

Dynamic Multi-Objective Optimization for Batch-wise Task Assignment and Resource Allocation in Mobile Edge Computing

Yingying Sun

National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University

Zhiming Dong

Liaoning Engineering Laboratory of Data Analytics and Optimization for Smart Industry

Gongshu Wang

wanggongshu@ise.neu.edu.cn

National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University

Research Article

Keywords: Mobile edge computing, Task assignment, Resource allocation, Dynamic multi-objective optimization, Evolutionary computation, Location-resource coupling

Posted Date: October 14th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-11200633/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Dynamic Multi-Objective Optimization for Batch-wise Task Assignment and Resource Allocation in Mobile Edge Computing

Yingying Sun^{1,2}, Zhiming Dong^{3,4}, Gongshu Wang^{1*}

¹National Frontiers Science Center for Industrial Intelligence and Systems Optimization, Northeastern University, Shenyang, 110819, China.

²Key Laboratory of Data Analytics and Optimization for Smart Industry (Northeastern University), Ministry of Education, Shenyang, 110819, China.

³Liaoning Engineering Laboratory of Data Analytics and Optimization for Smart Industry, Shenyang, 110819, China.

⁴Liaoning Key Laboratory of Manufacturing System and Logistics Optimization, Shenyang, 110819, China.

*Corresponding author(s). E-mail(s): wanggongshu@ise.neu.edu.cn;

Contributing authors: 2210350@stu.neu.edu.cn; dongzm@neu.edu.cn;

Abstract

In practical mobile edge computing systems, computation tasks arrive in successive batches while task attributes, wireless conditions, and available resources vary across decision slots. This paper formulates batch-wise task assignment and communication/computing resource allocation as a constrained dynamic bi-objective optimization problem and proposes a Location-Resource Coupled Dynamic NSGA-II (LRC-DNSGA-II). Its Location-Resource Coupled Variation operator captures the dependency between discrete assignment and continuous allocation by restricting assignment mutation to accessible execution sites and resetting associated resource-weight genes when locations change. A task-feature-matched dynamic response mechanism reuses relevant decision information across successive batches. Experiments under dynamic batch-wise MEC scenarios show that LRC-DNSGA-II significantly outperforms four comparison methods in hypervolume and inverted generational distance. Ablation studies indicate that coupled variation mainly improves final approximation quality, whereas dynamic response primarily accelerates adaptation after batch transitions.

Keywords: Mobile edge computing, Task assignment, Resource allocation, Dynamic multi-objective optimization, Evolutionary computation, Location-resource coupling

1 Introduction

With the rapid proliferation of the Internet of Things (IoT) and intelligent manufacturing, end devices increasingly generate computation-intensive tasks in successive batches, creating growing demands for low-latency and energy-efficient task processing [1–5]. However, the limited battery capacity and computational capability of end devices often make local execution inadequate for delay-sensitive applications [6, 7]. Offloading tasks to remote cloud servers can alleviate local resource limitations, but may introduce substantial transmission delay and backhaul congestion, while increasing the risk of exposing sensitive industrial data [8–10]. Mobile

edge computing (MEC) has therefore emerged as an effective paradigm by deploying computing resources closer to end devices, enabling computation offloading with reduced communication latency and improved responsiveness [11, 12].

Despite these advantages, practical MEC systems operate under inherently time-varying task and resource conditions [13]. The successive task batches generated by end devices form an online decision process over consecutive time slots, rather than a one-shot optimization problem defined over a fixed task set [14, 15]. For example, in industrial inspection, cameras and sensors may generate multiple image-recognition and data-processing

tasks within each inspection cycle [16]. Across consecutive slots, the number and attributes of newly arrived tasks, wireless conditions, and available communication and computing resources may all vary [17, 18]. Consequently, both the feasible decision space and the trade-offs among multiple optimization objectives can evolve over time [19–21]. Decisions optimized for one task batch and system state may therefore become ineffective or even infeasible for subsequent batches. This temporal evolution makes joint task assignment and resource allocation substantially more challenging than in static settings. Nevertheless, many existing MEC formulations are still based on a fixed task set and a static system state, limiting their ability to capture and adapt to batch-wise task arrivals and evolving operating conditions.

Beyond the temporal evolution of system states, the resulting optimization problem also exhibits a tightly coupled mixed discrete–continuous structure [22]. Task-assignment decisions determine whether a task is executed locally or offloaded to an edge server, thereby determining which communication and computing resources are involved in its execution. Meanwhile, unfinished tasks carried over from previous slots may continue to occupy part of the available edge computing capacity, further affecting the resources accessible to newly arrived tasks. Consequently, a change in task assignment can alter not only the feasible region of resource allocation but also the resource-sharing context in which the associated continuous variables take effect. The resulting problem therefore involves discrete assignment decisions, continuous communication and computing resource variables, nonlinear constraints, and time-varying feasibility conditions [23–25]. Such location-dependent coupling makes the search landscape highly sensitive to both system-state changes and assignment variations, substantially increasing the difficulty of exploring the mixed decision space effectively. Accordingly, jointly optimizing task assignment and resource allocation in dynamic MEC remains a challenging problem.

Existing dynamic multi-objective methods commonly adapt to environmental changes by reusing historical populations. However, their direct application to batch-wise MEC is challenging. Successive task batches may differ in size and attributes, weakening task-level correspondence across slots and reducing the effectiveness of direct population reuse. Meanwhile, changes in task assignment alter the associated resource-sharing context, potentially invalidating previously suitable resource-allocation variables. Effective optimization therefore requires both task-relevant cross-slot information transfer and consistent coordination between assignment and resource-allocation decisions. To address these challenges, we formulate batch-wise task assignment and resource allocation in MEC as

a constrained dynamic bi-objective optimization problem minimizing task completion delay and device energy consumption, and develop a Location-Resource Coupled Dynamic NSGA-II (LRC-DNSGA-II). The algorithm integrates task-relevant cross-slot reuse with location-resource coupled variation to adapt to evolving task batches while preserving consistency between assignment and resource-allocation decisions. The main contributions are summarized as follows:

1. We formulate batch-wise task assignment and resource allocation in MEC as a constrained dynamic bi-objective optimization problem that jointly minimizes task completion delay and device energy consumption. The formulation captures successive task-batch arrivals, time-varying communication and computing resources, location-dependent resource sharing, and the carry-over impact of unfinished tasks on subsequent decision slots.
2. We develop a Location-Resource Coupled Dynamic NSGA-II (LRC-DNSGA-II) for the resulting dynamic mixed discrete–continuous problem. Its core Location-Resource Coupled Variation operator restricts assignment mutation to accessible execution sites and resets the associated resource-weight genes when the execution location changes, thereby maintaining consistency between discrete assignment decisions and continuous resource allocation. A task-feature-matched dynamic response mechanism is further introduced to reuse task-relevant historical decision information across successive batches.
3. Extensive experiments demonstrate that LRC-DNSGA-II consistently achieves superior dynamic multi-objective optimization performance compared with representative baselines. Further ablation analysis confirms the complementary roles of its two key components: Location-Resource Coupled Variation primarily improves final approximation quality, while the task-feature-matched dynamic response enhances adaptation to environmental changes.

The remainder of this paper is organized as follows. Section 2 reviews the relevant literature. Section 3 introduces the system model and formulates the dynamic batch-wise task assignment and resource allocation problem. Section 4 details the proposed LRC-DNSGA-II and the deployment solution selection procedure. Section 5 describes the experimental setup and reports the performance evaluation and analysis. Finally, Section 6 concludes the paper.

2 Related Work

MEC task assignment and resource allocation have received extensive research attention [2, 25–29]. Existing studies can be broadly grouped into optimization under a given system state, dynamic MEC scheduling, and

dynamic multi-objective optimization for time-varying MEC environments.

2.1 MEC Optimization Under a Given System State

For a given task set and system state, existing MEC studies jointly optimize task placement, server selection, and communication and computing resource allocation [30, 31]. Conflicting objectives, such as task latency and device energy consumption, are often converted into scalar optimization problems through predefined preference parameters. When such preferences are not known in advance, Pareto-based methods preserve multiple trade-off solutions to support different operating requirements [32–35]. These studies establish important formulations and solution techniques for joint task assignment and resource allocation, but generally assume that the task set and system state remain fixed during optimization.

2.2 Dynamic MEC Scheduling

Dynamic MEC scheduling updates task and resource decisions as task demands, wireless conditions, user mobility, and resource availability evolve over time [36]. Prediction-based approaches exploit anticipated system states to support proactive scheduling and resource configuration. Plachy et al. [37] combined mobility and channel prediction to preallocate computing resources and select communication paths for real-time task assignment. Su et al. [38] incorporated idle mobile-device computing resources into cloud–edge collaboration and developed an online scheduling method for tasks with flexible execution windows.

Lyapunov optimization has been widely adopted for MEC problems involving long-term objectives and time-varying constraints. By transforming the original stochastic control problem into a sequence of per-slot optimization problems, it has been applied to joint task assignment and resource scheduling in energy-harvesting MEC systems [17], communication and computing coordination in vehicular MEC networks [39, 40], task assignment in UAV-assisted IoT–MEC systems [41], and resource coordination between asynchronous federated edge learning and local computing tasks [42].

Learning-based approaches instead learn scheduling policies from interactions with dynamic MEC environments. Deep reinforcement learning has been applied to decentralized task offloading and function offloading in edge computing systems [43, 44]. Multiagent, federated, and hierarchical reinforcement learning have further been used for distributed task coordination, service caching, workload allocation, and load balancing [45–47]. Recurrent and transformer-based models have also been introduced to capture temporal dependencies in task migration and edge–cloud resource scheduling [48, 49].

These approaches provide effective mechanisms for adapting MEC operations to time-varying system conditions. However, they typically seek a single scheduling policy or a scalarized objective under a specified preference structure. When multiple conflicting objectives must be optimized simultaneously and their trade-offs evolve with the system state, the problem is more naturally addressed from a dynamic multi-objective optimization perspective.

2.3 Dynamic Multi-Objective Optimization for MEC

In dynamic MEC systems, variations in newly arrived tasks, wireless conditions, and available computing resources can change not only the feasible decision space but also the trade-offs among multiple objectives. Dynamic multi-objective evolutionary algorithms address such changes by adapting the solution population as the optimization environment evolves. DMOEA/D-COPMEC characterizes environmental changes and combines similarity-aware memory, population prediction, and adaptive decomposition for dynamic computation offloading [50]. In parked-vehicle-assisted edge computing, DMOEA/D-MSFR employs population-center positioning, LSTM-based prediction, and adaptive weight adjustment to respond to changes in the task-assignment environment [51]. For vehicular edge computing, DCMOA coordinates task assignment and service caching through multi-population coevolution and a dynamic feedback mechanism [52].

2.4 Research Gap and Motivation

Existing studies have advanced MEC optimization under both static and time-varying environments. Dynamic scheduling methods adapt decisions to evolving system conditions, while dynamic multi-objective evolutionary algorithms maintain trade-off solutions as the environment changes. However, batch-wise task assignment and resource allocation introduce two challenges that remain insufficiently addressed.

First, successive task batches may differ in size and task attributes, weakening task-level correspondence across slots and reducing the effectiveness of directly reusing historical populations. Cross-slot knowledge transfer therefore requires identifying historical decision information that remains relevant to newly arrived tasks.

Second, discrete task assignment and continuous resource allocation are tightly coupled within each slot. Changing a task’s execution location alters the communication and computing resources involved and thus changes its resource-sharing context. Resource-allocation information suitable for one location may therefore become inappropriate after an assignment change.

Accordingly, dynamic optimization for batch-wise MEC requires both task-relevant cross-slot information transfer and location-aware coordination of assignment and resource allocation. These needs motivate the task-feature-matched dynamic response and Location-Resource Coupled Variation in LRC-DNSGA-II.

3 System Model and Problem Formulation

The dynamic MEC system architecture is illustrated in Fig. 1. It consists of a set of end devices $\mathcal{N} = \{1, \dots, N\}$ and a set of edge servers $\mathcal{M} = \{1, \dots, M\}$. The scheduling horizon is divided into T fixed-length decision slots indexed by $t \in \{1, \dots, T\}$, each with duration Δ . Computational tasks arrive online in successive batches, with a new batch becoming available at the beginning of each decision slot. After observing the current task and system states, the MEC controller jointly determines task assignment and communication/computing resource allocation for the current slot. Because task characteristics, wireless conditions, and available computing resources may vary across slots, the resulting decision problem evolves over time.

A task generated by device i is denoted by $\tau_{i,k}$, where k is its task index at that device. Its intrinsic attributes are

$$\mathcal{A}_{i,k} = (L_{i,k}, D_{i,k}, O_{i,k}), \quad (1)$$

where $L_{i,k}$ is the required number of CPU cycles, and $D_{i,k}$ and $O_{i,k}$ are the input and output data sizes, respectively. Tasks are assumed to be independent and non-splittable, and each task is executed entirely at one location. The execution-location set is $\mathcal{M}_0 = \{0\} \cup \mathcal{M}$, where 0 denotes local execution and $j \in \mathcal{M}$ denotes execution at edge server j .

3.1 Task and System State

At the beginning of decision slot t , the computation-active task set consists of newly arrived tasks $\mathcal{T}^{\text{new}}(t)$ and admitted unfinished tasks $\mathcal{T}^{\text{adm}}(t)$, and is defined as

$$\mathcal{T}^{\text{act}}(t) = \mathcal{T}^{\text{new}}(t) \cup \mathcal{T}^{\text{adm}}(t), \quad (2)$$

where $\mathcal{T}^{\text{new}}(t) \cap \mathcal{T}^{\text{adm}}(t) = \emptyset$.

For each active task $\tau_{i,k}$, $\ell_{i,k}(t)$ denotes its remaining computational workload at the beginning of slot t . The remaining workloads of the active tasks are represented by

$$\ell(t) = \{\ell_{i,k}(t) \mid \tau_{i,k} \in \mathcal{T}^{\text{act}}(t)\}. \quad (3)$$

A newly arrived task satisfies $\ell_{i,k}(t) = L_{i,k}$, whereas an admitted unfinished task satisfies $0 < \ell_{i,k}(t) \leq L_{i,k}$. Each task in $\mathcal{T}^{\text{adm}}(t)$ retains its previously assigned execution location and computing rate until completion. Only the tasks in $\mathcal{T}^{\text{new}}(t)$ participate in the optimization

performed at slot t . If $\mathcal{T}^{\text{new}}(t) = \emptyset$, no new optimization is invoked, and only the remaining-workload state is updated.

Let $\mathcal{L}_i^{\text{adm}}(t)$ and $\mathcal{E}_j^{\text{adm}}(t)$ denote the admitted unfinished tasks whose assigned execution locations are device i and edge server j , respectively. For each newly arrived task $\tau_{i,k} \in \mathcal{T}^{\text{new}}(t)$, let $x_{i,k,m}(t) \in \{0, 1\}$ indicate whether the task is assigned to execution location $m \in \mathcal{M}_0$, where $m = 0$ denotes local execution and $m = j \in \mathcal{M}$ denotes execution at edge server j . The indicator $\chi_{i,j}(t) \in \{0, 1\}$ specifies whether device i can access edge server j in slot t . Computing capacity released by a completed task during slot t becomes available for new admission at the beginning of slot $t+1$.

The observed system state at slot t is summarized by

$$\mathcal{S}(t) = \{\mathcal{T}^{\text{new}}(t), \mathcal{T}^{\text{adm}}(t), \ell(t), \chi(t), \mathbf{H}(t), \mathbf{B}(t), \mathbf{F}^{\text{max}}(t)\}, \quad (4)$$

Here, $\mathbf{H}(t)$ represents the wireless channel conditions, $\mathbf{B}(t)$ represents the available communication bandwidths, and $\mathbf{F}^{\text{max}}(t)$ represents the available local and edge computing capacities.

New task arrivals and exogenous changes in wireless channel conditions, communication bandwidths, and computing capacities cause $\mathcal{S}(t)$ to evolve across decision slots. In a stateful deployment, the deployed scheduling decision may further affect the next-slot state through the remaining workloads, admitted unfinished tasks, and reserved computing resources. Their evolution across consecutive slots is described by the inter-slot state-transition model below.

3.2 Local Execution Model

Among the tasks in $\mathcal{T}^{\text{new}}(t)$, those assigned to local execution at device i constitute the set

$$\mathcal{L}_i(t) = \{\tau_{i,k} \in \mathcal{T}^{\text{new}}(t) \mid x_{i,k,0}(t) = 1\}. \quad (5)$$

For each $\tau_{i,k} \in \mathcal{L}_i^{\text{adm}}(t)$, let $\bar{f}_{i,k}^{\text{loc}}(t)$ denote its previously assigned local computing rate.

The residual local computing capacity available to newly arrived tasks is

$$F_i^{\text{res}}(t) = F_i^{\text{max}}(t) - \sum_{\tau_{i,k} \in \mathcal{L}_i^{\text{adm}}(t)} \bar{f}_{i,k}^{\text{loc}}(t), \quad (6)$$

where $F_i^{\text{max}}(t)$ denotes the total local computing capacity of device i at slot t .

For each $\tau_{i,k} \in \mathcal{L}_i(t)$, $f_{i,k}^{\text{loc}}(t)$ denotes its assigned local computing rate. The time required to complete its remaining workload is

$$T_{i,k}^{\text{loc}}(t) = \frac{\ell_{i,k}(t)}{f_{i,k}^{\text{loc}}(t)}. \quad (7)$$

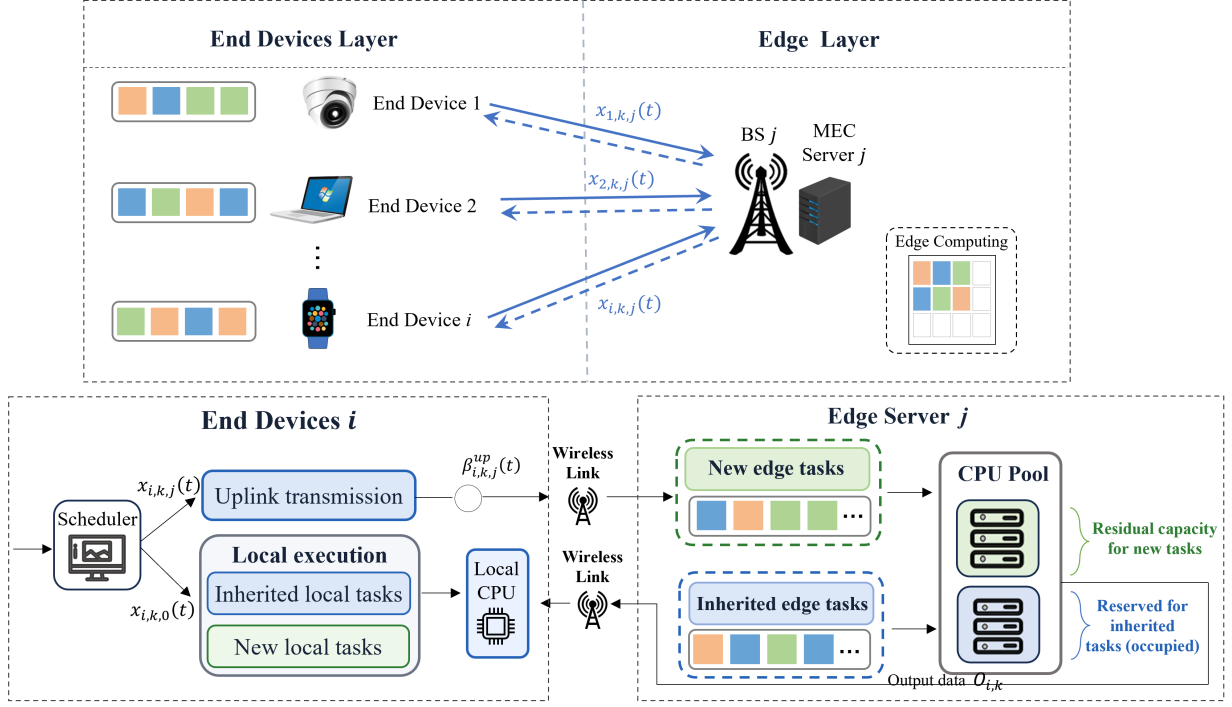


Fig. 1 Dynamic MEC System Architecture

The corresponding local computing energy consumption is

$$E_{i,k}^{\text{loc}}(t) = \kappa_i \ell_{i,k}(t) (f_{i,k}^{\text{loc}}(t))^2, \quad (8)$$

where κ_i is the effective switched capacitance coefficient of device i .

3.3 Edge Execution Model

For a task assigned to an edge server, the execution process consists of uplink transmission, edge computation, and result return. For a new task assigned to server j , the uplink transmission rate is

$$R_{i,k,j}^{\text{up}}(t) = \beta_{i,k,j}^{\text{up}}(t) B_j^{\text{up}}(t) \log_2 \left(1 + \frac{P_i^{\text{tx}} h_{i,j}(t)}{N_0 \beta_{i,k,j}^{\text{up}}(t) B_j^{\text{up}}(t)} \right), \quad (9)$$

where $\beta_{i,k,j}^{\text{up}}(t)$ is the fraction of server j 's uplink bandwidth allocated to the task, $B_j^{\text{up}}(t)$ is the available uplink bandwidth, P_i^{tx} is the transmission power, $h_{i,j}(t)$ is the channel gain, and N_0 is the noise power spectral density.

The corresponding uplink transmission duration is

$$T_{i,k,j}^{\text{up}}(t) = \frac{D_{i,k}}{R_{i,k,j}^{\text{up}}(t)}. \quad (10)$$

Uplink transmission is completed in the slot in which the task is assigned, whereas edge computation may span multiple slots. For each $\tau_{i,k} \in \mathcal{E}_j^{\text{adm}}(t)$, let $\bar{f}_{i,k,j}^{\text{edge}}(t)$ denote its previously assigned edge computing rate. The

residual computing capacity available to new tasks is

$$F_j^{\text{res}}(t) = F_j^{\text{max}}(t) - \sum_{\tau_{i,k} \in \mathcal{E}_j^{\text{adm}}(t)} \bar{f}_{i,k,j}^{\text{edge}}(t), \quad (11)$$

where $F_j^{\text{max}}(t)$ denotes the total computing capacity available to MEC tasks at server j in slot t .

Among the new tasks, those assigned to server j constitute

$$\mathcal{E}_j(t) = \{\tau_{i,k} \in \mathcal{T}^{\text{new}}(t) \mid x_{i,k,j}(t) = 1\}. \quad (12)$$

For each $\tau_{i,k} \in \mathcal{E}_j(t)$, $f_{i,k,j}^{\text{edge}}(t)$ denotes its assigned computing rate. Its computation delay is

$$T_{i,k,j}^{\text{cmp}}(t) = \frac{\ell_{i,k}(t)}{f_{i,k,j}^{\text{edge}}(t)}, \quad (13)$$

After edge computation is completed, the result is returned to the corresponding device. The result-return duration is

$$T_{i,k,j}^{\text{down}}(t) = \frac{O_{i,k}}{R_{i,j}^{\text{down}}(t)}, \quad (14)$$

where $R_{i,j}^{\text{down}}(t)$ denotes the effective downlink rate from edge server j to device i , which is determined by the underlying downlink communication system. The downlink rate is not optimized in the formulated problem.

The time required to complete task $\tau_{i,k}$ at edge server j is

$$T_{i,k,j}^{\text{edge}}(t) = T_{i,k,j}^{\text{up}}(t) + T_{i,k,j}^{\text{cmp}}(t) + T_{i,k,j}^{\text{down}}(t). \quad (15)$$

The device energy consumed during edge execution includes energy for uplink transmission, device idling during edge computation, and result reception:

$$E_{i,k,j}^{\text{edge}}(t) = P_i^{\text{tx}} T_{i,k,j}^{\text{up}}(t) + P_i^{\text{idle}} T_{i,k,j}^{\text{cmp}}(t) + P_i^{\text{rx}} T_{i,k,j}^{\text{down}}(t), \quad (16)$$

where P_i^{idle} and P_i^{rx} denote the idle and receiving powers of device i , respectively.

3.4 Dynamic Bi-Objective Optimization Problem

For each $\tau_{i,k} \in \mathcal{T}^{\text{new}}(t)$, the completion delay and device energy consumption are, respectively,

$$T_{i,k}(t) = \begin{cases} T_{i,k}^{\text{loc}}(t), & x_{i,k,0}(t) = 1, \\ T_{i,k,j}^{\text{edge}}(t), & x_{i,k,j}(t) = 1, \end{cases} \quad (17a)$$

$$E_{i,k}^{\text{dev}}(t) = \begin{cases} E_{i,k}^{\text{loc}}(t), & x_{i,k,0}(t) = 1, \\ E_{i,k,j}^{\text{edge}}(t), & x_{i,k,j}(t) = 1, \end{cases} \quad (17b)$$

When $|\mathcal{T}^{\text{new}}(t)| > 0$, the average task completion delay is defined as

$$\bar{T}(t) = \frac{\sum_{\tau_{i,k} \in \mathcal{T}^{\text{new}}(t)} T_{i,k}(t)}{|\mathcal{T}^{\text{new}}(t)|}, \quad (18)$$

and the average device energy consumption is defined as

$$\bar{E}^{\text{dev}}(t) = \frac{\sum_{\tau_{i,k} \in \mathcal{T}^{\text{new}}(t)} E_{i,k}^{\text{dev}}(t)}{|\mathcal{T}^{\text{new}}(t)|}. \quad (19)$$

Let the decision-variable sets for the newly arrived tasks in slot t be

$$\mathbf{X}(t) = \{x_{i,k,m}(t) \mid \tau_{i,k} \in \mathcal{T}^{\text{new}}(t), m \in \mathcal{M}_0\}, \quad (20a)$$

$$\boldsymbol{\beta}^{\text{up}}(t) = \{\beta_{i,k,j}^{\text{up}}(t) \mid \tau_{i,k} \in \mathcal{T}^{\text{new}}(t), j \in \mathcal{M}\}, \quad (20b)$$

$$\mathbf{F}^{\text{loc}}(t) = \{f_{i,k}^{\text{loc}}(t) \mid \tau_{i,k} \in \mathcal{T}^{\text{new}}(t)\}, \quad (20c)$$

$$\mathbf{F}^{\text{edge}}(t) = \{f_{i,k,j}^{\text{edge}}(t) \mid \tau_{i,k} \in \mathcal{T}^{\text{new}}(t), j \in \mathcal{M}\}. \quad (20d)$$

The dynamic bi-objective optimization problem at slot t is formulated as

$$\min_{\mathbf{X}(t), \boldsymbol{\beta}^{\text{up}}(t), \mathbf{F}^{\text{loc}}(t), \mathbf{F}^{\text{edge}}(t)} \Phi(t) = (\bar{T}(t), \bar{E}^{\text{dev}}(t)) \quad (21)$$

s.t. C1 : $x_{i,k,m}(t) \in \{0, 1\}, \forall \tau_{i,k} \in \mathcal{T}^{\text{new}}(t), m \in \mathcal{M}_0$,

$$\text{C2 : } \sum_{m \in \mathcal{M}_0} x_{i,k,m}(t) = 1, \forall \tau_{i,k} \in \mathcal{T}^{\text{new}}(t),$$

$$\text{C3 : } x_{i,k,j}(t) \leq \chi_{i,j}(t), \forall \tau_{i,k} \in \mathcal{T}^{\text{new}}(t), j \in \mathcal{M},$$

$$\text{C4 : } \varepsilon_{\beta} x_{i,k,j}(t) \leq \beta_{i,k,j}^{\text{up}}(t) \leq x_{i,k,j}(t), \\ \forall \tau_{i,k} \in \mathcal{T}^{\text{new}}(t), j \in \mathcal{M},$$

$$\text{C5 : } \sum_{\tau_{i,k} \in \mathcal{T}^{\text{new}}(t)} \beta_{i,k,j}^{\text{up}}(t) \leq 1, \forall j \in \mathcal{M},$$

$$\text{C6 : } T_{i,k,j}^{\text{up}}(t) \leq \Delta, \forall \tau_{i,k} \in \mathcal{E}_j(t), j \in \mathcal{M},$$

$$\text{C7 : } x_{i,k,0}(t) f_i^{\text{min}} \leq f_{i,k}^{\text{loc}}(t) \leq x_{i,k,0}(t) f_i^{\text{max}}, \\ \forall \tau_{i,k} \in \mathcal{T}^{\text{new}}(t),$$

$$\text{C8 : } \sum_{\tau_{i,k} \in \mathcal{L}_i(t)} f_{i,k}^{\text{loc}}(t) \leq F_i^{\text{res}}(t), \forall i \in \mathcal{N},$$

$$\text{C9 : } x_{i,k,j}(t) f_j^{\text{min}} \leq f_{i,k,j}^{\text{edge}}(t) \leq x_{i,k,j}(t) f_j^{\text{max}}, \\ \forall \tau_{i,k} \in \mathcal{T}^{\text{new}}(t), j \in \mathcal{M},$$

$$\text{C10 : } \sum_{\tau_{i,k} \in \mathcal{E}_j(t)} f_{i,k,j}^{\text{edge}}(t) \leq F_j^{\text{res}}(t), \forall j \in \mathcal{M}.$$

Constraint C1 defines the binary execution-location variables, while Constraints C2 and C3 require each newly arrived task to select exactly one execution location and exclude inaccessible edge servers, respectively. Constraints C4 and C5 link uplink-bandwidth allocation to edge assignment and enforce the bandwidth budget of each server, respectively. Constraint C6 requires each edge-assigned task to complete its input upload within the current decision slot. Constraints C7 and C9 impose the computing-rate bounds for local and edge execution, respectively, while Constraints C8 and C10 ensure that the aggregate computing rates do not exceed the residual capacities of the corresponding devices and edge servers.

The observed state must additionally satisfy

$$\sum_{\tau_{i,k} \in \mathcal{L}_i^{\text{adm}}(t)} \bar{f}_{i,k}^{\text{loc}}(t) \leq F_i^{\text{max}}(t), \forall i \in \mathcal{N}, \quad (\text{S1})$$

$$\sum_{\tau_{i,k} \in \mathcal{E}_j^{\text{adm}}(t)} \bar{f}_{i,k,j}^{\text{edge}}(t) \leq F_j^{\text{max}}(t), \forall j \in \mathcal{M}. \quad (\text{S2})$$

Conditions S1 and S2 ensure that the computing rates reserved for admitted unfinished tasks do not exceed the total capacities of the corresponding devices and edge servers.

3.5 Decision Deployment and Inter-Slot State Evolution

Let $\Omega^{\text{d}}(t)$ denote the feasible scheduling decision deployed at slot t , where the superscript d indicates the corresponding quantities under the deployed decision.

The workload evolution depends on both the execution location and whether a task is newly assigned or already admitted. A newly assigned local task can use the entire slot for computation. Therefore, for

$\tau_{i,k} \in \mathcal{L}_i(t)$, its remaining workload at the beginning of slot $t + 1$ is

$$\ell_{i,k}(t+1) = \max \left\{ L_{i,k} - f_{i,k}^{\text{loc,d}}(t)\Delta, 0 \right\}. \quad (22)$$

An admitted unfinished local task continues processing its remaining workload at its retained computing rate. For $\tau_{i,k} \in \mathcal{L}_i^{\text{adm}}(t)$,

$$\ell_{i,k}(t+1) = \max \left\{ \ell_{i,k}(t) - \bar{f}_{i,k}^{\text{loc}}(t)\Delta, 0 \right\}. \quad (23)$$

For a newly assigned edge task, edge computation begins after its input data have been uploaded. The deployed uplink transmission time $T_{i,k,j}^{\text{up,d}}(t)$ is obtained from the uplink transmission model using the deployed bandwidth allocation. Hence, the computing time available within the current decision slot is

$$\delta_{i,k,j}^{\text{cmp,d}}(t) = \max \left\{ \Delta - T_{i,k,j}^{\text{up,d}}(t), 0 \right\}. \quad (24)$$

Its remaining workload is then updated as

$$\ell_{i,k}(t+1) = \max \left\{ L_{i,k} - f_{i,k,j}^{\text{edge,d}}(t)\delta_{i,k,j}^{\text{cmp,d}}(t), 0 \right\}. \quad (25)$$

An admitted unfinished edge task has already completed its input upload and continues edge computation throughout the slot at its retained computing rate. For $\tau_{i,k} \in \mathcal{E}_j^{\text{adm}}(t)$,

$$\ell_{i,k}(t+1) = \max \left\{ \ell_{i,k}(t) - \bar{f}_{i,k,j}^{\text{edge}}(t)\Delta, 0 \right\}. \quad (26)$$

After these workload updates, only tasks with unfinished computation are carried forward. Accordingly,

$$\mathcal{T}^{\text{adm}}(t+1) = \left\{ \tau_{i,k} \in \mathcal{T}^{\text{act}}(t) \mid \ell_{i,k}(t+1) > 0 \right\}. \quad (27)$$

Each task in $\mathcal{T}^{\text{adm}}(t+1)$ retains its assigned execution location and computing rate. Together with the exogenous system conditions observed at slot $t+1$, these carried-forward tasks and their remaining workloads define the next system state $\mathcal{S}(t+1)$.

4 Proposed Algorithm

To address task-set changes across decision slots and the coupling between offloading locations and resource allocation, we develop LRC-DNSGA-II. The following subsections describe its main procedure and key components.

4.1 Design Motivation and Overview

As the set of newly arrived tasks changes across decision slots, the same chromosome position may correspond to different tasks, making position-based reuse of historical

solutions unreliable. Within a slot, the resource allocation of a task depends on its execution location. When location mutation changes the execution location of a task, the task becomes associated with a different set of shared communication and computing resources, and its current resource weights may no longer be suitable.

LRC-DNSGA-II is therefore centered on location-resource coupled variation (LRCV), with task-feature-matched dynamic response (TFM) used for cross-slot population adaptation. TFM establishes task correspondence across consecutive decision states and transfers task-relevant historical decision information to the current population. During within-slot evolution, LRCV restricts location mutation to accessible execution sites and resets the corresponding resource-weight genes only when a location mutation changes the execution location.

4.2 Overall Procedure

Algorithm 1 summarizes the slot-level procedure of LRC-DNSGA-II. At the beginning of slot t , the current system state $\mathcal{S}(t)$ is observed, and task features are constructed for the $K_t = |\mathcal{T}^{\text{new}}(t)|$ newly arrived tasks. If $K_t = 0$, no new optimization is performed and only the admitted unfinished tasks are advanced. If one or more intervening slots contain no newly arrived tasks, the TFM response at the next nonempty slot uses the population and task-feature matrix retained from the most recent preceding nonempty slot. Let $\phi_q(t)$ denote the feature vector of task q . The corresponding task-feature matrix is

$$\Psi(t) = [\phi_1(t), \dots, \phi_{K_t}(t)]^T, \quad (28)$$

where the construction of $\phi_q(t)$ is described in Section 4.4.

Let $\mathcal{X}_t^{(0)}$ denote the population used to initialize the optimization at slot t . For $t = 1$, the population is initialized as described in Section 4.3.1. For $t > 1$, TFM constructs $\mathcal{X}_t^{(0)}$ from the previous-slot population $\mathcal{X}_{t-1}$ using the cross-slot correspondence derived from $\Psi(t-1)$ and $\Psi(t)$. Historical objective values are not retained; all individuals in $\mathcal{X}_t^{(0)}$ are decoded and re-evaluated under the current state $\mathcal{S}(t)$.

Starting from $\mathcal{X}_t^{(0)}$, the algorithm performs G generations of within-slot evolution. In each generation, parent solutions are selected by binary tournament, and LRCV is applied to generate offspring. The offspring are decoded and evaluated under $\mathcal{S}(t)$, followed by standard NSGA-II environmental selection. For an infeasible individual, both objective values are assigned the common penalty $\Gamma_{\text{pen}} = 10^{12}$ before nondomination ranking and crowding-distance calculation.

After G generations, the resulting population is denoted by $\mathcal{X}_t$, and its feasible nondominated members form the approximation set $\mathcal{P}_t$. If $\mathcal{P}_t = \emptyset$, no

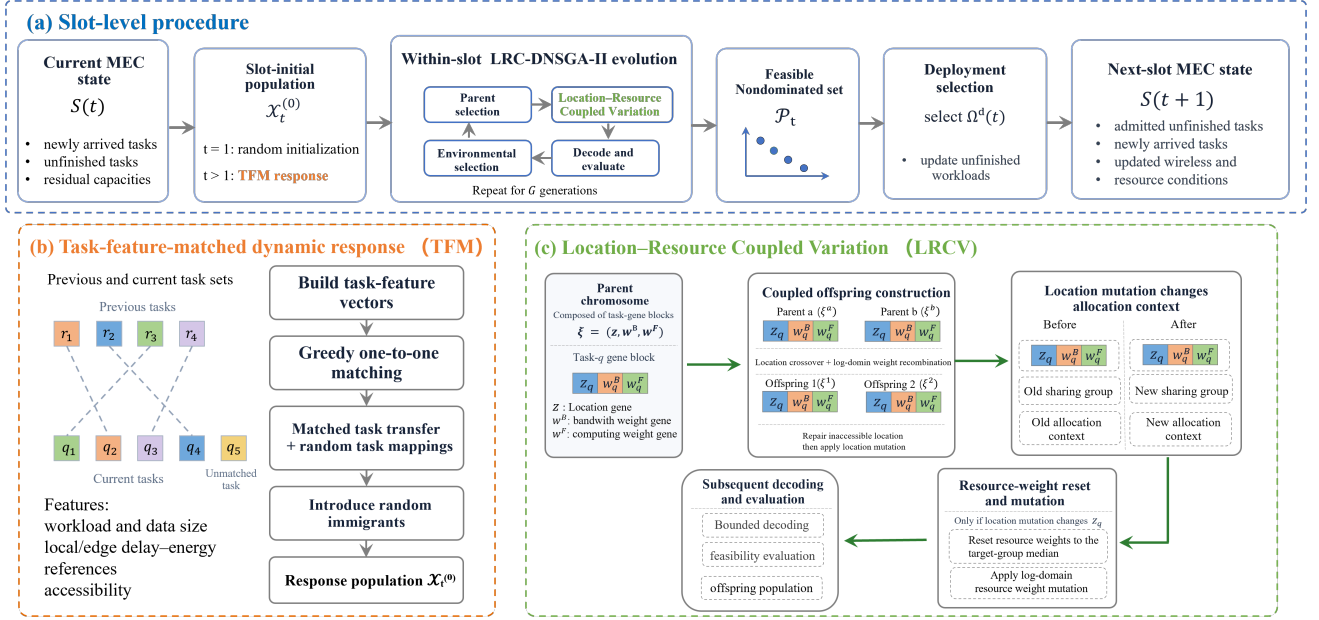


Fig. 2 Overall Procedure of LRC-DNSGA-II

new scheduling decision is deployed, and the previously admitted unfinished tasks continue their allocated execution.

For a nonempty $\mathcal{P}_t$, a single representative solution is selected for deployment. Since the delay and energy objectives have different scales and no prior preference is imposed between them, they are normalized over $\mathcal{P}_t$ and given equal importance:

$$\Omega^d(t) = \arg \min_{\Omega \in \mathcal{P}_t} \sum_{r=1}^2 \frac{f_r(\Omega) - f_r^{\min}(t)}{\max\{f_r^{\max}(t) - f_r^{\min}(t), \varepsilon_o\}} \quad (29)$$

where $r \in \{1, 2\}$ indexes the two objectives, $f_r(\Omega)$ denotes the value of objective r for solution Ω , with $f_1 = \bar{T}(t)$ and $f_2 = \bar{E}^{\text{dev}}(t)$. The quantities $f_r^{\min}(t)$ and $f_r^{\max}(t)$ are, respectively, the minimum and maximum values of objective r among the solutions in $\mathcal{P}_t$, and $\varepsilon_o > 0$ is a small constant used to avoid division by zero.

The selected decision $\Omega^d(t)$ is then applied to the MEC system, and the remaining workloads are updated according to Section 3.5. The final population $\mathcal{X}_t$ and task-feature matrix $\Psi(t)$ are retained for constructing the response population at slot $t + 1$.

4.3 Solution Representation and Decoding

4.3.1 Solution Representation and Initialization

At slot t , only newly arrived tasks are encoded, whereas admitted unfinished tasks retain their deployed execution locations and computing rates. Let $K_t = |\mathcal{T}^{\text{new}}(t)|$, and index the newly arrived tasks by $q \in \{1, \dots, K_t\}$. Each candidate solution is represented by the chromosome $\xi = (\mathbf{z}, \mathbf{w}^B, \mathbf{w}^F)$, whose q th components are z_q ,

w_q^B , and w_q^F , respectively. For task q , $z_q \in \mathcal{M}_0$ denotes its execution location, where $z_q = 0$ represents local execution and $z_q = j$, $j \in \mathcal{M}$, represents execution at edge server j . The positive genes w_q^B and w_q^F denote the uplink-bandwidth and computing-resource weights, respectively, with w_q^B used only for edge execution.

At $t = 1$, the initial population $\mathcal{X}_1^{(0)}$ consists of one feasibility-oriented seed and $N_p - 1$ randomly initialized individuals. The seed uses unit resource weights and assigns each task to the accessible execution location with the smallest reference delay. If the resulting solution is infeasible, an all-local solution is evaluated as an alternative. If neither solution is feasible, the original seed is retained and subsequently handled by the common penalty-based evaluation. The remaining individuals are generated with random location genes and positive resource weights, and are then decoded and evaluated under $\mathcal{S}(1)$.

4.3.2 Location Decoding and Resource Grouping

Let i_q denote the originating device of task q . Its accessible execution locations are

$$\mathcal{M}_q^{\text{acc}}(t) = \{0\} \cup \{j \in \mathcal{M} : \chi_{i_q, j}(t) = 1\}. \quad (30)$$

If $z_q \notin \mathcal{M}_q^{\text{acc}}(t)$, the location gene is reset to $z_q = 0$. The resulting z_q specifies the execution location of task q and determines the corresponding assignment variable in the system model. After location repair, the newly arrived tasks are grouped according to the communication and computing resources they share.

Algorithm 1 Overall Procedure of LRC-DNSGA-II

Input: N_p, T, G ;
TFM parameters $(\rho_m, \rho_i, \lambda)$;
LRCV parameters $(p_c, p_m^{\max}, c_m, p_m^{\text{res}}, \sigma)$

Output: $\{\mathcal{P}_t, \Omega^d(t)\}_{t=1}^T$

```

1: for  $t = 1$  to  $T$  do
2:   Observe  $\mathcal{S}(t)$  and set  $K_t \leftarrow |\mathcal{T}^{\text{new}}(t)|$ 
3:   if  $K_t = 0$  then
4:     Set  $\mathcal{P}_t, \Omega^d(t) \leftarrow \emptyset$ ; advance admitted unfinished tasks
5:     Retain the most recent nonempty  $(\mathcal{X}, \Psi)$  and continue
6:   end if
7:   Construct  $\Psi(t)$  for the newly arrived tasks
8:   if  $t = 1$  then
9:     Initialize  $\mathcal{X}_t^{(0)}$  according to Section 4.3.1
10:  else
11:    Construct  $\mathcal{X}_t^{(0)}$  from  $\mathcal{X}_{t-1}$  using TFM with  $\Psi(t-1)$  and  $\Psi(t)$ 
12:  end if
13:  Decode and evaluate  $\mathcal{X}_t^{(0)}$  under  $\mathcal{S}(t)$  using the common penalty rule; set  $\mathcal{X}_t \leftarrow \mathcal{X}_t^{(0)}$ 
14:  Assign nondomination ranks and crowding distances in  $\mathcal{X}_t$ 
15:  for  $g = 1$  to  $G$  do
16:    Repeatedly select parent pairs by binary tournament and apply LRCV (Algorithm 2) until  $N_p$  offspring  $\mathcal{O}_t$  are generated
17:    Decode and evaluate  $\mathcal{O}_t$  under  $\mathcal{S}(t)$  using the common penalty rule
18:     $\mathcal{X}_t \leftarrow \text{NSGAIISelect}(\mathcal{X}_t \cup \mathcal{O}_t, N_p)$ 
19:  end for
20:   $\mathcal{P}_t \leftarrow \text{ND}(\mathcal{X}_t)$  over feasible individuals
21:  if  $\mathcal{P}_t = \emptyset$  then
22:    Set  $\Omega^d(t) \leftarrow \emptyset$ 
23:    Advance previously admitted unfinished tasks and store  $\mathcal{X}_t, \Psi(t)$ 
24:    continue
25:  end if
26:  Select and deploy  $\Omega^d(t)$  using Eq. (29)
27:  Update remaining workloads using Section 3.5 and store  $\mathcal{X}_t, \Psi(t)$ 
28: end for
29: return  $\{\mathcal{P}_t, \Omega^d(t)\}_{t=1}^T$ 

```

For each edge server j , the set of newly arrived tasks assigned to server j is defined as

$$\mathcal{G}_j^{\text{edge}}(\mathbf{z}) = \{q \in \{1, \dots, K_t\} : z_q = j\}. \quad (31)$$

Similarly, the locally executed tasks at device i form

$$\mathcal{G}_i^{\text{loc}}(\mathbf{z}) = \{q \in \{1, \dots, K_t\} : i_q = i, z_q = 0\}. \quad (32)$$

Tasks in $\mathcal{G}_j^{\text{edge}}(\mathbf{z})$ share the uplink bandwidth and residual computing capacity of server j , whereas tasks in $\mathcal{G}_i^{\text{loc}}(\mathbf{z})$ compete only for the residual local computing capacity of device i .

4.3.3 Bounded Resource Decoding

After the execution locations are fixed, the resource-weight genes are converted into actual bandwidth and computing-resource allocations for the corresponding tasks.

Table 1 Bounded Resource Decoder

Resource	y_q	w_q	R	(a, b)	ρ
Uplink bandwidth	$\beta_{i_q, k_q, j}^{\text{up}}(t)$	w_q^B	1	$(\varepsilon_\beta, 1)$	1
Edge computing	$f_{i_q, k_q, j}^{\text{edge}}(t)$	w_q^F	$F_j^{\text{res}}(t)$	$(f_j^{\min}, f_j^{\max})$	1
Local computing	$f_{i_q, k_q}^{\text{loc}}(t)$	w_q^F	$F_i^{\text{res}}(t)$	$(f_i^{\min}, f_i^{\max})$	$\eta_i(t)$

Consider a nonempty task set $\mathcal{G}$ with $n = |\mathcal{G}|$. Let R denote the available resource, (a, b) the per-task lower and upper bounds, $\{w_q\}_{q \in \mathcal{G}}$ the positive resource weights, and $\rho \in [0, 1]$ the utilization factor. When $R \geq na$, the resource allocated to task q is

$$y_q = \min \left\{ b, a + \frac{w_q}{\sum_{r \in \mathcal{G}} w_r} \rho (\min\{R, nb\} - na) \right\}. \quad (33)$$

The term na accounts for the minimum allocation required by all tasks in $\mathcal{G}$. The remaining resource, $\min\{R, nb\} - na$, is distributed according to the relative weights, with ρ controlling the amount used for this allocation. The outer minimum limits each task to its upper bound b . Any resource left after this clipping is not redistributed. When $R < na$, the lower bounds cannot be satisfied, and the individual is treated as infeasible under the common penalty rule.

Table 1 gives the corresponding quantities for uplink bandwidth, edge computing, and local computing. Here, k_q denotes the task index of task q at device i_q . Since $\beta_{i_q, k_q, j}^{\text{up}}(t)$ is a bandwidth fraction, its total budget is normalized to $R = 1$. For edge and local computing, R is the residual computing capacity of the corresponding server or device.

Uplink bandwidth and edge computing use $\rho = 1$. For local computing, the utilization factor is

$$\eta_i(t) = \min \left\{ 1, \frac{1}{|\mathcal{G}_i^{\text{loc}}(\mathbf{z})|} \sum_{q \in \mathcal{G}_i^{\text{loc}}(\mathbf{z})} w_q^F \right\}. \quad (34)$$

The relative values of w_q^F determine how the local CPU is shared among the tasks, while their average magnitude determines, through $\eta_i(t)$, how much capacity is allocated beyond the minimum requirements. After location and resource decoding, each chromosome specifies the complete task-assignment and resource-allocation decision used for objective evaluation.

4.4 Task-Feature-Matched Dynamic Response

For $t > 1$, TFM is applied at the transition from slot $t-1$ to slot t to construct the response population $\mathcal{X}_t^{(0)}$.

Rather than transferring genes according to chromosome positions, it establishes correspondence between previous-slot and current tasks using intrinsic task attributes and state-dependent execution references.

For a current task q , the matching feature is

$$\phi_q(t) = \begin{bmatrix} \log(1 + L_{i_q, k_q}), \log(1 + D_{i_q, k_q}), \\ \log(1 + O_{i_q, k_q}), \log(1 + \hat{T}_q^{\text{loc}}(t)), \\ \log(1 + \hat{E}_q^{\text{loc}}(t)), (\chi_{i_q, j}(t), \log(1 + \hat{T}_{q, j}^{\text{edge}}(t)), \\ \log(1 + \hat{E}_{q, j}^{\text{edge}}(t)))_{j=1}^M \end{bmatrix}^T. \quad (35)$$

Here, $\hat{T}_q^{\text{loc}}(t)$ and $\hat{E}_q^{\text{loc}}(t)$ denote the reference delay and energy for local execution, while $\hat{T}_{q, j}^{\text{edge}}(t)$ and $\hat{E}_{q, j}^{\text{edge}}(t)$ denote the corresponding references for execution at server j . The three edge-side features are concatenated over $j = 1, \dots, M$. The reference computing rates are

$$\hat{f}_q^{\text{loc}}(t) = \max\left\{\min\left\{F_{i_q}^{\text{res}}(t), f_{i_q}^{\text{max}}\right\}, \varepsilon_{\text{num}}\right\}, \quad (36a)$$

$$\hat{f}_{q, j}^{\text{edge}}(t) = \max\left\{\min\left\{F_j^{\text{res}}(t), f_j^{\text{max}}\right\}, \varepsilon_{\text{num}}\right\}. \quad (36b)$$

The corresponding communication-rate references are

$$\hat{R}_{q, j}^{\text{up}}(t) = \max\left\{B_j^{\text{up}}(t) \log_2\left(1 + \frac{P_{i_q}^{\text{tx}} h_{i_q, j}(t)}{N_0 B_j^{\text{up}}(t)}\right), \varepsilon_{\text{num}}\right\}, \quad (37a)$$

$$\hat{R}_{q, j}^{\text{down}}(t) = \max\left\{R_{i_q, j}^{\text{down}}(t), \varepsilon_{\text{num}}\right\} \quad (37b)$$

where $\varepsilon_{\text{num}} = 10^{-12}$ is a numerical floor. The reference delay and energy terms in (35) are obtained by substituting these quantities into the local- and edge-execution models. The edge-side references are evaluated for all servers, while accessibility is represented separately by $\chi_{i_q, j}(t)$.

A current task q and a previous-slot task r are compared using

$$d_t(q, r) = \left[\sum_{d=1}^{d_\phi} \left(\frac{\phi_{q, d}(t) - \phi_{r, d}(t-1)}{s_d(t)} \right)^2 \right]^{1/2} + \lambda \mathbb{I}(i_q \neq i_r), \quad (38)$$

where $\phi_{q, d}(t)$ is the d th component of $\phi_q(t)$, $d_\phi = 5+3M$ is the feature dimension, and $s_d(t)$ is the standard deviation of component d computed from the pooled task features at slots $t-1$ and t , lower-bounded by $\varepsilon_\phi > 0$. The parameter $\lambda \geq 0$ penalizes cross-device matching, and $\mathbb{I}(\cdot)$ is the indicator function.

All current-previous task pairs are evaluated using (38) and ordered by increasing distance. Pairs are

accepted sequentially if neither task has been matched, yielding one-to-one task correspondences. Matched current tasks inherit the location and resource-weight genes of their previous-slot counterparts, while unmatched tasks are initialized with an accessible location and positive random resource weights. For random mapping, previous-slot task genes are assigned to current tasks without replacement when $K_t \leq K_{t-1}$ and with replacement otherwise. Any inherited edge location that is no longer accessible is reassigned to the accessible server with the largest current channel gain, or to local execution if none is accessible.

At slot t , the final population $\mathcal{X}_{t-1}$ is adapted to the current task set. Among the N_p individuals, $\text{round}(\rho_m N_p)$ use the matched task correspondence, while the remainder use random task mappings, where $\rho_m \in [0, 1]$ denotes the matched-transfer ratio. The adapted population is then partially replaced by $\max\{1, \text{round}(\rho_i N_p)\}$ random immigrants, where $\rho_i \in [0, 1]$ is the random-immigrant ratio. The resulting population forms $\mathcal{X}_t^{(0)}$ and is re-evaluated under $\mathcal{S}(t)$.

4.5 Location-Resource Coupled Variation

In the proposed chromosome, the resource weights of a task are interpreted together with its execution location. When location mutation changes the execution location, the task enters a different allocation context, and its current resource weights may no longer be suitable. LRCV therefore couples feasible location mutation with a mutation-triggered reset of resource-weight genes. Location mutation is restricted to accessible execution sites; when it changes the location, the bandwidth and computing resource-weight genes are reset according to the new allocation context. Since the resource weights are positive, their crossover and mutation are performed in the logarithmic domain.

4.5.1 Crossover and Location Variation

For a parent pair indexed by a and b , offspring 1 and 2 are initialized as copies of the parent chromosomes. Crossover is applied with probability $p_c \in [0, 1]$. Let $w_{q, s}^{(\nu)}$ denote the resource-weight component $\nu \in \{B, F\}$ of task q , where $s \in \{a, b\}$ denotes a parent and $s \in \{1, 2\}$ denotes an offspring.

When crossover occurs, uniform crossover is applied to the location vectors, while the resource weights are recombined as

$$w_{q, 1}^{(\nu)} = \exp\left[\alpha_q \log w_{q, a}^{(\nu)} + (1 - \alpha_q) \log w_{q, b}^{(\nu)}\right], \quad (39a)$$

$$w_{q, 2}^{(\nu)} = \exp\left[(1 - \alpha_q) \log w_{q, a}^{(\nu)} + \alpha_q \log w_{q, b}^{(\nu)}\right], \quad (39b)$$

where $\alpha_q \sim \mathcal{U}(0, 1)$. The same α_q is used for the two resource-weight components of task q .

The offspring locations are first checked against $\mathcal{M}_q^{\text{acc}}(t)$. If an inherited location is inaccessible, it is repaired by setting $z_q = 0$.

Since the number of newly arrived tasks varies across slots, using a fixed per-gene mutation probability would cause the number of location changes to increase with chromosome length. To keep the location variation at a comparable level across slots, the mutation probability is defined as

$$p_m^{\text{loc}}(t) = \min \left\{ p_m^{\text{max}}, \frac{c_m}{K_t} \right\}, \quad (40)$$

where $p_m^{\text{max}} \in [0, 1]$ limits the mutation probability of an individual location gene, and $c_m > 0$ controls the expected number of selected location genes per offspring. Accordingly, $K_t p_m^{\text{loc}}(t) \leq c_m$.

Each location gene is independently selected for mutation with probability $p_m^{\text{loc}}(t)$. To avoid generating inaccessible assignments, a new location is sampled directly from $\mathcal{M}_q^{\text{acc}}(t) \setminus \{z_q\}$ whenever this set is nonempty.

4.5.2 Resource-Weight Reset and Mutation

Whenever location mutation changes the execution location of task q , its bandwidth and computing resource-weight genes are reset before ordinary resource mutation is applied. As the location genes are processed in chromosome order, let $\mathbf{z}^{(q)}$ denote the current location vector after task q has been repaired and possibly mutated. The reference set for task q is

$$\mathcal{H}_q(\mathbf{z}^{(q)}) = \left\{ r \in \{1, \dots, K_t\} : z_r^{(q)} = z_q^{(q)} \right\}. \quad (41)$$

When a location mutation changes z_q , each resource weight is reset to the corresponding median in this set:

$$w_q^{(\nu)} \leftarrow \text{median} \left\{ w_r^{(\nu)} : r \in \mathcal{H}_q(\mathbf{z}^{(q)}) \right\}. \quad (42)$$

The median provides a reference value for the new location. Since task q itself belongs to $\mathcal{H}_q(\mathbf{z}^{(q)})$, its weight remains unchanged if no other task has the same encoded location. For local execution, $\mathcal{H}_q$ is used only for weight adaptation; the actual computing allocation is still decoded within the device-specific set in (32).

After the location-dependent adaptation, each resource weight is independently selected for mutation with probability $p_m^{\text{res}} \in [0, 1]$. When selected, $\epsilon_q^{(\nu)} \sim \mathcal{N}(0, \sigma^2)$ is applied as

$$w_q^{(\nu)} \leftarrow \exp \left(\log w_q^{(\nu)} + \epsilon_q^{(\nu)} \right), \nu \in \{B, F\}, \quad (43)$$

where $\sigma > 0$ controls the mutation strength. For numerical stability, the resulting resource weights are lower-bounded by $\varepsilon_w = 10^{-9}$.

Algorithm 2 Location–Resource Coupled Variation

Input: Parents $\xi^a, \xi^b, \{\mathcal{M}_q^{\text{acc}}(t)\}_{q=1}^{K_t}$, and LRCV parameters $(p_c, p_m^{\text{max}}, c_m, p_m^{\text{res}}, \sigma)$

Output: Offspring ξ^1, ξ^2

- 1: Copy the parents to the offspring
- 2: **if** crossover is activated with probability p_c **then**
- 3: Apply uniform crossover to the location genes and Eq. (39) to the resource weights
- 4: **end if**
- 5: Compute $p_m^{\text{loc}}(t)$ using Eq. (40)
- 6: **for** each offspring and $q = 1$ to K_t in chromosome order **do**
- 7: **if** $z_q \notin \mathcal{M}_q^{\text{acc}}(t)$ **then**
- 8: $z_q \leftarrow 0$
- 9: **end if**
- 10: $z_q^{\text{old}} \leftarrow z_q$
- 11: **if** location mutation is activated with probability $p_m^{\text{loc}}(t)$ and $\mathcal{M}_q^{\text{acc}}(t) \setminus \{z_q\} \neq \emptyset$ **then**
- 12: Sample z_q from $\mathcal{M}_q^{\text{acc}}(t) \setminus \{z_q\}$
- 13: **end if**
- 14: **if** $z_q \neq z_q^{\text{old}}$ **then**
- 15: Reset w_q^B, w_q^F using Eqs. (41)–(42)
- 16: **end if**
- 17: **for** $\nu \in \{B, F\}$ **do**
- 18: **if** resource mutation is activated with probability p_m^{res} **then**
- 19: Apply Eq. (43) to $w_q^{(\nu)}$
- 20: **end if**
- 21: Set $w_q^{(\nu)} \leftarrow \max\{w_q^{(\nu)}, \varepsilon_w\}$
- 22: **end for**
- 23: **end for**
- 24: **return** ξ^1, ξ^2

The resulting offspring are decoded under $\mathcal{S}(t)$ and evaluated in the within-slot evolutionary loop.

4.6 Computational Complexity

For $t > 1$, TFM is executed once at the slot transition. Its dominant cost comes from evaluating and sorting the $K_t K_{t-1}$ cross-slot task pairs, resulting in $O(K_t K_{t-1} [d_\phi + \log(K_t K_{t-1})])$.

During within-slot evolution, the worst-case cost of LRCV is $O(N_p K_t^2)$ per generation due to the location-dependent grouping used to reset resource weights after location mutation. Decoding and objective evaluation require $O(N_p K_t (N + M))$, while NSGA-II environmental selection requires $O(N_p^2)$. Therefore, the dominant complexity at slot $t > 1$ is

$$O \left(K_t K_{t-1} [d_\phi + \log(K_t K_{t-1})] + G [N_p K_t^2 + N_p K_t (N + M) + N_p^2] \right). \quad (44)$$

The one-time response-population adaptation and evaluation are of lower order than the repeated within-slot cost and are omitted from the dominant expression. At $t = 1$, the inter-slot matching term is absent.

5 Experimental Results and Analysis

We evaluate LRC-DNSGA-II in the dynamic MEC environment described above. The experiments examine its solution quality, adaptation to environmental changes, delay-energy trade-offs, scalability, and component effectiveness.

5.1 Experimental Setup and Comparison Algorithms

All algorithms were implemented in Python 3.7.1, and the experiments were conducted on a Dell Precision 3650 Tower running Windows 11, equipped with an Intel Core i7-11700 CPU at 2.50 GHz and 16 GB of RAM. The simulated MEC system consists of 50 end devices and 5 edge servers and evolves over six consecutive decision slots with temporally correlated system conditions. From S1 to S3, task attributes, wireless conditions, and available computing resources vary gradually. At S4, a persistent joint change is introduced relative to S3: task workload and input size are increased by 20% and 10%, respectively, while channel gain, uplink bandwidth, edge computing capacity, and local computing capacity are reduced by 35%, 15%, 20%, and 10%, respectively. These changed conditions are retained in S5 and S6, together with normal slot-to-slot variations.

LRC-DNSGA-II is compared with DNSGA-II [53], adapted DMOEA/D-COPMEC [50], DMOPSO [54], and TL-MOEA [55]. These methods represent different approaches to dynamic multi-objective optimization, including population inheritance, decomposition, swarm search, and learning-assisted adaptation. For each paired run, all algorithms are evaluated using the same task arrivals and system states, with the same decision encoding and objective evaluation procedure. Thirty runs are performed for each algorithm, with a population of 80 evolved for 30 generations in each decision slot.

The common algorithmic settings and the main MEC system parameters are summarized in Tables 2 and 3, respectively.

5.2 Parameter Selection

The parameter analysis considers ρ_m and ρ_i for TFM and c_m and σ for LRCV. The former two affect matched transfer and random immigration, whereas the latter two govern location variation and the perturbation of resource weights.

We begin with the response parameters, using a 3×3 grid with $\rho_m \in \{0.10, 0.20, 0.30\}$ and $\rho_i \in \{0.15, 0.20, 0.25\}$. In this experiment, $c_m = 5$, $\sigma = 0.30$, and $p_m^{\max} = p_m^{\text{res}} = 0.10$ are kept unchanged. The selected response setting is then used when examining $c_m \in \{3, 5, 7\}$ and $\sigma \in \{0.20, 0.30, 0.40\}$, with the two mutation probabilities still set to 0.10.

Table 2 Experimental and Algorithm Parameters

Parameter	Value
<i>General settings</i>	
Population size N_p	80
Generations per slot G	30
Decision slots T	6
Crossover probability p_c	0.90
<i>LRCV parameters</i>	
Maximum location-mutation probability $p_m^{\max}$	0.05
Location-mutation budget c_m	3
Resource-mutation probability p_m^{res}	0.05
Mutation strength σ	0.30
<i>TFM parameters</i>	
Matched-transfer ratio ρ_m	0.30
Random-immigrant ratio ρ_i	0.20
Cross-device matching penalty λ	0.75

Table 3 Main MEC System Parameters

Parameter	Value
Number of devices N	50
Number of edge servers M	5
Slot duration Δ	5 s
Newly arrived tasks per slot	[30, 60]
Workload	[300, 5000] Mcycles
Input data size	[0.5, 5.0] MB
Output data size	[0.05, 0.5] MB
Uplink bandwidth	[5, 20] MHz
Channel gain $h_{i,j}(t)$	$[10^{-9}, 10^{-6}]$
Noise power spectral density N_0	1.0×10^{-20} W/Hz
Downlink rate	[20, 100] Mbps
Local CPU capacity	[0.5, 2.0] GHz
Edge CPU capacities	{20, 25, 30, 35, 40} GHz
Transmit power	[0.3, 1.0] W
Receive power	[0.1, 0.3] W
Idle power	[0.05, 0.15] W
Effective capacitance κ_i	$[1.0, 5.0] \times 10^{-28}$

Each parameter setting is evaluated over 10 runs. Within a run, all settings use the same dynamic MEC sequence. For each decision slot, the settings are ranked separately by final HV and IGD, and the two ranks are averaged. The reported mean rank is obtained by averaging these values over all runs and slots. The results for the response parameters and LRCV parameters are shown in Fig. 3(a) and Fig. 3(b), respectively. When the mean ranks are close, aggregate HV and IGD are also considered.

For the response parameters, $(\rho_m, \rho_i) = (0.30, 0.20)$ gives the selected setting. The LRCV results favor $c_m = 3$. Although $\sigma = 0.20$ has a slightly better mean rank, $\sigma = 0.30$ produces higher aggregate HV and lower aggregate IGD, and is therefore used in the subsequent experiments. With $(c_m, \sigma) = (3, 0.30)$, the common mutation probability is further compared at 0.05, 0.10, and 0.15. Their mean ranks are 1.59, 2.03, and 2.38, respectively, leading to $p_m^{\max} = p_m^{\text{res}} = 0.05$.

5.3 Final Performance in HV and IGD

We use hypervolume (HV) and inverted generational distance (IGD) to compare the final Pareto approximations in each decision slot. For a given slot, the final

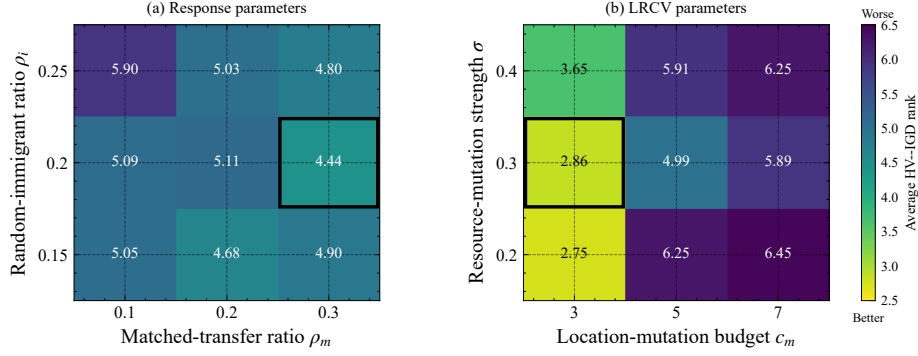


Fig. 3 Joint parameter analysis of LRC-DNSGA-II

feasible nondominated solutions from all algorithms and 30 runs are pooled, and the objective values are normalized using the componentwise minimum and maximum values of this set. The same pooled solutions are used to build the reference set for IGD. HV is computed in the normalized objective space with $\mathbf{r} = (1.1, 1.1)$ as the reference point.

Tables 4 and 5 give the mean and standard deviation of the final HV and IGD values over 30 runs. LRC-DNSGA-II is compared with each baseline using a two-sided paired Wilcoxon signed-rank test at $\alpha = 0.05$, followed by Holm correction. The symbols +, $\approx$, and – indicate significantly better, no significant difference, and significantly worse performance of LRC-DNSGA-II, respectively. Dark and light shading mark the best and second-best mean values in each slot.

LRC-DNSGA-II achieves the highest mean HV and the lowest mean IGD in all six slots, with significant differences from all comparison methods after Holm correction. Averaged over the six slots, its HV improvement ranges from 5.33% to 30.73%, while the IGD reduction ranges from 22.40% to 64.57%. The advantage is therefore observed in both metrics throughout the six-slot sequence.

Figs. 4 and 5 further show the run-level distributions underlying Tables 4 and 5. Across the six slots, the distributions of LRC-DNSGA-II remain consistently favorable, indicating that the observed improvements are not driven by a small number of runs.

5.4 Evolutionary Behavior under Dynamic Changes

Figs. 6 and 7 show how HV and IGD evolve over the six decision slots. Each curve gives the mean over 30 runs, with the shaded region showing the corresponding standard deviation.

From S2 onward, the slot transitions cause a temporary drop in solution quality as the task and resource conditions change. LRC-DNSGA-II recovers more quickly in most slots and subsequently maintains higher HV and lower IGD for much of the evolutionary process. The differences become clearer in the later slots,

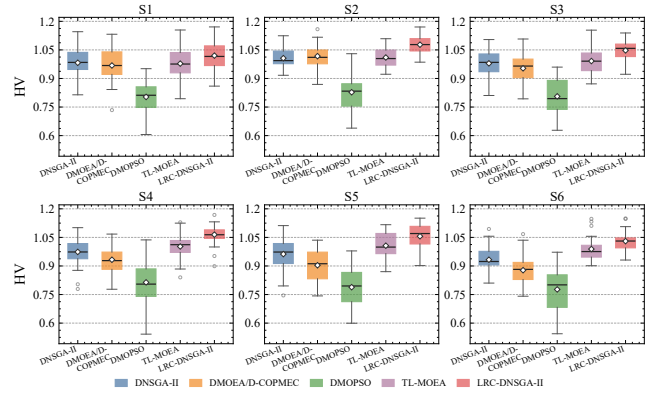


Fig. 4 Final HV distributions across decision slots

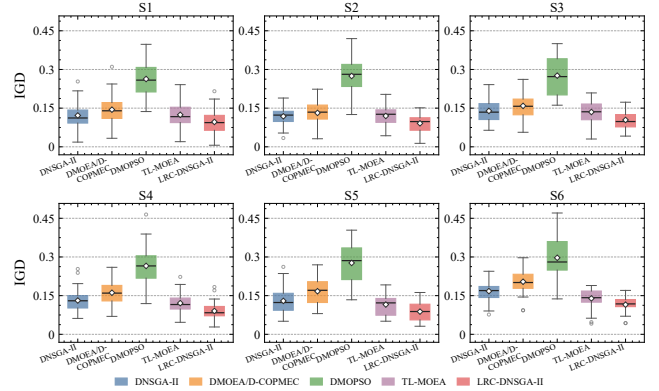


Fig. 5 Final IGD distributions across decision slots

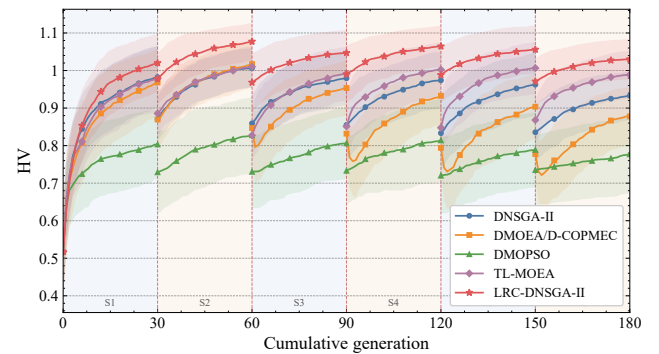


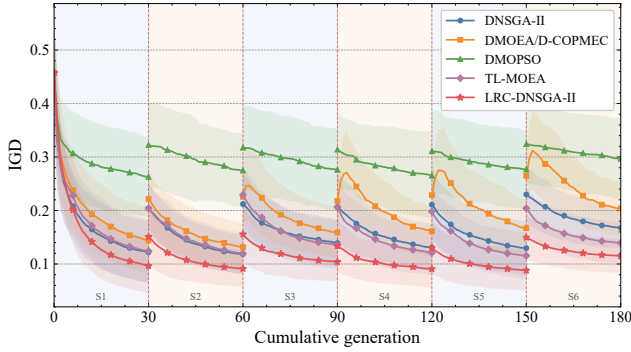
Fig. 6 HV convergence

Table 4 Final HV by Decision Slot

Slot	DNSGA-II	DMOE/D-COPMEC	DMOPSO	TL-MOE	LRC-DNSGA-II
S1	9.8247e-01 (8.34e-02) ⁺	9.6819e-01 (8.92e-02) ⁺	8.0304e-01 (9.24e-02) ⁺	9.7814e-01 (8.64e-02) ⁺	1.0201e+00 (7.74e-02)
S2	1.0064e+00 (5.26e-02) ⁺	1.0171e+00 (6.24e-02) ⁺	8.2766e-01 (1.02e-01) ⁺	1.0102e+00 (5.33e-02) ⁺	1.0774e+00 (4.91e-02)
S3	9.7941e-01 (7.20e-02) ⁺	9.5331e-01 (7.31e-02) ⁺	8.0601e-01 (9.58e-02) ⁺	9.9152e-01 (7.16e-02) ⁺	1.0474e+00 (6.08e-02)
S4	9.7391e-01 (7.47e-02) ⁺	9.3273e-01 (7.58e-02) ⁺	8.1391e-01 (1.09e-01) ⁺	1.0023e+00 (7.24e-02) ⁺	1.0646e+00 (5.41e-02)
S5	9.6239e-01 (8.56e-02) ⁺	9.0358e-01 (8.76e-02) ⁺	7.8857e-01 (1.00e-01) ⁺	1.0066e+00 (6.96e-02) ⁺	1.0560e+00 (6.48e-02)
S6	9.3275e-01 (6.36e-02) ⁺	8.7778e-01 (7.67e-02) ⁺	7.7685e-01 (1.03e-01) ⁺	9.8874e-01 (6.39e-02) ⁺	1.0304e+00 (5.38e-02)
+ / $\approx$ / -	6/0/0	6/0/0	6/0/0	6/0/0	-

Table 5 Final IGD by Decision Slot

Slot	DNSGA-II	DMOE/D-COPMEC	DMOPSO	TL-MOE	LRC-DNSGA-II
S1	1.2167e-01 (5.79e-02) ⁺	1.4413e-01 (6.22e-02) ⁺	2.6297e-01 (7.44e-02) ⁺	1.2422e-01 (5.88e-02) ⁺	9.6487e-02 (5.08e-02)
S2	1.1831e-01 (3.66e-02) ⁺	1.3159e-01 (4.53e-02) ⁺	2.7470e-01 (6.90e-02) ⁺	1.1973e-01 (3.88e-02) ⁺	9.1154e-02 (3.42e-02)
S3	1.3920e-01 (4.41e-02) ⁺	1.5877e-01 (4.83e-02) ⁺	2.7615e-01 (7.73e-02) ⁺	1.3500e-01 (4.50e-02) ⁺	1.0420e-01 (3.72e-02)
S4	1.3112e-01 (4.43e-02) ⁺	1.6122e-01 (4.84e-02) ⁺	2.6527e-01 (7.92e-02) ⁺	1.2090e-01 (4.23e-02) ⁺	9.0512e-02 (3.48e-02)
S5	1.2962e-01 (5.10e-02) ⁺	1.6701e-01 (5.46e-02) ⁺	2.7683e-01 (7.31e-02) ⁺	1.1543e-01 (4.37e-02) ⁺	8.8097e-02 (3.80e-02)
S6	1.6756e-01 (3.78e-02) ⁺	2.0394e-01 (5.01e-02) ⁺	2.9679e-01 (7.62e-02) ⁺	1.3927e-01 (3.83e-02) ⁺	1.1506e-01 (3.16e-02)
+ / $\approx$ / -	6/0/0	6/0/0	6/0/0	6/0/0	-

**Fig. 7** IGD convergence

where the changes in the system state have accumulated. DNSGA-II and TL-MOE also show recovery after each transition, but generally remain below LRC-DNSGA-II. DMOEA/D-COPMEC shows stronger fluctuations after the transitions, while DMOPSO tends to improve more slowly and maintains relatively high IGD.

The evolutionary trajectories therefore agree with the final HV and IGD results reported above, while also showing how the differences between the algorithms develop within each decision slot.

5.5 Trade-Offs Between Delay and Energy

Fig. 8 depicts the delay–energy trade-offs of the five algorithms over the six decision slots. Rather than showing a single run, each curve summarizes empirical attainment over the 30 paired runs. For a given delay threshold, the energy value from one run is the lowest energy among its final nondominated solutions that meet that threshold; a run with no such solution is regarded as unattained. The solid line shows the median attainable energy, whereas

the shaded region reflects the interquartile variation across runs.

At S1, before a previous population is available for transfer, the curves of LRC-DNSGA-II, DNSGA-II, and TL-MOE are relatively close. After the first environmental transition, LRC-DNSGA-II stays below the other methods throughout most of their common delay range, meaning that it can achieve lower device energy at comparable delay levels. This separation is particularly clear in S4–S6, where the task and resource conditions have changed more substantially. DNSGA-II and TL-MOE remain the closest competitors, while DMOEA/D-COPMEC and DMOPSO tend to require higher energy over most delay thresholds.

5.6 Scalability and Runtime

To examine the effect of task scale, the numbers of end devices and edge servers are kept at 50 and 5, while the number of newly arrived tasks per slot is set to 50, 75, and 90, corresponding to the Small, Medium, and Large settings, respectively. All three settings use the same six-slot experimental protocol, evolutionary budget, and 30 runs.

Tables 6 and 7 report the slot-wise HV and IGD results at the three task scales. LRC-DNSGA-II obtains the highest mean HV and the lowest mean IGD at every scale and decision slot. Nearly all pairwise comparisons are significant, with only one nonsignificant case and no significant loss. The differences from the comparison methods also become more evident as the number of tasks increases, showing that the performance advantage is maintained at larger task scales.

Table 8 reports the average wall-clock optimization time at the three task scales. The runtime of LRC-DNSGA-II increases from 7.04 to 8.34 s as the number of tasks increases from 50 to 90. Compared with

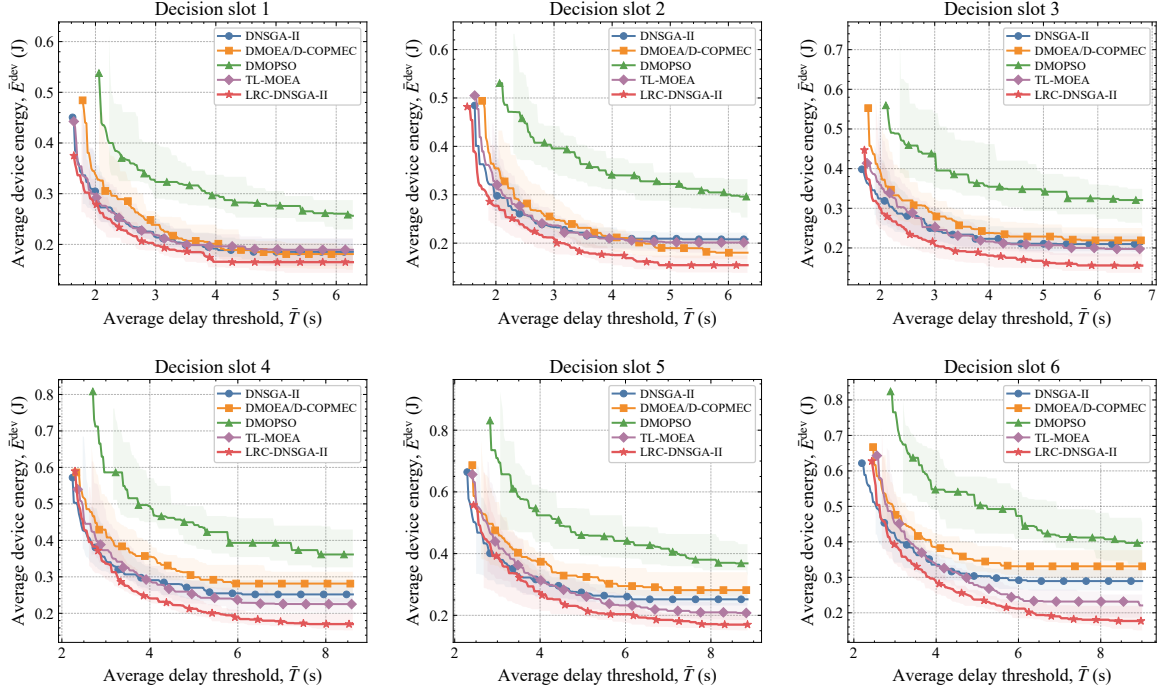


Fig. 8 Delay-energy trade-off profiles

DNSGA-II, the additional runtime ranges from 0.77 to 1.34 s, corresponding to an increase of 12.3%–20.1%. Its runtime remains close to that of TL-MOEA, while DMOPSO requires less computation time but shows clearly poorer HV and IGD performance.

5.7 Ablation Study

Two ablation variants are used to examine the effects of TFM and LRCV separately. LRC-DNSGA-II-w/o-TFM replaces task-feature matching with random historical-task mapping, while LRC-DNSGA-II-w/o-LRCV retains TFM but uses the standard crossover and mutation operators of DNSGA-II. DNSGA-II is also included as a reference without either mechanism. For each of the 30 paired runs, all variants use the same dynamic MEC sequence, with the remaining settings unchanged.

Table 9 compares the final approximation quality. Relative to w/o TFM, the full method improves HV by only 0.06% and reduces IGD by 0.79%, and neither difference is significant after Holm correction. In contrast, removing LRCV reduces HV by 4.25% and increases IGD by 14.24%, with both differences statistically significant. This shows that LRCV contributes most of the improvement in final approximation quality, whereas TFM has only a limited effect on the final solutions.

Table 10 further compares the search process. The early metrics are averaged over generations 0–10, the trajectory metrics use the normalized AUC over generations 0–30, and the final metrics are taken at generation 30. Each entry gives the numbers of +/≈/− outcomes over the six decision slots, indicating significantly better,

no significant difference, and significantly worse performance of LRC-DNSGA-II relative to the corresponding method, respectively. Compared with w/o TFM, significant differences appear mainly in the early generations and become less consistent over the full trajectory and at the final generation. Removing LRCV, however, leads to clear degradation from the early stage through the end of the search. Thus, TFM mainly helps at the beginning of a new slot, while the effect of LRCV persists throughout the within-slot search.

6 Conclusion

This work investigated batch-wise task assignment and resource allocation in dynamic MEC and formulated the problem as a constrained dynamic bi-objective optimization problem that jointly minimizes task completion delay and device energy consumption. To solve the resulting time-varying mixed discrete-continuous problem, we developed a Location-Resource Coupled Dynamic NSGA-II (LRC-DNSGA-II), which integrates task-feature-matched dynamic response with Location-Resource Coupled Variation. Experimental results over six temporally correlated decision slots demonstrated that LRC-DNSGA-II significantly outperformed all four comparison methods in both hypervolume (HV) and inverted generational distance (IGD). Scalability experiments further showed that its performance advantage was maintained as the task scale increased. Ablation studies confirmed that Location-Resource Coupled Variation was the primary contributor to the sustained improvement in final approximation quality, whereas the

Table 6 Slot-Wise HV Across Task Scales

Scale	Slot	DNSGA-II	DMOEA/D-COPMEC	DMOPSO	TL-MOEA	LRC-DNSGA-II
Small	S1	9.9990e-01 (8.25e-02) ⁺	9.7689e-01 (8.40e-02) ⁺	7.4706e-01 (1.02e-01) ⁺	9.9993e-01 (7.96e-02) ⁺	1.0283e+00 (8.39e-02)
	S2	9.4381e-01 (8.07e-02) ⁺	9.5324e-01 (7.11e-02) ⁺	6.7737e-01 (9.38e-02) ⁺	9.5840e-01 (7.45e-02) ⁺	1.0348e+00 (7.51e-02)
	S3	9.0033e-01 (8.24e-02) ⁺	9.0484e-01 (6.89e-02) ⁺	6.4232e-01 (8.40e-02) ⁺	9.5735e-01 (8.36e-02) ⁺	1.0260e+00 (7.24e-02)
	S4	8.9147e-01 (6.42e-02) ⁺	8.9077e-01 (6.62e-02) ⁺	6.8573e-01 (8.19e-02) ⁺	9.9231e-01 (6.57e-02) ⁺	1.0245e+00 (6.10e-02)
	S5	8.2600e-01 (7.40e-02) ⁺	7.9934e-01 (7.39e-02) ⁺	6.3672e-01 (9.75e-02) ⁺	9.6697e-01 (7.33e-02) ⁺	9.9330e-01 (6.68e-02)
	S6	8.2399e-01 (6.51e-02) ⁺	7.8989e-01 (6.39e-02) ⁺	6.7224e-01 (8.04e-02) ⁺	9.9769e-01 (7.63e-02) ⁺	1.0054e+00 (6.11e-02)
Medium	S1	9.1667e-01 (7.87e-02) ⁺	8.3912e-01 (8.59e-02) ⁺	6.9343e-01 (9.59e-02) ⁺	9.1435e-01 (7.80e-02) ⁺	9.6834e-01 (7.87e-02)
	S2	9.3449e-01 (6.61e-02) ⁺	9.0941e-01 (7.55e-02) ⁺	7.0496e-01 (7.07e-02) ⁺	9.6006e-01 (6.22e-02) ⁺	1.0311e+00 (5.97e-02)
	S3	8.9280e-01 (5.98e-02) ⁺	8.5957e-01 (6.21e-02) ⁺	6.8168e-01 (6.49e-02) ⁺	8.9326e-01 (5.67e-02) ⁺	9.9365e-01 (6.32e-02)
	S4	8.7653e-01 (6.51e-02) ⁺	8.3691e-01 (7.49e-02) ⁺	6.7349e-01 (7.28e-02) ⁺	8.9620e-01 (5.47e-02) ⁺	1.0084e+00 (5.69e-02)
	S5	8.3142e-01 (4.88e-02) ⁺	8.0024e-01 (6.74e-02) ⁺	6.3387e-01 (6.70e-02) ⁺	8.4177e-01 (4.51e-02) ⁺	9.7091e-01 (4.87e-02)
	S6	8.4041e-01 (4.98e-02) ⁺	8.0701e-01 (6.02e-02) ⁺	6.5397e-01 (6.29e-02) ⁺	8.6362e-01 (4.59e-02) ⁺	1.0092e+00 (4.74e-02)
Large	S1	8.9417e-01 (6.21e-02) ⁺	8.1286e-01 (7.32e-02) ⁺	7.3377e-01 (6.61e-02) ⁺	8.9382e-01 (6.69e-02) ⁺	9.9624e-01 (5.75e-02)
	S2	9.0549e-01 (5.00e-02) ⁺	8.5256e-01 (6.47e-02) ⁺	7.3445e-01 (6.43e-02) ⁺	9.2034e-01 (6.28e-02) ⁺	1.0343e+00 (5.11e-02)
	S3	8.7888e-01 (4.17e-02) ⁺	8.3560e-01 (4.60e-02) ⁺	7.4595e-01 (5.64e-02) ⁺	8.6841e-01 (5.69e-02) ⁺	1.0335e+00 (4.21e-02)
	S4	8.6111e-01 (4.32e-02) ⁺	8.0518e-01 (4.50e-02) ⁺	7.4721e-01 (6.12e-02) ⁺	8.3095e-01 (5.38e-02) ⁺	1.0090e+00 (3.69e-02)
	S5	8.7434e-01 (4.09e-02) ⁺	8.4093e-01 (4.60e-02) ⁺	7.7003e-01 (6.49e-02) ⁺	8.5918e-01 (4.88e-02) ⁺	1.0360e+00 (3.74e-02)
	S6	8.5485e-01 (3.94e-02) ⁺	8.4558e-01 (3.88e-02) ⁺	8.0029e-01 (6.97e-02) ⁺	8.4879e-01 (3.83e-02) ⁺	1.0504e+00 (3.54e-02)
Scale-slot +/ $\approx$ / -		18/0/0	18/0/0	18/0/0	17/1/0	-

Table 7 Slot-Wise IGD Across Task Scales

Scale	Slot	DNSGA-II	DMOEA/D-COPMEC	DMOPSO	TL-MOEA	LRC-DNSGA-II
Small	S1	1.1031e-01 (5.25e-02) ⁺	1.3948e-01 (5.48e-02) ⁺	3.0366e-01 (8.02e-02) ⁺	1.1324e-01 (5.31e-02) ⁺	1.0102e-01 (5.12e-02)
	S2	1.1623e-01 (4.79e-02) ⁺	1.3500e-01 (4.80e-02) ⁺	3.4986e-01 (8.62e-02) ⁺	1.1749e-01 (4.50e-02) ⁺	8.0283e-02 (4.07e-02)
	S3	1.4095e-01 (5.35e-02) ⁺	1.5068e-01 (4.65e-02) ⁺	3.8437e-01 (7.52e-02) ⁺	1.2556e-01 (4.79e-02) ⁺	7.8274e-02 (3.81e-02)
	S4	1.6105e-01 (4.31e-02) ⁺	1.6670e-01 (4.31e-02) ⁺	3.4914e-01 (6.78e-02) ⁺	1.1220e-01 (3.26e-02) ⁺	8.6392e-02 (2.93e-02)
	S5	1.7350e-01 (5.13e-02) ⁺	1.9980e-01 (5.02e-02) ⁺	3.7104e-01 (8.18e-02) ⁺	1.1719e-01 (3.17e-02) ⁺	8.3198e-02 (3.02e-02)
	S6	1.8324e-01 (4.66e-02) ⁺	2.0776e-01 (4.84e-02) ⁺	3.3253e-01 (6.43e-02) ⁺	9.1926e-02 (3.05e-02) ⁺	7.6611e-02 (2.80e-02)
Medium	S1	1.3699e-01 (5.77e-02) ⁺	1.8723e-01 (6.36e-02) ⁺	3.0463e-01 (8.13e-02) ⁺	1.3763e-01 (6.08e-02) ⁺	1.0864e-01 (5.46e-02)
	S2	1.3065e-01 (4.49e-02) ⁺	1.6423e-01 (4.95e-02) ⁺	3.1447e-01 (5.75e-02) ⁺	1.2516e-01 (4.32e-02) ⁺	8.5219e-02 (3.73e-02)
	S3	1.4273e-01 (4.21e-02) ⁺	1.8071e-01 (4.24e-02) ⁺	3.1572e-01 (5.85e-02) ⁺	1.5067e-01 (4.16e-02) ⁺	8.8035e-02 (3.82e-02)
	S4	1.0872e-01 (3.50e-02) ⁺	1.4892e-01 (4.28e-02) ⁺	2.6011e-01 (5.23e-02) ⁺	1.1170e-01 (3.05e-02) ⁺	6.4972e-02 (2.57e-02)
	S5	1.6241e-01 (3.29e-02) ⁺	1.8926e-01 (4.27e-02) ⁺	3.0182e-01 (4.76e-02) ⁺	1.5529e-01 (2.92e-02) ⁺	8.4380e-02 (2.64e-02)
	S6	1.4320e-01 (3.10e-02) ⁺	1.7040e-01 (3.72e-02) ⁺	2.7753e-01 (4.59e-02) ⁺	1.3158e-01 (2.58e-02) ⁺	6.5003e-02 (2.32e-02)
Large	S1	1.5488e-01 (3.69e-02) ⁺	2.0357e-01 (4.71e-02) ⁺	2.4977e-01 (4.68e-02) ⁺	1.5611e-01 (4.05e-02) ⁺	8.7048e-02 (2.77e-02)
	S2	1.3356e-01 (2.89e-02) ⁺	1.7073e-01 (4.06e-02) ⁺	2.5097e-01 (4.28e-02) ⁺	1.3078e-01 (3.62e-02) ⁺	7.6457e-02 (2.62e-02)
	S3	1.1409e-01 (2.52e-02) ⁺	1.4860e-01 (2.85e-02) ⁺	2.1056e-01 (3.61e-02) ⁺	1.2258e-01 (2.61e-02) ⁺	4.7299e-02 (1.87e-02)
	S4	1.5052e-01 (2.56e-02) ⁺	1.8148e-01 (3.05e-02) ⁺	2.0983e-01 (4.19e-02) ⁺	1.6516e-01 (3.57e-02) ⁺	6.0056e-02 (1.76e-02)
	S5	1.3500e-01 (2.43e-02) ⁺	1.5633e-01 (2.82e-02) ⁺	1.9657e-01 (4.04e-02) ⁺	1.4322e-01 (2.87e-02) ⁺	5.5501e-02 (1.91e-02)
	S6	1.2741e-01 (2.40e-02) ⁺	1.3530e-01 (2.13e-02) ⁺	1.6912e-01 (4.06e-02) ⁺	1.3093e-01 (2.23e-02) ⁺	4.2430e-02 (1.30e-02)
Scale-slot +/ $\approx$ / -		18/0/0	18/0/0	18/0/0	18/0/0	-

Table 8 Average Optimization Runtime per Slot (s)

Scale	DNSGA-II	DMOEA/D-COPMEC	DMOPSO	TL-MOEA	LRC-DNSGA-II
Small	6.27	5.00	2.95	6.92	7.04
Medium	6.66	6.13	3.54	7.54	8.00
Large	7.04	6.78	3.89	7.90	8.34

Table 9 Aggregate Effects of the Ablated Mechanisms

Comparison	HV gain (%)	IGD reduction (%)
Full vs. w/o TFM	+0.06	+0.79
Full vs. w/o LRCV	+4.25	+14.24

task-feature-matched dynamic response mainly accelerated adaptation following transitions between successive task batches.

Table 10 Ablation Results Across Search Stages

Indicator	DNSGA-II +/ $\approx$ / -	w/o TFM +/ $\approx$ / -	w/o LRCV +/ $\approx$ / -
Early HV	5/1/0	2/4/0	5/1/0
Early IGD	5/1/0	4/2/0	5/1/0
Trajectory HV AUC	6/0/0	1/5/0	5/1/0
Trajectory IGD AUC	6/0/0	3/3/0	5/1/0
Final HV	6/0/0	0/5/1	6/0/0
Final IGD	6/0/0	1/5/0	6/0/0

Future work will extend the current slot-to-slot dynamic optimization framework toward longer-horizon MEC operation using real-world workload and network traces, and will investigate more complex dynamic conditions, including stronger inter-slot dependencies, heterogeneous task arrivals, and larger-scale edge systems.

References

- [1] Duan, S., Wang, D., Ren, J., Lyu, F., Zhang, Y., Wu, H., Shen, X.: Distributed artificial intelligence empowered by end-edge-cloud computing: A survey. *IEEE Communications Surveys & Tutorials* **25**(1), 591–624 (2023) <https://doi.org/10.1109/COMST.2022.3218527>
- [2] Wang, G., Ling, G., Tang, L.: Optimization of energy-constrained task assignment and resource allocation under End-Edge-Cloud collaborative framework. *IEEE Internet of Things Journal* **12**(23), 50384–50398 (2025) <https://doi.org/10.1109/JIOT.2025.3609490>
- [3] Chu, W., Jia, X., Yu, Z., Lui, J.C.S., Lin, Y.: Joint service caching, resource allocation and task offloading for MEC-based networks: A multi-layer optimization approach. *IEEE Transactions on Mobile Computing* **23**(4), 2958–2975 (2024) <https://doi.org/10.1109/TMC.2023.3268048>
- [4] Hazra, A., Kalita, A., Gurusamy, M.: Meeting the requirements of internet of things: The promise of edge computing. *IEEE Internet of Things Journal* **11**(5), 7474–7498 (2024) <https://doi.org/10.1109/JIOT.2023.3339492>
- [5] Sharma, M., Tomar, A., Hazra, A.: Edge computing for industry 5.0: Fundamental, applications, and research challenges. *IEEE Internet of Things Journal* **11**(11), 19070–19093 (2024) <https://doi.org/10.1109/JIOT.2024.3359297>
- [6] Masoudi, M., Cavdar, C.: Device vs edge computing for mobile services: Delay-aware decision making to minimize power consumption. *IEEE Transactions on Mobile Computing* **20**(12), 3324–3337 (2021) <https://doi.org/10.1109/TMC.2020.2999784>
- [7] Mao, Y., Yu, X., Huang, K., Zhang, Y.-J.A., Zhang, J.: Green edge AI: A contemporary survey. *Proceedings of the IEEE* **112**(7), 880–911 (2024) <https://doi.org/10.1109/JPROC.2024.3437365>
- [8] Lin, L., Liao, X., Jin, H., Li, P.: Computation offloading toward edge computing. *Proceedings of the IEEE* **107**(8), 1584–1607 (2019) <https://doi.org/10.1109/JPROC.2019.2922285>
- [9] Sun, Z., Yang, H., Li, C., Yao, Q., Wang, D., Zhang, J., Vasilakos, A.V.: Cloud-edge collaboration in industrial internet of things: A joint offloading scheme based on resource prediction. *IEEE Internet of Things Journal* **9**(18), 17014–17025 (2022) <https://doi.org/10.1109/JIOT.2021.3137861>
- [10] Huang, B., Li, Z., Tang, P., Wang, S., Zhao, J., Hu, H., Li, W., Chang, V.: Security modeling and efficient computation offloading for service workflow in mobile edge computing. *Future Generation Computer Systems* **97**, 755–774 (2019) <https://doi.org/10.1016/j.future.2019.03.011>
- [11] Kong, X., Wu, Y., Wang, H., Xia, F.: Edge computing for internet of everything: A survey. *IEEE Internet of Things Journal* **9**(23), 23472–23485 (2022) <https://doi.org/10.1109/JIOT.2022.3200431>
- [12] He, Q., Lin, J., Fang, H., Wang, X., Huang, M., Yi, X., Yu, K.: Integrating IoT and 6G: Applications of edge intelligence, challenges, and future directions. *IEEE Transactions on Services Computing* **18**(4), 2471–2488 (2025) <https://doi.org/10.1109/TSC.2025.3586152>
- [13] Bi, R., Sun, L., Sun, Y., Han, M., Deng, Q.: Dynamic computation scheduling for hybrid energy mobile edge computing networks. *Journal of Systems Architecture* **155**, 103241 (2024) <https://doi.org/10.1016/j.sysarc.2024.103241>
- [14] Huang, H., Lin, W., Xu, M., Li, K.: AQESF: An adaptive QoS-enhanced scheduling framework for online batch of task scheduling. *Future Generation Computer Systems* **176**, 108174 (2026) <https://doi.org/10.1016/j.future.2025.108174>
- [15] Spatharakis, D., Pelekis, C., Dechouniotis, D., Papavasileiou, S.: A task offloading and batch scheduling framework for edge-assisted inference. *Future Generation Computer Systems* **175**, 108030 (2026) <https://doi.org/10.1016/j.future.2025.108030>
- [16] Nain, G., Pattanaik, K.K., Sharma, G.K.: Towards edge computing in intelligent manufacturing: Past, present and future. *Journal of Manufacturing Systems* **62**, 588–611 (2022) <https://doi.org/10.1016/j.jmsy.2022.01.010>
- [17] Zhao, F., Chen, Y., Zhang, Y., Liu, Z., Chen, X.: Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices. *IEEE Transactions on Network and Service Management* **18**(2), 2154–2165 (2021) <https://doi.org/10.1109/TNSM.2021.3069993>
- [18] Ale, L., Zhang, N., Fang, X., Chen, X., Wu, S., Li, L.: Delay-aware and energy-efficient computation offloading in mobile-edge computing using deep reinforcement learning. *IEEE Transactions on Cognitive Communications and Networking* **7**(3), 881–892 (2021) <https://doi.org/10.1109/TCCN.2021.3066619>

- [19] Xia, Y., Zhang, H., Tian, J., Yuan, D.: Dynamic task offloading and resource allocation for vehicular edge computing networks based on deep reinforcement learning. *IEEE Transactions on Vehicular Technology* **75**(3), 4977–4986 (2026) <https://doi.org/10.1109/TVT.2025.3612473>
- [20] Ma, X., Zhou, A., Zhang, S., Li, Q., Liu, A.X., Wang, S.: Dynamic task scheduling in cloud-assisted mobile edge computing. *IEEE Transactions on Mobile Computing* **22**(4), 2116–2130 (2023) <https://doi.org/10.1109/TMC.2021.3115262>
- [21] Qin, W., Chen, H., Wang, L., Xia, Y., Nascita, A., Pescapè, A.: MCOTM: Mobility-aware computation offloading and task migration for edge computing in industrial IoT. *Future Generation Computer Systems* **151**, 232–241 (2024) <https://doi.org/10.1016/j.future.2023.10.004>
- [22] Kim, M., Jang, J., Choi, Y., Yang, H.J.: Distributed task offloading and resource allocation for latency minimization in mobile edge computing networks. *IEEE Transactions on Mobile Computing* **23**(12), 15149–15166 (2024) <https://doi.org/10.1109/TMC.2024.3458185>
- [23] Shi, T., Zhang, T., Zhong, R., Liu, Y., Huang, R.: Cross-user dependent task offloading and resource allocation in spatial-temporal dynamic MEC networks. *IEEE Transactions on Vehicular Technology* **73**(10), 15584–15597 (2024) <https://doi.org/10.1109/TVT.2024.3411794>
- [24] Wang, L., Shen, B., Ma, L., Zhang, Y., Zhao, Y., Guo, H., Yu, Z., Guo, B.: Joint task offloading and migration optimization in UAV-enabled dynamic MEC networks. *IEEE Transactions on Services Computing* **18**(4), 2143–2157 (2025) <https://doi.org/10.1109/TSC.2025.3576644>
- [25] Ni, C., Kong, X., Du, J., Shen, G., Al-Asad, W., Min, G.: Transfer reinforcement learning for resource allocation in mobile edge computing systems: A survey. *IEEE Transactions on Services Computing* **19**(2), 1708–1727 (2026) <https://doi.org/10.1109/TSC.2026.3658306>
- [26] Shen, S., Yang, H., Yang, K., Wang, K., Zhang, G.: AoI-aware joint resource allocation in multi-UAV aided multi-access edge computing systems. *IEEE Transactions on Network Science and Engineering* **11**(3), 2596–2609 (2024) <https://doi.org/10.1109/TNSE.2023.3344667>
- [27] Yuan, P., Li, M., Zhao, X., Wang, C., Jin, H.: LOC-SAC: A Lyapunov-guided online collaborative offloading strategy with the SAC algorithm. *IEEE Transactions on Emerging Topics in Computing* **14**(2), 588–603 (2026) <https://doi.org/10.1109/TETC.2026.3687572>
- [28] Huang, P.-Q., Wang, Y., Wang, K., Liu, Z.-Z.: A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing. *IEEE Transactions on Cybernetics* **50**(10), 4228–4241 (2020) <https://doi.org/10.1109/TCYB.2019.2916728>
- [29] Huang, P.-Q., Wang, Y., Wang, K.: A divide-and-conquer bilevel optimization algorithm for jointly pricing computing resources and energy in wireless powered MEC. *IEEE Transactions on Cybernetics* **52**(11), 12099–12111 (2022) <https://doi.org/10.1109/TCYB.2021.3103840>
- [30] Xu, C., Guo, J., Li, Y., Zou, H., Jia, W., Wang, T.: Dynamic parallel multi-server selection and allocation in collaborative edge computing. *IEEE Transactions on Mobile Computing* **23**(11), 10523–10537 (2024) <https://doi.org/10.1109/TMC.2024.3376550>
- [31] Younis, A., Maheshwari, S., Pompili, D.: Energy-latency computation offloading and approximate computing in mobile-edge computing networks. *IEEE Transactions on Network and Service Management* **21**(3), 3401–3415 (2024) <https://doi.org/10.1109/TNSM.2024.3360850>
- [32] Li, J.-J., Chai, Z.-Y., Li, Y.-L., Zhou, Y.-B.: Reliable and efficient computation offloading for dependency-aware tasks in IIoT using evolutionary multi-objective optimization. *Future Generation Computer Systems* **174**, 107923 (2026) <https://doi.org/10.1016/j.future.2025.107923>
- [33] Gong, Y., Bian, K., Hao, F., Sun, Y., Wu, Y.: Dependent tasks offloading in mobile edge computing: A multi-objective evolutionary optimization strategy. *Future Generation Computer Systems* **148**, 314–325 (2023) <https://doi.org/10.1016/j.future.2023.06.015>
- [34] Tuo, S., Huyan, Y., Fan, T., Zhao, Y.: Membrane computing for IoT task offloading: An efficient multi-objective constrained optimization framework. *Applied Soft Computing* **169**, 112560 (2025) <https://doi.org/10.1016/j.asoc.2024.112560>
- [35] Liu, X., Wang, Z., Zhang, C., Cao, B., Fridenfalk, M., Kumar Singh, A.: Multiobjective resource allocation for Cloud-Edge-Terminal collaboration. *IEEE Internet of Things Journal* **12**(11), 17024–17033 (2025) <https://doi.org/10.1109/JIOT.2025.3539574>

- [36] Xu, S., Lu, Q., Zhang, Q., Liu, Q., Song, X.: Online dynamic resource allocation and trajectory scheduling for UAV-enabled wireless powered MEC. *IEEE Transactions on Vehicular Technology* **75**(7), 14550–14563 (2026) <https://doi.org/10.1109/TVT.2026.3656543>
- [37] Plachy, J., Becvar, Z., Strinati, E.C., di Pietro, N.: Dynamic allocation of computing and communication resources in multi-access edge computing for mobile users. *IEEE Transactions on Network and Service Management* **18**(2), 2089–2106 (2021) <https://doi.org/10.1109/TNSM.2021.3072433>
- [38] Su, Q., Li, W., Zhang, X., Hu, G., Zhang, X.: Cloud-edge collaborative task offloading and resource allocation based on mobile computility. *Future Generation Computer Systems* **178**, 108266 (2026) <https://doi.org/10.1016/j.future.2025.108266>
- [39] Su, J., Liu, Z., Li, Y., Lee, J., Guan, X.: Communication and computing balanced resource allocation in D2D-based vehicular MEC networks. *IEEE Internet of Things Journal* **11**(24), 40689–40701 (2024) <https://doi.org/10.1109/JIOT.2024.3454161>
- [40] Su, J., Liu, Z., Xie, Y., Li, Y., Ma, K., Guan, X.: UEE-Delay balanced online resource optimization for cooperative MEC-enabled task offloading in dynamic vehicular networks. *IEEE Internet of Things Journal* **11**(7), 11496–11507 (2024) <https://doi.org/10.1109/JIOT.2023.3330122>
- [41] Bai, Y., Zhang, Y.: Dynamic offloading based on Lyapunov optimization for UAV-assisted maritime IoT-MEC networks. *IEEE Transactions on Vehicular Technology* **74**(11), 17894–17906 (2025) <https://doi.org/10.1109/TVT.2025.3576090>
- [42] Zhao, Z., Zhang, H., Ding, H., Liu, W., Yuan, D.: Efficient asynchronous federated edge learning oriented tasks scheduling and resources allocation in dynamic multitasks MEC networks. *IEEE Transactions on Wireless Communications* **25**, 9214–9228 (2026) <https://doi.org/10.1109/TWC.2025.3648735>
- [43] Hou, J., Chen, M., Geng, H., Li, R., Lu, J.: GP-NFSP: Decentralized task offloading for mobile edge computing with independent reinforcement learning. *Future Generation Computer Systems* **141**, 205–217 (2023) <https://doi.org/10.1016/j.future.2022.11.025>
- [44] Yao, X., Chen, N., Yuan, X., Ou, P.: Performance optimization of serverless edge computing function offloading based on deep reinforcement learning. *Future Generation Computer Systems* **139**, 74–86 (2023) <https://doi.org/10.1016/j.future.2022.09.009>
- [45] Li, K., Wang, X., He, Q., Yi, B., Morichetta, A., Huang, M.: Cooperative multiagent deep reinforcement learning for computation offloading: A mobile network operator perspective. *IEEE Internet of Things Journal* **9**(23), 24161–24173 (2022) <https://doi.org/10.1109/JIOT.2022.3189445>
- [46] Xiao, H., Hu, Z., Zhang, X., Xu, A., Zheng, M., Li, K.: Federated deep reinforcement learning for task offloading in MEC-enabled heterogeneous networks. *IEEE Internet of Things Journal* **12**(8), 10238–10252 (2025) <https://doi.org/10.1109/JIOT.2024.3509893>
- [47] Zhang, W., Zhang, G., Mao, S.: Deep-reinforcement-learning-based joint caching and resources allocation for cooperative MEC. *IEEE Internet of Things Journal* **11**(7), 12203–12215 (2024) <https://doi.org/10.1109/JIOT.2023.3333826>
- [48] Li, C., Li, J., Gao, Y., Song, J., Wang, Z.: Multi-dimensional resource load-aware task migration in mobile edge computing. *Future Generation Computer Systems* **175**, 108091 (2026) <https://doi.org/10.1016/j.future.2025.108091>
- [49] Zhou, X., Yang, J., Li, Y., Li, S., Su, Z., Lu, J.: EC-TRL: Evolutionary-weighted clustering and transformer-augmented reinforcement learning for dynamic resource scheduling in edge cloud environments. *IEEE Internet of Things Journal* **12**(6), 7503–7517 (2025) <https://doi.org/10.1109/JIOT.2024.3496200>
- [50] Chai, Z.-Y., Liu, X., Li, Y.-L.: A computation offloading algorithm based on multi-objective evolutionary optimization in mobile edge computing. *Engineering Applications of Artificial Intelligence* **121**, 105966 (2023) <https://doi.org/10.1016/j.engappai.2023.105966>
- [51] Zhou, Y., Chai, Z.-Y., Li, Y.-L., Li, J.-J.: Parking vehicle-assisted task offloading in edge computing: A dynamic multi-objective evolutionary algorithm with multi-strategy fusion response. *Swarm and Evolutionary Computation* **94**, 101900 (2025) <https://doi.org/10.1016/j.swevo.2025.101900>
- [52] Qiu, Q., Ye, Y., Li, L., Xiao, Z., Lin, Q., Ming, Z.: Joint computation offloading and service caching in vehicular edge computing via a dynamic coevolutionary multiobjective optimization algorithm. *Expert Systems with Applications* **284**, 127821 (2025)

(2025) <https://doi.org/10.1016/j.eswa.2025.127821>

- [53] Deb, K., Rao N., U.B., Karthik, S.: Dynamic multi-objective optimization and decision-making using modified NSGA-II: A case study on hydro-thermal power scheduling. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) *Evolutionary Multi-Criterion Optimization. Lecture Notes in Computer Science*, vol. 4403, pp. 803–817. Springer, Berlin, Heidelberg (2007). https://doi.org/10.1007/978-3-540-70928-2_60
- [54] Aboud, A., Fdhila, R., Alimi, A.M.: Dynamic multi objective particle swarm optimization based on a new environment change detection strategy. In: Liu, D., Xie, S., Li, Y., Zhao, D., El-Alfy, E.-S.M. (eds.) *Neural Information Processing. Lecture Notes in Computer Science*, vol. 10637, pp. 258–268. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70093-9_27
- [55] Xie, H., Ding, D., Zhao, L., Kang, K.: Transfer learning based multi-objective evolutionary algorithm for dynamic workflow scheduling in the cloud. *IEEE Transactions on Cloud Computing* **12**(4), 1200–1217 (2024) <https://doi.org/10.1109/TCC.2024.3450858>

Data Availability

Data will be made available on request.

Statements and Declarations

Funding

This research was supported by the Major Program of National Natural Science Foundation of China (72192830, 72192835), the National Natural Science Foundation of China (62303102) and the 111 Project (B16009).

Competing Interests

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Author Contributions

Yingying Sun: Data curation; Formal analysis; Investigation; Methodology; Software; Validation; Visualization; Writing – original draft.

Zhiming Dong: Formal analysis; Investigation; Methodology; Resources; Software; Validation; Writing – review and editing.

Gongshu Wang: Conceptualization; Funding acquisition; Investigation; Project administration; Supervision; Writing – review and editing.