To mitigate this issue, a modification is made in the first iteration to prevent local nodes with a greedy selection bias, thus avoiding premature convergence to a suboptimal local solution. The formula for initializing pheromone distribution in this improved ACO is expressed as follows:

$$\tau_{ij}(0) = \begin{cases} \frac{d_{ij}\times m}{n} & i \neq j \\ 0 & i = j \end{cases} \tag{9}$$

4.2. Parameter Optimization of the ACO Based on PSO

The ACO algorithm involves several parameters, and the optimization results are directly influenced by the setting of these parameters. Manually selecting parameter values does not ensure optimal performance, making it crucial to find a reasonable parameter configuration before each iteration to enhance algorithm effectiveness. To address this challenge, this paper introduces a parameter optimization method based on PSO. This dynamic approach optimizes parameters based on the results of each iteration, providing an adaptive and efficient means of setting parameters for the ACO algorithm.

The PSO is a phenomenon that simulates birds searching for food. PSO has been widely used in continuous optimization problems and discrete optimization problems [36–38]. A population of m particles has its position in the multidimensional search space. The particles fly at a certain speed to simulate the migration process of birds. When searching for the target, focus on the best solution in this history search record and update the speed and position according to specific rules based on the best record of other particle history searches in this group.

The position of the d-th $(1 \leq d \leq m)$ particle in the i-th particle group is expressed as x_{id}, whose speed is expressed as v_{id}, and whose best history is expressed as p_{id}. The best record of all particles in the population is expressed as p_{gd}. The update of particle speed and position is based on the following two formulas:

$$\begin{aligned} v_{id} &= \omega \times v_{id} + c_1 \times \text{rand}() \times (p_{id} - x_{id}) \\ &\quad + c_2 \times \text{rand}() \times \left(p_{gd} - x_{id}\right) \end{aligned} \tag{10}$$

$$x_{id} = x_{id} + v_{id} \tag{11}$$

where ω is the inertia weight, whose value determines how much of the past value of the particle remains. Whether ω is appropriate can determine whether the final solution is enough for optimization. c_1 and c_2 are called learning factors, equivalent to measuring the speed of particles approaching the optimal solution. The learning factor has the effect of evaluating itself and learning other particles. rand() is a pseudorandom number in the interval $[0, 1]$.

In this paper, the parameters α,β of the traditional ACO are optimized by training. Moreover, a mutation strategy is set so that when the ant selects a new node, the probability of selecting the next hop node through the pheromone concentration is no longer an inevitable event but has a certain probability of modifying the probability of selection.

In this algorithm, the total number of N_a ants is divided into several populations, and the number of ants in each population is the same N_{sa}. Each population maintains its pheromone matrix, which records the path between each node and the migration cost. Then, position x and speed v are converted into parameters $\{\alpha, \beta\}$; The PSO is used to optimize these two parameters after each iteration of the ant colony algorithm as the initial parameters of the next iteration. The value range of α is set to $[1, 2]$, and the value range of β is set to $[1, 2]$. So the lower bound of $\{\alpha, \beta\}$ is $\{1, 1\}$, and σ is the difference between the upper and lower bound. Set the range of particle position to $[-50, 50]$ and particle speed to $[-60, 60]$.

Suppose X_k is the k-th position element of the particle; then, the particle position is converted into a parameter by the following formula.

$$\{\alpha, \beta\} = R_{min} + \sigma * \frac{X_k - X_{min}}{X_{max} - X_{min}} \tag{12}$$

The traditional PSO is improved, and ω is optimized to make it dynamic.

$$\begin{aligned} v_{id} &= \omega^{(t)} \times v_{id} + c_1 \times \text{rand}() \times (p_{id} - x_{id}) \\ &\quad + c_2 \times \text{rand}() \times \left(p_{gd} - x_{id}\right) \end{aligned} \tag{13}$$

where $\omega^{(t)} = \frac{(\omega_{ini} - \omega_{end})(G_k - g)}{G_k + \omega_{end}}$.

Parameter G_k is the maximum number of iterations, ω_{ini} is the initial inertia weight, and ω_{end} is the inertial weight value when the number of iterations reaches the maximum. Dynamic ω can achieve better results than fixed values and has more search options. Moving in a specific direction in parameter updating is not easy, but it has more significant opportunities to search unknown fields.

4.3. Improvement of Selection Probability

To solve the problem that the ACO can easily fall into the local optimum and converge to a local solution quickly, we propose the idea of selective mutation. Set a variation index (VI). Random variation will occur when the probability of selecting the next node is more significant than VI. She mutates the selection probability, which is half of the original probability, and the selection probability of other nodes should also be changed accordingly. The probability formula for selecting the next node is:

$$p_{ij-new}^{k}(t) = \begin{cases} p_{ij}^{k}(t) \times \frac{1}{1+\frac{\delta}{t}} & p_{ij}^{k}(t) > \text{VI} \\ p_{ij}^{k}(t) & \text{otherwise} \end{cases} \tag{14}$$

where $p_{ij}^{k}(t)$ refers to the probability of selecting the next node, while $p_{ij-new}^{k}(t)$ refers to the probability of selecting the next node after selection and variation. δ indicates the probability decay rate of the next node. t represents the current number of iterations. After selecting one node changes, the probability of selecting other nodes will be normalized accordingly; that is, the probability distribution of the final roulette selection can be obtained.

According to the Formula (14), when the value of t is minimal, that is, in the first few iterations, the value of $p_{ij}^{k}(t)$ has a significant attenuation. With the increasing value of t, the probability of selecting the next node will be closer to the original unchanged probability. The advantage of this is that when the number of iterations is small, it can better weaken the probability of selecting the next node so that the algorithm has better breadth and randomness when searching for nodes and avoid quickly entering the optimal local value. With the increase in the number of iterations, the probability of modifying the selection of the next node becomes smaller and smaller and converges faster. It can ensure the breadth of ants searching for nodes and accelerate the speed of searching for the global approximate optimal solution.

This approach is not entirely blind and random to find new results, which cannot be called a random search. Because this selection mutation operation only weakens the probability that the path with too much influence is selected, rather than discarding this probability, it makes up the defect that the algorithm converges to the optimal local solution too quickly.

4.4. Pheromone Updating and Improvement

The improvement of pheromone updating rules and pheromone incremental updating rules interact. The traditional ACO for pheromone incremental updating rules converges too slowly. The reason is that the pheromone updating is too slow, which makes the algorithm unable to converge to the path at a low cost quickly. After analyzing the original pheromone incremental update, Formula (8), even though the cost of different nodes on the path is very different, the pheromone increment calculated by the formula is not significantly different. It is meaningless to choose a better path and cannot change the final selection strategy. Therefore, some improvements have been made to Formula (8) to increase the difference of pheromone increments on the path with significant cost difference so that it has a more vital ability to select multiple paths. The improved formula is as follows:

$$\Delta\tau_{ij}^{k} = \begin{cases} \frac{Q}{e^{L_k - Avg}} & \text{if edge}(i,j)\text{in its tour} \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where Q is constant; L_k is the length of the current path; Avg is the average cost of all ants' paths in this round. When the cost of an ant's path is small, the concentration of the pheromone update is greater than that of Q. Furthermore, when the cost of an ant's path is high, the concentration of the pheromone update is much smaller than that of Q, which can highlight the difference in the pheromone update value and increase the convergence speed of the algorithm. We have improved the pheromone updating rule to prevent the algorithm's convergence rate from exceeding expectations.

$$\tau_{ij}(t+n) = (1-\rho) \times \tau_{ij}(t) + \rho \times \Delta\tau_{ij} \tag{16}$$

In Algorithm 1, the heuristic algorithm flow designed in this paper is as follows:

1. Initialize the parameters, randomly generate the parameters of the particle swarm (including the speed and position of each particle), and convert the parameters into $\{\alpha, \beta\}$.
2. Initialize the pheromone and other parameters of the ACO; set N_{sa} ants of each ant population on these nodes.
3. Make all ants in each ant population search for all nodes according to Formula (14). Select the target node to apply for resources, modify the scheduling table, and calculate the total cost of scheduling resources.
4. Calculate the total scheduling cost of each population scheduling resource, and record the ant path corresponding to the optimal scheduling cost. The optimal scheduling generation value is used as the fitness value of the particle. The particle with the optimal fitness value selected is the best record of the current iteration number of the population. Update the best record of each particle's history and the best record of the population's history. After updating the speed and position according to Formula (13), convert the position parameters to new $\{\alpha, \beta\}$ according to Formula (12). Update the value of pheromone according to Formula (16).
5. If the number of iterations has reached the target number of iterations, stop the iteration; if the number of iterations does not meet the requirements, re-execute step (3).

Algorithm 1 Adaptive dynamic migration algorithm for large-scale distributed resources.

```
Require: Large-scale distributed network P
Ensure: Resource scheduling optimization scheme R
 1: Initialize parameters of particle swarm $v_{id}$, $x_{id}$, $tabu_k$, $N_{sa}$
 2: $\{\alpha, \beta\} <- v_{id}, x_{id}$
 3: for k <- 1 to $N_{sa}$ do
 4:     for i in P do
 5:         ant $k$ choose next park
 6:         modify taboo list $tabu_k$ of ant k
 7:         if $L(i) < L(known - best)$ then
 8:             replace best route
 9:             $L(best) <- L(i)$
10:             Shortest route $<- L(\mathrm{i})_{route}$
11:         end if
12:         activity of pheromone updating is executed
13:         activity of $v_{id}$,$x_{id}$ updating is excuted
14:     end for
15:     R updating is excuted
16: end for
17: Return R
```

5. Experimental Analysis

Given the characteristics of the global resource scheduling problem in distributed edge computing, dataset preprocessing is essential to better reflect the model's characteristics and structure. A dataset was constructed, comprising 50 nodes located in different positions. To assess the algorithm's performance, several algorithms were employed for comparison, including the greedy algorithm [39], the basic ant colony optimization (ACO) [40], the genetic algorithm (GA) [41], and the hybrid ACO-GA algorithm (H3AGA) [42]. The selective mutation ACO (SMACA), proposed in this paper, is then compared against this dataset.

The current operations on this dataset include two aspects:

1. Accuracy comparison
   To validate the accuracy of the algorithm proposed in this paper, a comparison is made based on the objective function values of the scheduling results obtained by

different algorithms. A higher objective function value after scheduling indicates that the results of this algorithm are closer to the optimal solution.

2. Iteration cost comparison
The iteration cost of the algorithm refers to the number of iterations, serving as a crucial performance metric. We establish the iteration times of different algorithms under the same target conditions for comparison.
3. Throughput analysis
The topology of this experiment consists of 50 nodes, with connections between each pair of nodes, forming a graph-like data structure. The edges between any two nodes have weights, which carry different meanings in various application scenarios. As illustrated in Figure 1, in the context of IoT edge computing, the edge weights can represent latency, whereas in the intelligent transportation scenario, the edge weights may denote distance. The weights of edges play a crucial role in the results of task migration scheduling. This experiment primarily focuses on the system's task processing throughput, represented by the ratio of successfully scheduled tasks within a unit of time.

Table 2 shows the algorithm parameter setting information of this experiment.

Table 2. The parameter setting of the experiment.

Id	Algorithm	Parameter
1	ACO	$\tau = 1, \rho = 0.2, \alpha = 1, \beta = 2, m = 50, T = 1000$
2	GA	The crossing probability value is set as 0.75, the variation probability is 0.05, the population size is 1000, and the chromosome length is 50.

The accuracy of the algorithm is assessed based on the objective function value. A higher objective function value indicates better accuracy for the algorithm under the same iteration number. In Figure 4, the x-axis represents the iteration number of the algorithm, while the y-axis represents the objective function value of different algorithms under the same iteration number. In the initial iterations, the accuracy of the GA is notably lower than the other three algorithms. However, with a gradual increase in the number of iterations, the accuracy of the GA improves. Compared with the three algorithms, the SMACA proposed in this paper demonstrates significantly better convergence speed than the ACO and H3AGA. It approaches the maximum target value at the fastest rate. With the increase in the number of iterations, the objective function value of the ACO continues to rise, and its convergence time is the longest among the ant colony algorithms. The H3AGA converges before the 400th iteration, while the SMACA requires about 200–300 iterations to converge. In summary, the SMACA can substantially reduce the iteration time cost, exhibits the fastest convergence, and has a notable advantage in accuracy. This experimental result is attributed to SMACA dynamically adjusting when selecting the next node and balancing the convergence rate during pheromone updates. The result of the greedy algorithm is independent of multiple iterations, making it incomparable with other heuristic algorithms in terms of convergence speed and accuracy.

Because the results of a single experiment may need to be more accurate due to randomness, this paper will repeat each experiment 20 times. The results of 20 experiments will be displayed in a boxplot, as shown in Figure 5. The abscissa in the figure represents different algorithms, and the ordinate represents the objective function value. In the statistics of large samples, the experimental results of the GA are very dependent on the randomness of mutation and the population size. The highest value of the GA algorithm is 2.04, the lowest value is 1.88, with a difference of 0.16. In contrast, SMACA has a highest value of 2.07 and a lowest value of 2.04, resulting in a difference of only 0.03. The experimental results of the ACO and H3AGA are evenly distributed. Compared with other algorithms, the experimental results of SMACA are more concentrated and less random. Therefore, the algorithm proposed in this paper has excellent stability.

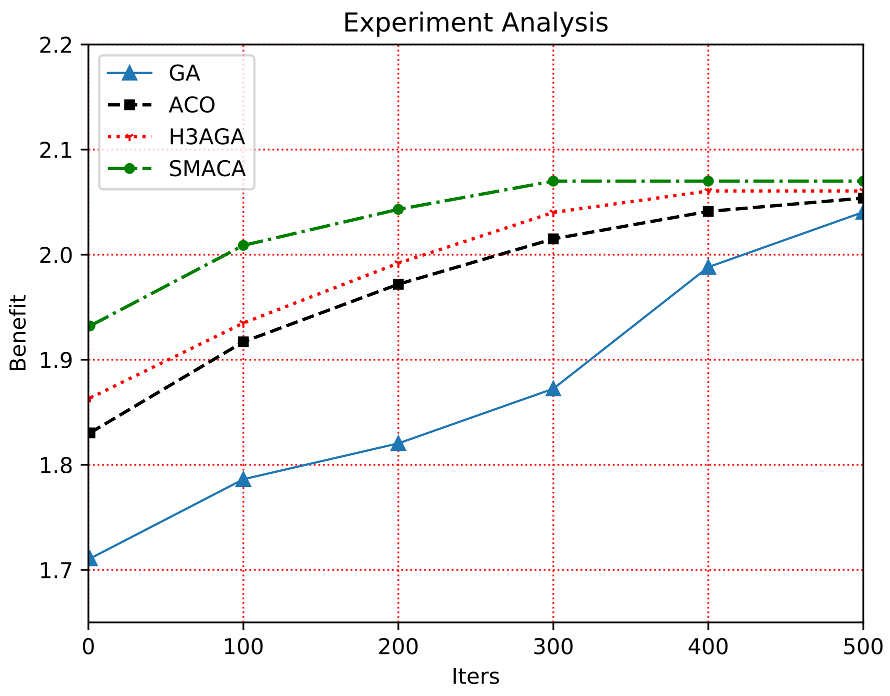

Figure 4. Comparison of algorithm minimum cost.

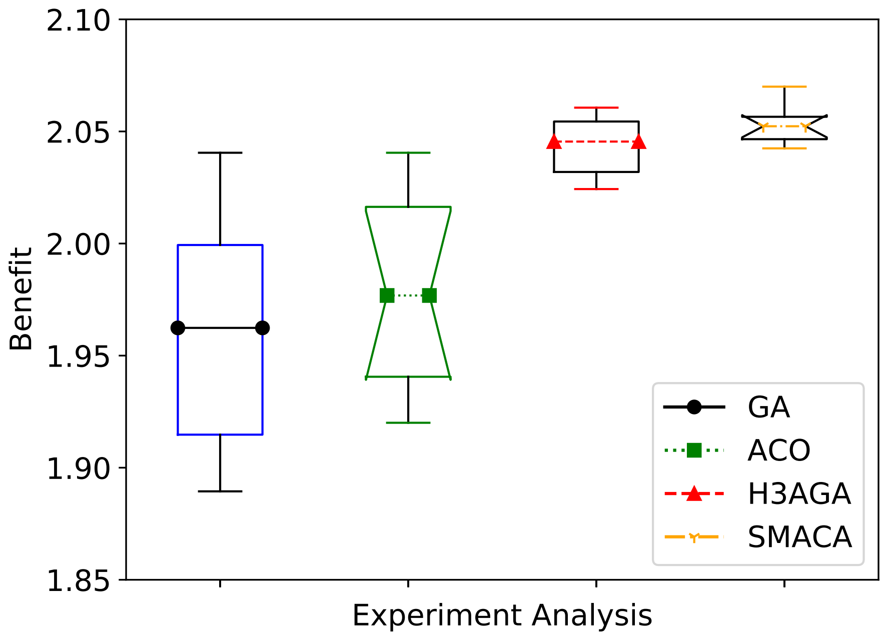

Figure 5. Distribution of multiple experimental results of different algorithms.

In the quest for the minimum scheduling cost, the criterion for assessing the merits of an algorithm is its ability to approach the actual minimum value. Due to the problem's large scale and other complexities, obtaining the true minimum value might be impractical. Therefore, a new concept, precision, is introduced here. Precision is defined as the difference between the minimum value obtained by the algorithm and the smaller minimum value obtained by the next iteration. A smaller difference indicates a closer approximation to the minimum value. Consequently, when various algorithms achieve the same accuracy, the algorithm requiring fewer iterations is closer to the actual minimum value. Applying this criterion to the problem at hand, we can formulate the following principle: when the iteration times of various algorithms are equal, the algorithm with a smaller accuracy is considered better.

Due to the inherent uncertainty in the results of the GA, a precise comparison cannot be made. This paper focuses on comparing the accuracy of the other three algorithms. Figures 6 and 7 depict the actual performance comparison of each algorithm in two different formats. The x-axis represents the number of iterations, and the y-axis represents the precision value of the algorithm. As the number of iterations increases, the accuracy value of each algorithm improves. In Figure 6, as the number of iterations increases, the performance differences among different algorithms become more apparent. When the iteration count reaches 400, the precision differences between the SMACA algorithm and the other two algorithms are 130 and 180, respectively. Throughout the iterations, the accuracy of the SMACA consistently surpasses that of the ACO and H3AGA. It can achieve lower accuracy values with fewer iterations, indicating that the SMACA exhibits superior convergence speed.

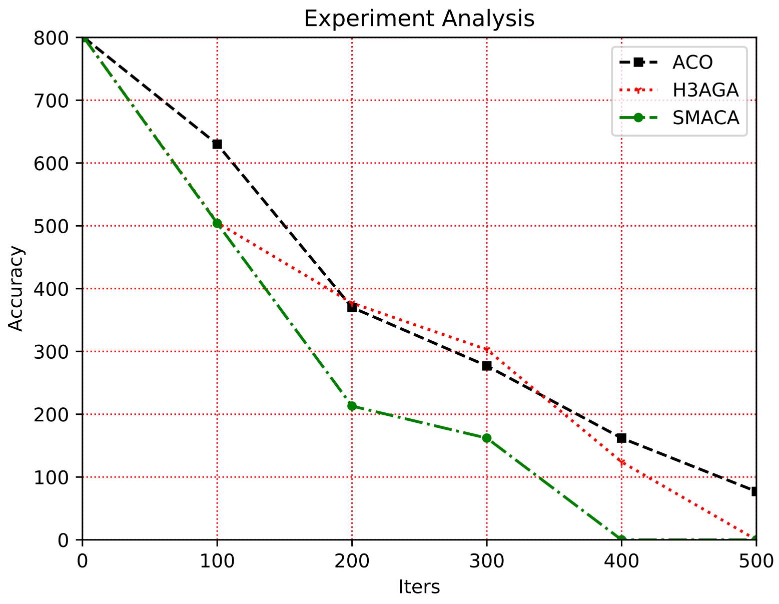

Figure 6. Accuracy comparison plot of different algorithms.

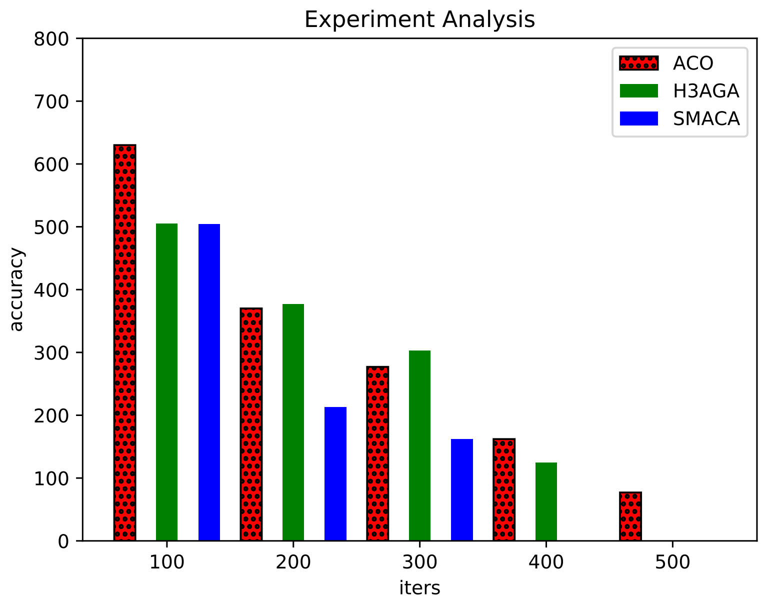

Figure 7. Accuracy comparison histogram of different algorithms.

The number of nodes is a crucial factor influencing the algorithm's performance. This paper conducts a comparative experiment on algorithm performance under different node numbers. In Figure 8, the x-axis represents the number of clusters (each cluster contains multiple nodes), and the y-axis represents the migration cost. As the number of nodes increases, the cost of resource migration also increases. This is because, with an increase in the number of nodes, the resource scheduling of any node is determined based on the objective function and constraint conditions, not only considering nodes that are close to each other. Therefore, this increase in cost is inevitable. When the number of nodes is 40, the difference between the SMACA algorithm and other algorithms is most significant, with the cost of the GA algorithm reaching 10,000. Compared with other algorithms, the SMACA proposed in this paper demonstrates superior performance in terms of average migration cost. The changing trend of the interval distance for each curve in Figure 9 indicates that the convergence time of the four algorithms is the same when the node size is small. However, with an increasing number of nodes, the advantages of SMACA become more apparent, making it more suitable for large-scale datasets. The improvement in pheromone updating rules significantly expands the difference in pheromone updating amounts on paths with different costs. This makes it easier to choose paths with low costs, greatly optimizing the convergence speed of the algorithm.

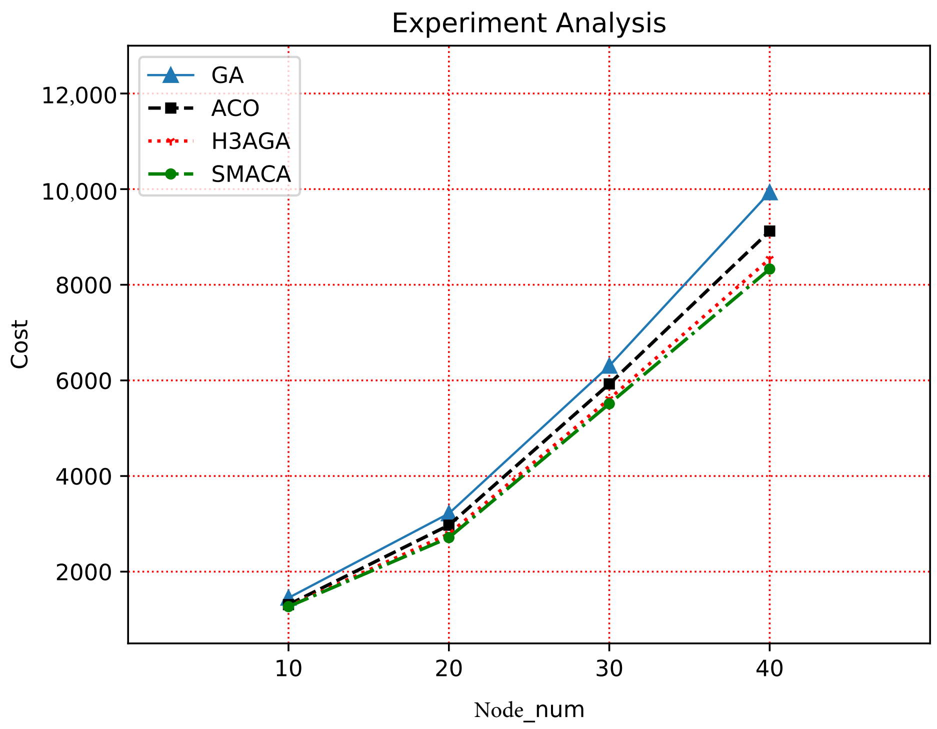

Figure 8. Cost comparison of different algorithm scales.

In the throughput experiment, we recorded the total number of task scheduling executions within the same time interval (one minute) (the ratio of scheduled tasks to the total number of tasks). As the execution results of heuristic algorithms such as ACO exhibit a certain degree of randomness, this study repeated the same set of experiments 20 times. The boxplot tool was used for statistical analysis of the experimental data. Figure 10 shows the results of the throughput experiment.

As shown in Figure 10, among the four compared algorithms, the proposed SMACA algorithm exhibits superior throughput. This is primarily attributed to the algorithm in this paper optimizing the ACO parameters through PSO, leading to a higher scheduling success rate and fewer iterations, resulting in higher throughput within a unit of time. Additionally, the stability of the SMACA algorithm is the highest, as indicated by the concentrated experimental result data. This suggests that the algorithm proposed in this paper has a higher probability of achieving favorable scheduling results when dealing with different application scenarios.

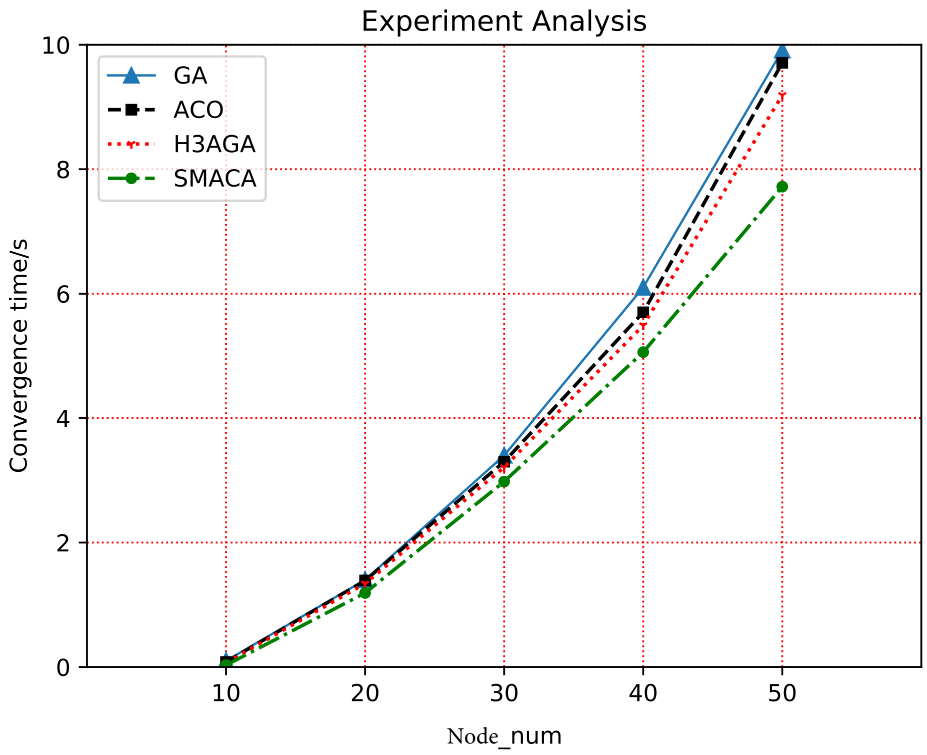

Figure 9. Comparison of convergence time of different algorithms based on scale.

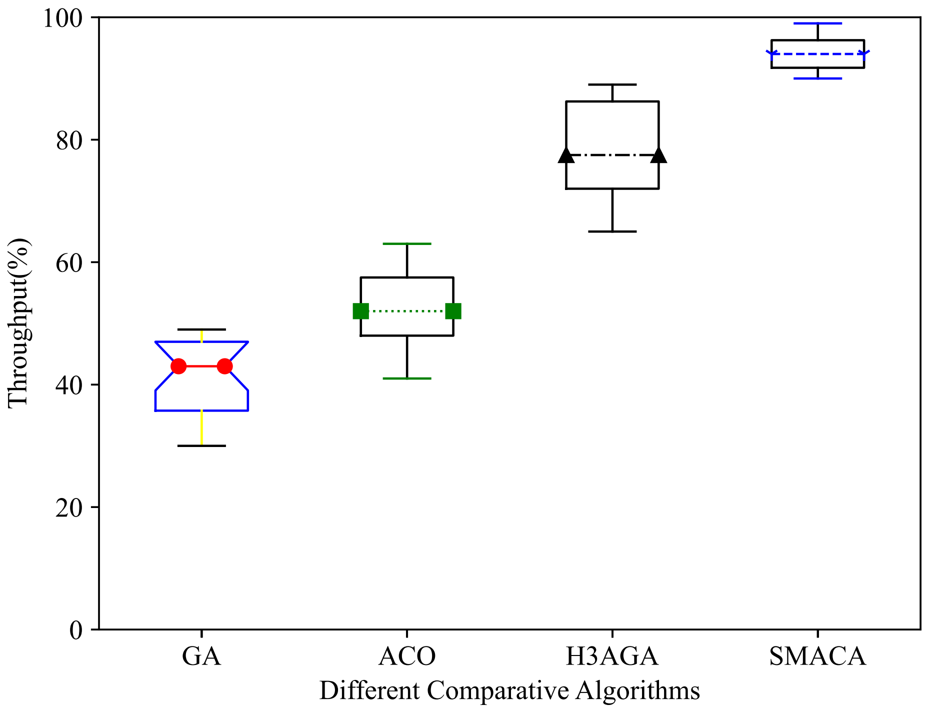

Figure 10. Throughput comparative experiment.

Based on the experimental results mentioned above, SMACA demonstrates superior performance in the adaptive migration of large-scale distributed resources.

6. Conclusions

This paper explores the application of adaptive dynamic migration in large-scale distributed resources, aiming to overcome current research limitations. The principal contribution lies in the formulation of a mathematical model for a resource scheduling problem in distributed edge computing. The computational complexity of this problem is analyzed, providing evidence that it is NP-Hard and applicable to various scenarios. To tackle this challenge, the paper proposes a solution algorithm based on ACO, complemented by an optimization method using PSO. In the experimental section, the proposed methods are thoroughly assessed for performance, revealing that the algorithm introduced in this paper adeptly addresses the stated problem. A key advantage is the ability to achieve near-optimal results with a reduced number of iterations. Furthermore, the algorithm demonstrates robust stability, with minimal errors observed across multiple experiments. These findings suggest that the algorithm is well-suited for diverse scenarios.

The paper primarily focuses on mathematical modeling and complexity analysis for complex engineering problems. While the algorithm utilizes basic PSO to enhance ACO, improvements in ACO steps are made to address large-scale distributed resource scheduling. However, optimal solution guarantees remain a challenge, and future work should explore algorithm design enhancements for optimal solutions.

Author Contributions: Writing—original draft preparation, A.T. and Y.L.; writing—review and editing, Y.W.; data curation, Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by National Key R&D Program of China (2019YFB1406002).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Stray, V.; Moe, N.B.; Vedal, H.; Berntzen, M. Using objectives and key results (OKRs) and slack: A case study of coordination in large-scale distributed agile. *TechRxiv* **2021**. [CrossRef]
2. Kang, P.; Deng, H.; Wang, X. Research on Multi-Equipment Collaborative Scheduling Algorithm under Composite Constraints. *Processes* **2022**, *10*, 1171. [CrossRef]
3. Deshmukh, S.; Thirupathi Rao, K.; Shabaz, M. Collaborative learning based straggler prevention in large-scale distributed computing framework. *Secur. Commun. Netw.* **2021**, *2021*, 8340925. [CrossRef]
4. Jia, Y.H.; Chen, W.N.; Gu, T.; Zhang, H.; Yuan, H.Q.; Kwong, S.; Zhang, J. Distributed cooperative co-evolution with adaptive computing resource allocation for large scale optimization. *IEEE Trans. Evol. Comput.* **2018**, *23*, 188–202. [CrossRef]
5. Askarizade Haghighi, M.; Maeen, M.; Haghparast, M. An energy-efficient dynamic resource management approach based on clustering and meta-heuristic algorithms in cloud computing IaaS platforms. *Wirel. Pers. Commun.* **2019**, *104*, 1367–1391. [CrossRef]
6. Yuan, M.; Li, Y.; Zhang, L.; Pei, F. Research on intelligent workshop resource scheduling method based on improved NSGA-II algorithm. *Robot. Comput.-Integr. Manuf.* **2021**, *71*, 102141. [CrossRef]
7. Zhang, R.; Shi, W. Research on workflow task scheduling strategy in edge computer environment. *J. Phys. Conf. Ser.* **2021**, *1744*, 032215. [CrossRef]
8. Rjoub, G.; Bentahar, J.; Abdel Wahab, O.; Saleh Bataineh, A. Deep and reinforcement learning for automated task scheduling in large-scale cloud computing systems. *Concurr. Comput. Pract. Exp.* **2021**, *33*, e5919. [CrossRef]
9. Li, B.; Pang, R.; Sainath, T.N.; Gulati, A.; Zhang, Y.; Qin, J.; Haghani, P.; Huang, W.R.; Ma, M.; Bai, J. Scaling end-to-end models for large-scale multilingual ASR. In Proceedings of the 2021 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Cartagena, Colombia, 13–17 December 2021; pp. 1011–1018.
10. Muhtadi, A.; Pandit, D.; Nguyen, N.; Mitra, J. Distributed energy resources based microgrid: Review of architecture, control, and reliability. *IEEE Trans. Ind. Appl.* **2021**, *57*, 2223–2235. [CrossRef]
11. Liu, S.; Hennequin, S.; Roy, D. Enterprise Platform of Logistics Services Based on a Multi-Agents Mechanism and Blockchains. *IFAC-PapersOnLine* **2021**, *54*, 825–830. [CrossRef]

12. Liu, L.; Feng, J.; Mu, X.; Pei, Q.; Lan, D.; Xiao, M. Asynchronous Deep Reinforcement Learning for Collaborative Task Computing and On-Demand Resource Allocation in Vehicular Edge Computing. *IEEE Trans. Intell. Transp. Syst.* **2023**, 1–14. [CrossRef]
13. Masdari, M.; Gharehpasha, S.; Ghobaei-Arani, M.; Ghasemi, V. Bio-inspired virtual machine placement schemes in cloud computing environment: taxonomy, review, and future research directions. *Clust. Comput.* **2020**, *23*, 2533–2563. [CrossRef]
14. Sheng, J.; Hu, Y.; Zhou, W.; Zhu, L.; Jin, B.; Wang, J.; Wang, X. Learning to schedule multi-NUMA virtual machines via reinforcement learning. *Pattern Recognit.* **2022**, *121*, 108254. [CrossRef]
15. Goudarzi, S.; Soleymani, S.A.; Wang, W.; Xiao, P. UAV-Enabled Mobile Edge Computing for Resource Allocation Using Cooperative Evolutionary Computation. *IEEE Trans. Aerosp. Electron. Syst.* **2023**, *59*, 5134–5147. [CrossRef]
16. Guo, D.; Olesen, J.E.; Manevski, K.; Ma, X. Optimizing irrigation schedule in a large agricultural region under different hydrologic scenarios. *Agric. Water Manag.* **2021**, *245*, 106575. [CrossRef]
17. Cui, L.; Liu, X.; Lu, S.; Jia, Z. A variable neighborhood search approach for the resource-constrained multi-project collaborative scheduling problem. *Appl. Soft Comput.* **2021**, *107*, 107480. [CrossRef]
18. Wu, Y.; Sun, X. Optimization and simulation of enterprise management resource scheduling based on the radial basis function (RBF) neural network. *Comput. Intell. Neurosci.* **2021**, *2021*, 9754050. [CrossRef]
19. Yuan, M.; Cai, X.; Zhou, Z.; Sun, C.; Gu, W.; Huang, J. Dynamic service resources scheduling method in cloud manufacturing environment. *Int. J. Prod. Res.* **2021**, *59*, 542–559. [CrossRef]
20. Wang, Z.J.; Zhan, Z.H.; Yu, W.J.; Lin, Y.; Zhang, J.; Gu, T.L.; Zhang, J. Dynamic group learning distributed particle swarm optimization for large-scale optimization and its application in cloud workflow scheduling. *IEEE Trans. Cybern.* **2019**, *50*, 2715–2729. [CrossRef]
21. Jiang, F.; Dong, L.; Wang, K.; Yang, K.; Pan, C. Distributed resource scheduling for large-scale MEC systems: A multiagent ensemble deep reinforcement learning with imitation acceleration. *IEEE Internet Things J.* **2021**, *9*, 6597–6610. [CrossRef]
22. Envelope, M.; Envelope, M.; Envelope, M. Task offloading using GPU-based particle swarm optimization for high-performance vehicular edge computing. *J. King Saud Univ. Comput. Inf. Sci.* **2022**, *34*, 10356–10364.
23. Zhang, X.; Du, K.J.; Zhan, Z.H.; Kwong, S.; Zhang, J. Cooperative Coevolutionary Bare-Bones Particle Swarm Optimization With Function Independent Decomposition for Large-Scale Supply Chain Network Design With Uncertainties. *IEEE Trans. Cybern.* **2019**, *50*, 4454–4468. [CrossRef] [PubMed]
24. Sajid, M.; Mittal, H.; Pare, S.; Prasad, M. Routing and scheduling optimization for UAV assisted delivery system: A hybrid approach. *Appl. Soft Comput.* **2022**, *126*, 109225. [CrossRef]
25. Jahic, A.; Plenz, M.; Eskander, M.; Schulz, D. Route scheduling for centralized electric bus depots. *IEEE Open J. Intell. Transp. Syst.* **2021**, *2*, 149–159. [CrossRef]
26. Li, H.; Wu, X.; Kun, P.K.; Hou U, L. Near-optimal fixed-route scheduling for crowdsourced transit system. In Proceedings of the 2021 IEEE 37th International Conference on Data Engineering (ICDE), Chania, Greece, 19–22 April 2021; pp. 2273–2278.
27. Mousa, M.H.; Hussein, M.K. Effcient UAV-Based MEC Using GPU-Based PSO and Voronoi Diagrams. *Comput. Model. Eng. Sci.* **2022**, *133*, 413–434.
28. Farahbakhsh, F.; Shahidinejad, A.; Ghobaei-Arani, M. Multiuser context-aware computation offloading in mobile edge computing based on Bayesian learning automata. *Trans. Emerg. Telecommun. Technol.* **2021**, *32*, e4127. [CrossRef]
29. Agiwal, M.; Kwon, H.; Park, S.; Jin, H. A survey on 4G-5G dual connectivity: road to 5G implementation. *IEEE Access* **2021**, *9*, 16193–16210. [CrossRef]
30. Son, J.; Buyya, R. Latency-aware virtualized network function provisioning for distributed edge clouds. *J. Syst. Softw.* **2019**, *152*, 24–31. [CrossRef]
31. Zhao, Z.; Barijough, K.M.; Gerstlauer, A. Deepthings: Distributed adaptive deep learning inference on resource-constrained iot edge clusters. *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.* **2018**, *37*, 2348–2359. [CrossRef]
32. Hu, J.; Liu, X.; Shahidehpour, M.; Xia, S. Optimal operation of energy hubs with large-scale distributed energy resources for distribution network congestion management. *IEEE Trans. Sustain. Energy* **2021**, *12*, 1755–1765. [CrossRef]
33. Zhang, Y.; Wu, J.; Liu, M.; Tan, A. TSN-based routing and scheduling scheme for Industrial Internet of Things in underground mining. *Eng. Appl. Artif. Intell.* **2022**, *115*, 105314. [CrossRef]
34. Peng, Y.; Ning, Z.; Tan, A.; Wang, S.; Obaidat, M.S. A Delay-Sensitive Multibase-Station Multichannel Access System for Smart Factory. *IEEE Syst. J.* **2022**, *17*, 188–199. [CrossRef]
35. Hartmanis, J. Computers and intractability: a guide to the theory of np-completeness (michael r. garey and david s. johnson). *Siam Rev.* **1982**, *24*, 90. [CrossRef]
36. Jian, J.R.; Chen, Z.G.; Zhan, Z.H.; Zhang, J. Region Encoding Helps Evolutionary Computation Evolve Faster: A New Solution Encoding Scheme in Particle Swarm for Large-Scale Optimization. *IEEE Trans. Evol. Comput.* **2021**, *25*, 779–793. [CrossRef]
37. Xia, X.; Gui, L.; Yu, F.; Wu, H.; Zhan, Z.H. Triple Archives Particle Swarm Optimization. *IEEE Trans. Cybern.* **2019**, *50*, 4862–4875. [CrossRef] [PubMed]
38. Li, J.Y.; Zhan, Z.H.; Liu, R.D.; Wang, C.; Zhang, J. Generation-Level Parallelism for Evolutionary Computation: A Pipeline-Based Parallel Particle Swarm Optimization. *IEEE Trans. Cybern.* **2020**, *51*, 4848–4859. [CrossRef] [PubMed]
39. Edmonds, J. Matroids and the greedy algorithm. *Math. Program.* **1971**, *1*, 127–136. [CrossRef]

40. Dorigo, M.; Birattari, M.; Stutzle, T. Ant colony optimization. *IEEE Comput. Intell. Mag.* **2006**, *1*, 28–39. [CrossRef]
41. Whitley, D. A genetic algorithm tutorial. *Stat. Comput.* **1994**, *4*, 65–85. [CrossRef]
42. Liu, J.; Xu, S.; Zhang, F.; Wang, L. A hybrid genetic-ant colony optimization algorithm for the optimal path selection. *Intell. Autom. Soft Comput.* **2017**, *23*, 235–242. [CrossRef]