Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.068248

ARTICLE

High-Dimensional Multi-Objective Computation Offloading for MEC in Serial Isomerism Tasks via Flexible Optimization Framework

Zheng Yao* **and Puqing Chang**

School of Robotics and Automation, Hubei University of Automotive Technology, Shiyan, 442002, China

*Corresponding Author: Zheng Yao. Email: yaozheng@huat.edu.cn

Received: 23 May 2025; Accepted: 22 August 2025; Published: 10 November 2025

ABSTRACT: As Internet of Things (IoT) applications expand, Mobile Edge Computing (MEC) has emerged as a promising architecture to overcome the real-time processing limitations of mobile devices. Edge-side computation offloading plays a pivotal role in MEC performance but remains challenging due to complex task topologies, conflicting objectives, and limited resources. This paper addresses high-dimensional multi-objective offloading for serial heterogeneous tasks in MEC. We jointly consider task heterogeneity, high-dimensional objectives, and flexible resource scheduling, modeling the problem as a Many-objective optimization. To solve it, we propose a flexible framework integrating an improved cooperative co-evolutionary algorithm based on decomposition (MOCC/D) and a flexible scheduling strategy. Experimental results on benchmark functions and simulation scenarios show that the proposed method outperforms existing approaches in both convergence and solution quality.

KEYWORDS: Edge computing offload; serial Isomerism applications; many-objective optimization; flexible resource scheduling

1 Introduction

The Internet of Things (IoT) generates numerous high-density, low-latency tasks—such as augmented reality and gaming—on mobile devices [1,2]. However, limited computational power and energy severely constrain device performance [3,4], hindering IoT development. While cloud computing offers elastic resources [5], it suffers from latency and congestion due to long-distance transmission. Mobile Edge Computing (MEC) mitigates these issues by enabling low-latency, localized computation through edge servers [6]. Computation offloading, a core MEC component, poses significant challenges due to task dependencies, resource constraints, and conflicting objectives. Recent studies tackle challenges like complex task topologies and conflicting goals such as minimizing delay versus energy consumption [7,8]. Subtasks may have serial or parallel dependencies, requiring careful offloading decisions. Most research, however, focuses on low-dimensional problems and rarely considers scenarios with more than four objectives, common in real MEC systems. This gap motivates the need for scalable and robust offloading algorithms.

To tackle these challenges, this paper investigates high-dimensional multi-objective computation offloading for serial isomerism tasks, emphasizing serial task structure, multi-objective complexity, and flexible scheduling. We formulate the problem as a Many-objective optimization model and propose a hybrid framework integrating an improved MOCC/D algorithm with a task-aware scheduling strategy. Experiments on benchmarks and simulations validate the effectiveness of the proposed method over existing approaches. The main contributions of this paper are summarized as follows.

- We propose a flexible offloading model for MEC with serial isomerism tasks, tackling high-dimensional optimization across makespan, delay, energy consumption, and energy balance, while capturing task dependencies, resource heterogeneity, and objective conflicts.
- We propose a new offloading solution framework that integrates an improved MOCC/D algorithm with a flexible scheduling strategy. The MOCC/D incorporates decomposition-based co-evolution, cumulative-probability-guided mating selection, and HV-based evolution switching to improve search performance in high-dimensional spaces. The scheduling module introduces a task-aware double-layer encoding, greedy decoding, randomized initialization, and crossover operations tailored to serial task execution.
- Experimental results on benchmarks and simulations confirm that our approach outperforms existing methods in convergence, diversity, and resource efficiency for high-dimensional offloading tasks.

The paper has six sections: Section 2 reviews related work; Section 3 presents the model; Section 4 introduces the algorithm strategy; Section 5 shows experiments; Section 6 concludes the work.

2 Related Works

2.1 Application Topology for Edge Computation Offloading

Application topologies in MEC offloading fall into two categories: full-offload and partial-offload. The full-offload model handles tightly coupled applications as a whole and is typically NP-hard [9], often solved via heuristics such as artificial fish swarm algorithms [10] or optimization of CPU frequency, transmission power, and UAV path planning [11]. Partial-offload divides tasks into subtasks, forming streaming or workflow structures. Prior work has explored parallel streaming under mobility reinforcement learning with energy harvesting [12], and dependency-aware grouping to reduce latency [13]. More recent approaches include distributed deep reinforcement learning for scalable offloading [14] and hybrid bio-inspired models for intrusion detection [15]. However, flexible offloading for **serial isomerism tasks**—sequential subtasks differing in type, load, and hardware compatibility—remains underexplored, especially in multi-user, multi-server MEC settings. This paper addresses this gap by jointly considering structural diversity and execution constraints in dynamic edge environments.

2.2 Optimization Objectives for Edge Computation Offloading

Computation offloading decisions often aim to minimize makespan, or balance makespan and energy. Wu et al. [16] proposed a layered algorithm to reduce total makespan, while Gorlatova et al. [17] used probabilistic modeling for makespan minimization. Time-delay constraints linked to deadlines were addressed by Meng et al. [18] through an online strategy maximizing deadline satisfaction, and by Verba et al. [19] with penalty terms for violations. To balance makespan and energy, Wang et al. [20] used a weighted sum objective solved via an improved genetic algorithm, and Li et al. [21] introduced a cooperative edge caching strategy. However, high-dimensional multi-objective computation offloading in MEC remains underexplored. We propose an intelligent offloading algorithm based on a flexible optimization framework, combining an improved multi-objective cooperative co-evolutionary algorithm based on decomposition (MOCC/D) with a flexible scheduling strategy. This hybrid framework offers enhanced exploration capabilities for solving high-dimensional multi-objective offloading problems.

3 Problem Modeling

In this section, we present a generic computation offloading system of serial isomerism tasks as illustrated in Fig. 1. The considered computation offloading system supports the cooperation among multiple

edge servers and can be applied to the real-world scenario with highly random and dynamic request, e.g., smart transportation, smart buildings, smart home. This network is a powerful infrastructure to handle complex and changeable demand in IOT system.

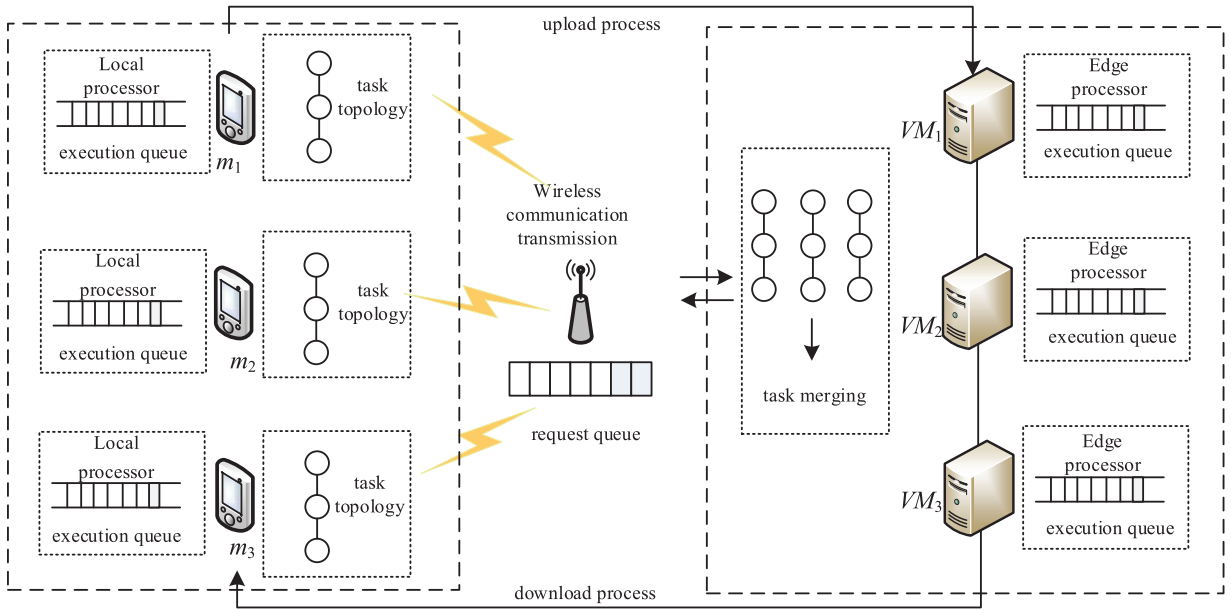

Figure 1: The diagram of system model

3.1 MEC Architecture

The proposed offloading system adopts a hierarchical architecture comprising mobile terminals, a single wireless access point, and multiple edge servers. At the terminal layer, mobile devices are heterogeneous, large-scale, and geographically dispersed, functioning collectively in IoT applications where a single device failure may impact overall performance. Given the prevalence of latency-sensitive and computation-intensive tasks, energy-efficient management of low-battery devices is critical. These terminals generate workflows of serial isomerism tasks—subtasks that differ in type and resource demands but execute sequentially—modeled via function reference graphs to capture dependencies. At the wireless layer, communication between terminals and the access point uses Orthogonal Frequency Division Multiple Access (OFDMA), with negligible transmission loss due to the co-location of the access point and edge servers. At the edge layer, multiple servers collaboratively process offloaded tasks. Since each server supports only a subset of software types, verifying compatibility between subtasks and edge resources is essential. Therefore, an efficient and flexible scheduling strategy is needed to enable parallel execution of tasks.

3.2 System Modeling

In view of the above MEC architecture, mobile terminals of IoT system can be expressed as $MD_s = \{MD_1, MD_2, \ldots, MD_N\}$, in which MD_i can be denoted by an eight tuple:

$$MD_i = \left\{c_i^m, e_i, P_i^c, P_i^n, P_i^s, P_i^b, P_i^d, d_i\right\} \tag{1}$$

where $i \in [1, N]$ denotes the terminal number; N is the total number of terminals; c_i^m is the terminal computing capacity; e_i is the terminal remaining battery; P_i^c, P_i^n, P_i^s, and P_i^b represent the power consumption

in computing, idle, sending, and receiving states, respectively; P_i^d is the communication idle power; d_i is the distance between terminal and its nearest gateway node.

Workflow applications of sequential chain can be expressed as $Task_s = \{Task_1, Task_2, \ldots, Task_N\}$, in which $Task_i$ can be denoted by a four tuple:

$$Task_i = \left\{deadline_i, task_{i,j}, data_{i,j}, type_{i,j}\right\} \tag{2}$$

where $deadline_i$ is the soft deadline of $Task_i$ and represents the relative urgency; $task_{i,j}$ is the j sub-task of i task, which satisfies $j \in [1, O_i]$; O_i represents the sub-task number of i task; $data_{i,j}$ is the input data volume required of $task_{i,j}$; $type_{i,j}$ is the task type of $task_{i,j}$, which only execute on the server installed with corresponding application software.

Edge server cluster can be expressed as $VM_s = \{VM_1, VM_2, \ldots, VM_M\}$, in which VM_k can be denoted by a two tuple:

$$VM_k = \{c_k^v, type_k^v\} \tag{3}$$

where k is the server number, which satisfies $k \in [1, M]$; M is the total number of servers; c_k^v is the server computing capacity; $type_k^v$ is the tuple of application software type installed on server VM_k as the edge server computing resources are limited.

3.3 Time Computing Model

When $task_{i,j}$ is executed locally, the start time and end time of it are expressed as $S_{i,j}^m$ and $F_{i,j}^m$, respectively, and its calculation formula is as follows:

$$F_{i,j}^m = S_{i,j}^m + T_{i,j}^m \tag{4}$$

$$S_{i,j}^m = \begin{cases} 0, pred(i,j) = \varnothing \\ F_{i,j-1}^m + T_{i,j-1,j}, pred(i,j) \neq \varnothing \end{cases} \tag{5}$$

$$T_{i,j}^m = \frac{data_{i,j}}{c_i^m} \tag{6}$$

$$T_{i,j-1,j} = \begin{cases} 0, others \\ \frac{data_{i,j}}{r_i}, x_{i,j} \cdot x_{i,j-1} = 0 \ and \ x_{i,j} + x_{i,j-1} \neq 0 \end{cases} \tag{7}$$

where $x_{i,j}$ is the offload decision variable, $x_{i,j} = 0$ represents that $task_{i,j}$ is executed on the MD_i, while $x_{i,j} = k$ represents that $task_{i,j}$ is executed on the VM_k; $pred\,(i,j)$ is the precursor set of $task_{i,j}$; $Tm_{i,j}$ is the local execution time of $task_{i,j}$; $T_{i,j-1,j}$ is the transfer time between $task_{i,j}$ and $task_{i,j-1}$; r_i is the channel transmission rate. Furthermore, the calculation formula of r_i is as follows:

$$r_i = B \log_2 \left(1 + \frac{P_i^s h_i}{\sigma^2}\right) \tag{8}$$

$$h_i = \rho {d_i}^{-\eta} \tag{9}$$

where B is the fixed maximum channel bandwidth; σ^2 is the fixed channel noise power; h_i is the channel status information; ρ is the channel power gain at the reference distance; η is the path loss objective.

When $task_{i,j}$ is executed at the edge server, the start time and end time of it are expressed as $S_{i,j}^{v}$ and $F_{i,j}^{v}$, respectively, and its calculation formula is as follows:

$$F_{i,j}^{v} = S_{i,j}^{v} + T_{i,j}^{v} \tag{10}$$

$$S_{i,j}^{v} = \begin{cases} T_{i,j-1,j}, pred(i,j) = \varnothing \\ \max\left(F_{i,j-1}^{v} + T_{i,j-1,j}, R_{k,i,j}\right) pred(i,j) \neq \varnothing \end{cases} \tag{11}$$

$$Tv_{i,j} = \frac{data_{i,j}}{c_k^v} \tag{12}$$

where $Tv_{i,j}$ is the edge execution time of $task_{i,j}$; $R_{k,i,j}$ is the earliest time that $task_{i,j}$ can execute on server VM_k, and it depends on $task_{i,j}$ edge sorting variables $y_{k,i,j}$ on server VM_k. Therefore, the makespan and time-delay are expressed as:

$$Makespan_i = FT_{i,exit} - ST_{i,entry} \tag{13}$$

$$Timedelay_i = |Makespan_i - deadline_i| \tag{14}$$

where $FT_{i,exit}$ is the end time of the last sub-task of $Task_i$, and $ST_{i,entry}$ is the start time of the first sub-task of $Task_i$.

Makespan represents the total time to complete the task workflow, reflecting system efficiency and throughput. Minimizing makespan reduces processing time, improving resource utilization and performance. In contrast, time-delay measures the deviation from a task's deadline, indicating the system's ability to meet real-time requirements. Minimizing time-delay ensures timely execution and enhances user satisfaction by reducing deadline violations. These metrics provide a comprehensive view of time performance: makespan addresses global scheduling efficiency, while time-delay focuses on meeting individual task deadlines. Optimizing both simultaneously enables more effective time-related system performance.

3.4 Energy Computing Model

The energy consumption of a mobile terminal includes both computational energy and communication energy, which are further divided into active and idle states. We consider energy usage comprehensively to optimize two critical system-level objectives: minimizing total energy consumption and ensuring energy balance across devices. The computing module of mobile terminal includes running and idle states, and the calculation of them is as follows:

$$E_i^c = P_i^c \sum_{j=1}^{O_i} \frac{data_{i,j} \max\left(1 - x_{i,j}, 0\right)}{c_i^m} \tag{15}$$

$$E_i^n = P_i^n \left(Makespan_i - \sum_{j=1}^{O_i} \frac{data_{i,j} \max\left(1 - x_{i,j}, 0\right)}{c_i^m} \right) \tag{16}$$

$$E_i^1 = E_i^c + E_i^n \tag{17}$$

where E_i^c is the energy consumption at the running state; E_i^n is the energy consumption at the idle state; E_i^1 is the total energy consumption from computing module.

The communication module of mobile terminal includes send, back and idle states, and the calculation of them is as follows:

$$E_i^s = P_i^s \sum_{j=1}^{y_{k,i,j}} T_{i,j-1,j}, x_{i,j-1} = 0 \ and \ x_{i,j} \neq 0 \tag{18}$$

$$E_i^b = P_i^b \sum_{j=1}^{y_{k,i,j}} T_{i,j-1,j}, x_{i,j-1} \neq 0 \ and \ x_{i,j} = 0 \tag{19}$$

$$E_i^d = P_i^d \left(Makespan_i - \sum_{j=1}^{y_{k,i,j}} T_{i,j-1,j} \right) \tag{20}$$

$$E_i^2 = E_i^s + E_i^b + E_i^d \tag{21}$$

where E_i^s is the energy consumption at the send state; E_i^b is the energy consumption at the back state; E_i^d is the energy-consumption at the idle state; E_i^2 is the total energy consumption of communication module.

Therefore, the total energy consumption E_i and energy-balance SOC are as follows:

$$E_i = E_i^1 + E_i^2 \tag{22}$$

$$SOC = \sqrt{\frac{1}{N} \sum_{i=1}^{N} \left((e_i - E_i) - \frac{1}{N} \sum_{i=1}^{N} (ey_i - E_i) \right)^2} \tag{23}$$

3.5 Optimizing Objectives

To achieve the purpose of prolonging the lifetime of IoT system, we introduce the high-dimensional multi-objective computation offloading problem in terms of combine makespan, time-delay, energy-consumption and energy-balance optimization objectives. The following optimization objectives are designed:

$$\begin{cases} \min f_1(x, y) = \sum_{i=1}^{N} (Makespan_i) \\ \min f_2(x, y) = \sum_{i=1}^{N} (\text{Timedelay}_i) \\ \min f_3(x, y) = \sum_{i=1}^{N} (E_i) \\ \min f_4(x, y) = SOC \end{cases} \tag{24}$$

where $f_1(x, y)$ is the total makespan of mobile terminals; $f_2(x, y)$ is the total time-delay of mobile terminals; $f_3(x, y)$ is the total energy-consumption of mobile terminals; $f_4(x, y)$ is the remaining power-balance of the mobile terminals. And variable constraints include: the offload decision variable is constrained by $x_{i,j} \in [0, 1]$; the edge sorting variables is constrained by $y_{k,i,j} \leq \sum_{i=1}^{N} O_i$.

4 Model Solution Strategy

To address these challenges, we model the computation offloading problem as a Many-objective optimization task and propose an intelligent offloading algorithm based on a flexible optimization framework. The proposed MOCC/D incorporates cooperative co-evolution, cumulative probability-based mating selection, and multi-evolutionary switching using HV. The proposed flexible scheduling strategy feature a double-layer coding scheme, pluggable greedy decoding, randomized population initialization, and line-order crossover, to effectively solve the scheduling of serial isomerism tasks. Source code for the proposed framework is available at https://github.com/Changpuqing/CMC_MEC.git (accessed on 21 August 2025).

The algorithm functions as an offline optimization engine, intended to be deployed on resource-relatively-abundant edge servers or cloudlets. Its purpose is to generate pre-computed task-offloading strategies or periodic resource allocation plans (e.g., re-optimized every few minutes or hours based on predicted network states and task loads) for the underlying terminal devices.

4.1 Proposed MOCC/D

High-dimensional multi-objective problems with over four objectives challenge MOEAs due to poor convergence and rising computational complexity from many non-dominated solutions [22]. Although methods like MOEA/D [23], RVEA [24], BiGE [25], and MEMO [26], balancing convergence and diversity remains difficult. We propose MOCC/D, an improved cooperative co-evolution algorithm that decomposes the problem into single-objective subproblems to reduce complexity and enable subpopulation cooperation. A mating selection alternates between nearest and farthest individuals using cumulative probability to balance convergence and diversity. Two evolutionary strategies switch adaptively based on the hypervolume (HV) of an extra population to enhance convergence to the Pareto front.

4.1.1 Cooperative Co-Evolution Strategy Based on Decomposition

The cooperative co-evolution strategy adopts a divide-and-conquer approach to significantly reduce the search space and computational complexity. Specifically, the complete variable set is divided into multiple segments, and the population is partitioned into corresponding subpopulations. Optimal individuals from each subpopulation are combined to form complete solutions for fitness evaluation. Subpopulations then evolve independently to generate new populations. To address multi-objective problems and further reduce computational complexity, a cooperative co-evolution strategy based on decomposition is designed. This transforms multiple objectives into a series of single-objective subproblems, enabling cooperation among subpopulations. Uniform weight vectors are generated using the simplex lattice method, each associated with a subproblem. Subproblems are evaluated via the Chebyshev scalarization function. Finally, Euclidean distances between weight vectors define neighborhoods, allowing subproblems to evolve independently. Algorithm 1 presents the pseudocode of this strategy.

Algorithm 1: Cooperative co-evolution strategy based on decomposition

Input:
M: The size of objective;
N: The size of population;
NP: The size of variable;
$Group$: The size of variable group;
Output: External archive guidance population EP;
1: Initialize $Group$ sub-population;
2: Initialize target weight vector $\lambda_1, \lambda_2, \ldots, \lambda_N$;
3: Initialize the neighborhood $B(i) = \{\lambda_{i_1}, \ldots, \lambda_{i_T}\}$ of $\lambda_i, i \in \{1, \ldots, N\}$;
4: Initialize reference vector $z = (z_1, \ldots, z_M)^T = \min_{1,2,..N} f_j(p_k), (j = 1, 2, \ldots, M)$;
5: Initialize external archive guidance population $EP = \varnothing$;
6: **while** *the stopping criteria is unsatisfied* **do**
7: **for** $m = 1$: $Group$ **do**
8: **for** $n = 1$: $Group$ && $m \neq n$ **do**
9: The variables of Pop(m) and optimal individual of Pop(n) are spliced;
10: **end for**

(Continued)

Algorithm 1 (continued)

```
for k = 1: N/Group do
    Randomly choose an individual p_r in mating pool and set it to be p_k spouse;
    Generate a new offspring pc_k by using genetic operators;
    Update reference vector z:
        if z_j < f_j(pc_k), z_j = f_j(pc_k)(j = 1, 2, ..., M);
    Update optimal individual of neighborhood B(i):
        if g^te(pc_k|λ_i, z) < g^te(p_k|λ_i, z), p_k = pc_k;
    Update external archive guidance population EP
end for
end for
end while
```

The time complexity of the algorithm is approximately $O\left(T \times N \times \frac{NP}{Group}\right)$, where T is the number of iterations, N is the population size, NP is the variables, and $Group$ is the variable groups.

4.1.2 Mating Individual Selection Strategy Based on Cumulative Probability

Subproblems traditionally select crossover partners randomly within their neighborhoods, making it difficult to balance convergence and diversity. To address this, we propose a mating selection strategy that alternates between the nearest and farthest individuals in the neighborhood. The nearest mating individual is chosen based on the smallest Euclidean distance, while the farthest is based on the largest distance. Additionally, a switching mechanism driven by cumulative probability controls the alternation. Fig. 2 illustrates this mating selection strategy.

Specifically, the mathematical expression of the switching method based on cumulative probability is as follows:

$$Object_{mating\_pool} = \begin{cases} \text{Far\_Nearest}_{mating\_pool}, \text{if rand} < P_m^i \\ \text{Random}_{mating\_pool}, \text{ otherwise} \end{cases} \tag{25}$$

where $\text{Far\_Nearest}_{mating\_pool}$ is the strategy for selecting the farthest and nearest mating object, while $\text{Random}_{mating\_pool}$ is the strategy for selecting two random mating objects; P_m^i is the selection probability, which satisfies $0 < P_m^i < 1$ and follows normal distribution. When initializing, μ is set as 0.5 and σ is set as 1. When the number of iterations reached the set threshold Th_{mating_pool}, μ is updated and P_m^i is updated synchronously. The set threshold $Threshold_{mating\_pool}$ is set as 10 in this paper, and the μ update formula is as follows:

$$u = \frac{\sum_{P_m^i \in PMAT} P_m^i}{|PMAT|} \tag{26}$$

where $\sum_{P_m^i \in PMAT} P_m^i$ is the cumulative probability of P_m^i; $|PMAT|$ is the number of P_m^i to get better performance. The $PMAT$ update formula is as follows:

$$PMAT = PMAT \bigcup P_m^i, if f_{x_i^p} > f_{x_i^c} \tag{27}$$

where $f_{x_i^p}$ is the Chebyshev value of parent individual x_i^p; $f_{x_i^c}$ is the Chebyshev value of the offspring individual x_i^c.

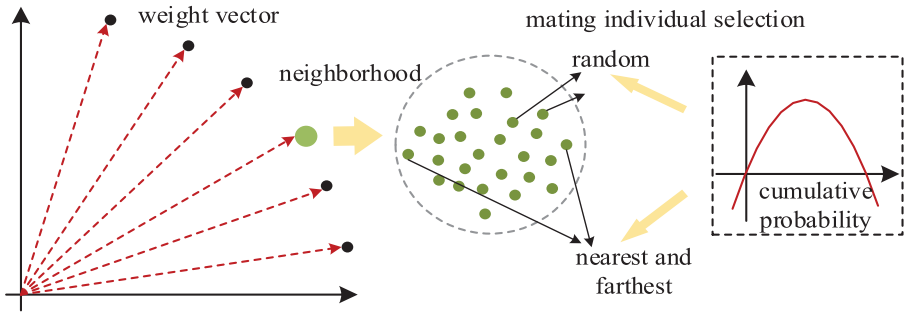

Figure 2: The diagram of mating individual selection strategy based on cumulative probability

4.1.3 Multi-Evolutionary Switching Strategy Based on HV

Cooperative co-evolution reduces search space and computational complexity but weakens local search. To better approach the Pareto front, we introduce a genetic evolution strategy based on cost values. An external population preserves non-dominated solutions, and switching between two evolutionary strategies is guided by its hypervolume (HV). When the non-dominated set exceeds capacity, a neighborhood screening strategy is applied. Fig. 3 illustrates this multi-evolutionary switching strategy.

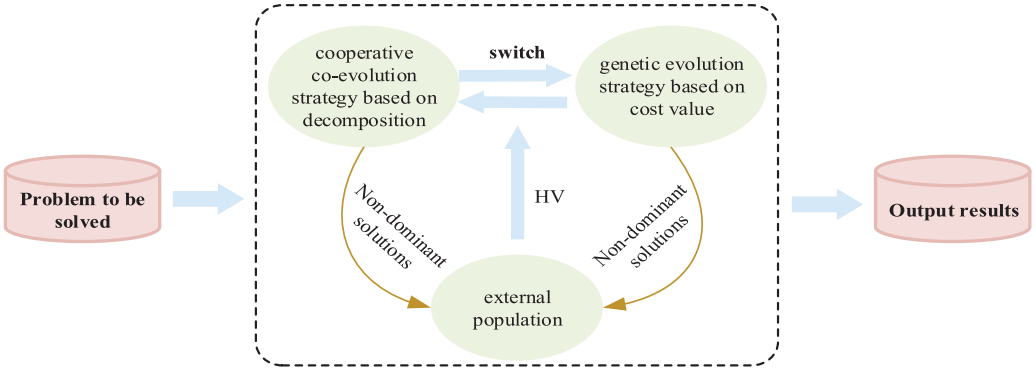

Figure 3: The diagram of multi-evolutionary switching strategy based on HV

When the external population's HV declines continuously past a threshold, the current strategy is deemed ineffective and switches to another. The mathematical form is:

$$num_{\text{dec}} = \begin{cases} num_{\text{dec}} + 1, \text{ if } \text{HV}_{gen} < \text{HV}_{gen-1} \\ 0, \text{ otherwise} \end{cases} \tag{28}$$

$$\text{flag}_{\text{switch}} = \begin{cases} \text{true}, \text{ if } num_{\text{dec}} = TH_{\text{switch}} \\ \text{false}, \text{ otherwise} \end{cases} \tag{29}$$

where HV_{gen} is the HV value in gen iterative; HV_{gen-1} is the HV value in $gen-1$ iterative. num_{dec} is the continuous declines number of HV. TH_{switch} is the switching threshold, which is set as 5 in this paper. $\text{flag}_{\text{switch}}$ is the switch flag bit. The calculation formula of HV is as follows:

$$HV\left(P^*\right) = volume\left(\bigcup_{f \in P^*} [f_1, y_1^*] \times \ldots \times [f_m, y_m^*]\right) \tag{30}$$

where P^* is the approximate solution set of Pareto frontier, $y^* = (y_1^*, y_2^*, \ldots, y_m^*)$ is the reference point, which is the 1.1 times of the maximum value of each objective in P^*. $volume$ is the volume composed of P^* and y^*. The larger HV is, the closer P^* is to the real Pareto front.

4.1.4 The Procedure of Proposed MOCC/D

In summary, MOCC/D consists of cooperative co-evolution strategy based on decomposition, mating individual selection strategy based on cumulative probability and multi-evolutionary switching strategy based on HV. The pseudocode of MOCC/D is as follows (Algorithm 2).

Algorithm 2: MOCC/D

Input: parameter;
Output: External archive guidance population EP;
Initialize parameter;
while *the stopping criteria is unsatisfied* **do**
  if $\text{flag}_{\text{switch}} = Ture$ **then**
    Execute cooperative co-evolution strategy based on decomposition, where an individual p_r in mating pool is selected by mating individual selection strategy based on cumulative probability:
    if $rand < P_m^i$ **then**
      Execute $\text{Far\_Nearest}_{mating\_pool}$;
    else
      Execute $\text{Random}_{mating\_pool}$;
    end
    if $m = Th_{mating\_pool}$ **then**
      Update $PMAT$, u and P_m^i;
    end
    Update m;
  else
    Execute genetic evolution strategy based on based on cost value;
  end
  Update external archive guidance population EP;
  Calculate HV value of EP;
  if $\text{HV}_{gen} > \text{HV}_{gen-1}$ **then**
    Update num_{dec};
  end
  if $num_{\text{dec}} = Th_{mating\_pool}$ **then**
    Switch $\text{flag}_{\text{switch}}$;
  end
end while

4.2 Flexible Scheduling Strategy

Furthermore, based on solving Many-objective optimization with MOCC/D, we proposed a flexible scheduling strategy, designing a double-layer coding strategy, a pluggable greedy decoding strategy, a randomized population initialization strategy, and a line order cross evolution strategy, which efficiently solves the flexible scheduling problem of serial isomerism tasks.

4.2.1 Double-Layer Coding Strategy

Unlike traditional chromosome coding, a double-layer coding strategy based on task ID and position order is proposed. Chromosomes are arranged by task ID and subtask order, defining execution sequence and positions. The task ID chromosome length equals the total number of subtasks; each gene's integer indicates the task ID, and its appearance order determines subtask execution order. The position order chromosome, of the same length, encodes each subtask's execution position within its feasible set, ensuring valid solutions. Fig. 4 illustrates a sequence: subtasks T21, T11, T31, T32, T41, T12, T22, T42 with execution positions VM3, VM4, VM1, VM3, VM2, VM4, VM2, VM1.

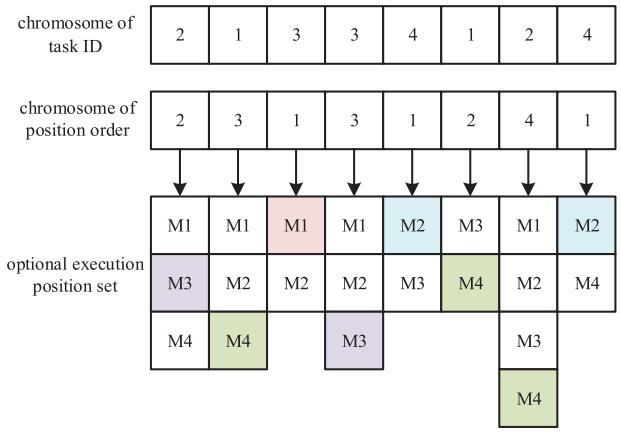

Figure 4: The diagram of double-layer coding strategy based on task ID and position order

4.2.2 Pluggable Greedy Decoding Strategy

To reduce the idle time of local terminals and edge servers and shorten the task execution time, a plug-in greedy decoding strategy is designed, which fetches the execution order and execution position of sub-tasks in turn, and then inserts the sub-tasks into the available execution position idle interval. Specifically, the previous sub-task end time of the same task ID (TP) and the previous sub-task end time of the same execution location (TM) are compared. If the time gap between TP and TM is greater than the execution time of the current sub-task and TM is greater than TP, the current sub-task can be inserted into the gap, and the start time is TM. Otherwise, the current sub-task cannot be inserted into the gap and the start time is TP.

4.2.3 Randomized Population Initialization Strategy

To ensure the diversity of individuals in the population, a randomized population initialization strategy is designed. Specifically, it supposes the chromosome length based on task ID is $\sum_{j=1}^{n} N_j$, where N_j represents the number of the sub-tasks of task j. When population initialization, task j is repeated by N_j times, and then the vector based on task ID is rearranged. On the other hand, it supposes the optional execution positions set of sub-tasks is $VM(i) = \{VM_1, VM_2, \ldots, VM_m\}$, where m represents the number of optional execution positions of sub-task i. When population initialization, the sub-task i randomly selects an integer from $[1, m]$, and then the vector based on position order is rearranged.

4.2.4 Line Order Cross Evolution Strategy

To enhance the global convergence of the evolution strategy, the gene of task ID is defined as a dominant gene, and the gene of position order code is an invisible gene. The invisible gene is adjusted according to the change of the dominant gene. Therefore, the line order cross evolution strategy is adopted in the chromosome

of the task ID. The specific steps are as follows. Firstly, it randomly generates two crossing positions and exchanges the fragments of two parent individuals between the two crossing positions. Secondly, it deletes the same gene in the original parent individual as the gene exchanged from another parent individual. Thirdly, from the first gene position, it fills in the remaining genes outside the two cross positions in turn.

5 Experimental Analysis

To validate the proposed method for Many-objective optimization and computing offloading, benchmark function tests and simulation experiments were conducted. Experiments ran on MATLAB 2020b with an Intel i7-7700 CPU (3.6 GHz) and 16 GB RAM under Windows 10.

5.1 Benchmark Function Experiments

5.1.1 Parameter Setting

The MaF1–MaF6 benchmark functions from the CEC2018 competition were selected with 3, 5, and 7 objectives to represent Many-objective optimization problems. MaF1 and MaF2 feature rule-based frontiers; MaF3 and MaF4 have multi-modal frontiers; MaF5 presents unevenly distributed frontiers; and MaF6 involves degenerate convex frontiers. Evaluation uses IGD and HV metrics: HV measures convergence and diversity, while IGD assesses convergence and uniformity. IGD is defined as:

$$\mathrm{IGD}\left(P^*, P\right) = \frac{\sum_{o \in P^*} d\left(o, P\right)}{\left|P^*\right|} \tag{31}$$

where P^* is the true Pareto front, P the approximate set, and $d\left(o, P\right)$ the minimum distance from o to point set P, where any point o belongs to the P^*. Lower IGD indicates closer and more uniformly covered approximations. Four state-of-the-art high-dimensional MOEAs—MOEA/D, RVEA, BiGE, and MEMO—were selected for comparison. Common parameters for MOCC/D and these algorithms are in Table 1. MOCC/D uses an additional population size of 240, while the others are from the EMO platform. The hyperparameters employed in our study (e.g., population size, crossover and mutation probabilities) were chosen primarily based on standard practices and default settings widely adopted.

Table 1: Parameter settings

Parameter	Settings
Population size	240
Fitness evaluations	D × 10^4 (D is number of decision variables)
Coding method	real encoding
Crossover operator	Simulated binary crossover (SBX)
Crossover probability	0.8
Mutation operator	Polynomial mutation
Mutation probability	1/D

5.1.2 Simulation Results and Analysis

Experiments were run 30 times independently for fair comparison. Tables 2 and 3 present mean and variance results for IGD and HV metrics, with best performances highlighted. MOCC/D consistently outperforms four benchmark algorithms across most test cases. For MaF1 and MaF2 (regular fronts),

MOCC/D excels at 3, 5, and 7 objectives, showing strong global and local search. For MaF3 and MaF4 (multi-modal fronts), MOCC/D leads at 3 objectives and remains competitive at higher dimensions, though MEMO slightly outperforms it at 5 and 7 objectives due to complex front adaptation. For MaF5 (uneven fronts), MOCC/D is less consistent, especially against MOEA/D, indicating challenges in maintaining diversity. For MaF6 (convex and degenerate fronts), MOCC/D performs best across all dimensions, demonstrating robustness. Overall, results validate MOCC/D's effectiveness: decomposition-based co-evolution reduces complexity, cumulative-probability mating balances search, and HV-guided multi-evolution improves uniformity and Pareto proximity, confirming its competitive edge on diverse high-dimensional problems.

Table 2: The IGD results after 30 runs of the five algorithms

Fun.	M	MOEA/D	BiGE	RVEA	MEMO	MOCC/D
MaF1	3	4.44E−02 (**6.24E−13**)	3.73E−02 (2.46E−06)	4.94E−02 (3.48E−08)	2.71E−02 (7.26E−08)	**2.38E−02** (2.58E−08)
	5	1.26E−01 (5.35E−07)	1.22E−01 (1.27E−05)	2.78E−01 (2.48E−04)	1.07E−01 (6.51E−07)	**9.85E−02** (**2.22E−07**)
	7	3.74E−01 (1.75E−03)	1.90E−01 (2.81E−05)	5.00E−01 (1.06E−02)	1.72E−01 (**1.83E−06**)	**1.68E−01** (5.20E−04)
MaF2	3	2.45E−02 (1.46E−07)	3.06E−02 (1.66E−06)	2.78E−02 (9.38E−07)	1.90E−02 (4.41E−08)	**1.67E−02** (**1.69E−08**)
	5	1.11E−01 (**2.03E−08**)	1.17E−01 (1.66E−05)	1.16E−01 (4.59E−07)	8.89E−02 (1.97E−06)	**8.11E−02** (1.11E−06)
	7	1.66E−01 (**5.65E−07**)	1.72E−01 (5.49E−05)	1.78E−01 (1.21E−05)	1.21E−01 (2.00E−06)	**1.17E−01** (4.81E−06)
MaF3	3	4.47E−02 (**2.87E−06**)	4.28E−02 (4.01E−04)	5.84E−02 (5.96E−03)	2.27E−02 (5.36E−06)	**2.12E−02** (9.67E−06)
	5	1.08E−01 (7.32E−06)	2.90E−01 (1.77E−02)	8.76E−02 (3.36E−03)	**5.60E−02** (**1.02E−06**)	6.31E−02 (1.69E−04)
	7	1.50E−01 (**2.83E−06**)	8.61E+02 (1.78E+06)	1.19E−01 (9.40E−03)	8.83E−02 (1.15E−05)	**7.57E−02** (2.08E−04)
MaF4	3	6.07E−01 (5.90E−04)	3.21E−01 (1.73E−01)	4.64E−01 (9.74E−02)	1.61E−01 (**7.42E−06**)	**1.56E−01** (6.72E−05)
	5	8.12E+00 (2.68E+00)	2.15E+00 (3.27E−03)	1.99E+00 (8.12E−03)	**1.73E+00** (**8.09E−04**)	2.62E+00 (2.43E+00)
	7	4.29E+01 (1.69E−01)	9.49E+00 (1.46E−01)	8.89E+00 (**8.18E−02**)	**7.13E+00** (9.31E−02)	1.21E+01 (2.72E+01)
MaF5	3	4.86E−01 (9.05E−01)	2.21E−01 (6.27E−05)	**1.56E−01** (**3.31E−07**)	1.60E−01 (3.02E−06)	9.04E−01 (2.26E+00)
	5	8.12E+00 (2.68E+00)	2.15E+00 (3.27E−03)	1.99E+00 (8.12E−03)	**1.73E+00** (**8.09E−04**)	2.62E+00 (2.43E+00)
	7	4.29E+01 (1.69E−01)	9.49E+00 (1.46E−01)	8.89E+00 (**8.18E−02**)	**7.13E+00** (9.31E−02)	1.21E+01 (2.72E+01)
MaF6	3	2.04E−02 (**2.76E−11**)	6.40E−03 (4.26E−07)	3.32E−02 (4.65E−05)	1.91E−03 (2.62E−09)	**1.55E−03** (8.28E−10)
	5	7.23E−02 (1.87E−02)	1.08E−02 (8.54E−06)	7.35E−02 (2.48E−04)	1.92E−03 (3.19E−09)	**1.58E−03** (**1.20E−09**)
	7	9.65E−02 (3.75E−02)	1.75E−02 (4.17E−05)	8.04E−02 (1.42E−04)	2.38E−03 (6.26E−06)	**1.55E−03** (**6.77E−10**)

Table 3: The HV results after 30 runs of the five algorithms

Fun.	M	MOEA/D	BiGE	RVEA	MEMO	MOCC/D
MaF1	3	2.96E−01 (**8.93E−12**)	2.99E−01 (4.96E−06)	2.85E−01 (1.45E−06)	3.13E−01 (9.86E−08)	**3.18E−01** (4.37E−08)
	5	1.80E−02 (**2.12E−08**)	1.82E−02 (2.33E−07)	5.46E−03 (3.12E−07)	2.06E−02 (4.47E−08)	**2.24E−02** (4.89E−08)
	7	1.12E−04 (1.79E−09)	4.20E−04 (8.24E−10)	3.11E−05 (**2.53E−10**)	4.77E−04 (1.12E−09)	**5.67E−04** (3.37E−09)
MaF2	3	2.20E−01 (1.27E−07)	2.19E−01 (3.21E−07)	2.13E−01 (1.02E−06)	2.24E−01 (6.91E−08)	**2.25E−01** (**3.04E−08**)
	5	4.99E−02 (**8.56E−09**)	5.47E−02 (1.14E−07)	4.71E−02 (2.41E−07)	5.50E−02 (1.21E−07)	**5.54E−02** (1.21E−07)
	7	1.12E−04 (5.48E−07)	**5.69E−02** (1.94E−07)	4.81E−02 (4.72E−07)	5.59E−02 (**1.92E−07**)	5.56E−02 (2.98E−07)
MaF3	3	1.28E+00 (**2.21E−06**)	1.27E+00 (2.14E−04)	1.25E+00 (1.06E−02)	1.29E+00 (8.44E−06)	**1.29E+00** (1.16E−05)
	5	1.59E+00 (4.92E−06)	1.38E+00 (6.32E−02)	1.60E+00 (1.64E−03)	**1.61E+00** (**1.86E−09**)	1.61E+00 (7.41E−06)
	7	1.91E+00 (1.22E−05)	1.94E−02 (1.13E−02)	1.91E+00 (3.54E−02)	**1.95E+00** (**1.67E−10**)	1.95E+00 (2.54E−06)
MaF4	3	4.49E+01 (**4.03E−02**)	4.58E+01 (1.76E+01)	4.18E+01 (3.09E+01)	4.72E+01 (7.62E−02)	**4.74E+01** (7.39E−02)
	5	5.76E+02 (**3.37E+04**)	6.83E+03 (5.72E+03)	2.03E+03 (4.23E+05)	**6.59E+03** (5.10E+04)	4.20E+03 (1.32E+06)
	7	2.91E+04 (**4.90E+07**)	9.16E+06 (8.05E+11)	9.62E+04 (2.72E+09)	**6.50E+06** (3.20E+11)	1.93E+06 (2.34E+12)
MaF5	3	4.52E+01 (8.48E+01)	4.81E+01 (1.41E−02)	**4.92E+01** (3.42E−03)	4.91E+01 (**1.19E−03**)	4.22E+01 (1.88E+02)
	5	2.46E+04 (3.73E+07)	4.19E+04 (2.71E+04)	4.27E+04 (9.26E+05)	**4.28E+04** (**3.52E+03**)	4.03E+04 (1.56E+07)
	7	2.25E+08 (1.87E+15)	4.68E+08 (2.24E+12)	4.75E+08 (2.55E+13)	**4.76E+08** (**5.75E+11**)	4.20E+08 (3.09E+15)
MaF6	3	1.26E−01 (**1.90E−12**)	1.32E−01 (1.26E−07)	1.18E−01 (1.49E−05)	1.34E−01 (6.63E−10)	**1.34E−01** (2.10E−10)
	5	8.16E−03 (5.95E−06)	9.12E−03 (9.46E−10)	8.31E−03 (1.58E−08)	9.24E−03 (**4.11E−10**)	**9.26E−03** (6.01E−10)
	7	1.95E−04 (1.83E−09)	2.09E−04 (6.54E−13)	1.90E−04 (6.67E−12)	2.11E−04 (3.77E−12)	**2.11E−04** (**3.09E−13**)

5.2 Model Simulation Experiment

5.2.1 Data Preparation

Based on [27], we simulate realistic data for the MEC system. Firstly, 10 mobile terminals are randomly deployed in the communication network, with the computing capacity value ranges from 0.1×10^9 MIPS to 0.2×10^9 MIPS, the computing power value in the working state and idle state respectively ranges from 100 to 300 mW and 5 to 10 mW, the communication power value in the transmission state and idle state ranges from 50 to 150 mW and 8 to 15 mW, the remaining battery value ranges from 0.05 to 01 J and the network radius value from 10 to 100 m. Secondly, the number sub-task of workflow applications ranges from 3 to 7, the data volume value of sub-task ranges from 10 to 50 MB, the type of sub-task ranges from 1 to 5 and the time deadline of task ranges from 0.1 to 2 s. Finally, the communication network based on routing node is presented, with the fixed maximum channel bandwidth value of 5×10^6 HZ, the fixed channel noise power value of 10^{-13}, the channel power gain value of 2 and the path loss objective value of 10^{-4}. Meanwhile, 6 edge servers are randomly deployed in the routing node, with the computing capacity value ranges from 5×10^9 MIPS to 6×10^9 MIPS and the number of soft type ranges from 2 to 3.

5.2.2 Model Solution Results and Analysis

For fair comparison, experiments were run 30 times independently. Table 4 and Fig. 5 show IGD and HV comparing MOCC/D with MOEA/D, RVEA, BiGE, and MEMO, with best results highlighted. MOCC/D outperforms all others in the multi-objective offloading model, improving mean IGD by 119.44%, 31.31%, 47.98%, and 38.64%, while reducing variance by 62.66%, 20.8%, 75.7%, and 23.56%. For HV, MOCC/D improves mean performance by 3, 2, 2, and 1 orders of magnitude, though variance is not always better. Overall, MOCC/D delivers solutions closer to the true Pareto front with more even distribution.

Table 4: The IGD and HV results after 30 runs of the five algorithms

S		MOEA/D	BiGE	RVEA	MEMO	MOCC/D
IGD	Mean	8.69E−01	5.20E−01	5.76E−01	5.49E−01	**3.96E−01**
	Var	6.49E−02	4.82E−02	7.01E−02	4.93E−02	**3.99E−02**
	Best	3.82E−01	2.36E−01	2.39E−01	2.10E−01	**9.00E−02**
HV	Mean	8.52E−08	8.23E−07	1.98E−07	2.71E−06	**1.07E−05**
	Var	**1.56E−13**	4.34E−12	5.19E−13	1.03E−11	1.00E−10
	Best	2.15E−06	**1.04E−05**	3.09E−06	1.04E−05	9.45E−06

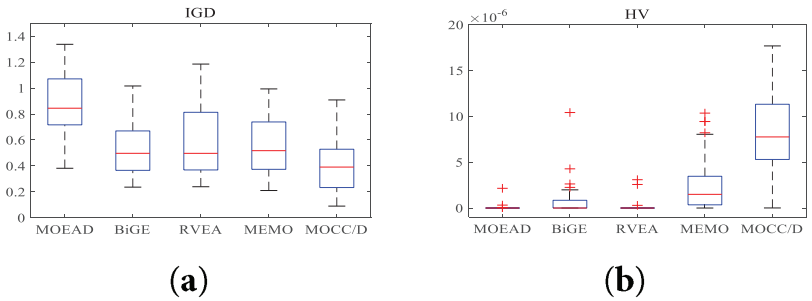

Figure 5: The IGD and HV box diagram after 30 runs of the five algorithms. (**a**) The IGD box diagram. (**b**) The HV box diagram

Table 5 shows the mean values of each objective for the IGD optimal runs over 30 iterations, with the best results highlighted. MOCC/D outperforms others in average delay, energy consumption, and energy balance, reducing delay by 24.11%, 5.11%, 6.12%, and 15.78%, and energy consumption by 18.96%, 1.11%, 4.0%, and 15.93%. Energy balance improves by 126.55%, 90.06%, 95.76%, and 130.77%. These results demonstrate MOCC/D's suitability for time-sensitive MEC offloading, delivering accurate task matching and stable user experience. Its delay is only 3.6% higher than the optimal, which is acceptable. Overall, MOCC/D exhibits stronger global and local search performance than the four compared algorithms.

Table 5: The optimization objectives mean value of the optimal IGD in the 30 times

S	MOEA/D	BiGE	RVEA	MEMO	MOCC/D
Makespan	2.2015E+00	**1.9616E+00**	2.0372E+00	2.1233E+00	2.0321E+00
Time-delay	1.0827E+00	9.0169E−01	9.2057E−01	1.0100E+00	**8.7234E−01**
Energy-consumption	4.5975E−02	3.8849E−02	4.0015E−02	4.4590E−02	**3.8646E−02**
Energy-balance	1.4633E−03	1.2271E−03	1.2644E−03	1.5013E−03	**6.4591E−04**

5.2.3 Model Parameter Analysis

To analyze convergence, Fig. 6 shows the population evolution of model objectives using the proposed MOCC/D. Table 5 indicates good convergence, with makespan, time-delay, and energy consumption stabilizing after about 40 iterations, and energy balance stabilizing after approximately 125 iterations Fig. 6 shows the evolution of optimization objectives using the proposed MOCC/D algorithm. Makespan, time-delay, and energy consumption stabilize after about 40 iterations, while energy balance converges around 125 iterations, indicating good overall convergence.

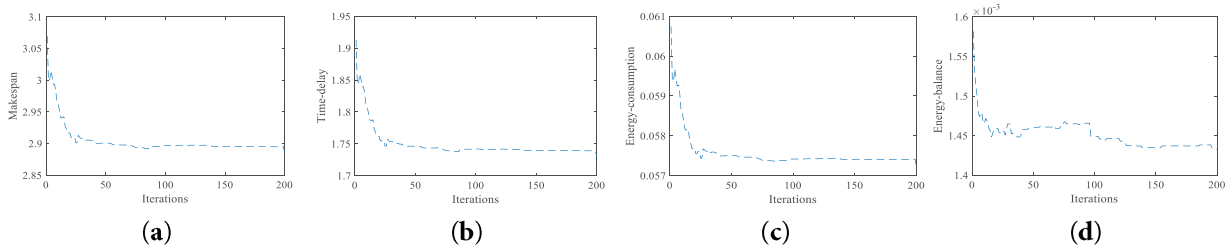

Figure 6: The convergence analysis of model optimization objective. (**a**) Makespan. (**b**) Time-delay. (**c**) Energy-consumption. (**d**) Energy-balance

Fig. 7 illustrates the impact of the energy balance strategy on device battery levels. Without it, devices MD1, MD2, MD5, and MD7 experience significant depletion. With the strategy, the remaining energy of MD2, MD5, and MD7 improves by 12.98%, 2.14%, and 0.86%, respectively, effectively protecting low-energy nodes. Although MD1 shows a slight 0.42% drop, it remains within acceptable limits. Devices with higher initial energy—MD3, MD4, MD6, and MD8—show increases of 1.8%, 8.6%, 47.98%, and 0.9%, respectively. These results confirm that the strategy redistributes workload to balance energy consumption and prolong overall system lifetime.

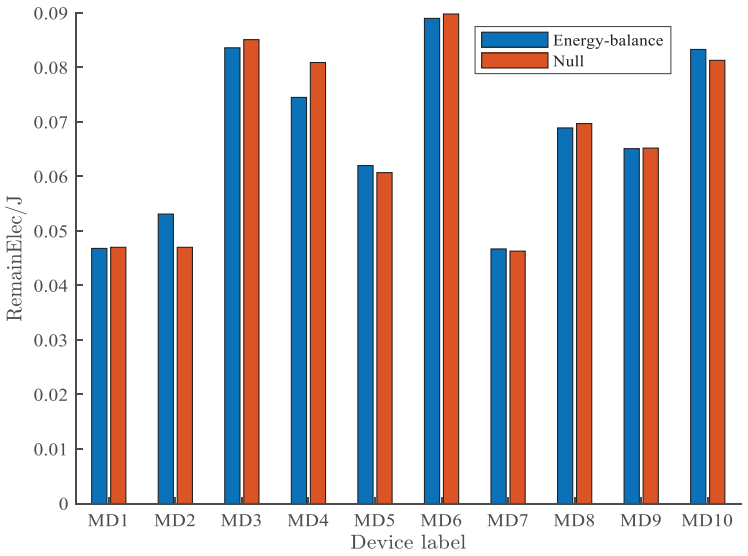

Figure 7: The distribution of the terminals remaining battery under two conditions

6 Conclusion

This paper addresses high-dimensional multi-objective computation offloading in MEC for serial isomerism tasks, considering task characteristics, complex objectives, and flexible resource scheduling. We model the problem as a Many-objective optimization and propose an intelligent offloading algorithm within a flexible optimization framework, featuring an improved multi-objective cooperative co-evolutionary algorithm based on decomposition (MOCC/D) and a flexible scheduling strategy. Evaluation on benchmark functions and simulations demonstrates the superior performance of the proposed approach.

Acknowledgement: The authors are deeply grateful to all team members involved in this research.

Funding Statement: Funding: This work was supported by Youth Talent Project of Scientific Research Program of Hubei Provincial Department of Education under Grant Q20241809, Doctoral Scientific Research Foundation of Hubei University of Automotive Technology under Grant 202404.

Author Contributions: Study conception and design: Zheng Yao; data collection: Zheng Yao and Puqing Chang; analysis and interpretation of results: Zheng Yao and Puqing Chang; draft manuscript preparation: Zheng Yao. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: Data available on request from the authors.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Zhang Y, Cui X, Zhao Q. A multi-objective joint task offloading scheme for vehicular edge computing. Comput Mater Contin. 2025;84(2):2355–73. doi:10.32604/cmc.2025.065430.
2. Bhaskaran S, Muthuraman S. A comprehensive study of resource provisioning and optimization in edge computing. Comput Mater Contin. 2025;83(3):5037–70. doi:10.32604/cmc.2025.062657.

3. Hasan MK, Jahan N, Ahmad Nazri MZ, Islam S, Khan MA, Alzahrani AI, et al. Federated learning for computational offloading and resource management of vehicular edge computing in 6G-V2X network. IEEE Trans Consum Electron. 2024;70(1):3827–47. doi:10.1109/TCE.2024.3357530.
4. Dong S, Tang J, Abbas K, Hou R, Kamruzzaman J, Rutkowski L, et al. Task offloading strategies for mobile edge computing: a survey. Comput Netw. 2024;254(6):110791. doi:10.1016/j.comnet.2024.110791.
5. Wu G, Chen X, Shen Y, Xu Z, Zhang H, Shen S, et al. Combining Lyapunov optimization with actor-critic networks for privacy-aware IIoT computation offloading. IEEE Internet Things J. 2024;11(10):17437–52. doi:10.1109/JIOT.2024.3357110.
6. Patel M, Naughton B, Chan C, Sprecher N, Abeta S, Neal A. Mobile-edge computing—introductory technical white paper. In: Sophia antipolis. France: Mobile-Edge Computing (MEC) Industry Initiative; 2014.
7. Wu H, Lu Y, Ma H, Xing L, Deng K, Lu X. A survey on task type-based computation offloading in mobile edge networks. Ad Hoc Netw. 2025;169(2):103754. doi:10.1016/j.adhoc.2025.103754.
8. Zhang S, Yi N, Ma Y. A survey of computation offloading with task types. IEEE Trans Intell Transp Syst. 2024;25(8):8313–33. doi:10.1109/TITS.2024.3410896.
9. Li QP, Zhao JH, Gong Y. Computation offloading and resource management scheme in mobile edge computing. Telecommun Sci. 2019;35(3):36–46. (In Chinese).
10. Yang L, Zhang H, Li M, Guo J, Ji H. Mobile edge computing empowered energy efficient task offloading in 5G. IEEE Trans Veh Technol. 2018;67(7):6398–409. doi:10.1109/TVT.2018.2799620.
11. Zhou F, Wu Y, Hu RQ, Qian Y. Computation rate maximization in UAV-enabled wireless-powered mobile-edge computing systems. IEEE J Sel Areas Commun. 2018;36(9):1927–41. doi:10.1109/JSAC.2018.2864426.
12. Min M, Xiao L, Chen Y, Cheng P, Wu D, Zhuang W. Learning-based computation offloading for IoT devices with energy harvesting. IEEE Trans Veh Technol. 2019;68(2):1930–41. doi:10.1109/TVT.2018.2890685.
13. Kang MC, Li X, Ji H, Zhang HL. Collaborative computation offloading exploring task dependencies in small cell networks. J Beijing Univ Posts Telecommun. 2021;44(1):72–8. (In Chinese). doi:10.13190/j.jbupt.2020-115.
14. Darchini-Tabrizi M, Roudgar A, Entezari-Maleki R, Sousa L. Distributed deep reinforcement learning for independent task offloading in mobile edge computing. J Netw Comput Appl. 2025;240(6):104211. doi:10.1016/j.jnca.2025.104211.
15. Saheed YK, Abdulganiyu OH, Tchakoucht TA. Modified genetic algorithm and fine-tuned long short-term memory network for intrusion detection in the Internet of Things networks with edge capabilities. Appl Soft Comput. 2024;155(4):111434. doi:10.1016/j.asoc.2024.111434.
16. Wu Y, Qian LP, Ni K, Zhang C, Shen X. Delay-minimization nonorthogonal multiple access enabled multi-user mobile edge computation offloading. IEEE J Sel Top Signal Process. 2019;13(3):392–407. doi:10.1109/JSTSP.2019.2893057.
17. Gorlatova M, Inaltekin H, Chiang M. Characterizing task completion latencies in multi-point multi-quality fog computing systems. Comput Netw. 2020;181(6):107526. doi:10.1016/j.comnet.2020.107526.
18. Meng J, Tan H, Li XY, Han Z, Li B. Online deadline-aware task dispatching and scheduling in edge computing. IEEE Trans Parallel Distrib Syst. 2020;31(6):1270–86. doi:10.1109/TPDS.2019.2961905.
19. Verba N, Chao KM, Lewandowski J, Shah N, James A, Tian F. Modeling industry 4.0 based fog computing environments for application analysis and deployment. Future Gener Comput Syst. 2019;91(1):48–60. doi:10.1016/j.future.2018.08.043.
20. Wang Y, Ge HB, Feng AQ. Computation offloading strategy in cloud-assisted mobile edge computing. Comput Eng. 2020;46(8):27–34. doi:10.1109/icccbda49378.2020.9095689.
21. Li C, Zhang Y, Gao X, Luo Y. Energy-latency tradeoffs for edge caching and dynamic service migration based on DQN in mobile edge computing. J Parallel Distrib Comput. 2022;166(4):15–31. doi:10.1016/j.jpdc.2022.03.001.
22. Cao Y, Mao H. High-dimensional multi-objective optimization strategy based on directional search in decision space and sports training data simulation. Alex Eng J. 2022;61(1):159–73. doi:10.1016/j.aej.2021.04.077.
23. Zhang Q, Li H. MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput. 2007;11(6):712–31. doi:10.1109/TEVC.2007.892759.

24. Cheng R, Jin Y, Olhofer M, Sendhoff B. A reference vector guided evolutionary algorithm for many-objective optimization. IEEE Trans Evol Comput. 2016;20(5):773–91. doi:10.1109/TEVC.2016.2519378.
25. Li M, Yang S, Liu X. Bi-goal evolution for many-objective optimization problems. Artif Intell. 2015;228:45–65. doi:10.1016/j.artint.2015.06.007.
26. Yuan J, Liu HL, Gu F. A cost value based evolutionary many-objective optimization algorithm with neighbor selection strategy. In: 2018 IEEE Congress on Evolutionary Computation (CEC); 2018 Jul 8–13; Rio de Janeiro, Brazil. doi:10.1109/CEC.2018.8477649.
27. Rui L, Yang Y, Gao Z, Qiu X. Computation offloading in a mobile edge communication network: a joint transmission delay and energy consumption dynamic awareness mechanism. IEEE Internet Things J. 2019;6(6):10546–59. doi:10.1109/JIOT.2019.2939874.