Egyptian Informatics Journal 24 (2023) 100418

Contents lists available at ScienceDirect

Egyptian Informatics Journal

journal homepage: www.sciencedirect.com

Full Length Article

Workflow migration in uncertain edge computing environments based on interval many-objective evolutionary algorithm

Zhenyu Shi, Tianhao Zhao, Qi Li, Zhixia Zhang, Zhihua Cui *

Shanxi Key Laboratory of Big Data Analysis and Parallel Computing, Taiyuan University of Science and Technology, Taiyuan, 030024, Shanxi, China

ARTICLE INFO

Keywords:
Edge computing
Workflow migration
Interval many-objective optimization
Evolutionary algorithm

ABSTRACT

In edge computing (EC), when the edge server (ES) is processing tasks delivered by the mobile devices (MDs), the MDs move outside the coverage of the ES, where task migration is required to ensure service continuity. Most current research on task migration ignores inter-task dependencies and uncertain computing environments, and it focuses mainly on migration scenarios where MDs have a one-to-one or many-to-one relationship with ESs. Aiming at the problem of workflow migration with multi-MDs and multi-ESs in uncertain environments, this paper proposes an interval many-objective optimized workflow migration in uncertain environments (I-MaOWMUE) model that considers transforming uncertainty factors into interval parameters for processing, along with the migration delay, maximum completion time, energy consumption, and load balancing as an objective function, and at the same time, utilize real-time priority scheduling strategies to achieve the fast response of the tasks. Considering the dependency of tasks and the changing characteristics of ES load in a migration environment, this paper designs a migration-based interval many-objective evolutionary algorithm (MI-MaOEA), which adopts an interval confidence strategy to improve algorithm convergence and formulates an objective-value-dominated hierarchical sorting and dual-migration selection strategy based on the migration delay and the success rate of the migration to improve the diversity of the populations. Simulation results show that MI-MaOEA optimizes 27%, 35%, 14%, and 80% in solving the four objective values of I-MaOWMUE, and enables the solution to have faster converse speed and better distribution.

1. Introduction

Wireless network communications and intelligent sensory processing technologies are driving the swift development of the Internet of Things (IoT) [1], and IoT-based fine-processing applications, such as video data analytics and augmented/virtual reality (AR/VR), need to deal with the exploding number of tasks in a real-time manner [2]. However, MD's limited hardware resources and battery life make it unable to meet users' real-time requirements and provide them with overloaded services. EC as a new network computing paradigm pushes computing resources to the edge of the network [3], and resource-poor MDs can offload tasks to ES to relieve the computational pressure. In most EC scenes, it is usually assumed that the MD is at rest, when in fact the positional state changes of the MD are not to be ignored. Considering the limited service scope of ES, when the MD exceeds the coverage of ES, service interruption may occur, and to ensure the continuity of service, the unexecuted tasks need to be migrated to other available ESs within the coverage [4]. Existing task migration scenarios are mainly aimed at migrating tasks from single or multiple MDs to a single ES [5,6]. However, the computational resources of a single ES are still limited and cannot satisfy the computational requests of a large number of users, and collaborative computing (CC) can be adapted to unite multiple ESs to provide migration services [7]. In multi-MDs and multi-ESs migration scenarios, it is necessary to consider the switching of network connections, the control of migration latency, and the state information of server clusters in the new region, etc., and formulate a reasonable migration strategy for the originally unexecuted tasks to improve the user quality of experience (QoE).

In current computing migration scenarios, it is often overlooked that there are many uncertainties present, which include uncertain computational and network environments [8], all of which can have an impact on the processing efficiency of a task. However, in real applications, network congestion, excessive load leading to task execution failure, and

* Corresponding author.
E-mail addresses: shizy19990102@163.com (Z. Shi), zhaotianhao1015@163.com (T. Zhao), liqi2575564568@163.com (Q. Li), 15634969919@163.com (Z. Zhang), cuizhihua@gmail.com (Z. Cui).

https://doi.org/10.1016/j.eij.2023.100418
Received 20 August 2023; Received in revised form 13 October 2023; Accepted 5 November 2023
Available online 21 November 2023

other problems are unavoidable. In particular, when migrating multiple tasks at the same time, the increasing migration link resource contention and the unavoidable serial interference caused between links. Therefore, the uncertainty factor in the computational migration environment has an important impact on the efficiency and stability of task processing. In the vast majority of computing migration scenarios, scholars consider independent task migration, which somewhat weakens the dependency relationship and execution order between tasks. However, in practical application scenarios, tasks do not exist in isolation, and there are often mutual constraint relationships between tasks [9], such as medical diagnosis and treatment, logistics, and supply chain management. Therefore, we apply workflow to a computational migration scenario to achieve task passing and parallel processing between different Ess [10], which implies that task constraint relationships need to be considered comprehensively when making task migration decisions.

A lot of research has been done on task computation and resource allocation in EC today, aiming to formulate the computational problem as an optimization model to minimize objectives such as total delay, total energy consumption, and total cost [11,12]. However, most of them consider one optimization objective, explore the optimal solution in a one-dimensional decision space [13], or consider multiple objectives but weigh them to reduce the dimensionality, but this way of weighting with subjective preferences may lead to errors in the results [14]. In the actual task computation process, in the face of different user requirements, it is necessary to comprehensively consider the trade-offs between multiple objectives [15].

Multi-MDs and multi-ESs workflow migration problems in uncertain environments face two key challenges. On the one hand, migrating tasks from the original ES cluster to the target ES cluster requires comprehensive consideration of the dependencies and requirements of the tasks, the ES cluster resource status, energy consumption, and other factors, as well as uncertainties such as network congestion and ES resource availability, etc. Through real-time detection and analysis, in complex migration scenarios, a rational migration strategy is developed to optimize multiple objectives so that service requests from multiple MDs can be executed in parallel and efficiently and to ensure effective resource utilization. On the other hand, each ES needs to handle tasks uploaded by users in the region and tasks migrated from neighboring regions due to user location changes. Although ESs release a certain amount of storage space through computation, unreasonable migration strategies may lead to the overloading of some ESs and even excessive task lag time. For ES, due to untimely processing, there may be excessive loads resulting in data loss or even system crashes. To ensure that tasks are processed promptly and reduce additional delay expenses, it is necessary to comprehensively consider the dependence of the task, load changes, and storage limitations, according to the emergency degree of the task to formulate a real-time priority scheduling strategy, which is necessary to determine the priority of the task execution relationship, thereby improving user satisfaction, so that the EC system is efficiently executed.

At present, interval many-objective evolutionary algorithms (I-MaOEAs) can effectively solve uncertain many-objective optimization problems (U-MaOPs). In this paper, the above problem is modeled as an I-MaOWMUE model, and MI-MaOEA is proposed for this model, the algorithm adopts the interval confidence strategy to solve the uncertainty distress and optimize the interval many-objective, and at the same time, it weighs the consideration of ES load situation and inter-task dependency in the migration environment, and takes the migration delay and the success rate of the migration as the basis of the selection to formulate the objective-value-dominated hierarchical sorting and dual-migration selection strategies, so that these selected vectors are uniformly distributed on the Pareto Frontier (PF), thus providing the decision makers with appropriate migration strategies. The main contributions of this paper are as follows:

- In this paper, we discuss the multi-MDs and multi-ESs workflow migration problem in uncertain environments and propose the MaOWMUE model, where we consider uncertainties such as network bandwidth and server computing power to be transformed into interval parameters, and migration delay, maximum completion time, energy consumption, and load balancing as optimization objectives.
- To achieve a rational allocation of resources and rapid response to tasks, a real-time priority scheduling strategy is proposed, which uses information from real-time surveillance based on task dependencies and characteristics to assign different task priorities and degrees of urgency.
- To solve the I-MaOWMUE model, this paper designs MI-MaOEA, which adopts an interval confidence strategy to improve the convergence of the algorithm and proposes objective-value-dominated hierarchical sorting and dual-migration selection strategies to increase the diversity of the population.
- Simulation results show that MI-MaOEA optimizes the objective of the I-MaOWMUE model much better compared to other algorithms, and the solution set has a significant advantage over other algorithms in terms of mean, maximum, and minimum values.

The rest of the paper is organized as follows. In section 2, related work is presented. In section 3, the I-MaOWMUE model is constructed. In section 4, MI-MaOEA is proposed to solve the I-MaOWMUE model. In section 5, simulation experiments are performed and the results are summarised and analyzed. In section 6, the paper is summarised and conclusions are drawn.

2. Related work

MD transfers the generated tasks to ES execution by means of computational offloading, exploiting terminal-edge collaboration to improve QoE. Zakaryia et al. [16] consider offloading tasks from mobile devices to cloudlets for execution and achieving efficient offloading of tasks through the strategy of the queuing networks and an evolutionary algorithm. Liu et al. [17] acquire energy through hybrid access points (HAPs) and choose to execute the task locally or offload it to a single fog/cloud server for execution. They propose the Generalized Bending Decomposition (GBD) method to maximize the minimum energy balance among users. However, the limited resources of a single server are not enough to meet the real-time demands of MD. To this end, Do-Duy et al. [18] considered multiple MDs offloading tasks to multiple ESs to reduce latency in the presence of limited computational and service resources. Ding et al. [19] proposed two computing architectures, hierarchical end-edge-cloud computing (Hi-EECC) and horizontal end-edge-cloud computing (Ho-EECC), and proposed two potential game algorithms based on this architecture. The above computational approaches consider terminal-edge and edge-edge collaboration for computational tasks while developing suitable computational strategies to improve the overall effectiveness of the system. However, these studies consider the state of the MD to be stationary and do not take into account the effect of changes in the user's location on the computational decision.

In fact, the MD may change its position during EC computation, and to ensure service continuity, the unexecuted tasks need to be migrated to realize the edge-edge collaborative computation. Kim et al. [20] offload the user tasks to a nearby ES, and as the user moves, the tasks are migrated, and heuristics are proposed to solve it to reduce the computational cost and service latency. Similarly, in [21], vehicles carrying 6g network in boxes (NIBs) can communicate with other NIBs in real time to reduce the energy consumption and cost incurred during the service migration process through the NIB task migration method (NTM) and to develop a suitable migration strategy through the strength Pareto evolutionary algorithm (SPEA2). However, the above studies ignore the impact of environmental changes on the computational results during the task computation process, and at the same time, they focus on in-

dividual tasks from time to time in computational migration scenarios while ignoring the characteristics of inter-task constraint relationships.

In real EC systems, uncertain computing and network environments will affect the processing efficiency of tasks. In [22], a constraint mechanism is proposed to cope with the uncertainty in the processing cycle of the task and minimize the energy consumption while satisfying the deadlines and designing an online selection scheme to solve the problem. In [23], the channel and ES statistical characteristics are constantly changing, and by sequentially selecting ESs and using historical time and energy consumption to make new offloading decisions. Xu et al. [24] address uncertainty issues such as resource competition and link outages in the Internet of Vehicles (IoV), for which a software-defined network-based service management framework for IoV is proposed. Therefore, considering the uncertainty factor in the migration environment, it is more in line with practical computing scenarios.

In practical applications, tasks do not exist independently and need to be endowed with task states, attributes, and user requirements [25,26]. In [27], the focus is on the division of tasks for different application types and the development of suitable offloading strategies for joint optimization. He et al. [28] designed a hybrid task offloading problem with hard and soft deadlines and proposed the CONFECT offloading method to handle it. Sun et al. [29] considered the limited capacity of MD and the dependencies between tasks, proposed a series of task allocation strategies for different types of tasks in complex network environments, and found feasible solutions that satisfied the constraints. Huang et al. [30] formulated risk-constrained workflow scheduling as a Markov Decision Process (MDP) and designed a reinforcement learning-based security-aware workflow scheduling (SAWS) scheme. The study of task flows with constraint relationships is of practical interest by considering the states of the tasks and the strong connections between them. However, the above studies have not developed reasonable computation strategies and resource allocation strategies for the characteristics of dependency tasks to improve the overall system efficiency.

Nowadays, multi-objective evolutionary algorithms can effectively solve multi-objective optimization problems [31–35]. However, for many-objective optimization problems, the scale of the problem increases as the number of objectives increases, and the complexity of searching for and evaluating the solution increases, which requires an appropriate many-objective evolutionary algorithm to solve the problem [36–38]. Cui et al. [39] propose a many-objective evolutionary algorithm based on three-way decision (MaOEA-TWD) to solve the problem of convergence and diversity conflict as the number of objectives increases. Bozorgchenani et al. [40] consider that in mobile edge computing (MEC) and fog computing (FC), the task is offloaded from the client to the ES or other clients to minimize latency and energy consumption. For this purpose, the problem is modeled as a constrained multi-objective optimization problem (CMOP), and an evolutionary algorithm is designed to solve it. However, uncertain optimization problems often place higher demands on the solution capabilities of traditional many-objective evolutionary algorithms, which makes it difficult to obtain satisfactory solutions.

There is a growing tendency among scholars to adopt methods such as random, fuzzy, and interval variables to deal with multi-objective optimization problems that are fraught with uncertainty [41,42]. In practical scenarios, determining the exact probability distribution of a random variable or the exact affiliation function of a fuzzy number is often challenging, whereas upper and lower bounds, or midpoints and radii of interval parameters, are relatively easier to obtain and provide more reliable information. Because of this, the use of interval-based approaches in uncertain optimization problems has become highly sought-after due to their ability to better solve complex problems in practice. To solve the interval multi-objective optimization problem, Jin et al. [43] proposed a decomposition-based interval multi-objective evolutionary algorithm with adaptive adjustment of weight vectors and neighborhoods. He et al. [44] proposed a multi-objective interval evolutionary algorithm for knee joint decision-making for a multi-objective optimization problem with concurrent risk-benefit. Therefore, I-MaOEAs can be used to solve U-MaOPs.

In this paper, we consider a computational migration scenario where the user location changes, in which multiple MDs are located in the overlapping region of multiple BSs. When there is a user location change, the unexecuted tasks need to be migrated to multiple servers within the coverage area based on the system state. In addition, this paper considers the dependent task computation problem in uncertain environments, such as network bandwidth and server computing power, and models it as the I-MaOWMUE model to optimize the four objectives of migration delay, maximum completion time, energy consumption, and load balancing. Meanwhile, a real-time priority scheduling strategy is formulated for workflow characteristics to achieve reasonable resource allocation and fast task response. To solve the I-MaOWMUE model, MI-MaOEA is designed in this paper, which solves the uncertainty-troubling problem and ensures the convergence and diversity of the solution.

3. The proposed I-MaOWMUE model

3.1. System model

As shown in Fig. 1, we consider a multi-area EC system consisting of multiple base stations (BAs) and multiple MDs (e.g., computers, mobile phones, tablets, etc.). We denote by $BM = \{1, 2, ..., m, ..., M\}$ the set of base stations and $MU = \{1, 2, ..., u, ..., U\}$ the set of users. Multiple BAs and multiple MDs are randomly distributed within each area, and each base station is equipped with an ES. Each ES has different state information, which is denoted by SI, i.e., $SI = (cp, ss, co)$, cp, ss, and co denote the computing power, storage space, and coverage of the ES, respectively. Each MD generates multiple task requests and offloads tasks to the ES cluster within the co to provide computing services to it. Different link connections (e.g., wifi and 5G) are used for communication between users and base stations (MD-BA) and between base stations and base stations (BA-BA).

In this paper, we focus on EC scenarios for workflow migration in uncertain environments, and we will introduce the workflow model in Section 3.2, the migration model in Section 3.3, and the optimization objective in Section 3.4.

3.2. Workflow model

Multiple MDs within the coverage area of each ES locally generate $K(K \geq 0)$ heterogeneous workflows $W = \{w_1, w_2, ..., w_k, ..., w_K\}$. The heterogeneity of the workflows is mainly reflected in the different dependencies and number of tasks of the workflows. As shown in Fig. 2, usually w_k consists of a set of tasks and dependencies between them, which we represent by a directed acyclic graph (DAG). For each w_k, it consists of the quintuple $D = (T_k^c, Pre_k, Su_k, N_k, PS_k)$, where T_k^c denotes the completion time of the terminal node of the workflow w_k, which represents the completion of the whole workflow, Pre_k denotes the set of predecessor nodes of the workflow w_k, Su_k denotes the set of successor nodes of the workflow w_k, N_k denotes the number of tasks of the workflow w_k, and PS_k denotes the strong constraint relationship between the two sets Pre_k and Su_k. For each workflow, there are often many-to-many dependencies between tasks. There may exist one or more predecessor nodes for each child node except the start node, and one or more successor nodes for each child node except the terminal node. For each subtask in the workflow w_k, it consists of the quintuple $WT = (w_{k,n}, us_{k,n}, act_{k,n}, \chi_{k,n}^m, \omega_{k,n})$, where $w_{k,n}$ denotes the nth task of the workflow w_k, $us_{k,n}$ denotes the upload size of the task $w_{k,n}$, $act_{k,n}$ denotes the activation time of the task $w_{k,n}$, except for the start node, and the task is activated when and only when the execution of all the predecessor nodes is completed, $\chi_{k,n}^m$ denotes the priority level of the task $w_{k,n}$ in the ES m, and $\omega_{k,n}$ denotes the degree of urgency of the task $w_{k,n}$.

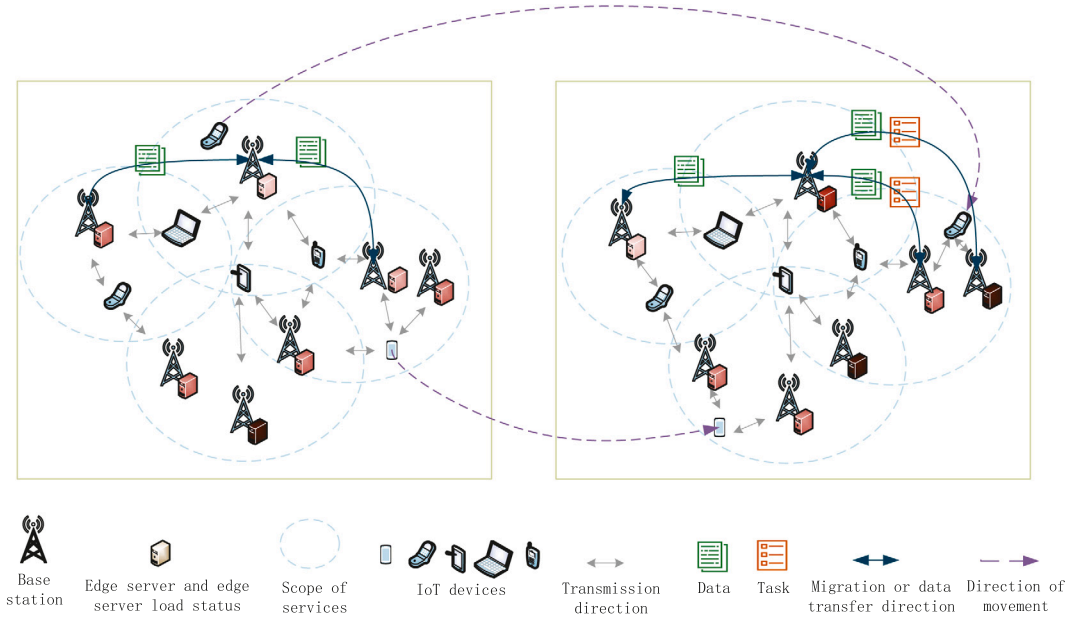

Fig. 1. Migration model framework.

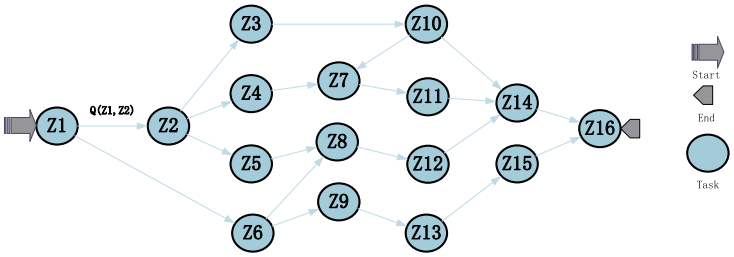

Fig. 2. An example of a workflow.

3.3. Migration model

As shown in Fig. 1, the location state of the MD may change during the execution of the offloading task by the ES, and when its location is not in the co of the ES, the ES that originally served it cannot continue to serve it, and at the same time, considering the limitations of the server cp, a part of the tasks are not processed promptly during the offloading phase. To ensure the continuity of the service and to improve the QoE, the sub-tasks of each workflow that were not executed and partially executed but not completed by the original ES were migrated to different ESs using cross-area edge-edge collaborative computation. However, each ES is unable to support strong computational demand with limited cp due to receiving tasks offloaded within the co as well as tasks migrated from multiple areas, which may lead to overloading. A migration strategy can be utilized to select appropriate servers for migrating tasks that have not finished executing based on the current system state information, and the ES can also utilize real-time priority scheduling policies to allocate resources to tasks based on the degree of urgency of the task to meet the four objectives of migration delay, maximum completion time, energy consumption, and load balancing.

3.3.1. Queuing model

As shown in Fig. 3, to ensure that tasks in ES can be processed quickly, this paper formulates a corresponding real-time priority scheduling strategy based on the priority level $\chi_{k,n}^{m}$ and degree of urgency $\omega_{k,n}$ of the tasks, with the higher $\chi_{k,n}^{m}$ and the higher $\omega_{k,n}$ being executed first. In the migration phase, except for the start node, $act_{k,n}$ is redefined as the maximum value of the execution completion time of all the predecessor nodes and the time of task migration to ES, and an orderly sorting of the execution order of tasks in the server is performed. $act_{k,n}$ will be described in detail in Section 3.4 Maximum completion time. For each server m, the specific steps of the real-time priority scheduling strategy are as follows:

(1) Pre-processing: tasks that have not been executed by ES are placed in the set S_L. Some tasks are migrated, and the tasks that are migrated away need to be deleted from S_L and the migrated over tasks put into S_L;

(2) Assigning $\chi_{k,n}^{m}$ and ordering:

1) If there are tasks without predecessors in S_L, assign the highest priority level $\chi_{k,n}^{m}=\chi_{1}^{m}$, and put it into the set R_L. For tasks that match χ_{1}^{m}, compare the migration delay $T_{k,n}^{mig}$ as $\omega_{k,n}$ of these tasks, with the smaller $\omega_{k,n}$ is, the higher the order of execution, $\omega_{k,n}$ will be introduced in section 3.3.2;

2) If the server has a task in progress, assign that task a priority level $\chi_{k,n}^{m}=\chi_{2}^{m}$;

3) For tasks that have been partially executed and migrated and put into the set R_L, assign these tasks a priority level $\chi_{k,n}^{m}=\chi_{3}^{m}$, compare $act_{k,n}$ as their $\omega_{k,n}$, with the smaller $act_{k,n}$ is, the higher the order of execution;

4) For other tasks waiting to be executed, assign the lowest priority level $\chi_{k,n}^{m}=\chi_{4}^{m}$. Real-time monitoring of the server, if the server temporarily has no task or a task and task execution is completed, check whether there is a task in the set S_L at this time has been activated, if it exists, then put into the set R_L, and according to $act_{k,n}$ to sort, the smaller $act_{k,n}$, the higher the degree of urgency $\omega_{k,n}$, the higher the order of execution. Therefore, $\omega_{k,n}$ is calculated as follows:

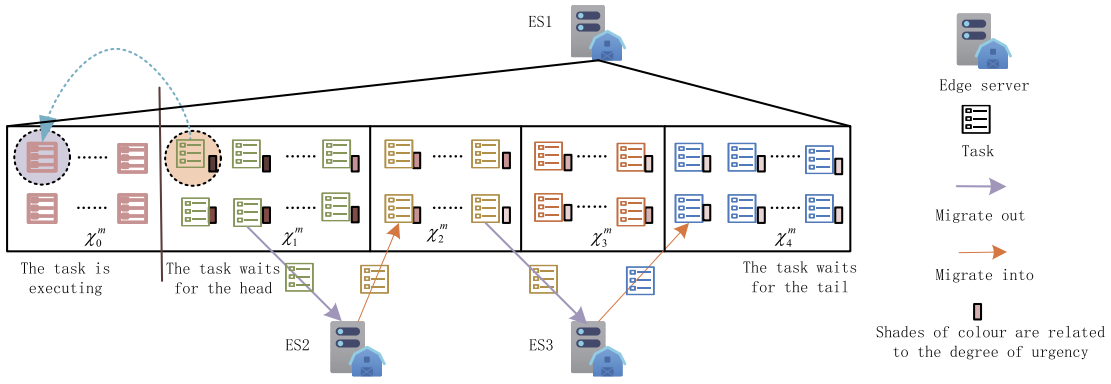

Fig. 3. Real-time priority scheduling model.

$$\omega_{k,n}=\begin{cases}\frac{1}{1+T_{k,n}^{mig}}, & \chi_{k,n}^m=\chi_1^m\\ 0, & \chi_{k,n}^m=\chi_2^m\\ \frac{1}{1+\mathrm{act}_{k,n}}, & \chi_{k,n}^m=\chi_3^m \text{ or } \chi_{k,n}^m=\chi_4^m\end{cases} \tag{1}$$

(3) Repeat step (2) until the task execution in server m is complete.

3.3.2. Migration cycle model

The life cycle of each $w_{k,n}$ in the migration process contains 3 or 4 phases, denoted as $T=\{T_1,T_2,T_3,T_4\}$ which denotes the migration phase, the service waiting phase, the execution phase, and the data transfer phase, respectively, where T_1 is optional, and this phase is ignored when the task is not migrating. If the server is in the process of processing user-delivered tasks and the user location has exceeded the *co* of the server, then the unexecuted and partially implemented but not completed tasks need to be migrated and the time spent is the migration delay $T_{k,n}^{mig}$, T_2 denotes the waiting delay $T_{k,n}^{wait}$ of task $w_{k,n}$ at the server, T_3 denotes the execution delay $T_{k,n}^{exe}$ of task $w_{k,n}$ at the server, and T_4 denotes the data transfer delay $T_{k,n}^{dt}$ used by task $w_{k,n}$ to transfer the generated data to the successor node after its execution. The completion time $T_{k,n}^{c}$ for each subtask $w_{k,n}$ for a complete workflow w_k is denoted as:

$$T_{k,n}^{c}=T_{k,n}^{mig}+T_{k,n}^{wait}+T_{k,n}^{exe}+T_{k,n}^{dt} \tag{2}$$

3.3.3. Uncertainty model

Uncertain communication link environments and unstable smart device execution capabilities will certainly affect the efficiency of task execution. Communications noise, communications interference, communications strength, etc. are the uncertainty factors affecting the transmission of task communication links when the intelligent device requests a relatively large number of tasks, the channel link resource contention and serial interference between the links lead to poor communication status, on the contrary, the communication link is more stable when there are fewer tasks grabbing resources. Bursty computational requests often lead to an insufficient supply of ES resources, resulting in ES corruption directly affecting the computational capability of the ES, which may lead to slower or even inoperable task processing. Therefore, we model it in the form of an interval to represent the uncertainty factors that affect the channel state and computational power. Since task $w_{k,n}$ cannot complete its computation in the server where it was originally offloaded, it needs to be migrated, and the bandwidth of the migrated link can be denoted as $\overline{B^{mig}}=[\overset{\vee}{B}^{mig},\hat{B}^{mig}]$ during the migration process. The data transfer bandwidth of the predecessor node to transfer the data to the server where the successor node is located after calculating the task is denoted as $\overline{B^{dt}}=[\overset{\vee}{B}^{dt},\hat{B}^{dt}]$. The number of CPU cycles per second that can be computed by server m is denoted as $\overline{R^m}=[\overset{\vee}{R}^m,\hat{R}^m]$. Since this paper focuses on the migration phase and does not consider the effect of uncertainties in the offloading phase on the migration phase, the transmission and computation process before migration is considered as a deterministic problem in this paper. The completion time $\overline{T_{k,n}^{c}}$ of task $w_{k,n}$ can be redefined as:

$$\overline{T_{k,n}^{c}}=[\overset{\vee}{T}_{k,n}^{c},\hat{T}_{k,n}^{c}]=\overline{T_{k,n}^{mig}}+\overline{T_{k,n}^{wait}}+\overline{T_{k,n}^{exe}}+\overline{T_{k,n}^{dt}} \tag{3}$$

where $\overset{\vee}{T}_{k,n}^{c}$ and $\hat{T}_{k,n}^{c}$ denote the lower and upper bounds on the completion time of task $w_{k,n}$, respectively.

3.4. Objective function

(1) Migration delay

The MD position changes during the process of calculating the uploaded tasks of local users by the ES, and when the MD position exceeds the co of the ES and the uploaded tasks are not processed in time, it is necessary to migrate the task $w_{k,n}$ and mark it with $\phi_{k,n}=1$, or $\phi_{k,n}=0$ otherwise. $T_{k,n}^{mig}$ for task $w_{k,n}$ is calculated as follows:

$$\overline{T_{k,n}^{mig}}=[\overset{\vee}{T}_{k,n}^{mig},\hat{T}_{k,n}^{mig}]=\frac{\phi_{k,n}\cdot TS_{k,n}\cdot D_{k,n}^{mm'}}{\overline{S_{k,n}^{mig}}} \tag{4}$$

$$\overline{S_{k,n}^{mig}}=[\overset{\vee}{S}_{k,n}^{mig},\hat{S}_{k,n}^{mig}]=\overline{B^{mig}}\cdot\log_2(1+\frac{p_{k,n}\cdot g_{k,n}}{\delta^2+p_{k,n}\cdot I}) \tag{5}$$

$$TS_{k,n}=\begin{cases}us_{k,n}, & \lambda_{k,n}=0\\ ms_{k,n}, & \lambda_{k,n}=1\end{cases} \tag{6}$$

$$\phi_{k,n}\in\{0,1\},\forall w_{k,n}\in w_k$$

$$\lambda_{k,n}\in\{0,1\},\forall w_{k,n}\in w_k$$

where $TS_{k,n}$ denotes the size of task $w_{k,n}$ in the migration phase; $D_{k,n}^{mm'}$ denotes the link migration distance between servers m and m'; $\overline{S_{k,n}^{mig}}$ denotes the migration speed between the servers serving the task before and after the migration; $p_{k,n}$ denotes the upload power of task $w_{k,n}$; $g_{k,n}$ denotes the communications gain of task $w_{k,n}$; δ denotes the communications noise; I denotes the communications interference; $\lambda_{k,n}$ denotes whether the task was partially executed before the migration but was not completed, yes then $\lambda_{k,n}=1$, otherwise $\lambda_{k,n}=0$; and $ms_{k,n}$ denotes task size of the remaining unexecuted portion of task $w_{k,n}$ tagged by the server m before the migration starts, which is calculated as follows:

$$ms_{k,n}=us_{k,n}\cdot(1-\frac{\tau-T_{k,n}^{r}}{T_{k,n}^{exe}}) \tag{7}$$

where τ denotes the time interval of the offloading phase, and $T_{k,n}^{r}$ denotes the response time of task $w_{k,n}$.

Objective $\overline{f_1}$ is to minimize the migration delay for all migration tasks:

$$\min \overline{f_1}=min[\overset{\vee}{f_1},\overset{\wedge}{f_1}]=\min\left\{\sum_{v=1}^{V}\sum_{a=1}^{A}\overline{T_{v,a}^{mig}}\right\} \tag{8}$$

(2) Maximum completion time

After the task $w_{k,n}$ is delivered to the ES, it is sorted according to the real-time priority scheduling strategy, and the task starts to execute when the task execution condition is satisfied, and the execution delay $\overline{T_{k,n}^{exe}}$ of task $w_{k,n}$ is denoted as:

$$\overline{T_{k,n}^{exe}}=[\overset{\vee}{T_{k,n}^{exe}},\overset{\wedge}{T_{k,n}^{exe}}]=\begin{cases}\dfrac{TS_{k,n}\cdot \mathrm{cc}_{k,n}^{m}}{\overline{R_{k,n}^{m}}}, & \phi_{k,n}=0\\ \dfrac{TS_{k,n}\cdot \mathrm{cc}_{k,n}^{m'}}{\overline{R_{k,n}^{m'}}}, & \phi_{k,n}=1\end{cases} \tag{9}$$

$$\overset{\wedge}{R_{k,n}^{m}}\le R_{k,n}^{m,\max},\forall w_{k,n}\in w_k$$

$$\overset{\wedge}{R_{k,n}^{m'}}\le R_{k,n}^{m',\max},\forall w_{k,n}\in w_k$$

where $\mathrm{cc}_{k,n}^{m}$ denotes the CPU cycles required by server m to compute task $w_{k,n}$; $\overline{R_{k,n}^{m}}$ denotes the number of CPU cycles per second that can be computed by the server m on which task $w_{k,n}$ resides; and $\overline{R_{k,n}^{m'}}$ denotes the number of CPU cycles per second that can be computed by the server m' on which task $w_{k,n}$ resides.

In the workflow w_k, for each task $w_{k,n}$ with a successor node, the resulting data is transferred to each subtask of its successor set $Su_{k,n}$ after the task execution, $\overline{T_{k,n}^{dt}}$ denoted by:

$$\overline{T_{k,n}^{dt}}=[\overset{\vee}{T_{k,n}^{dt}},\overset{\wedge}{T_{k,n}^{dt}}]=\rho_{k,n}\cdot\sum_{su=1}^{Su_{k,n}}\overline{DT_{k,n}^{su}} \tag{10}$$

$$\overline{DT_{k,n}^{su}}=[\overset{\vee}{DT_{k,n}^{su}},\overset{\wedge}{DT_{k,n}^{su}}]=\frac{v_{k,n}^{su}\cdot D_{k,n}^{mm',su}\cdot\sum_{h=1}^{H}TDS_{k,n}^{su,h}}{\overline{S_{k,n}^{dt}}} \tag{11}$$

$$\overline{S_{k,n}^{dt}}=[\overset{\vee}{S_{k,n}^{dt}},\overset{\wedge}{S_{k,n}^{dt}}]=\overline{B^{dt}}\cdot\log_2(1+\frac{p_{k,n}\cdot g_{k,n}}{\delta^2+p_{k,n}I}) \tag{12}$$

$$\rho_{k,n}\in\{0,1\},\forall w_{k,n}\in w_k$$

$$\kappa_{k,n}\in\{0,1\},\forall w_{k,n}\in w_k$$

$$v_{k,n}^{su}\in\{0,1\},PS_{k,n}^{su}\in PS_k$$

where $\rho_{k,n}$ denotes whether the task is a terminal task, $\rho_{k,n}=0$ denotes that the task is a terminal task, otherwise $\rho_{k,n}=1$; $\overline{DT_{k,n}^{su}}$ denotes the delay of data transmission to the server where the suth successor node is located; $v_{k,n}^{su}=0$ denotes that the predecessor and successor tasks are executed on the same server, otherwise $v_{k,n}^{su}=1$; $D_{k,n}^{mm',su}$ denotes the distance between the server where the predecessor task $w_{k,n}$ is located and the server where the corresponding zero or suth successor task is located; $TDS_{k,n}^{su,h}$ denotes the size of the hth data generated after the execution of the predecessor task $w_{k,n}$; and $\overline{S_{k,n}^{dt}}$ denotes the transmission rate of the link between the servers where the predecessor and successor tasks reside.

Since the start node has no predecessor task, it is always active i.e. $\overline{act_{k,n}}=0$, for other subtasks $w_{k,n}$ activation requires two conditions: (1) Ensure that the tasks have been migrated to the server at this point. (2) Since there are one or more predecessor tasks for $w_{k,n}$ and each predecessor task generates one or more pieces of data when it is completed, all predecessor tasks need to transfer data to the server where the successor task is located after completion of the task, with the above conditions, task $w_{k,n}$ can be activated, so except for the start node, the activation of the task time $\overline{act_{k,n}}$ is the migration delay and the maximum value of the last feedback time for all precursor tasks:

$$\overline{act_{k,n}}=[\overset{\vee}{act_{k,n}},\overset{\wedge}{act_{k,n}}]=ws_{k,n}\cdot\max\left\{\overline{T_{k,n}^{mig}},\max_{j\in Pre_{k,n}}\left\{\overline{T_{k,n}^{ect,j}}+\overline{DT_{k,n}^{j}}\right\}\right\} \tag{13}$$

$$ws_{k,n}\in\{0,1\},\forall w_{k,n}\in w_k$$

where $ws_{k,n}$ denotes whether task $w_{k,n}$ is the start node, if yes then $ws_{k,n}=0$, otherwise $ws_{k,n}=1$; $Pre_{k,n}$ denotes the set of predecessor tasks of task $w_{k,n}$; $\overline{T_{k,n}^{ect,j}}$ denotes the execution completion time of the jth predecessor task of task $w_{k,n}$; and $\overline{DT_{k,n}^{j}}$ denotes the data transfer time of the jth predecessor task to transfer data to task $w_{k,n}$.

Due to the limited co of the ES, when there is a large number of tasks in demand, it may not be able to serve them in time, the task arrives at the server even if it is activated may still have to wait in the server, the waiting time $\overline{T_{k,n}^{wait}}$ for task $w_{k,n}$ can be calculated as:

$$\overline{T_{k,n}^{wait}}=[\overset{\vee}{T_{k,n}^{wait}},\overset{\wedge}{T_{k,n}^{wait}}]=\overline{T_{k,n}^{r}}-\overline{act_{k,n}} \tag{14}$$

Therefore, for workflow w_k the maximum completion time is:

$$\overline{T_k^c}=[\overset{\vee}{T_k^c},\overset{\wedge}{T_k^c}]=\max_{o\in w_k}\overline{T_{k,o}^{c}} \tag{15}$$

Objective $\overline{f_2}$ is to minimize the average of the maximum completion times of all workflows W:

$$\min\overline{f_2}=min[\overset{\vee}{f_2},\overset{\wedge}{f_2}]=min\left\{\frac{1}{K}\cdot\sum_{k=1}^{K}\overline{T_k^c}\right\} \tag{16}$$

(3) Energy consumption

In be sustainable and use resources wisely, energy consumption is one of the key objectives considered in the EC environment. T_1 phase some tasks need to be migrated, the migration process task $w_{k,n}$ generates energy consumption for $\overline{E_{k,n}^{mig}}$, T_3 phase task $w_{k,n}$ generates execution energy consumption during its execution as $\overline{E_{k,n}^{exe}}$, T_4 phase task $w_{k,n}$ after the completion of the computation of the data generated by the transmission of the data to the successor task inevitably generates the data transmission energy consumption for $\overline{E_{k,n}^{dt}}$, for the total energy consumption for the task $w_{k,n}$ is:

$$\overline{E_{k,n}^{tot}}=[\overset{\vee}{E_{k,n}^{tot}},\overset{\wedge}{E_{k,n}^{tot}}]=\begin{cases}\overline{E_{k,n}^{exe}}+\overline{E_{k,n}^{dt}}, & \phi_{k,n}=0\\ \overline{E_{k,n}^{mig}}+\overline{E_{k,n}^{exe}}+\overline{E_{k,n}^{dt}}, & \phi_{k,n}=1\end{cases} \tag{17}$$

The energy consumption generated at each stage can be expressed as:

$$\overline{E_{k,n}^{mig}}=[\overset{\vee}{E_{k,n}^{mig}},\overset{\wedge}{E_{k,n}^{mig}}]=\overline{T_{k,n}^{mig}}\cdot P_{k,n}^{mig} \tag{18}$$

$$\overline{E_{k,n}^{dt}}=[\overset{\vee}{E_{k,n}^{dt}},\overset{\wedge}{E_{k,n}^{dt}}]=\overline{T_{k,n}^{dt}}\cdot p_{k,n}^{dt} \tag{19}$$

$$\overline{E_{k,n}^{exe}}=[\overset{\vee}{E_{k,n}^{exe}},\overset{\wedge}{E_{k,n}^{exe}}]=\begin{cases}\overline{T_{k,n}^{exe}}\cdot p_{k,n}^{m,exe}, & \phi_{k,n}=0 and\ \lambda_{k,n}=0\\ \overline{T_{k,n}^{exe}}\cdot p_{k,n}^{m',exe}, & \phi_{k,n}=1 and\ \lambda_{k,n}=0\\ [\tau-T_{k,n}^{r}]\cdot p_{k,n}^{m,exe}+\dfrac{ms_{k,n}\cdot \mathrm{cc}_{k,n}^{m'}}{\overline{R_{k,n}^{m'}}}\cdot p_{k,n}^{m',exe}, & \phi_{k,n}=1 and\ \lambda_{k,n}=1\end{cases} \tag{20}$$

where $P_{k,n}^{mig}$ denotes the power to migrate the task $w_{k,n}$; $p_{k,n}^{dt}$ denotes the power in which task $w_{k,n}$ transfers the data; $p_{k,n}^{m,exe}$ denotes the execution power of the server m to perform the task $w_{k,n}$; $p_{k,n}^{m',exe}$ denotes the execution power of the server m' to perform the task $w_{k,n}$.

Objective $\overline{f_3}$ is to minimize the total energy consumption:

$$\min\overline{f_3}=\min[\overset{\vee}{f_3},\overset{\wedge}{f_3}]=\min\sum_{k=1}^{K}\sum_{n=1}^{N}\overline{E_{k,n}^{tot}} \tag{21}$$

(4) Load balancing

The migration controller obtains the status information of each area server cluster, including the computing capacity of each server, the impact of the degree of completion of server tasks on load before migration, and joint consideration of the impact on server load of inter-area migration of tasks generated by users in surrounding areas, and chooses to migrate the various subtasks of the workflow to a suitable new server farm within MD's coverage area. To ensure efficient execution while balancing the workload and avoiding overloading to reduce operational efficiency, the specific calculation of load balancing $\overline{L}$ is as follows:

$$\overline{L} = [\overset{\vee}{L}, \hat{L}] = \frac{\sum_{m=1}^{M} [\overline{L_m} - \overline{L_a}]^2}{\overline{L_a}} \tag{22}$$

$$L_a = [\overset{\vee}{L_a}, \overset{\wedge}{L_a}] = \frac{\sum_{m=1}^{M} \overline{L_m}}{M} \tag{23}$$

$$0 \le \overset{\wedge}{L_m} \le ss_m, \forall m \in M$$

where $\overline{L_m}$ denotes the storage space occupied by server m during the migration phase and $\overline{L_a}$ denotes the average load of all ESs, calculated as follows:

$$\overline{L_m} = [\overset{\vee}{L_m}, \overset{\wedge}{L_m}] = \frac{L_m^{off} - L_m^c + L_m^{mt} - L_m^{ma}}{\overline{R_{k,n}^m}} \tag{24}$$

where L_m^{off} denotes the amount of tasks offloaded to server m; L_m^c denotes the amount of tasks that server m finished performing before migration; L_m^{mt} denotes the amount of tasks that have been migrated to server m from other areas; and L_m^{ma} denotes the amount of tasks that have been migrated away from server m. The calculation is as follows:

$$L_m^{off} = \sum_{g=1}^{GO_m} us_{m,g} \tag{25}$$

$$L_m^c = \sum_{g=1}^{GE_m} [\ell_{m,g} \cdot (us_{m,g} - ms_{m,g}) + (1 - \ell_{m,g}) \cdot us_{m,g}] \tag{26}$$

$$L_m^{mt} = \sum_{m'=1 and m' \ne m}^{M} \sum_{g=1}^{G_{m'm}} [\psi_{m',g} \cdot ms_{m',g} + (1 - \psi_{m',g}) \cdot us_{m',g}] \tag{27}$$

$$L_m^{ma} = \sum_{m'=1 and m' \ne m}^{M} \sum_{g=1}^{G_{mm'}} [\xi_{m',g} \cdot ms_{m',g} + (1 - \xi_{m',g}) \cdot us_{m',g}] \tag{28}$$

where GO_m denotes the number of tasks offloaded to server m; GE_m denotes the number of tasks executed by server m; $G_{m'm}$ denotes the number of tasks migrated from m' to m; $G_{mm'}$ denotes the number of tasks migrated from m to m'; $\ell_{m,g}$ denotes whether the task has been partially executed before the migration or not, respectively, and if yes, then $\ell_{m,g} = 1$, otherwise it means that the task execution is completed, at this time $\ell_{m,g} = 0$; and $\psi_{m',g}$ denotes whether the task has been partially executed before the migration or not, and if yes, then $\psi_{m',g} = 1$. Otherwise, it means that the task has not been executed yet, when $\psi_{m',g} = 0$. $\xi_{m',g}$ is consistent with the definition of $\psi_{m',g}$. Also needs to be satisfied:

$$\ell_{m,g} \in \{0,1\}, \forall g \in GE_m$$

$$\psi_{m',g} \in \{0,1\}, \forall g \in G_{m'm}$$

$$\xi_{m',g} \in \{0,1\}, \forall g \in G_{mm'}$$

Objective $\overline{f_4}$ is to minimize load balancing:

$$min\overline{f_4} = min[\overset{\vee}{f_4}, \overset{\wedge}{f_4}] = min\overline{L} \tag{29}$$

In this paper, the constructed I-MaOWMUE model is defined as a minimization optimization problem, defined as follows:

$$minF(X) = min\left\{\overline{f_1}, \overline{f_2}, \overline{f_3}, \overline{f_4}\right\} \tag{30}$$

4. Proposed MI-MaOEA

4.1. Algorithmic framework

In this paper, MI-MaOEA is proposed for solving the I-MaOWMUE model, and Algorithm 1 demonstrates the algorithmic framework of MI-MaOEA. First, the population P_t is randomly initialized. Then, the mating pool is constituted utilizing P_t. Similar to other evolutionary algorithms, the offspring Q_{tm} is generated by simulated binary crossover (SBX) and polynomial mutation (PM). Finally, P_t is merged with Q_{tm} to select the next-generation individuals through an objective-value-dominated hierarchical sorting and dual-migration selection strategy. The above steps are repeated until the maximum number of iterations is reached and MI-MaOEA ends.

Algorithm 1 MI-MaOEA

Input: Population size N, Maximum number of iterations t_{max}
Output: Last generation of populations: $P_{t_{max}}$
1: Initialize: $P_t = \{P_1, P_2, P_3, ..., P_N\}$
2: **for** $t \leftarrow 1$ to t_{max} **do**
3: P_{tm} = Mating selection(N, P_t) // Refer to Algorithm 2
4: Q_{tc} = SBX(N, P_t, P_{tm})
5: Q_{tm} = PM(N, Q_{tc}, P_{tm})
6: $Z = P_t \cup Q_{tm}$
7: P_{t+1} = Environmental selection(N,Z) // Refer to Algorithm 3
8: **end for**

4.2. Mating selection

Comparison of each individual through Pareto non-dominance sorting to determine the relationship between them. Diversity of solutions and flexibility in decision-making can be ensured by dividing candidate solutions into dominated and non-dominated solutions and obtaining relatively optimal non-dominated solutions. However, the above methods do not allow direct comparison of interval objective values, and in order to measure the quality of optimal solutions to interval many-objective optimization problems, this paper performs a comparison of dominance relationships by means of the interval confidence strategy.

For the interval values $\kappa_1 = [\overset{\vee}{\kappa_1}, \overset{\wedge}{\kappa_1}]$ and $\kappa_2 = [\overset{\vee}{\kappa_2}, \overset{\wedge}{\kappa_2}]$, we use $\kappa = [\overset{\vee}{\kappa}, \overset{\wedge}{\kappa}]$ as the minimum interval, where $\overset{\vee}{\kappa}$ and $\overset{\wedge}{\kappa}$ are the minimum and second smallest values of the $\overset{\vee}{\kappa_1}$, $\overset{\wedge}{\kappa_1}$, $\overset{\vee}{\kappa_2}$ and $\overset{\wedge}{\kappa_2}$, respectively, and we represent the confidence level of the intervals in which κ_1 is less than κ_2 as:

$$P(\kappa_1 < \kappa_2) = \frac{d(\kappa_2, \kappa)}{d(\kappa_1, \kappa) + d(\kappa_2, \kappa)} \tag{31}$$

where $d(\varepsilon_1, \varepsilon_2)$ denotes the distance from the objective value of the interval and $d(\varepsilon_1, \varepsilon_2)$ is calculated as follows:

$$d(\varepsilon_1, \varepsilon_2) = \sqrt{(\overset{\vee}{\varepsilon_1} - \overset{\vee}{\varepsilon_2})^2 + (\overset{\wedge}{\varepsilon_1} - \overset{\wedge}{\varepsilon_2})^2} \tag{32}$$

It is known from the interval confidence that the solution x_1 interval dominates the solution $x_2 (x_1 \prec x_2)$ if and only if the following conditions are met:

$$\begin{cases} \forall l \in \{1,2,3,...,\varpi\}, P(f_l(x_1,g) < f_l(x_2,g)) \ge 0.5 \\ \exists j \in \{1,2,3,...,\varpi\}, P(f_j(x_1,g) < f_j(x_2,g)) > 0.5 \end{cases} \tag{33}$$

where ϖ denotes the objective number.

In Algorithm 2 we give a matching selection method that randomly selects two solutions x_1, x_2 of the parent and compares their relations of domination: 1) if $x_1 \prec x_2$ add solution x_1 to the mating pool, 2) if $x_2 \prec x_1$ add solution x_2 to the mating pool, and 3) otherwise randomly select a solution to be added to the mating pool.

4.3. Environmental selection

Considering that the interval objective values cannot be compared between individuals in the same non-dominated layer, they can be com-

Algorithm 2 Mating Selection

Input: Population size N, Parent P_t
Output: mating pool P_{tm}

$P_{tm} = []$
for $i \leftarrow 1$ to N **do**
  x_1, x_2 = random select(P_t)
  if $x_1 \prec x_2$ **then**
    $P_{tm}.append(x_1)$
  else if $x_2 \prec x_1$ **then**
    $P_{tm}.append(x_2)$
  else
    s = random select(x_1,x_2)
    $p_{tm}.append(s)$
  end if
end for

pared using the interval crowding distance, which will be presented in Section 4.3.1. For the interval many-objective optimization problem of computational migration, this paper proposes a dual-migration selection strategy, which will be introduced in Section 4.3.2.

4.3.1. Interval crowding distance

Individuals in stratum F_h were sorted according to interval confidence to determine the relationship of individuals. In general, we measure the interval crowding distance in terms of the denseness of the distribution of the solution x to the two nearest individuals on either side of it, and we make use of the interval crowding distance O_{dis} for the selection, computed as follows:

$$O_{\text{dis}} = \begin{cases} \sum_{i=1}^{\varpi} \frac{d(f_i^{o+1}, f_i^{o-1})}{d(f_i^{\max}, f_i^{\min})}, & \Upsilon_O = 0 \\ +\infty, & \Upsilon_O = 1 \end{cases} \tag{34}$$

$$\Upsilon_O \in \{0,1\}, \forall O \in F_h$$

where f_i^{o+1} and f_i^{o-1} denote the ith objective value of the $O+1$th and $O-1$th solution of the Oth individual of the sorted set, respectively. $f_i^{\max}$ and $f_i^{\min}$ denote the maximum and minimum values of the ith objective function. $\Upsilon_O = 0$ means that the individual is not a boundary solution, otherwise $\Upsilon_O = 1$.

4.3.2. Dual-migration selection strategy

In the migration phase, the excessive migration delay of the task correspondingly increases the migration cost, and even though the server computing time is shortened, it is not enough to compensate for the additional consumption caused by the migration process, which is not permissible for the whole system. For ES, it needs to receive migration tasks from multiple neighboring regions at the same time. Even if ES releases part of the storage resources during the computation process, the limited server cp and ss cannot support a huge amount of task migration. The insufficient supply of computational capacity is likely to lead to server damage. To improve the overall effectiveness of the server, it is necessary to seek a suitable method to formulate a reasonable migration strategy. For this reason, we propose a dual-migration selection strategy. We chose to use individuals with small migration delays and low migration success as an important basis for selection at the final level. The small migration delays mean that a large number of tasks are selected to be migrated to the nearby servers, which significantly reduces task migration delay costs and energy consumption. Although the above approach makes the migration delay shorter, the servers near the users may be overloaded, which reduces the migration success rate, which is not the best solution for task migration. However, from another perspective, although the server is overburdened, there may be a situation where tasks of the same workflow are migrated to the same server, which in turn significantly reduces the data transfer delay and energy consumption, etc., and the servers, although in a state of imbalance for a short period, reduce the time for the completion of the workflow.

Specifically, in the last level of selection, to increase the search range of feasible solutions and ensure the diversity of solutions, we complete the selection according to the following steps: (1) Select $qq = \lfloor \gamma \cdot J \rfloor$ individuals based on interval crowding distance, where $\gamma(\gamma \geq 1)$ denotes the environmental factor for interval crowding distance selection in layer F_h. If qq is greater than the number of individuals in layer F_h, step 2 is performed, otherwise, step 3 is performed; (2) Use O_{dis} to sort the individuals in stratum F_h in descending order and select G individuals; (3) Sort the individuals of stratum F_h in descending order using O_{dis} and select qq individuals from them, and then select J individuals are then selected from them using S ascending order. We take the midpoint of $\overline{f_1}$ to provide a diversity of choice support. S is calculated as follows:

$$S = f_1 \cdot SR \tag{35}$$

$$SR = \frac{\sum_{m=1}^{M} \sum_{g=1}^{G_{mig}} \Gamma_{m,g} \cdot TS_{m,g}}{\sum_{m=1}^{M} \sum_{g=1}^{G_{mig}} TS_{m,g}} \cdot 100\% \tag{36}$$

where SR denotes the migration success rate and $\Gamma_{m,g}$ denotes whether the migration of the gth task to the mth server is successful or not, if the migration is successful then $\Gamma_{m,g} = 1$, otherwise $\Gamma_{m,g} = 0$.

Algorithm 3 gives the overall process of environmental selection. First, the Z individuals from the merger of the parent P_t and the offspring Q_{tm} are sorted according to the interval dominance, and divided into l layers, where layer 1 is the lowest and preferred to be selected, followed by layer 2, and so on. Then, individuals of size N are then selected layer by layer for the next generation. When the selected individuals belong to the same non-dominated stratum, we select individuals by interval crowding distance and dual-migration selection strategy as the selection basis.

Algorithm 3 Environment Selection

Input: Population size N, Parental generation P_t merges with paternal generation Q_{tm}: Z, $Y = 0$, $h = 1$, $A = []$, $B = []$, $P_{t+1} = []$
Output: The next generation of populations: P_{t+1}

$F = F_1, F_2, ..., F_h, ..., F$ = Non-dominated ordering descending(Z)
while $Y < N$ **do**
  $Y = Y \cup F_h$, $h = h + 1$
end while
$P_{t+1} = P_{t+1} \cup F_1, F_2, ..., F_{h-1}$, $Y = Y - F_h$
The number of individuals selected in the final layer: $J = N - Y$
$qq = \lfloor \gamma \cdot J \rfloor$
if $qq >$ length(F_h) **then**
  $G = J$
else
  $G = qq$
end if
for each individual in F_h **do**
  Sort O_{dis} by eq. (34) in descending order and put the result into A
end for
for $i \leftarrow 1$ to G **do**
  $B.append(A[i])$
end for
for each individual in B **do**
  Sort S by eq. (35) in ascending order and put the result into B
end for
for $i \leftarrow 1$ to J **do**
  $P_{t+1}.append(B[i])$
end for

5. Experiments

In order to evaluate the performance of MI-MaOEA on the I-MaOWMUE model, we need to conduct validation experiments and compare it with the MI-MaOEA algorithm using state-of-the-art interval multi-objective evolutionary algorithms, DI-μMOGA [45], InMaOEA

Table 1
Parameters of the MI-MaOEA algorithm.

Variable	Description	Value
N	Population size	100
M	Objective number	4
t_{max}	Maximum number of iterations	100
η	Distribution factor	20
p_m	Crossover probability	1.0
q_m	Mutation probability	0.01

[46], and II-MOEA [47]. DI-μMOGA proposes to use Monte Carlo stochastic simulation methods to seek optimization objective intervals and uses the degree of interval constraint violation to deal with constraints. InMaOEA uses this interval credibility strategy to improve the convergence of the algorithm and the interval congestion distance strategy to improve population diversity. II-MOEA is a classical method for solving U-MaOPs problems, which defines the dominance relation through the interval confidence level and the crowding distance through the location and volume of the hyper-cuboids.

5.1. Simulation environment and parameter settings

Simulation environment: Windows 11 Home Edition; AMD Ryzen 7 4800H with Radeon Graphics with 2.9 GHz; NVIDA GeForce GTX 1650 with 8 GB memory; MATLAB R2023a development platform.

Table 1 provides information about the parameter settings of the MI-MaOEA algorithms in this simulation experiment. To make the comparison of the algorithms fairer, for all the algorithms we use the same parameter settings, setting the population size to 100, the number of objectives to 4, and the maximal number of iterations to 100. In addition, the parameter setting sizes for η, p_m, and q_m are also listed in Table 1, and we will verify in Section 4.2 that the reasonableness of the parameter size settings, and the rest of the parameter size settings are based on the original literature. Table 2 gives the constraints on the important parameter settings for constructing the I-MaOWMUE model, and we randomly assign values to each parameter within the given range of values to apply to different migration scenarios and computational conditions.

We simulate task migration in real IoT environments using five differently structured benchmark workflows for discussion and evaluation of the performance of EC systems from well-known scientific applications [48], namely the Montage astronomy workflow, Epigenomics workflow, CyberShake earthquake hazard characterization workflow, SIPHT workflow that searches for small untranslated RNAs, and Inspiral physics workflow, which are widely used to evaluate the performance of workflow scheduling problems.

5.2. Results and analysis

The parameters of the proposed MI-MaOEA algorithm should be adjusted and evaluated according to the proposed I-MaOWMUE model to find the combinations that perform well on the I-MaOWMUE model. The setting of the values of the three important parameters of the distribution index η, the crossover probability p_m, and the mutation probability q_m in the MI-MaOEA algorithm directly affects the search efficiency and the quality of the solution. In general, a higher η increases the probability of the offspring approximating the parent, and the range of values is generally controlled between 5 and 20. Higher p_m increases population diversity, but occasionally over-convergence occurs, and the range of values was kept between 0.7 and 1.0. Higher q_m increases stochasticity, but unstable values may occur and keep the range of values between 0.001 and 0.01. For this reason, this paper conducts several sets of experiments to determine the most appropriate parameter settings so that the solution can satisfy faster convergence and better diversity.

Table 3 shows the impact of the MI-MaOEA algorithm on the four optimization objectives of migration delay, maximum completion time, energy consumption, and load balancing in the I-MaOWMUE model for different parameter value settings, and we compare the mean, maximum, and minimum values on the four optimization objectives respectively, where the average value is a clearer expression of the degree to which the algorithm is good or bad, and our goal is to minimize each objective function value, and the best results are shown in bold. By analyzing the table, it can be concluded that when the parameter η is set to 20, parameter p_m is set to 1, and parameter q_m is set to 0.01, the average and minimum values of migration delay, maximum completion time, and energy consumption are optimal as compared to the other parameter combinations of the value settings, and the upper and lower bounds of each objective are smaller than those of the other parameter combinations, which is a desirable optimization result. Although it fails to outperform the other algorithms at the maximum value of energy consumption and load balancing, it shows outstanding performance in terms of migration time and maximum completion time. In summary, adopting the parameter combination settings of $\eta = 20$, $p_m = 1$, and $q_m = 0.01$ makes the algorithm MI-MaOEA more superior and robust in solving the I-MaOWMUE model.

In Table 4, in order to compare the performance effects of different algorithms on the four optimization objectives of the I-MaOWMUE model, we compare the mean, maximum, and minimum values on the four optimization objectives, where the mean value more clearly expresses the degree of goodness of the algorithms, and for each objective, the smaller the objective value represents the better the performance effect and the best results are marked in bold. From Table 4, it can be seen that the MI-MaOEA algorithm shows consistent and superior

Table 2
I-MaOWMUE model parameter constraints.

Variable	Description	Value
	Number of mobile devices	5
	Number of base stations	20
	Workflow type	5
	Area 1 communications coverage	6 km*6 km
	Area 2 communications coverage	4 km*6 km
$p_{k,n}$	Upload power for task $w_{k,n}$	[100,500] W
$us_{k,n}$	Task $w_{k,n}$ size	[50,500] MB
δ	Communications noise	[-100-10] W
I	Communications interference	[0.01,0.02]
$\frac{g_{k,n}}{B^{mig}}$	Communication gain of task $w_{k,n}$	[0.2,0.4] Mbps
	Migration bandwidth for task $w_{k,n}$	[5,20] Mbps
$\overline{B^{dt}}$	Task $w_{k,n}$ data transmission bandwidth	[1.5,3] Mbps
$\overline{R^m}$	The number of CPU cycles per second that can be computed by server m	[400,450] cycle/s
$P_{k,n}^{mig}$	Migration energy consumption	[5,6] MJ
$p_{k,n}^{dt}$	Data transfer energy consumption	[0,1] MJ
$p_{k,n}^{m,exe}$	Energy consumed by task $w_{k,n}$ to compute 1 CPU cycle at server m	[2,3] MJ

Table 3
Parameter setting experiment.

	η p_m q_m	Migration Delay	Maximum Completion Time	Energy Consumption	Load Balancing
Mean	20 1.0 0.01	**[29.711161,30.1336498]**	**[22.804685,23.3616373]**	**[136.7282042,137.3782608]**	**[0.0848622,0.7509751]**
	15 0.9 0.007	[30.284265,30.669576]	[23.5883838,24.0716393]	[141.3749482,141.851065]	[0.3398962,0.837589]
	10 0.8 0.004	[32.1544108,32.3979383]	[27.6639578,28.0678613]	[140.8662545,141.3589961]	[0.4072463,0.9352577]
	5 0.7 0.001	[32.6213206,33.155733]	[27.6949102,28.2571371]	[146.9854565,147.5287044]	[0.4445313,1.0148102]
Max	20 1.0 0.01	**[31.3272123,32.1932377]**	**[24.2524596,25.2522005]**	[146.3478425,147.2138679]	[0.6957946,1.6380555]
	15 0.9 0.007	[32.5683788,33.2341189]	[29.0679496,29.3812936]	[147.6913113,147.8724276]	[0.8311289,**1.4845266]**
	10 0.8 0.004	[33.4876592,33.9678162]	[31.8904347,32.7328054]	**[142.5541255,143.5517943]**	[2.5855946,3.4025645]
	5 0.7 0.001	[34.1015124,34.6210964]	[27.923119,28.9228598]	[154.170455,154.9874249]	**[0.5135437**, 1.5008733]
Min	20 1.0 0.01	**[29.3096505,29.4005854]**	**[21.6134043,21.7945205]**	**[135.9613069,136.0468097]**	**[0.0053804,0.1112645]**
	15 0.9 0.007	[29.9449029,30.0307611]	[22.1152358,22.2061706]	[140.9512022,141.2042048]	[0.1001481,0.1456627]
	10 0.8 0.004	[31.8792113,31.9692206]	[24.627043,24.7179778]	[140.3557766,140.59765]	[0.1603905,0.2965571]
	5 0.7 0.001	[32.3079131,32.4440797]	[27.3135959,27.6434887]	[145.7673041,146.0092017]	[0.3796501,0.4705849]

Table 4
Comparison of the objective values of four algorithms on the I-MaOWMUE model.

	Algorithm	Migration Delay	Maximum Completion Time	Energy Consumption	Load Balancing
Mean	MI-MaOEA	**[29.711161,30.1336498]**	**[22.804685,23.3616373]**	**[136.7282042,137.3782608]**	**[0.0848622,0.7509751]**
	DI-μMOGA	[40.090148,40.7592748]	[34.9260306,35.5662294]	[147.8948972,148.5279753]	[1.4806598,2.0857283]
	II-MOEA	[37.5503117,38.1812611]	[34.6287875,35.2204535]	[159.0845489,159.7607118]	[1.6637146,2.3347694]
	InMaOEA	[37.3343011,37.9819818]	[34.0898393,34.7578041]	[155.9052669,156.539373]	[1.8459827,2.4281237]
Max	MI-MaOEA	**[31.3272123,32.1932377]**	**[24.2524596,25.2522005]**	**[146.3478425,147.2138679]**	**[0.6957946,1.6380555]**
	DI-μMOGA	[50.5688727,51.0086075]	[39.6541748,39.6541748]	[175.1968137,175.1968137]	[2.7586038,3.2907886]
	II-MOEA	[46.3692592,47.159135]	[39.9270916,40.8959008]	[210.1066616,210.5868185]	[2.6972205,3.6495187]
	InMaOEA	[46.7995969,47.465337]	[42.1384334,43.1175175]	[191.1862898,191.367406]	[3.849449,4.79171]
Min	MI-MaOEA	**[29.3096505,29.4005854]**	**[21.6134043,21.7945205]**	**[135.9613069,136.0468097]**	**[0.0053804,0.1112645]**
	DI-μMOGA	[32.1034223,32.6558227]	[28.385888,29.3732177]	[139.2552522,139.6373911]	[0.1126588,0.3824556]
	II-MOEA	[33.4377613,33.8401625]	[31.5321978,32.0698001]	[139.062606,139.1081206]	[1.2233632,1.3595299]
	InMaOEA	[33.1676769,33.566078]	[28.9984236,29.8863088]	[141.1859596,142.0938643]	[1.1911343,1.3270775]

performance in the four optimization objectives of migration delay, maximum completion time, energy consumption, and load balancing for the I-MaOWMUE model, and it achieves the optimal results in terms of the mean, maximum and minimum values compared to the DI-μMOGA, II-MOEA and InMaOEA algorithms. MI-MaOEA algorithm can greatly shorten the migration delay and maximum completion time of the task, and ensure that the task can be more evenly distributed to the various servers to achieve load balancing, for the more distant servers, even if it increases the migration energy consumption, but also to ensure that the rapid processing of the dependent tasks, reducing the corresponding energy consumption of the execution, which indicates that the MI-MaOEA algorithm to maintain the balance of the four conflicting objectives at the same time to ensure that the quality of the solution, to avoid falling into the local optimum, which is due to the fact that we use the dual-migration selection strategy, so that the iterative process can be jumped out of the local optimum, to ensure that the diversity of the solution. This indicates that the MI-MaOEA algorithm can effectively solve the I-MaOWMUE problem and provide better migration strategies for decision-makers. In addition, when the MI-MaOEA algorithm solves the problem, the upper bound of the interval for each objective is smaller than the lower bound of the interval for the corresponding objective of the other algorithms, which indicates that the MI-MaOEA algorithm not only achieves better results overall but also provides better performance on each objective. In summary, based on the analysis of the results in Table 4, the MI-MaOEA algorithm shows the best performance on all four optimization objectives of the I-MaOWMUE model, with significant advantages.

Fig. 4 shows the effect of all the algorithms on the optimization of each objective value of the I-MaOWMUE model in the first 100 iterations to demonstrate the convergence of the MI-MaOEA algorithm. To give credibility to the comparison results, we set the same parameters for all the algorithms as a way to assess the convergence speed and objective optimization effect of the different algorithms, while we took the objective values every 10 generations and averaged these objective values to plot them in Fig. 4. From Fig. 4, we can see that when MI-MaOEA, DI-μMOGA, II-MOEA, and InMaOEA solve the problem, the value of the objective function of the problem gradually tends to stabilize as the number of iterations increases. In particular, in the first 30 iterations, the MI-MaOEA algorithm obtained the most significant changes in the objective values, with faster convergence, which means that the MI-MaOEA algorithm obtains the optimal solution faster during the optimization process and obtains the best four objective values compared to the other three algorithms. In the environment selection phase, we adopt an individual-based interval objective value hierarchical sorting strategy, which is ordered according to the individual objective value and can more effectively screen out the excellent individuals and introduce them to the next generation, and the MI-MaOEA algorithm utilizes this strategy to exert greater selection pressure, thus pushing the algorithm to converge to the optimal solution more quickly in the convergence phase. In the middle of the iterations (30-60), the convergence of the MI-MaOEA algorithm slows down as the number of iterations increases, good individuals are selected frequently, and poorer individuals are progressively eliminated, leading to a decrease in the diversity in the population. As the objectives become more and more conflicting with each other, making it unable to sort individuals for non-dominance, the convergence of the MI-MaOEA algorithm slows down. Since the algorithm MI-MaOEA employs a dual-migration selection strategy to provide new possible solutions to the decision space and to avoid falling into local optima, the optimal values of the four objective values obtained are better than those of the other algorithms and reach the optimal values after 60 iterations. In summary, based on the analysis of the results in Fig. 4, the MI-MaOEA algorithm has a faster convergence rate compared to other algorithms.

Fig. 5 obtains the box diagrams of the four objective function values in the model as a way of observing the distribution of the overall solutions of all algorithms solving the I-MaOWMUE model, which includes

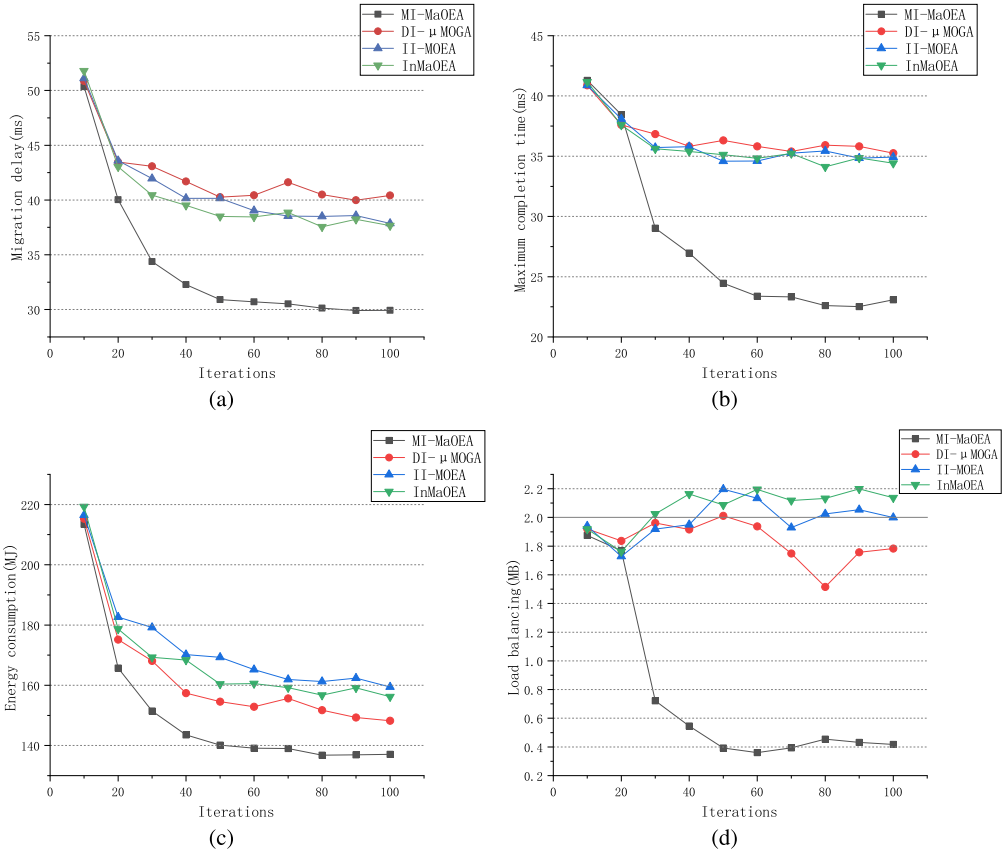

Fig. 4. Trends in the search process for each objective value.

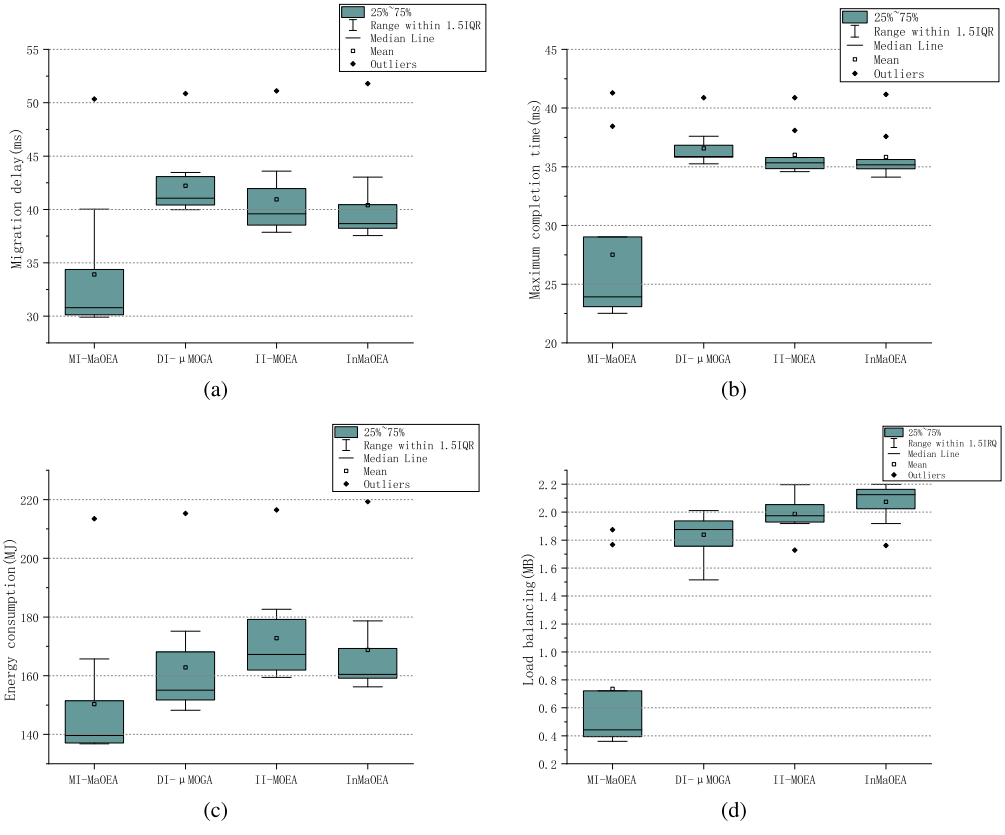

Fig. 5. Box diagrams of the four algorithmic solutions for each objective function.

the outliers, maximum, 1/4 and 3/4 distributions, mean, median, and minimum values of the overall solutions. From Fig. 5, it can be seen that among the four objective values obtained by each algorithm, the performance of the overall solution of the MI-MaOEA algorithm is better than the other algorithms, which indicates that the optimization effect of the MI-MaOEA algorithm is better. The results in Fig. 5(a), (b), and (d) show that the MI-MaOEA algorithm has more solution distributions in terms of migration delay, maximum completion time, and load balancing, which is because we use a dual-migration selection strategy to provide a new direction of choice for decision-making schemes in the process of non-dominated sorting. Fig. 5(c) shows that the MI-MaOEA algorithm has a similar overall solution distribution to DI-μMOGA and II-MOEA in terms of the energy consumption objective, but the overall solution is optimal for each value. The effect of the variational operator makes the solution have outliers, causing it to deviate from the overall solution. In conclusion, the proposed MI-MaOEA algorithm provides the best solution set optimization compared to other algorithms.

6. Conclusion

In this paper, we consider the problem of computing migration in an uncertain environment and transform the uncertain environmental factors into interval parameters, while taking into account the relevance of the tasks and constraints such as server computing power, capacity limitations, and service scope, the task migration strategy and real-time priority scheduling strategy are proposed to achieve a rational allocation of resources and fast response to dependent tasks, and the four optimization objectives of migration delay, maximum completion time, energy consumption, and load balancing are jointly optimized, and the I-MaOWMUE model is established. For this reason, this paper designs the MI-MaOEA algorithm, which uses interval confidence to represent the interval dominance relationship and serves as an important basis for matching selection, which in turn ensures the convergence of the solution, and utilizes objective-value-dominated hierarchical sorting and dual-migration selection strategies to ensure diversity of solutions. To evaluate the effectiveness of MI-MaOEA on the I-MaOWMUE model, MI-MaOEA is compared with three algorithms, DI-μMOGA, II-MOEA, and InMaOEA, respectively. The solution set shows advantages in the mean, maximum, and minimum values and has a better convergence rate for the four optimization objectives. The solutions have a better distribution and maintain the best performance, which can provide a better migration solution for decision-makers.

In future work, we will focus on privacy and security issues in migration scenarios by using encrypted communication and authentication to achieve reliable transmission of tasks and prevent malicious users from accessing the private information of ordinary users. In addition, we will extend the constructed interval optimization model by introducing new uncertainty factors that are consistent with migration scenarios to provide feasible resource allocation schemes for IoT systems and introduce dynamic factors to make our model more flexible to adapt to dynamically changing computing environments in terms of user requirements and resources.

Declaration of competing interest

No conflict of interest exists in the submission of this manuscript, and the manuscript is approved by all authors for publication.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China under [Grant No.61806138]; in part by the China University Industry-University-Research Collaborative Innovation Fund (Future Network Innovation Research and Application Project) [Grant 2021FNA04014]; in part by the Key R&D program of Shanxi Province, under Grant No. 202202020101012; in part by Taiyuan University of Science and Technology Scientific Research Initial Funding (TYUST SRIF), No. 20232087; and in part by the Shanxi University Science and Technology Innovation Funding, No. 2023L177.

References

[1] Islam MM, Nooruddin S, Karray F, Muhammad G. Internet of things: device capabilities, architectures, protocols, and smart applications in healthcare domain. IEEE Int Things J 2022;10:3611–41.
[2] Bai J, Huang G, Zhang S, Zeng Z, Liu A. Ga-dctsp: an intelligent active data processing scheme for uav-enabled edge computing. IEEE Int Things J 2022;10:4891–906.
[3] Kong X, Wu Y, Wang H, Xia F. Edge computing for internet of everything: a survey. IEEE Int Things J 2022;9:23472–85.
[4] Wang H, Lv T, Lin Z, Zeng J. Energy-delay minimization of task migration based on game theory in mec-assisted vehicular networks. IEEE Trans Veh Technol 2022;71:8175–88.
[5] Liang Z, Liu Y, Lok T-M, Huang K. Multi-cell mobile edge computing: joint service migration and resource allocation. IEEE Trans Wirel Commun 2021;20:5898–912.
[6] Liu C, Tang F, Hu Y, Li K, Tang Z, Li K. Distributed task migration optimization in mec by extending multi-agent deep reinforcement learning approach. IEEE Trans Parallel Distrib Syst 2020;32:1603–14.
[7] Lin R, Xie T, Luo S, Zhang X, Xiao Y, Moran B, et al. Energy-efficient computation offloading in collaborative edge computing. IEEE Int Things J 2022;9:21305–22.
[8] Ji T, Luo C, Yu L, Wang Q, Chen S, Thapa A, et al. Energy-efficient computation offloading in mobile edge computing systems with uncertainties. IEEE Trans Wirel Commun 2022;21:5717–29.
[9] Chen J, Yang Y, Wang C, Zhang H, Qiu C, Wang X. Multitask offloading strategy optimization based on directed acyclic graphs for edge computing. IEEE Int Things J 2021;9:9367–78.
[10] Yao S, Wang M, Qu Q, Zhang Z, Zhang Y-F, Xu K, et al. Blockchain-empowered collaborative task offloading for cloud-edge-device computing. IEEE J Sel Areas Commun 2022;40:3485–500.
[11] Maleki EF, Mashayekhy L, Nabavinejad SM. Mobility-aware computation offloading in edge computing using machine learning. IEEE Trans Mob Comput 2021;22:328–40.
[12] Chen R, Wang X. Maximization of value of service for mobile collaborative computing through situation aware task offloading. IEEE Trans Mob Comput 2021.
[13] Xia X, Chen F, Grundy J, Abdelrazek M, Jin H, He Q. Constrained app data caching over edge server graphs in edge computing environment. IEEE Trans Serv Comput 2021;15:2635–47.
[14] Tao O, Chen X, Zhou Z, Li L, Tan X. Adaptive user-managed service placement for mobile edge computing via contextual multi-armed bandit learning. IEEE Trans Mob Comput 2021.
[15] Luo Q, Li C, Luan TH, Shi W. Minimizing the delay and cost of computation offloading for vehicular edge computing. IEEE Trans Serv Comput 2021;15:2897–909.
[16] Zakaryia SA, Ahmed SA, Hussein MK. Evolutionary offloading in an edge environment. Egypt Inform J 2021;22:257–67.
[17] Liu J, Xiong K, Ng DWK, Fan P, Zhong Z, Letaief KB. Max-min energy balance in wireless-powered hierarchical fog-cloud computing networks. IEEE Trans Wirel Commun 2020;19:7064–80.
[18] Do-Duy T, Van Huynh D, Dobre OA, Canberk B, Duong TQ. Digital twin-aided intelligent offloading with edge selection in mobile edge computing. IEEE Wirel Commun Lett 2022;11:806–10.
[19] Ding Y, Li K, Liu C, Li K. A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing. IEEE Trans Parallel Distrib Syst 2021;33:1503–19.
[20] Kim T, Sathyanarayana SD, Chen S, Im Y, Zhang X, Ha S, et al. Modems: optimizing edge computing migrations for user mobility. IEEE J Sel Areas Commun 2022;41:675–89.
[21] Xu X, Yao L, Bilal M, Wan S, Dai F, Choo K-KR. Service migration across edge devices in 6g-enabled internet of vehicles networks. IEEE Int Things J 2021;9:1930–7.
[22] Li S, Li C, Huang Y, Jalaian BA, Hou YT, Lou W. Enhancing resilience in mobile edge computing under processing uncertainty. IEEE J Sel Areas Commun 2023;41:659–74.
[23] Ghoorchian S, Maghsudi S. Multi-armed bandit for energy-efficient and delay-sensitive edge computing in dynamic networks with uncertainty. IEEE Trans Cogn Commun Netw 2020;7:279–93.
[24] Xu X, Huang Q, Zhu H, Sharma S, Zhang X, Qi L, et al. Secure service offloading for internet of vehicles in sdn-enabled mobile edge computing. IEEE Trans Intell Transp Syst 2020;22:3720–9.
[25] Xue F, Hai Q, Gong Y, You S, Cao Y, Tang H. Rvea-based multi-objective workflow scheduling in cloud environments. Int J Bio-Inspir Comput 2022;20:49–57.
[26] Dong T, Zhou L, Chen L, Song Y, Tang H, Qin H. A hybrid algorithm for workflow scheduling in cloud environment. Int J Bio-Inspir Comput 2023;21:48–56.
[27] Chen H, Qin W, Wang L. Task partitioning and offloading in iot cloud-edge collaborative computing framework: a survey. J Cloud Comput 2022;11:86.
[28] He X, Zheng J, He Q, Dai H, Liu B, Dou W, et al. Online computation offloading for deadline-aware tasks in edge computing. Wirel Netw 2022:1–20.

[29] Sun J, Yin L, Zou M, Zhang Y, Zhang T, Zhou J. Makespan-minimization workflow scheduling for complex networks with social groups in edge computing. J Syst Archit 2020;108:101799.
[30] Huang B, Xiang Y, Yu D, Wang J, Li Z, Wang S. Reinforcement learning for security-aware workflow application scheduling in mobile edge computing. Secur Commun Netw 2021;2021:1–13.
[31] Wang S, Ma D, Ren Z, Qu Y, Wu M. An adaptive multi-objective particle swarm optimisation algorithm based on fitness distance to streamline repository. Int J Bio-Inspir Comput 2022;20:209–19.
[32] Das G. Techno-economic analysis of novel multi-objective soft computing technique. Int J Bio-Inspir Comput 2022;20:172–82.
[33] Xiao S, Wang W, Wang H, Huang Z. A new multi-objective artificial bee colony algorithm based on reference point and opposition. Int J Bio-Inspir Comput 2022;19:18–28.
[34] Zhao T, Wu L, Wu D, Li J, Cui Z. Multi-factor evolution for large-scale multi-objective cloud task scheduling. KSII Trans Int Inf Syst 2023;17.
[35] Wu L, Wu D, Zhao T, Cai X, Xie L. Dynamic multi-objective evolutionary algorithm based on knowledge transfer. Inf Sci 2023;636:118886.
[36] Cui Z, Xue Z, Fan T, Cai X, Zhang W. A many-objective evolutionary algorithm based on constraints for collaborative computation offloading. Swarm Evol Comput 2023;77:101244.
[37] Cui Z, Wen J, Lan Y, Zhang Z, Cai J. Communication-efficient federated recommendation model based on many-objective evolutionary algorithm. Expert Syst Appl 2022;201:116963.
[38] Cui Z, Zhang Z, Hu Z, Geng S, Chen J. A many-objective optimization based intelligent high performance data processing model for cyber-physical-social systems. IEEE Trans Netw Sci Eng 2021;9:3825–34.
[39] Cui Z, Li B, Lan Z, Xu Y. Many-objective evolutionary algorithm based on three-way decision. Egypt Inform J 2023;24:100388.
[40] Bozorgchenani A, Mashhadi F, Tarchi D, Monroy SAS. Multi-objective computation sharing in energy and delay constrained mobile edge computing environments. IEEE Trans Mob Comput 2020;20:2992–3005.
[41] Han D, Du W, Jin Y, Du W, Yu G. A fuzzy constraint handling technique for decomposition-based constrained multi-and many-objective optimization. Inf Sci 2022;597:318–40.
[42] Dai J, Wang Z, Huang W. Interval-valued fuzzy discernibility pair approach for attribute reduction in incomplete interval-valued information systems. Inf Sci 2023;642:119215.
[43] Jin Y, Zhang Z, Xie L, Cui Z. Decomposition-based interval multi-objective evolutionary algorithm with adaptive adjustment of weight vectors and neighborhoods. Egypt Inform J 2023;24:100405.
[44] He Q, He Z, Duan S, Zhong Y. Multi-objective interval portfolio optimization modeling and solving for margin trading. Swarm Evol Comput 2022;75:101141.
[45] Liu G, Liu S. Direct method for uncertain multi-objective optimization based on interval non-dominated sorting. Struct Multidiscip Optim 2020;62:729–45.
[46] Zhang Z, Zhao M, Wang H, Cui Z, Zhang W. An efficient interval many-objective evolutionary algorithm for cloud task scheduling problem under uncertainty. Inf Sci 2022;583:56–72.
[47] Gong D-w, Qin N-n, Sun X-y. Evolutionary algorithms for multi-objective optimization problems with interval parameters. In: 2010 IEEE fifth international conference on bio-inspired computing: theories and applications (BIC-TA). IEEE; 2010. p. 411–20.
[48] Juve G, Chervenak A, Deelman E, Bharathi S, Mehta G, Vahi K. Characterizing and profiling scientific workflows. Future Gener Comput Syst 2013;29:682–92.