A Dual-Population Evolutionary Computation Framework for Two-Stage Task Scheduling in Mobile Edge Computing

Lu Yin, Jin Sun, *Member, IEEE*, Junlong Zhou, *Member, IEEE*, Zhihui Wei, *Member, IEEE*, and Keqin Li, *Fellow, IEEE*

***Abstract*—Task scheduling in mobile edge computing (MEC) is critical for managing latency and energy consumption, especially under user mobility, which poses additional challenges to scheduling decisions. This paper studies the mobility-aware two-stage task scheduling (MTTS) problem in MEC systems to minimize energy consumption at the mobile edge under task deadline constraints. First, the MTTS problem is formulated as a two-stage optimization model consisting of two interrelated sub-problems, associated with task offloading and result downloading, respectively. Then, a dual-population-based evolutionary computation (DORA) framework is proposed that can solve MTTS-type two-stage problems by incorporating a wide range of swarm intelligence algorithms (SIAs). The proposed DORA framework employs two populations that run in parallel, each dedicated to solving one of the two interrelated sub-problems, to enable effective exploration of the solution space and improve computational efficiency. The evolutionary process of each population in DORA consists of three fundamental algorithmic components: mapping, evaluation, and updating. In particular, the mapping component establishes the link between individual space in SIAs and solution space in the MTTS problem, with the critical parameter derived through rigorous theoretical analysis. Furthermore, the inter-population collaboration is realized through an asynchronous solution transfer mechanism from the stage-1 population to the stage-2 population, guiding the search toward high-quality final scheduling solutions. Extensive experiments are conducted on a real-world mobile device trajectory dataset, incorporating four well-established SIAs to demonstrate the applicability and effectiveness of DORA.**

***Index Terms*—Mobile edge computing, user mobility, task scheduling, swarm intelligence algorithm, evolutionary computation framework.**

I. Introduction

BY DEPLOYING computing resources and services at the network edge, mobile edge computing (MEC) has emerged as a highly promising computing paradigm. In MEC systems, mobile devices (MDs) can offload computation-intensive tasks to edge servers, thereby alleviating the constraints of their limited computing resources and battery capacity. By computation offloading, MDs can also handle more sophisticated and resource-demanding applications that were previously unfeasible due to hardware limitations. To improve computation offloading performance, task scheduling has attracted a great deal of attention in MEC-related research. Effective scheduling strategies can significantly enhance the quality of experience for mobile users by delivering low-latency and energy-efficient computing services [1], [2], [3].

Despite extensive research on task scheduling in MEC, several important challenges remain inadequately addressed. Many existing studies, in particular, rely on simplified scheduling models that typically neglect user mobility and its implications [4], [5]. When user mobility is considered, task scheduling becomes complicated due to the fluctuating distance between the MD and the edge server located at the base station (BS) to which the MD is connected. Such distance variations would lead to changes in the MD's uplink and downlink transmission rates, consequently impacting critical factors such as the task transmission duration, the MD's communication energy consumption, and the timely completion of tasks. Therefore, ignoring user mobility when designing task scheduling algorithms inevitably undermines the effectiveness of scheduling decisions, ultimately degrading the quality of user experience. In the context of mobility-aware MEC scheduling, a fundamental challenge lies in acquiring accurate knowledge of the user's trajectory, whether through prediction or predefined paths. In some practical scenarios, the MD's trajectory can be predetermined, such as in environmental monitoring, road cruising, and the agricultural industry. Moreover, recent studies have developed advanced trajectory prediction methods that achieve prediction accuracies exceeding 90% [6]. In light of this, research on mobility-aware scheduling algorithms can be designed based on known or predicted trajectories [7], [8].

Beyond mobility-related challenges, a commonly overlooked aspect in task scheduling problems is the computation result downloading stage [9], [10]. Motivated by the assumption that the data amount of computation results (edge server $\Rightarrow$ MD) is much smaller than that of task inputs (MD $\Rightarrow$ edge server), a considerable number of studies disregard the transmission time

Received 5 June 2025; revised 22 December 2025; accepted 4 March 2026. Date of publication 10 March 2026; date of current version 6 July 2026. This work was supported in part by Jiangsu Provincial Key Research and Development Program under Grant BE2022065-2. Recommended for acceptance by D. Michalopoulos. *(Corresponding author: Jin Sun.)*

Lu Yin is with the School of Computer Engineering, Nanjing Institute of Technology, Nanjing 211167, China (e-mail: ylu@njit.edu.cn).

Jin Sun, Junlong Zhou, and Zhihui Wei are with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China (e-mail: sunj@njust.edu.cn; jlzhou@njust.edu.cn; gswei@njust.edu.cn).

Keqin Li is with the Department of Computer Science, State University of New York, New Paltz, NY 12561 USA (e-mail: lik@newpaltz.edu).

Digital Object Identifier 10.1109/TMC.2026.3672119

1536-1233 © 2026 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

and energy consumption incurred by result downloading [11], [12], [13], [14], [15]. This assumption limits the applicability of such approaches to a wide range of application scenarios, including high-definition video processing, media format conversion, remote sensing data interpretation, and mathematical calculations, where the computation results involve substantial data and cannot be ignored. This highlights the need for task scheduling algorithms that take into account both task offloading and result downloading. By addressing task off-and-downloading problems, more efficient and broadly applicable scheduling strategies can be developed, enhancing performance and user experience across diverse applications.

With the aforementioned challenges in mind, this work investigates the mobility-aware two-stage task scheduling (MTTS) problem and develops a dual-population evolutionary computation framework tailored for two-stage scheduling optimization. First, we develop an optimization model for MTTS that jointly considers task offloading and result downloading stages, explicitly incorporating user mobility to achieve energy-efficient scheduling under deadline constraints. Then, we propose a dual-population evolutionary computation (DORA) framework that can solve the two-stage MTTS problem based on the search strategies of swarm intelligence algorithms (SIAs). The two populations in DORA are evolved in a parallel manner and are responsible for solving the offloading sub-problem and downloading sub-problem in MTTS, respectively. Accordingly, the two populations are named stage-1/offloading population and stage-2/downloading population. In DORA, the evolutionary process of each population comprises three essential algorithmic operations: mapping, evaluation, and updating. Specifically, the mapping operation links the individual space in SIAs with the solution space in the MTTS problem, where the critical parameter is determined through rigorous theoretical analysis. The evaluation operation quantifies the fitness of each individual, reflecting its effectiveness in solving the corresponding sub-problem. The updating operation updates the individual attributes based on the specific SIA that has been chosen. More importantly, the inter-population collaboration is realized through an asynchronous solution transfer mechanism from the stage-1 population to the stage-2 population. Once a better offloading solution is identified by the stage-1 population, it is transferred to the stage-2 population to guide its search process, ensuring continuous refinement across the two scheduling stages. Through this iterative co-evolutionary process, DORA is ultimately capable of producing high-quality scheduling solutions to the MTTS problem. The main contributions of this paper are as follows.

1) A two-stage optimization model is formulated for the mobility-aware scheduling problem MTTS in MEC systems, with the objective of minimizing the MD's energy consumption under task deadline constraints.
2) A dual-population evolutionary computation framework termed DORA is developed to solve the MTTS problem by leveraging the search strategies of many SIAs.
3) The two populations in DORA evolve in parallel, each dedicated to the offloading or downloading sub-problem, thereby enabling complementary and efficient exploration of the solution space of the MTTS problem.
4) The asynchronous solution transfer mechanism allows the stage-1 population to guide the search of the stage-2 population, such that high-quality solutions can be obtained through inter-population collaboration.
5) The mapping operator in DORA links the individual space to the solution space, with its critical parameter rigorously analyzed and experimentally validated.

In experiments, four highly-regarded SIAs, including bat algorithm (BA), firefly algorithm (FA), particle swarm optimization (PSO), and cuckoo search (CS), are incorporated into DORA. Experimental results demonstrate the superior effectiveness of DORA compared to several state-of-the-art approaches.

This paper is organized into the following sections. Section II reviews the existing related works. Section III presents the optimization model of MTTS. Section IV provides the proposed DORA. Section V shows the experimental results. Section VI draws the conclusion.

II. Related Work

This section reviews related work from the perspectives of both problems and problem-solving approaches, i.e., MEC-related task scheduling problems and dual-population-based swarm intelligence methodologies.

A. Task Scheduling in MEC Environments

Task scheduling methods are commonly divided into static and dynamic approaches. Dynamic scheduling approaches handle stochastic task arrivals and time-varying system conditions through sequential decision-making. Recent studies on dynamic scheduling have explored learning-based approaches, particularly reinforcement learning [25]. Motivated by scenarios in which a batch of tasks is known in advance, this paper focuses on static scheduling due to its clear advantages in maximizing resource efficiency and ensuring reliable performance in predictable MEC environments. In this context, we review representative scheduling methods targeting diverse optimization objectives and constraints in MEC systems.

Wang et al. [16] designed an energy-efficient task scheduling scheme based on traffic mapping in a heterogeneous edge computing architecture for Green IoT scenarios. Parallel tasks are assigned to heterogeneous edge servers to jointly improve resource utilization and reduce energy consumption. Li et al. [17] investigated the task scheduling problem in an MEC environment with multiple cloud-assisted edge servers, and developed scheduling algorithms that take into account the heterogeneity of computation and communication speeds. Lian et al. [18] focused on the scheduling problem of security-critical tasks in MEC systems, and proposed metaheuristic algorithms based on the grey wolf optimizer for searching scheduling solutions. Zhang et al. [11] studied a security-critical task scheduling problem in a resource-limited MEC system to optimize tasks' makespan and MD's energy consumption jointly. Zhao et al. [12] introduced a convex programming-based approach for offloading dependent tasks under the limited service caching capacity of edge nodes. Huang et al. [19] built a security overhead model for edge servers, and searched for scheduling solutions to minimize

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

MD's energy consumption under task deadline constraints. Lin et al. [20] optimized the data transmission time for a scientific workflow while satisfying the storage capacity constraint of edge servers. The work by Li et al. [13] focuses on optimizing communication and computation resources in deadline-constrained task offloading scenarios. Lin et al. [21] addressed the task scheduling problems for deep neural network-based applications, proposing a metaheuristic to optimize the system cost. However, the studies discussed above generally assume a fixed location for the MD to simplify the scheduling model. In reality, user mobility introduces considerable complexity to task scheduling problems due to the varying distance between the MD and the BS. Distance variation would lead to changes in the MD's transmission rates, which subsequently affect the task transmission duration, the MD's communication energy consumption, and the timely completion of tasks. Therefore, it is essential to focus on mobility-aware task scheduling to address these challenges effectively.

Considering mobility-aware task scheduling, the MD's trajectory can be predetermined in some practical scenarios, such as environmental monitoring or smart agriculture. Furthermore, recent advancements in trajectory prediction techniques have achieved satisfactory accuracy, supporting the feasibility of relying on known trajectories. Within this context, Saleem et al. [7] proposed a genetic algorithm-based scheduling strategy to minimize task offloading latency. Zhan et al. [14] studied the offloading decision and resource allocation problem in MEC, which assumes that all MDs are moving straight with a constant speed. Ding et al. [15] proposed three algorithms for task offloading and service migration, tailored to users with different mobility patterns. Hua et al. [22] concentrated on the joint optimization of data transmission and offloading decisions in cloud-edge collaborative systems, aiming to minimize MDs' energy consumption while considering user mobility. Zhu et al. [23] addressed the task scheduling problem under deadline constraints and developed approximation algorithms applicable to various system settings. Wang et al. [24] focused on the task assignment problem of divisible tasks, aiming to reduce latency and improve resource utilization. While acknowledging the valuable contributions of the aforementioned studies, several limitations remain unaddressed. First, while user mobility is taken into account, many existing approaches use simplified mobility models (e.g., straight-line or constant-speed movement) and assume a constant data transmission rate throughout the offloading or downloading process of an individual task [7], [23], [24]. Such assumptions deviate from practical MEC settings, in which the transmission rate may vary over a task's transmission interval. This deviation can significantly affect data-intensive tasks, as these tasks' prolonged transmission durations have a pronounced impact on both latency and energy consumption [7], [14], [15], [22]. For this reason, this work adopts a fine-grained interval-based transmission model that allows a task's data transmission to span multiple slots and reflect trajectory-induced rate variations. More importantly, in most mobility-aware scheduling approaches, result downloading latency is either neglected or treated as a simple additive delay after task execution. Thus, these models cannot explicitly capture the feasibility of downloading decisions that depend on prior offloading decisions, nor enable effective collaboration between the two stages. These modeling oversimplifications would lead to suboptimal task scheduling and resource allocation, thereby limiting the practical applicability of these methods in MEC systems.

Table I summarizes the main characteristics of the aforementioned task scheduling studies. Only a limited number of studies have taken into account both concerns of user mobility and result downloading. Moreover, most formulations treat result downloading as a secondary delay or cost component, rather than as an explicit scheduling stage. Consequently, the inherent two-stage structure of the task lifecycle is not fully captured. These observations motivate the need for a two-stage scheduling formulation that explicitly accounts for the interdependence between offloading and downloading under mobility, together with a dedicated dual-population evolutionary framework to solve the formulated model.

B. Dual-Population-Based Methodologies

In the literature, numerous methodologies have been developed in the field of multi-population SIAs [26]. Dual-population SIAs, as a particular type in the multi-population field, have garnered attention for enhancing optimization performance. We then investigate some existing dual-population SIAs from diverse perspectives and delineate the distinctions between this paper and previous studies. Specifically, based on the mechanism of algorithmic search, the dual-population-based methods can be classified into three categories.

In the first category, two populations are responsible for exploration (global search) and exploitation (local search), respectively [27], [28], [29], [30], [31], [32]. For example, Zhong et al. [27] developed a dual-population evolution algorithm, which adopts different parameter settings to promote the diversity of one population and the convergence of the other population. In the second category, particularly for constrained optimization problems (COPs) and constrained multi-objective optimization problems (CMOPs), two populations search the solution space with and without considering constraints, respectively [33], [34], [35], [36]. Specifically, Ming et al. [33] developed a dual-population-based evolutionary algorithm that explores the solution space by making good use of informative infeasible solutions obtained from populations that ignore constraints. In the third category, each population evolves to optimize its respective objective optimization [37], [38], [39], [40], [41]. For instance, Gao et al. [39] presented a dual-population differential evolution. By treating COPs as a bi-objective optimization problem, one population deals with the actual optimization function, and the other aims to minimize the degree of constraint violations.

Table II summarizes the main characteristics of several existing approaches. While acknowledging the significance of these works, this study goes beyond prior efforts in both problem complexity and methodological generality. First, the existing works mainly focus on solving standard COPs or CMOPs, whose encoding methods and optimization functions would be difficult to apply to complex MEC scheduling scenarios. Second, to the best of our knowledge, there is still a lack of a generic

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

TABLE I
SUMMARY OF EXISTING STUDIES ON TASK SCHEDULING IN MEC ENVIRONMENTS. ("+" INVOLVED; "-": NOT INVOLVED).

Ref.	Task Model			Mobility		Offloading		Download		Constraint			Object			Methodology
	Independent	Dependent	Divisible	Aware	Unaware	Full	Partial	Considered	Neglected	Energy	Deadline	Security	Time	Energy	Cost	
[7]	+	-	-	+	-	-	+	-	+	+	+	-	+	-	-	Genetic Algorithm
[11]	+	-	-	-	+	+	-	-	+	-	-	+	+	+	-	Particle Swarm Optimization
[12]	-	+	-	-	+	-	+	-	+	-	-	-	+	-	-	Convex Programming
[16]	-	+	-	-	+	+	-	-	+	+	-	-	+	-	-	Heuristic
[17]	+	-	-	-	+	-	+	-	+	+	-	-	+	-	-	Heuristic
[18]	+	-	-	-	+	+	-	-	+	-	-	-	+	-	-	Grey Wolf Optimizer
[19]	-	+	-	-	+	-	+	+	-	-	+	+	-	+	-	Genetic Algorithm
[20]	-	+	-	-	+	+	-	+	-	-	-	-	+	-	-	Particle Swarm Optimization
[13]	-	-	+	-	+	-	+	-	+	-	+	-	-	+	-	Convex Programming
[21]	-	-	+	-	+	+	-	+	-	-	+	-	-	-	+	Particle Swarm Optimization
[14]	+	-	-	+	-	-	+	-	+	+	-	-	+	+	-	Heuristic
[15]	+	-	-	+	-	-	+	-	+	+	+	-	+	+	-	Approximate Algorithm
[22]	+	-	-	+	-	-	+	-	+	-	+	-	-	+	-	Heuristic
[23]	+	-	-	+	-	-	+	+	-	-	+	-	-	+	-	Approximate Algorithm
[24]	-	-	+	+	-	-	+	+	-	-	-	-	+	-	-	Heuristic
Ours	+	-	-	+	-	+	-	+	-	-	+	-	-	+	-	Evolutionary Computation Framework

TABLE II
SUMMARY OF EXISTING WORKS ON DUAL-POPULATION-BASED INTELLIGENCE ALGORITHMS

Pop. 1	Pop. 2	Ref.	SIA	Problem
Global Search	Local Search	[27]	DE	Railway Timetable Scheduling
		[28]	GA+SA	Flow Shop Scheduling
		[29]	GA	Benchmark Functions
		[30]	DE	Economic Power Dispatch
		[31]	EA	CMOP
		[32]	HS	Dynamic Optimization
With Constraints	Without Constraints	[33]	EA	CMOP
		[34]	GA	CMOP
		[35]	EA	CMOP
		[36]	GA	CMOP
Objective 1	Objective 2	[37]	GA	Facility Location Problem
		[38]	EA	Service Selection Problem
		[39]	DE	COP
		[40]	EA	Multi-Objective Optimization
		[41]	EA	Multi-Objective Optimization

[1] Swarm intelligence algorithm (SIA): differential evolution (DE), genetic algorithm (GA), simulated annealing (SA), ant colony optimization (ACO), harmony search (HS).
[2] Problem: constrained multi-objective optimization problem (COMP), constrained optimization problem (COP).

dual-population evolutionary framework capable of supporting a wide range of swarm intelligence algorithms (SIAs). Realizing this, we develop an evolutionary computation framework based on dual-population, which can deploy a wide range of SIAs to solve mobility-aware task scheduling problems in MEC.

III. PROBLEM DESCRIPTION

We formulate the scheduling problem as a mathematical optimization model in this section. The superscripts "$\Uparrow$" and "$\Downarrow$" indicate offloading and downloading stages, respectively.

A. MEC System Model

We consider an MEC system that deploys multiple edge servers, each of which is connected to a BS. Assume that there are M edge servers in the MEC system, i.e., $S = \{s_1, s_2, \ldots, s_M\}$. For any edge server s_j $(j = 1, 2, \ldots, M)$, its location is represented by (x_j, y_j), and its CPU operating frequency is f_j (in CPU cycles/second). Benefiting from virtual machine deployment and service migration technologies, tasks offloaded to one edge node would be transmitted to another for execution to support flexible computation placement. This work considers an intra-region MEC deployment, where multiple edge servers are interconnected via high-speed wired or optical backhaul links. The backhaul transmission rate (in Gbps) is generally several orders of magnitude higher than the wireless access rate between the mobile device and base stations (in Mbps) [42]. Accordingly, the inter-edge transmission latency is not explicitly modeled in this work, as it is typically much smaller than the wireless access latency in intra-region MEC systems. Under this assumption, edge servers form a shared computing resource pool, with base stations serving as access points for the MD. For larger-scale federated or geo-distributed MEC systems, in which inter-edge latency may become significant, this assumption may no longer hold and will be explored in future work. Moreover, consistent with [11], [43], the MD is assumed to access only one BS at a time.

B. Mobility Model

Following existing studies, a slotted operational mechanism is adopted, with a one-second slot length to accurately capture the MD's mobility [7]. Assuming that the trajectory of the MD is available over time slots $[1, T]$, a series of 2D coordinates is used to represent it, i.e., $traj = \{<x_1, y_1>, <x_2, y_2>, \ldots, <x_T, y_T>\}$, where the subscripts represent the time indexes. This study considers a scheduling environment in which mobile trajectories are known or can be accurately predicted prior to scheduling. This assumption arises naturally in many practical engineering deployments, where mobile entities operate along predefined or highly predictable trajectories due to fixed routes or scheduled operational patterns. As such, trajectories are assumed to be known in advance or reliably estimated over the

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

scheduling horizon, enabling effective offline scheduling and optimization. Specifically, for a timeslot $t \in \{1, 2, \ldots, T\}$, the distance between the MD and the BS it accesses is

$$dis_t = \min_{j=1,2,\ldots,M} \{dis_{t,j}\}, \tag{1}$$

where $dis_{t,j}$ denotes the distance between the MD and the BS b_j at timeslot t, i.e.,

$$dis_{t,j} = \sqrt{(x_t - x_j^s)^2 + (y_t - y_j^s)^2}. \tag{2}$$

For the t-th timeslot, the data transmission rate of the MD's uplink and downlink can be determined by

$$r_t^{\Uparrow} = b \log_2 \left(1 + \frac{g_0 (dis^*/dis_t)^{\theta} p^{\Uparrow}}{bN_0}\right), \tag{3}$$

$$r_t^{\Downarrow} = b \log_2 \left(1 + \frac{g_0 (dis^*/dis_t)^{\theta} p^{\Downarrow}}{bN_0}\right). \tag{4}$$

The descriptions of relevant parameters b, N_0, g_0, θ, and dis^* are provided in [11]. $p^{\Uparrow}$ and $p^{\Downarrow}$ are the MD's uplink and downlink data transmission powers, respectively.

C. Task Model

Along a given trajectory, the MD is required to offload a series of independent tasks, each subject to its own deadline constraint. These tasks originate from various applications or services and are assumed to be independent of each other, meaning that there is no data dependency or execution order between them. Each task is atomic in nature and cannot be further partitioned into smaller subtasks for parallel execution. Let N denote the total number of tasks, with the task set denoted by $\Gamma = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_N\}$. For a specific task ς_i $(i = 1, 2, \ldots, N)$, it can be represented by $\varsigma_i =< DS_i^{\Uparrow}, DS_i^{\Downarrow}, CI_i, DL_i >$, where $DS_i^{\Uparrow}$ is input data size (in bits), $DS_i^{\Downarrow}$ is output data size (in bits), CI_i is computation intensity (in CPU cycles per bit), and DL_i is deadline. These parameters collectively provide both the computational and communication requirements of each task. This work focuses on a static scheduling setting in which the set of tasks and their associated attributes are available at the time of scheduling. This assumption is widely adopted in many practical batch-oriented MEC scenarios (e.g., video analytics and healthcare monitoring), where the workload for a scheduling window can be collected in advance [12], [18], [44].

D. Offloading Model (MTTS-Stage1)

Let $ST_i^{\Uparrow}$ be the offloading start time of ς_i, then the offloading finish time of ς_i is

$$FT_i^{\Uparrow} = \arg\min \left\{ x \,\middle|\, \sum_{t=ST_i^{\Uparrow}}^{x} r_t^{\Uparrow} \geq DS_i^{\Uparrow} \right\}. \tag{5}$$

Since the MD can only offload one task at a time, for any two tasks ς_i and ς_j, we have

$$ST_j^{\Uparrow} \geq FT_i^{\Uparrow}, \forall i, j \in \{1, 2, \ldots, N\}, i \neq j, ST_j^{\Uparrow} > ST_i^{\Uparrow}. \tag{6}$$

Then, the offloading energy consumption of ς_i is

$$E_i^{\Uparrow} = p^{\Uparrow} \times (FT_i^{\Uparrow} - ST_i^{\Uparrow}). \tag{7}$$

Accordingly, the MD's offloading energy consumption is

$$E^{\Uparrow} = \sum_{i=1}^{N} E_i^{\Uparrow}. \tag{8}$$

E. Downloading Model (MTTS-Stage2)

Let $ST_i^{\Downarrow}$ be the downloading start time of ς_i. Considering that the result data of ς_i can be transmitted back only after its execution is finished, $ST_i^{\Downarrow}$ should satisfy

$$ST_i^{\Downarrow} > FT_i^{\text{exe}}, \tag{9}$$

where FT_i^{exe} is the execution finish time of ς_i that can be determined by

$$FT_i^{\text{exe}} = ST_i^{\text{exe}} + \frac{DS_i^{\Uparrow} CI_i}{f_{x_i}}, \tag{10}$$

where x_i indicates the index of the edge node where ς_i is executed.

For a task to start execution, the following two conditions must be satisfied: a) the offloading of the task's input data is completed, and b) the edge server assigned to the task is available. Therefore, the execution start time of ς_i is

$$ST_i^{\text{exe}} = \max\{FT_i^{\Uparrow}, FT_{x_i}^{\text{pre}}\}, \tag{11}$$

where $FT_{x_i}^{\text{pre}}$ represents the execution finish time of ς_i's previous task on s_{x_i}. Then, ς_i's downloading finish time is

$$FT_i^{\Downarrow} = \arg\min \left\{ x \,\middle|\, \sum_{t=ST_i^{\Downarrow}}^{x} r_t^{\Downarrow} \geq DS_i^{\Downarrow} \right\}. \tag{12}$$

Since the MD can only download one task at a time, for any two tasks ς_i and ς_j, we have

$$ST_j^{\Downarrow} \geq FT_i^{\Downarrow}, \forall i, j \in \{1, 2, \ldots, N\}, i \neq j, ST_j^{\Downarrow} > ST_i^{\Downarrow}. \tag{13}$$

The downloading energy consumption of ς_i is

$$E_i^{\Downarrow} = p^{\Downarrow} \times (FT_i^{\Downarrow} - ST_i^{\Downarrow}). \tag{14}$$

Accordingly, the MD's downloading energy consumption is

$$E^{\Downarrow} = \sum_{i=1}^{N} E_i^{\Downarrow}. \tag{15}$$

F. The Optimization Model for MTTS

Taken together, the MTTS problem studied in this work can be formulated as the following optimization model:

$$\text{minimize} \quad E = E^{\Uparrow} + E^{\Downarrow} \tag{16}$$

$$\text{subject to} \quad FT_i^{\Downarrow} \leq DL_i \tag{17}$$

$$ST_j^{\Uparrow} \geq FT_i^{\Uparrow}, \forall ST_j^{\Uparrow} > ST_i^{\Uparrow} \tag{18}$$

$$ST_j^{\Downarrow} \geq FT_i^{\Downarrow}, \forall ST_j^{\Downarrow} > ST_i^{\Downarrow} \tag{19}$$

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

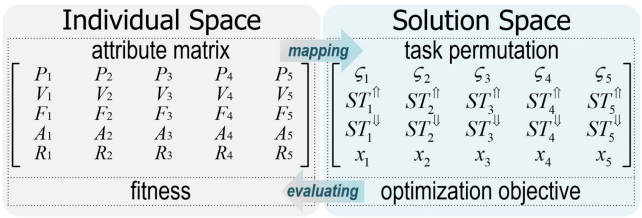


Fig. 1. The individual space and solution space in DORA, taking BA as an example. (The attributes P_i, V_i, F_i, A_i, and R_i indicate the position, velocity, frequency, loudness, and pulse rate of the individual's i-th dimension).

$$ST_i^{\Downarrow} \geq FT_i^{\text{exe}} \tag{20}$$

$$\text{variables} \quad ST_i^{\Uparrow} \in [1, T], ST_i^{\Downarrow} \in [1, T], x_i \in [1, M] \tag{21}$$

IV. The DORA Framework

The proposed DORA framework is a dual-population evolutionary computation framework that can integrate various SIAs for solving the MTTS problem. The following provides a detailed explanation of each component within DORA.

A. Solution Representation and Initialization

As a dual-population evolutionary computational framework, DORA begins by initializing two populations, with each population composed of multiple individuals. Specifically, the stage-1 population (denoted by $P^{\Uparrow}$) and the stage-2 population (denoted by $P^{\Downarrow}$) are denoted by $P^{\Uparrow} = \{\Phi_1^{\Uparrow}, \Phi_2^{\Uparrow}, \ldots, \Phi_I^{\Uparrow}\}$ and $P^{\Downarrow} = \{\Phi_1^{\Downarrow}, \Phi_2^{\Downarrow}, \ldots, \Phi_I^{\Downarrow}\}$, respectively. Here, I is the population size.

As shown in Fig. 1, the DORA framework explores two complementary search spaces:

- *Individual space:* Stores the SIA-specific biological attributes that drive the algorithmic operators. Taking BA as an example, an individual is described by its position P_i, velocity V_i, frequency F_i, loudness A_i, etc.
- *Solution space:* Stores the scheduling solution, represented by the task sequence in this work, attached to each individual. Given N tasks to be scheduled, i.e.,

$$\Phi_i^{\Uparrow/\Downarrow} = \{\phi_1, \phi_2, \ldots, \phi_N\}, \tag{22}$$

For example, assume that the subscript of each element in Φ is precisely the task index, that is,

$$\phi_j \leftrightarrow \text{task } j = \varsigma_j, \qquad j = 1, \ldots, N. \tag{23}$$

Consequently, we can obtain

$$\Phi_i^{\Uparrow/\Downarrow} = \{\varsigma_1, \varsigma_2, \ldots, \varsigma_N\}. \tag{24}$$

Throughout the search process, the update rules first refine the attribute vector stored in the individual space. A dedicated mapping operator then converts these updated attributes into the task sequence that resides in the solution space. The details of the proposed mapping operator will be discussed in Section IV-C. Benefiting from this attribute-driven update scheme and the dual-space representation, the proposed DORA framework remains compatible with a broad array of swarm-intelligence algorithms and can effectively solve a wide range of MTTS-type two-stage problems.

B. Inter-Population Cooperative Mechanism

In DORA, the stage-1 population $P^{\Uparrow}$ focuses on exploring the solution space of the offloading sub-problem, while the stage-2 population $P^{\Downarrow}$ is dedicated to the downloading sub-problem. In contrast to the three representative inter-population cooperation strategies commonly adopted in existing dual-population algorithms (as summarized in Table II), our DORA framework introduces a novel cooperative mechanism based on asynchronous solution transfer, which is elaborated below.

Fundamentally, the cooperative mechanism in DORA employs solution transfer to address the interdependence between the two sub-problems. The reason lies in the dependence of the two sub-problems, i.e., the start time of result downloading in the stage-2 sub-problem cannot be earlier than the execution finish time in the stage-1 sub-problem. As a result, a large number of infeasible scheduling decisions would occur, seriously degrading the algorithm's performance. If the two populations evolve independently in their respective search spaces and simply combine the two solutions obtained by them as the final solution, a large number of infeasible scheduling decisions would occur, seriously degrading the algorithm's performance. For this consideration, $P^{\Downarrow}$ evolves based on the solution obtained by the $P^{\Uparrow}$, thereby named the solution transfer-based cooperative mechanism, with the transfer direction from $P^{\Uparrow}$ to $P^{\Downarrow}$.

Complementing the solution transfer design, the cooperation in DORA is implemented in an asynchronous fashion, enabling dynamic adaptation to evolving search states. This asynchronous behavior is facilitated through a shared cooperative module, in which $P^{\Uparrow}$ records every newly discovered better solution as the current best stage-1 solution. This shared record serves as the decision cue for $P^{\Downarrow}$: after each complete round of $G_{\max}^{\Downarrow}$ iterations, $P^{\Downarrow}$ reads the stored stage-1 current best solution and decides whether to launch the next round. Here, one round of iterations denotes a full round of iterations that reaches the maximum number of iterations $G_{\max}^{\Downarrow}$ set by $P^{\Downarrow}$. If the acquired current best stage-1 solution is newly updated, $P^{\Downarrow}$ starts one round of iterations based upon this solution. If not, $P^{\Downarrow}$ keeps waiting until a new stage-1 solution is generated, or finishes as the evolution of the stage-1 population ends.

In brief, DORA enables the two populations to evolve concurrently while interacting asynchronously through such a cooperative mechanism, which ensures coordinated progress and feasibility across interdependent sub-problems.

C. Mapping Operator

To link the individual space and the solution space introduced above, we design an exponential probability-based mapping operator (MAP) that generates candidate solutions for individuals. The widely used mapping method, known as the ranked-order value (ROV) rule, relies solely on an individual's position [45]. Unlike ROV, the proposed MAP takes into account the solution of the current best individual. By inheriting the "good genes" inside of the current best solution in a probabilistic sense, MAP

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

is capable of exploring high-quality solutions and enhancing the effectiveness of DORA.

For any population, let I^* denote the current best individual, and Π^* denotes I^*'s solution (i.e., the current best solution). For any individual I_i to be mapped, MAP has the following two operations to generate Π_i:

A) copy some tasks from Π^* to the same location of Π_i;
B) place the remaining tasks on the empty cells of Π_i in random order.

Regarding operation(A), for any dimension q in Π_i ($q = 1, 2, \ldots, N$), the probability that q-th task copied from Π^* is

$$prob = \overline{P}_{iq}{}^{\varphi}, \tag{25}$$

where $\varphi \in (0, 1)$ is a critical parameter that will be further investigated later, and $\overline{P}_{iq}$ is the normalized value of position P_{iq}, i.e.,

$$\overline{P}_{iq} = \frac{P_{iq} - P_{\min}}{P_{\max} - P_{\min}}, \tag{26}$$

where $P_{\min}$ and $P_{\min}$ denote the lower and upper bounds of the position in individual space, respectively. Based on (25), apparently, the value of $prob$ is within [0,1] regardless of the position's value domain.

To investigate the impact of the critical parameter φ on the search performance of MAP, we introduce the concept of template [46], which indicates the good genes hidden in the high-quality scheduling solution. For simplicity, in what follows, the scheduling solutions represented by task permutations are regarded as integer sequences formed by various possible orderings of task indexes. Accordingly, for any population, the template $\mathcal{S}$ is a sequence of N elements, consisting of the task indexes {1,2,. . . , N} and the symbol #. Here, # can be repeated multiple times, whereas the same integer (or task index) can appear at most once. Taking a scheduling solution containing five tasks as an example, suppose that the optimal solution is Π^*={1,2,3,4,5} and the template $\mathcal{S}$ is {1,#,3,#,5}. Hence, {1,4,3,2,5} is a template-matched solution, whereas {2,1,3,4,5} is a non-template-matched solution.

During the evolutionary process, let $\Pi_u^{\Uparrow}$ represent an offloading solution from $P^{\Uparrow}$ and be recorded in the cooperative module. And $\Pi_u^{\Downarrow}$ represents a downloading solution in the $P^{\Downarrow}$. Then, we have $\Omega^{\Uparrow} = \{\Pi_u^{\Uparrow} | u = 1, 2, \ldots, \Gamma\}$ and $\Omega_t^{\Downarrow} = \{\Pi_v^{\Downarrow} | v = 1, 2, \ldots, \Gamma\}$ to represent the solution set of all individuals in $P^{\Uparrow}$ and $P^{\Downarrow}$, respectively. Regarding the current best solution in $P^{\Uparrow}$ and $P^{\Downarrow}$ as $\mathcal{S}^{\Uparrow}$ and $\mathcal{S}^{\Downarrow}$, respectively, we have the following theorem.

Theorem 1: $E[|(\mathcal{S}^{\Downarrow} \cap \Omega^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega^{\Uparrow})|] \geq \Gamma(\frac{\alpha^{N\alpha}}{N^{N-N\alpha}})^2$, where $\alpha = \overline{P}_{iq}{}^{\varphi}$.

Proof: The symbol $\cap$ indicates taking the intersection set. Based on the solution transfer mechanism, $(\mathcal{S}^{\Downarrow} \cap \Omega^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega^{\Uparrow})$ denotes the template-matched solutions obtained by $P^{\Downarrow}$. Accordingly, $E[|(\mathcal{S}^{\Downarrow} \cap \Omega^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega^{\Uparrow})|]$ is the expectation number of template-matched solutions.

As detailed above, MAP involves two operations to map an individual onto a solution. For $\Pi_u^{\Uparrow}$, assume that L tasks are copied to $\Pi_u^{\Uparrow}$ in operation(A). To ensure that the final generated $\Pi_u^{\Uparrow}$ matches $\mathcal{S}^{\Uparrow}$, the remaining $N-L$ tasks must be placed at their original locations in operation(B). Obviously, the probability of this case is

$$\mathcal{P}_L\left(\Pi_u^{\Uparrow}, \mathcal{S}^{\Uparrow}\right) = \delta_L C_N^L \prod_{\forall t \in T_L^{\Uparrow}} \overline{P}_{iq}{}^{\varphi} \prod_{\forall t \in (T - T_L^{\Uparrow})} \left(1 - \overline{P}_{iq}{}^{\varphi}\right), \tag{27}$$

where $T_L^{\Uparrow}$ denotes the set of L tasks that are copied to $\Pi_u^{\Uparrow}$ in operation(A). Clearly, $T_L^{\Uparrow}$ has a total of C_N^L possible scenarios. In other words, C_N^L indicates the number of combinations of $T_L^{\Uparrow}$. δ_L is the probability that all remaining tasks are inserted into their original locations in operation(B), so we have $\delta_L = 1/A_N^{N-L}$. Here, A_N^{N-L} is the number of permutations for selecting $N - L$ tasks from N tasks.

Thus, the probability of $\Pi_u^{\Uparrow}$ matching the template $\mathcal{S}^{\Uparrow}$ is the sum of $\mathcal{P}_L(\Pi_u^{\Uparrow}, \mathcal{S}^{\Uparrow})$ for all possible L values:

$$\mathcal{P}\left(\Pi_u^{\Uparrow}, \mathcal{S}^{\Uparrow}\right) = \sum_{L=0}^{N} \mathcal{P}_L\left(\Pi_u^{\Uparrow}, \mathcal{S}^{\Uparrow}\right). \tag{28}$$

In a similar manner, the probability of template matching for an individual in $P^{\Downarrow}$ is

$$\mathcal{P}\left(\Pi_v^{\Downarrow}, \mathcal{S}^{\Downarrow}\right) = \sum_{L=0}^{N} \mathcal{P}_L\left(\Pi_v^{\Downarrow}, \mathcal{S}^{\Downarrow}\right), \tag{29}$$

where

$$\mathcal{P}_L\left(\Pi_v^{\Downarrow}, \mathcal{S}^{\Downarrow}\right) = \delta_L C_N^L \prod_{\forall t \in T_L^{\Downarrow}} \overline{P}_{iq}{}^{\varphi} \prod_{\forall t \in (T - T_L^{\Downarrow})} \left(1 - \overline{P}_{iq}{}^{\varphi}\right). \tag{30}$$

Based on the preceding derivations, the probability of template matching in terms of the final solution is

$$\begin{aligned} \mathcal{P}\left((\mathcal{S}^{\Downarrow} \cap \Omega^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega^{\Uparrow})\right) &= \mathcal{P}\left(\Pi_u^{\Uparrow}, \mathcal{S}^{\Uparrow}\right) \times \mathcal{P}\left(\Pi_v^{\Downarrow}, \mathcal{S}^{\Downarrow}\right) \\ &= \left(\sum_{L=0}^{N} \mathcal{P}_L\left(\Pi_u^{\Uparrow}, \mathcal{S}^{\Uparrow}\right)\right)^2. \end{aligned} \tag{31}$$

Then, we can derive that

$$\begin{aligned} &E\left[\left|(\mathcal{S}^{\Downarrow} \cap \Omega^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega^{\Uparrow})\right|\right] \\ &\quad = \Gamma \times \mathcal{P}\left((\mathcal{S}^{\Downarrow} \cap \Omega^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega^{\Uparrow})\right) \\ &\quad = \Gamma \left(\sum_{L=0}^{N} \mathcal{P}_L\left(\Pi_u^{\Uparrow}, \mathcal{S}^{\Uparrow}\right)\right)^2. \end{aligned} \tag{32}$$

According to the probability threshold in MAP defined in (25), the expectation value of L can be determined as

$$E[L] = N \times \overline{P}_{iq}{}^{\varphi}. \tag{33}$$

Taken together, we have

$$\begin{aligned} &E\left[\left|(\mathcal{S}^{\Downarrow} \cap \Omega^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega^{\Uparrow})\right|\right] \\ &\quad = \Gamma \left(\sum_{L=0}^{N} \left(\delta_L C_N^L \prod \overline{P}_{iq}{}^{\varphi} \prod \left(1 - \overline{P}_{iq}{}^{\varphi}\right)\right)\right)^2 \\ &\quad = \Gamma \left(\frac{\prod \overline{P}_{iq}{}^{\varphi} \prod \left(1 - \overline{P}_{iq}{}^{\varphi}\right)}{\left(N - N\overline{P}_{iq}{}^{\varphi}\right)!}\right)^2. \end{aligned} \tag{34}$$

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

Let $\alpha = \overline{P}_{iq}{}^{\varphi}$, $E[|(\mathcal{S}^{\Downarrow} \cap \Omega^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega^{\Uparrow})|]$ can be further derived as

$$E\left[\left|(\mathcal{S}^{\Downarrow} \cap \Omega_t^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega_t^{\Uparrow})\right|\right] \geq \Gamma\left(\frac{\alpha^{N\alpha}(1-\alpha)^{N-N\alpha}}{(N-N\alpha)^{N-N\alpha}}\right)^2 = \Gamma\left(\frac{\alpha^{N\alpha}}{N^{N-N\alpha}}\right)^2. \quad (35)$$

■

The derivations in Theorem 1 obtain the lower bound of $E[|(\mathcal{S}^{\Downarrow} \cap \Omega_t^{\Downarrow}) \Leftarrow (\mathcal{S}^{\Uparrow} \cap \Omega_t^{\Uparrow})|]$, which is denoted by E_{lb} for simplicity. In what follows, we discuss the monotonicity of E_{lb} with respect to the critical parameter φ.

Lemma 1: $E_{\text{lb}}(\alpha) = \Gamma(\frac{\alpha^{N\alpha}}{N^{N-N\alpha}})^2$, where $\alpha = \overline{P}_{iq}{}^{\varphi}$, is a monotonically increasing function with regard to φ.

Proof: Since $E_{\text{lb}}(\alpha) > 0$, we have

$$E_{\text{lb}}(\alpha) = e^{\ln \Gamma\left(\frac{\alpha^{N\alpha}}{N^{N-N\alpha}}\right)^2}. \quad (36)$$

Let $g(\alpha) = \ln(\frac{\Lambda\alpha^{2N\alpha}}{(\hat{N})^{2N-2N\alpha}})$, the monotonicity of $E_{\text{lb}}(\alpha)$ and $g(\alpha)$ will be consistent. Therefore, we investigate the monotonicity of $g(\alpha)$ next, which can be further derived by

$$\begin{aligned} g(\alpha) &= \ln \Gamma + 2\left(\ln\left(\alpha^{N\alpha}\right) - \ln\left(N^{N-N\alpha}\right)\right) \\ &= \ln \Gamma + 2N\alpha \ln \alpha - 2N \ln N + 2N\alpha \ln N. \end{aligned} \quad (37)$$

Obviously, when $g'(\alpha) > 0$, we have

$$\begin{aligned} g'(\alpha) > 0 &\Rightarrow 2N(\ln \alpha + \ln N + 1) > 0 \\ &\Rightarrow \ln \alpha > -1 - \ln N \\ &\Rightarrow \alpha > \exp -1 - \ln N \\ &\Rightarrow \alpha > \frac{1}{Ne}. \end{aligned} \quad (38)$$

Since N is generally a large value in the context of static scheduling, $g'(\alpha) > 0$ holds, which means $g(\alpha)$ and $E_{\text{lb}}(\alpha)$ are monotonically increasing with regard to α. Moreover, $\alpha = \overline{P}_{iq}{}^{\varphi}$ is monotonically decreasing with regard to φ. Referring to the monotonicity principle of composite functions, E_{lb} is monotonically decreasing with regard to φ. ■

As demonstrated in Lemma 1, E_{lb} is monotonically decreasing with regard to φ. Therefore, the smaller the value of φ, the more the number of template-matched individuals in the population, leading to better effectiveness of MAP. However, the value of φ should not be too small. The reason is that a tiny value of φ would lose population diversity, resulting in the degradation of exploration capability.

D. Fitness Evaluation Operator

The fitness evaluation operator of stage-1 population (FEO-P$^{\Uparrow}$) accepts the individual's task permutation as input, and outputs the optimization objective value of stage-1 sub-problem in MTTS. As shown in Algorithm 1, FEO-P$^{\Uparrow}$ first makes the data offloading decision for each task by utilizing the transmission rate variations at all timeslots along an MD's trajectory to minimize the offloading energy consumption, and then makes the edge server assignment decision by utilizing the limited computing resources in the MEC system. Specifically, FEO-P$^{\Uparrow}$ traverses all elements of the input task permutation in turn. For each task, Algorithm 3 is called to obtain the energy-efficiency offloading timeslots, which is a sliding window strategy based on a start time pointer and an end time pointer. Then, the execution time of each task is determined by finding the edge server that can finish it earliest. Also, the obtained execution finish time of the task will be recorded in a list, which will be acquired by the stage-2 population and play an essential role in the cooperative mechanism.

Algorithm 1: Stage-1 Fitness Evaluation (FEO-P$^{\Uparrow}$).

Input: i) individual's solution: $\Phi = \{\phi_1, \phi_2, \ldots, \phi_N\}$;
ii) uplink transmission rate: $R^{\Uparrow} = \{r_1^{\Uparrow}, r_2^{\Uparrow}, \ldots, r_T^{\Uparrow}\}$;
iii) uplink transmission power: $p^{\Uparrow}$;
iv) edge servers: $S = \{s_1, s_2, \ldots, s_M\}$.
Output: i) optimization objective: $obj^{\Uparrow}$;
ii) execution finish time: FT^{Exe}=$\{ft_1^{\text{Exe}}, ft_2^{\text{Exe}}, \ldots, ft_N^{\text{Exe}}\}$.

for ϕ_i *in* Φ **do**
  /*`offloading time determination`*/
  Get offloading data size $DS_i^{\Uparrow}$;
  Calculate latest offloading start time $LST_i^{\Uparrow}$;
  Get offloading timeslots $[ST_i^{\Uparrow}, FT_i^{\Uparrow}]$ by Algorithm 3 taking $(DS_i^{\Uparrow}, R^{\Uparrow}, p^{\Uparrow}, [0, LST_i^{\Uparrow}])$ as input;
  Label $[ST_i^{\Uparrow}, FT_i^{\Uparrow}]$ in $R^{\Uparrow}$ as non-available;
  Calculate $E_i^{\Uparrow}$ by Eq.(7);
  $obj^{\Uparrow} \leftarrow obj^{\Uparrow} + E_i^{\Uparrow}$;
  /*`execution time determination`*/
  Find edge server $s_x \in S$ that can finish the execution earliest;
  Get execution timeslots $[ST_i^{\text{Exe}}, FT_i^{\text{Exe}}]$;
  Record FT_i^{Exe} in the corresponding element in FT^{Exe};
  Label $[ST_i^{\text{Exe}}, FT_i^{\text{Exe}}]$ of s_x as non-idle;
return $obj^{\Uparrow}$, FT^{Exe};

Algorithm 2: Stage-2 Fitness Evaluation (FEO-P$^{\Downarrow}$).

Input: i) individual's solution: $\Phi = \{\phi_1, \phi_2, \ldots, \phi_N\}$;
ii) downlink transmission rate: $R^{\Downarrow} = \{r_1^{\Downarrow}, r_2^{\Downarrow}, \ldots, r_T^{\Downarrow}\}$;
iii) downlink transmission power: $p^{\Downarrow}$;
iv) execution finish time: FT^{Exe}=$\{ft_1^{\text{Exe}}, ft_2^{\text{Exe}}, \ldots, ft_N^{\text{Exe}}\}$.
Output: $obj^{\Downarrow}$.

for ϕ_i *in* Φ **do**
  /*`downloading time determination`*/
  Get downloading data size $DS_i^{\Downarrow}$;
  Calculate latest downloading start time $LST_i^{\Downarrow}$;
  Get execution finish time FT_i^{Exe};
  Get offloading timeslots $[ST_i^{\Downarrow}, FT_i^{\Downarrow}]$ by Algorithm 3 taking $(DS_i^{\Downarrow}, R^{\Downarrow}, p^{\Downarrow}, [FT_i^{\text{Exe}}, LST_i^{\Downarrow}])$ as input;
  Label $[ST_i^{\Downarrow}, FT_i^{\Downarrow}]$ in $R^{\Downarrow}$ as non-available;
  Calculate $E_i^{\Downarrow}$ by Eq.(14);
  $obj^{\Downarrow} \leftarrow obj^{\Downarrow} + E_i^{\Downarrow}$;
return $obj^{\Downarrow}$;

The fitness evaluation operator of stage-2 population (FEO-P$^{\Downarrow}$) accepts the individual's task permutation as input, and outputs the optimization objective value of stage-2 sub-problem in MTTS. As shown in Algorithm 2, FEO-P$^{\Downarrow}$ makes the data downloading decision for each task by utilizing the transmission rate variations at all timeslots along an MD's trajectory to minimize the downloading energy consumption. Specifically, FEO-P$^{\Downarrow}$ traverses all elements of the input task permutation

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

Algorithm 3: Off/Downloading Strategy.

Input: i) data size: DS;
ii) transmission rate: R;
iii) transmission power: p;
iv) time range: $[T_{\text{Start}}, T_{\text{End}}]$.
Output: offloading/downloading time: $[T^*_{\text{Start}}, T^*_{\text{End}}]$.
/* P_{S}:start time pointer, P_{E}:end time pointer */
$P_{\text{S}} \leftarrow T_{\text{Start}}$;
while $P_S \leq T_{End}$ **do**
  $P_{\text{E}} \leftarrow P_{\text{S}} + 1$;
  while *true* **do**
    if $\sum_{t=P_S}^{P_E} r_t \geq DS_i$ **then**
      $E_{\text{temp}} \leftarrow p(P_{\text{E}} - P_{\text{S}})$;
      if $E_{temp} < E_{best}$ **then**
        $E_{\text{best}} \leftarrow E_{\text{temp}}$;
        $T^*_{\text{Start}} \leftarrow P_{\text{S}}$, $T^*_{\text{End}} \leftarrow P_{\text{E}}$;
        break;
    else if r_{P_E} *is equal to 0* **then**
      $P_{\text{S}} \leftarrow P_{\text{E}} + 1$;
      break;
    $P_{\text{E}} \leftarrow P_{\text{E}} + 1$;
  $P_{\text{S}} \leftarrow P_{\text{S}} + 1$;
return $[T^*_{\text{Start}}, T^*_{\text{End}}]$;

Algorithm 4: Adaptive Iteration Controller (AIC).

Input: i) population $\delta \in \{\Uparrow, \Downarrow\}$;
ii) number of tasks N;
iii) related parameters: scaling factors (ι, κ), threshold ε, and window L^δ.
Output: terminated iteration count G^δ.
if $\delta = \Uparrow$ **then**
  $G^\delta_{\max} \leftarrow \iota N$;
else
  $G^\delta_{\max} \leftarrow \kappa N$;
$g \leftarrow 0$, $c \leftarrow 0$;
while $g < G^\delta_{\max}$ **do**
  $g \leftarrow g + 1$;
  Run one evolutionary iteration for population P^δ;
  Obtain current best fitness $F^{\text{new}}_{\text{best}}$;
  $\Delta F \leftarrow F^{\text{old}}_{\text{best}} - F^{\text{new}}_{\text{best}}$;
  if $\Delta F \leq \varepsilon |F^{\text{old}}_{\text{best}}|$ **then**
    $c \leftarrow c + 1$;
  $c \leftarrow 0$;
  if $c \geq L^\delta$ **then**
    break;
  $F^{\text{old}}_{\text{best}} \leftarrow F^{\text{new}}_{\text{best}}$;
return $G^\delta = g$;

in turn. For each task, Algorithm 3 is called to obtain the energy-efficient downloading timeslots. Here, the task's execution finish time, which is obtained by the stage-1 population and transferred to the stage-2 population, is set as the start pointer, whereas the latest downloading start time is set as the end pointer. After traversing all the tasks, the optimization objective value of MTTS's stage-2 sub-problem is obtained, which is also the fitness of the current individual in stage-2 population.

It is worth noting that this scheduling framework is developed specifically for independent tasks, but can be extended to accommodate precedence-constrained tasks of workflow applications. To this end, the scheduling decision for each task must be restricted by the completion times of its predecessor tasks to comply with precedence relations. A feasible extension is to incorporate a precedence-aware refinement mechanism into the decision-making process. Prior to determining the offloading, execution, or downloading time of a single task, the scheduler would derive its earliest available start time from the DAG structure and exclude any scheduling intervals that would violate precedence relations. This mechanism would enable the proposed framework to accommodate workflow applications while preserving the overall structure of evolutionary computation.

E. Adaptive Iteration Controller

To enhance computational efficiency and ensure that the search process adapts to the problem scale and the evolutionary progress, an adaptive iteration controller (AIC) is incorporated into both the offloading and downloading populations. The controller adjusts the iteration budget based on two components: (i) scale-aware initial iteration allocations and (ii) progress-based early stopping mechanism.

The detailed procedure of the AIC controller is provided in Algorithm 4. First, the scale-aware initial iteration allocations specify the maximum number of iterations for each population based on the problem size. Let N denote the number of tasks. The upper bounds of the two populations are set as $G^{\Uparrow}_{\max} = \iota N$ and $G^{\Downarrow}_{\max} = \kappa N$, where $\iota > \kappa > 0$ are global constants. In this way, the initial iteration allocations grow linearly with the task scale while preserving a larger search capacity for the offloading population. This design also avoids scenario-specific configuration and provides scale-aware hard limits on the maximum iteration budgets.

Second, both populations continuously monitor the evolutionary progress in solution quality and can terminate early when the improvement becomes negligible. For the offloading population, the controller evaluates the improvement magnitude $\Delta F = F^{\text{old}}_{\text{best}} - F^{\text{new}}_{\text{best}}$ at the end of each iteration, and compares it with a relative threshold ε. If $\Delta F \leq \varepsilon |F^{\text{old}}_{\text{best}}|$ persists for $L^{\Uparrow}$ consecutive iterations, the offloading evolution terminates before reaching $G^{\Uparrow}_{\max}$. A similar mechanism is applied to the downloading population within each round, enabling each round to end early when progress is negligible.

By combining these two components, the AIC controller enables DORA to allocate iterations to the evolutionary phases where meaningful improvement is still achievable, and to reduce redundant iterations when the search becomes stagnant. Accordingly, the computational cost is adjusted automatically according to both the task scale and the observed optimization behavior, rather than being fixed a priori. Overall, the AIC controller relieves the requirement for manual configuration, while improving computational efficiency and maintaining solution quality across different problem instances and swarm intelligence algorithms in the DORA framework.

F. Descriptions of DORA Framework

As illustrated in Fig. 2, DORA begins with determining the evolutionary algorithms, and completing the initialization

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

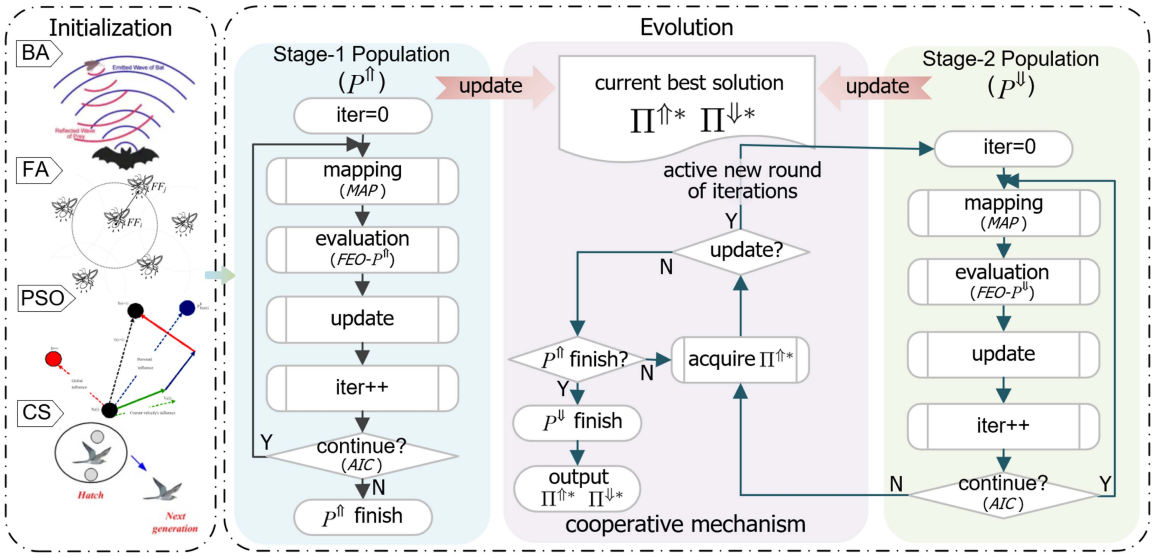


Fig. 2. The proposed evolutionary computation framework DORA.

of population, initial solution, and relevant parameters. After that, DORA enters the iterative process, where $P^{\Uparrow}$ population performs one round of iterations and $P^{\Downarrow}$ population might perform multiple rounds of iterations, according to the asynchronous solution transfer mechanism (detailed in Section IV-B). During each iteration, DORA first generates the corresponding solution for each individual, which is based upon the proposed mapping method (detailed in Section IV-C). Next, the fitness of each individual (i.e., the optimization objective of each individual's corresponding solution) will be calculated based on the problem-dependent evaluation algorithm (detailed in Section IV-D). Then, DORA updates the population following the search mechanisms of the predetermined SIA. Overall, DORA iteratively performs the solution generation, fitness evaluation, and population updating to obtain the final solution to the considered problem.

From a methodological perspective, the proposed dual-population framework is motivated by the growing disparity between the substantially enlarged solution space, which is induced by incorporating the result downloading decision into the scheduling problem, and the increasing complexity of effective solution exploration. The interdependence between offloading and downloading decisions further exacerbates the difficulty in exploring high-quality schedules in the solution space. By employing two populations dedicated to task offloading and result downloading decisions, upon which an inter-population collaboration mechanism is further established, the solution exploration capability of the proposed DORA can be significantly enhanced.

G. Complexity Analysis

The following presents an analysis of the computational complexity of the proposed DORA framework. First, we discuss the complexity of each process within DORA. In the initialization step, the individuals in the two populations are initialized randomly across each dimension, resulting in a computational complexity of $\mathcal{O}(N \cdot \Gamma)$. During each iteration in the evolutionary process, three steps are performed for each individual in any population: the mapping operation, the evaluation operation, and the update operation. The mapping method MAP has a computational complexity of $\mathcal{O}(N)$, while the evaluation method has a complexity of $\mathcal{O}(N \cdot \log E[DL])$. The individual update operation depends on the specific SIA deployed in the DORA framework and is not analyzed here. At the end of each iteration, all individuals need to be checked to determine the current best individual in the population, with a computational complexity of $\mathcal{O}(\Gamma)$.

Next, we need to consider the total number of iterations of DORA. The total number of iterations of the offloading population $P^{\Uparrow}$ is a fixed value $G^{\Uparrow}_{\max}$, while the total number of iterations of the downloading population $P^{\Downarrow}$ throughout the algorithm's lifecycle is not a fixed value $G^{\Downarrow}_{\max}$. This is because $P^{\Downarrow}$ initiates multiple rounds of iterations based on the inter-population collaboration mechanism. Therefore, the upper and lower bounds of the total number of iterations of $P^{\Downarrow}$ are discussed next, denoted by $G^{\Downarrow}_{\text{lb}}$ and $G^{\Downarrow}_{\text{ub}}$, respectively. Firstly, consider the situation for $G^{\Downarrow}_{\text{lb}}$, where $T^{\Uparrow}_{\text{round}} > \omega T^{\Downarrow}_{\text{round}}$, with $T^{\Uparrow}_{\text{round}}$ and $T^{\Downarrow}_{\text{round}}$ representing the time for $P^{\Uparrow}$ and $P^{\Downarrow}$ to complete one round of iterations, and ω being the number of iteration rounds conducted by $P^{\Downarrow}$. Specifically, this situation indicates that $P^{\Uparrow}$ does not update the current optimal offloading solution in the later stages of the iteration process. Consequently, after completing ω rounds of iterations, $P^{\Downarrow}$ remains in a waiting state until $P^{\Uparrow}$ completes, ending the entire algorithm, thus $G^{\Downarrow}_{\text{lb}} = G^{\Uparrow}_{\max}$. The underlying reason is that the key processes in $P^{\Uparrow}$ and $P^{\Downarrow}$ are highly similar, making the time for a single iteration nearly identical. Next, consider the situation for $G^{\Downarrow}_{\text{ub}}$, where $T^{\Uparrow}_{\text{round}} \leq \omega T^{\Downarrow}_{\text{round}}$. This situation indicates that $P^{\Uparrow}$ updates the current optimal offloading solution in the last iteration, while $P^{\Downarrow}$ has just initiated a new round of iterations before the last iteration of $P^{\Uparrow}$. Thus, $G^{\Downarrow}_{\text{ub}} = G^{\Uparrow}_{\max} - 1 + 2G^{\Downarrow}_{\max}$. Since the number of iterations is generally

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

TABLE III
PARAMETER SETTINGS FOR SWARM INTELLIGENCE ALGORITHMS USED FOR EVALUATION

General Settings		BA			
Parameter	Value	Parameter	Value	Parameter	Value
Γ	30	$f_{\min}$	0	$f_{\max}$	1
$P_{\min}$	0	$v_{\min}$	-2	$v_{\max}$	2
$P_{\max}$	10	f_0	rand(0, 1)	v_0	0
—	—	A_0	rand(0.7, 1)	α	0.98
—	—	r_0	rand(0, 0.4)	γ	0.98
FA		PSO		CS	
Parameter	Value	Parameter	Value	Parameter	Value
β_0	1	w_0	0.9	pa	0.25
γ	1	c_1	2	α	1
α	rand(0, 1)	c_2	2	β	1
—	—	$w_{\min}$	0.4	—	—

large, the constant terms can be ignored, leading to $G^{\Downarrow}_{\text{ub}} = G^{\Uparrow}_{\max} + 2G^{\Downarrow}_{\max}$.

As a result, the lower computational amount of the DORA framework is $2N \cdot \Gamma + G^{\Uparrow}_{\max}(\Gamma(N + N \cdot \log E[DL]) + \Gamma) + G^{\Downarrow}_{\text{lb}}(\Gamma(N + N \cdot \log E[DL]) + \Gamma)$, and the lower bound of the computational complexity is

$$\underline{\mathcal{O}(\text{DORA})} = \Gamma \cdot N \cdot G^{\Uparrow}_{\max} \cdot \log E[DL]. \tag{39}$$

The upper computational amount is $2N \cdot \Gamma + G^{\Uparrow}_{\max}(\Gamma(N + N \cdot \log E[DL]) + \Gamma) + G^{\Downarrow}_{\text{ub}}(\Gamma(N + N \cdot \log E[DL]) + \Gamma)$, and the upper bound of DORA's computational complexity can be given by

$$\overline{\mathcal{O}(\text{DORA})} = \begin{cases} \Gamma \cdot N \cdot G^{\Uparrow}_{\max} \cdot \log E[DL], & \text{if } G^{\Uparrow}_{\max} > 2G^{\Downarrow}_{\max} \\ \Gamma \cdot N \cdot G^{\Downarrow}_{\max} \cdot \log E[DL], & \text{otherwise.} \end{cases} \tag{40}$$

It is worth mentioning that the actual number of iterations, for either the offloading or downloading population, can be substantially lower than the pre-defined upper-bound value ($G^{\Downarrow}_{\max}$ or $G^{\Uparrow}_{\max}$), as the adaptive iteration controller can terminate the evolutionary process early when appropriate.

V. EXPERIMENTAL RESULTS

In this section, we first describe the experimental setup, including test instances, parameter settings, and evaluation metrics. Then, we analyze the impact of key parameters in DORA, such as the mapping parameter φ and the maximum iteration counts assigned to the two populations. This analysis also serves to validate the theoretical findings in Section IV-C. Subsequently, we assess the performance of DORA through comparative experiments with several existing SIAs and evaluate its effectiveness across different configurations.

A. Experimental Setup

We develop a real-world MD trajectory dataset MDT-2023,[1] which contains 80 trajectories under three modes of mobility. The trajectory collection was conducted on the campus of Nanjing University of Science and Technology, Nanjing 210094, China. UniStrong GPS G1 handheld receiver[2] can provide a location accuracy of better than 1.7 meters for 95 percent of the time. By using this receiver, the MDT-2023 records the MD's trajectory as a sequence of time-stamped GPS information (i.e., latitude and longitude) per second apart. We establish a testing instance set (TIS) consisting of six groups of testing instances with different numbers of tasks, i.e., {50, 100, 200, 300, 600, 1000}. Each group contains five testing instances, resulting in a total of 30 testing instances. The task characteristics, including the input data size, output data size, and average computation workload, are profiled from the real-world Google cluster-usage traces dataset "clusterdata-2011-2"[3], which provides detailed information about the resource utilization in Google's data centers.

The MEC system in the experiment consists of ten MEC nodes, each of which deploys an edge server with a 3.0 GHz operating frequency. The parameters in (3) and (4) are set identically to those in [11], [47]. Specifically, $b^{\Uparrow} = 10$ MHz, $b^{\Downarrow} = 20$ MHz, $g_0 = -40$ dB, $dis^* = 1$ m, $\theta = 4$, and $N_0 = -174$ dBm/Hz. The MD's transmission powers of both uplink and downlink are set as 100 mW. The parameter settings of SIAs are shown in Table III. Experiments were carried out on a machine with a 10-core 2.8 GHz CPU and 32 GB of memory. The evaluation system was fully implemented in Java.

Considering the stochastic nature of the evolutionary algorithms, we use the well-known analysis of variance (ANOVA) method to analyze the scheduling results in a statistical manner. Accordingly, relative error (RE) is introduced as an index to quantify the algorithm's effectiveness, which is defined as

$$RE = \frac{1}{R}\left(\sum_{r=1}^{R} \frac{obj_r - obj^*}{obj^*}\right) \times 100\%, \tag{41}$$

where R is the number of runs per testing instance, with $R = 5$ in our experiments. obj_r is the objective value obtained by the r-th run, and obj^* denotes the best objective value among all competing methods during all rounds. Accordingly, a smaller RE value implies better solution quality.

B. Evaluation of Critical Operators

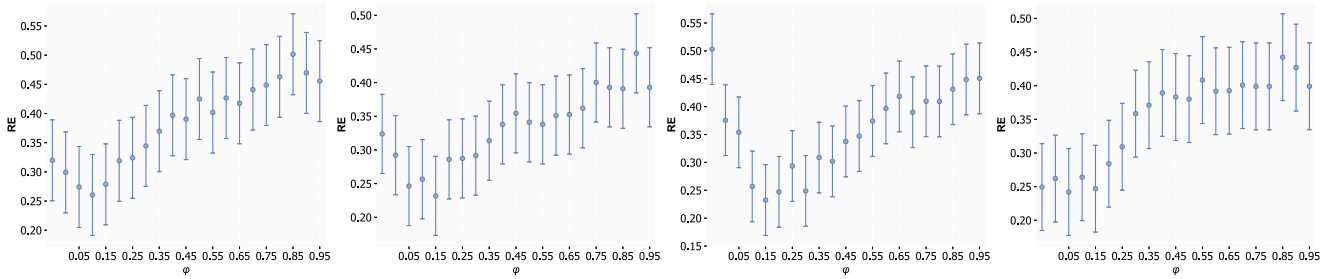
1) Parameter φ in MAP: To validate the theoretical analysis of the critical parameter involved in the mapping operator MAP, we integrate four classical SIAs into the DORA framework and evaluate the resulting REs under different φ values. Fig. 3 plots the mean values and 95% least significant difference (LSD) intervals of REs under different φ values of BA, FA, PSO, and CS, respectively. The results demonstrate that the value of φ generally has the same influence on the searching effectiveness of these SIAs. The greater the φ, the worse the RE, indicating MAP requires a small φ to achieve better effectiveness. However, when φ is extremely small (i.e., $\varphi \leq 0.1$), RE becomes worse,

[1] https://github.com/YinLu-NJUST/MDT-2023

[2] http://www.navearth.com

[3] https://github.com/google/cluster-data/blob/master/ClusterData2011_2.md

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.



(a) BA (b) FA (c) PSO (d) CS

Fig. 3. Mean REs and 95% LSD intervals obtained by BA, FA, PSO, and CS with different φ values.

TABLE IV
COMPARISON OF DORA WITH FIXED AND ADAPTIVE ITERATION CONFIGURATIONS ACROSS DIFFERENT TASK SCALES

N	Method	Avg. energy (J)	Avg. runtime (s)	Avg. iters$^{\Uparrow}$	Avg. iters$^{\Downarrow}$	Speedup
50	Fixed	18.956	0.61756	250.00	252.00	–
	Adaptive	18.968	0.29816	110.56	94.24	**1.93**×
100	Fixed	37.200	2.38204	500.00	528.00	–
	Adaptive	37.232	1.05544	221.92	214.64	**1.80**×
200	Fixed	76.040	10.17664	1000.00	1224.00	–
	Adaptive	76.112	5.12136	235.00	675.60	**2.01**×
300	Fixed	122.584	24.91444	1500.00	2268.00	–
	Adaptive	122.260	15.61688	938.40	1357.12	**2.68**×
600	Fixed	231.800	213.60540	3000.00	4608.00	–
	Adaptive	231.744	132.65920	1828.08	2538.48	**2.64**×
1000	Fixed	395.908	869.33612	5000.00	8160.00	–
	Adaptive	396.608	566.91304	3473.92	4659.68	**2.87**×

revealing that a tiny φ leads to poor effectiveness due to the loss of population diversity. Based on the above investigation, we verify the theoretical analysis provided in Section IV-C. Since these four SIAs all achieve satisfactory effectiveness when φ is 0.15, we set φ as 0.15in the following experiments.

2) Adaptive Iteration Controller in DORA: To evaluate the effectiveness of the AIC controller, we evaluate DORA under different task scales, in both fixed-iteration and adaptive iteration configurations. In experiments, the AIC controller adopts the settings $\iota = 5$, $\kappa = 3$, $\varepsilon^{\Uparrow} = \varepsilon^{\Downarrow} = 10^{-4}$, $L^{\Uparrow} = 0.1G^{\Uparrow}_{\max}$, and $L^{\Downarrow} = 0.1G^{\Downarrow}_{\max}$. All other algorithmic settings, parameter configurations, and testing instances were kept identical across the fixed-iteration and adaptive iteration configurations.

As shown in Table IV, the adaptive iteration configuration with AIC controller achieves substantial efficiency improvements. Across all task scales, the runtime is reduced by approximately 35%–56%, indicating 1.80×-2.87× speedups. This demonstrates that, once the search has stabilized, the AIC controller effectively prevents unnecessary iterations. Importantly, the optimization quality is preserved, i.e., for all task sizes, the average energy consumption of the adaptive version differs by less than 0.5% from that of the fixed iteration version. These observations confirm that the AIC controller successfully auto-tunes the computational cost according to instance difficulty while maintaining robustness and solution quality.

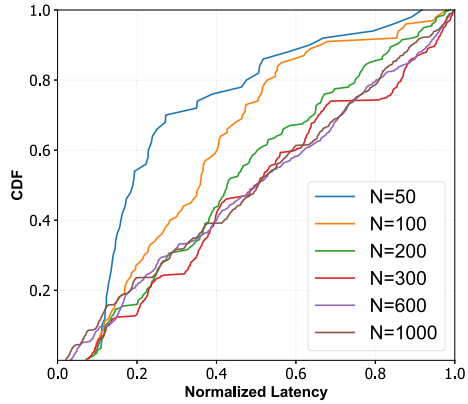


Fig. 4. Cumulative distribution of normalized task completion latency for different task scales.

C. Performance Evaluation

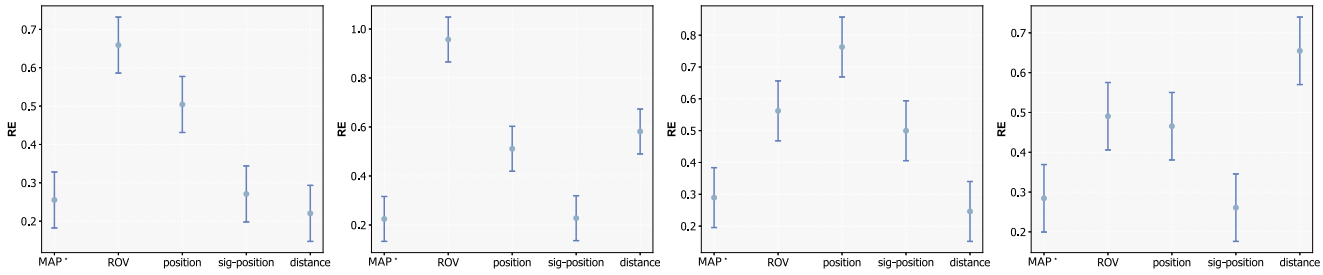
1) Latency Distribution Analysis: In order to characterize the latency distribution, the latency of each task is defined by a normalized metric:

$$L_i = \frac{FT_i^{\Downarrow}}{DL_i}, \tag{42}$$

where $FT_i^{\Downarrow}$ denotes the downloading completion time of task i and DL_i denotes its deadline. This normalized latency enables comparison across tasks with heterogeneous deadlines. A value of L_i close to zero indicates early completion with abundant slack, whereas a value close to one indicates completion approaching the deadline.

Fig. 4 illustrates the latency distribution in terms of the cumulative distribution function (CDF) of normalized latency under different task counts. As the problem scale increases, the CDF curves shift gradually to the right, indicating that a larger proportion of tasks complete closer to their deadlines due to the increased competition for limited transmission and execution resources. An important observation is that the normalized latency values in all CDF curves reach 1.0, indicating no deadline violations across all testing instances. Overall, the obtained latency distributions are consistent with the design principle of DORA,

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.



(a) BA (b) FA (c) PSO (d) CS

Fig. 5. Mean REs and 95% LSD intervals of DORA obtained by BA, FA, PSO, and CS when incorporating different mapping methods.

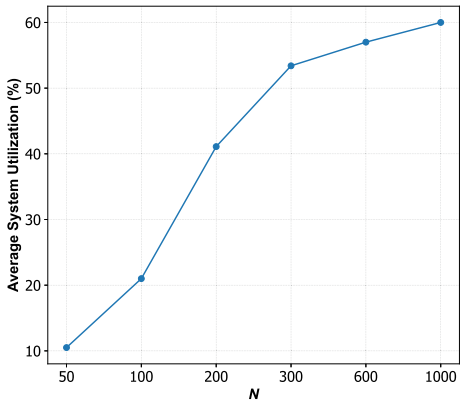


Fig. 6. Average server utilization under different task scales.

leveraging available deadline slack to allocate transmissions to energy-efficient time slots while satisfying deadline constraints.

2) Server Utilization Evaluation: Regarding server utilization, we define a utilization metric to evaluate server utilization under the obtained scheduling solution. For each edge server s_j, let B_j denote the total time period during which the server is executing tasks, which can be calculated by summing the execution intervals of all tasks assigned to it. The time horizon H is defined as the latest execution completion time among all tasks. The utilization rate of server s_j is then given by:

$$U_j = \frac{B_j}{H}.$$

Accordingly, the overall system utilization is defined as

$$U_{\text{avg}} = \frac{1}{M}\sum_{j=1}^{M} U_j,$$

where M represents the number of edge servers. In experiments, for each task count N, DORA is executed on all testing instances with multiple independent runs per instance. The scheduling decision for every single run is used to compute U_{avg}. The U_{avg} values are then aggregated across all instances and runs to obtain the average system utilization.

Fig. 6 reports the utilization evaluation results for different problem scales. As the number of tasks N increases from 50 to 300, system utilization rises steadily, indicating that a heavier workload leads to a higher fraction of time during which the servers are actively executing workloads. When N further increases to 600 and 1000, system utilization continues to rise but at a slower rate and remains within a moderate range. This behavior can be attributed to the fact that the growing workload also extends the overall execution horizon, resulting in a relatively balanced increase in both busy time and total duration and, in turn, a gradual rise in utilization rate. Overall, the results in Fig. 6 demonstrate that the proposed scheduling approach yields reasonable and stable server utilization across a wide range of workload patterns.

3) Comparison With Heuristics: To further evaluate the advantages of the proposed evolutionary framework, four representative heuristics are implemented as baselines, including a) an earliest-deadline-first (EDF) rule that sorts and schedules tasks in the ascending order of deadline values; b) a smallest-data-first (SDF) rule that prioritizes tasks with smaller uplink–downlink data volumes; c) a random-ordering (Random) method that schedules tasks in a uniformly shuffled order; and d) a greedy-early (GE) method that assigns each task to the earliest available time. These heuristics represent common non-evolutionary scheduling principles, ranging from deadline-driven and workload-driven rules to random and greedy strategies.

Table V reports the scheduling results, in terms of average energy consumption and feasibility rate, obtained by DORA and the heuristics across different task scales. We observe that DORA consistently achieves the lowest energy while maintaining feasibility across all problem scales. EDF and SDF lead to energy values close to DORA at $N = 50$. However, their energy consumption increases substantially at larger scales. They both become infeasible when the task count exceeds 300. When N reaches 1000, EDF is only feasible on 2 out of 5 testing instances. The random heuristic exhibits large performance fluctuations and, in turn, low feasibility. GE remains feasible but incurs significantly higher energy consumption, especially on testing instances of large scale. The comparison results justify that conventional heuristics are inadequate to effectively handle the complex constraints and enlarged solution space induced by joint offloading, execution, and downloading decisions, and further demonstrate the necessity and advantages of DORA's evolutionary search and dual-population collaboration mechanism.

4) Comparative Experiments on MAP: As an essential component of DORA, the mapping operator MAP is compared with the other four mapping methods in the existing works. The

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

TABLE V
COMPARISON BETWEEN DORA AND REPRESENTATIVE HEURISTICS ACROSS DIFFERENT TASK SCALES. (N DENOTES THE TASK SCALE, ENERGY IS MEASURED IN JOULES (J), AND "FEAS." INDICATES THE NUMBER OF FEASIBLE INSTANCES, WITH FIVE INSTANCES IN TOTAL).

Algorithm	$N = 50$		$N = 100$		$N = 200$		$N = 300$		$N = 600$		$N = 1000$	
	Energy	Feas.	Energy	Feas.	Energy	Feas.	Energy	Feas.	Energy	Feas.	Energy	Feas.
DORA	18.940	5/5	37.140	5/5	75.696	5/5	120.460	5/5	229.912	5/5	400.200	5/5
EDF	19.020	5/5	39.260	5/5	95.920	5/5	144.280	4/5	239.220	4/5	472.932	2/5
SDF	19.020	5/5	44.260	5/5	87.180	5/5	146.600	4/5	235.400	5/5	453.075	4/5
Random	23.630	5/5	47.340	5/5	96.400	4/5	156.850	2/5	233.833	3/5	522.140	4/5
GE	32.100	5/5	60.200	5/5	110.280	5/5	175.820	5/5	347.880	5/5	570.100	5/5

TABLE VI
SUMMARY OF MAPPING METHODS PROPOSED IN THE RELATED WORK

Work	Mapping Scheme	SIA
Ref. [11]	position-based mapping (position)	PSO
Ref. [45]	ranked-order value (ROV)	FA
Ref. [48]	distance-based mapping (distance)	WOA
Ref. [49]	sigmoid-position-based mapping (sig-position)	FA

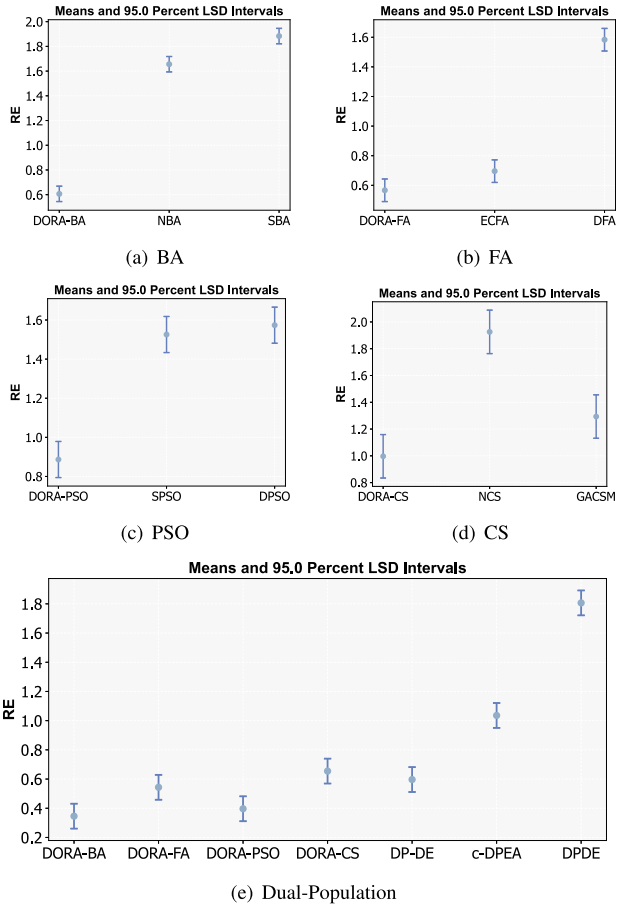
[1] Whale Optimization Algorithm (WOA)

details of the competing methods are provided in Table VI. Note that, each of these competing methods is developed particularly for one SIA. Aiming to evaluate these mapping methods in a comprehensive sense, we incorporate them into several different SIAs.

Fig. 5 provides the mean values and LSD intervals of REs by employing MAP and the other four competing mapping methods on BA, FA, PSO, and CS, respectively. It can be seen that the effectiveness of the mapping method varies across different SIAs, with this variation being particularly apparent for the distance method. To be specific, the distance method performs well in BA and PSO but significantly worse in CS, whereas the sig-position method performs well in FA but relatively worse in PSO. Compared to the four competing methods, the proposed MAP achieves stable and strong performance across the four SIAs, demonstrating its effectiveness in high-quality solution exploration.

5) Effectiveness Evaluation of DORA: To evaluate the effectiveness of DORA, we compare it with some existing SI-based approaches as follows. The BA-based approaches: novel bat algorithm (NBA) [50] and simulated annealing-based bat algorithm (SBA) [51]. The FA-based approaches: discrete firefly algorithm (DFA) [45] and efficient convergent firefly algorithm (ECFA) [49]. The PSO-based approaches: slow-movement particle swarm optimization (SPSO) [11] and distributed particle swarm optimization (DPSO) [52]. The CS-based approaches: novel cuckoo search algorithm (NCS) [53] and Gaussian random walk and adaptive discovery probability-based cuckoo search algorithm (GACSM) [54]. The dual-population-based approaches: dual-population differential evolution (DP-DE) [27], cooperative dual-population-based evolutionary algorithm (c-DPEA) [34], and dual-population differential evolution (DPDE) [39].

Fig. 7 presents mean values and LSD intervals of REs obtained by the DORA framework and the competing approaches in



(a) BA (b) FA (c) PSO (d) CS (e) Dual-Population

Fig. 7. Mean values and LSD intervals of REs obtained by the DORA framework and competing approaches.

the FA-based, PSO-based, CS-based, and dual-population-based categories. Fig. 7(a) and (c) show that the non-overlapping LSD intervals indicate that the performance of DORA is significantly better than that of competing approaches. Despite the overlapping LSD intervals in Fig. 7(b) and (d), DORA-FA and DORA-CS still achieve the best mean REs in their respective categories. Moreover, the evaluation results in Fig. 7(e) indicate that DORA performs much better than DPEA and c-DPEA, which are also based on the dual-population search scheme. This set of experiments justifies that DORA achieves promising solution exploration performance as a general evolutionary computational framework. The reasons for achieving the above performance advantages are twofold: the mapping operator of

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

DORA facilitates the solution exploration capability, and the cooperative mechanism effectively conveys decision information in high-quality solutions and achieves excellent inter-population division of labor and collaboration.

VI. Conclusion and Discussions

This paper investigates a mobility-aware two-stage task scheduling problem in an MEC system, referred to as MTTS, considering user mobility and both the task offloading and result downloading stages. To solve this task scheduling problem, we propose a dual-population evolutionary computation framework DORA, capable of deploying a wide range of SIAs for solution exploration. The proposed DORA framework features two parallel-evolving populations, responsible for solving the offloading and downloading sub-problems in MTTS, respectively. The evolutionary process of each population involves a mapping operator to link the individual space to the solution space, a fitness evaluation operator to assess the quality of each individual, and an update operator to modify individual attributes based on the search strategy of a specific SIA. In particular, we conduct rigorous theoretical analysis regarding the critical parameter setting in the mapping operator, further validated by experimental results. Inter-population collaboration is realized through an asynchronous solution transfer mechanism, where the current best solution is transferred from the stage-1 population to the stage-2 population, guiding the search toward high-quality final scheduling solutions. Experimental results using a real-world MD trajectory dataset demonstrate the superior performance of DORA over state-of-the-art SIA-based methods.

Future research may be directed toward the following aspects. On the one hand, the mobility model in the proposed DORA assumes that mobile trajectories are known or can be accurately predicted prior to scheduling. To enhance the adaptability of DORA to highly dynamic, unpredictable mobility in real-world deployment, one practical approach is to incorporate a rolling-window, re-planning mechanism, in which scheduling decisions are periodically re-optimized over short horizons using the most recent system observations and trajectory updates. On the other hand, to extend the present framework to large-scale MEC environments involving many devices, additional system-level factors become increasingly important considerations, including wireless channel contention among devices, queuing effects at shared edge resources, and cross-device fairness constraints. Incorporating these factors into DORA would require a redesign of both the system model and the scheduling algorithm. These directions indicate promising extensions for advancing the proposed framework beyond its current assumptions and toward more complex and dynamic MEC systems.

References

[1] K. Li, "Scheduling precedence constrained tasks for mobile applications in fog computing," *IEEE Trans. Services Comput.*, vol. 16, no. 3, pp. 2153–2164, May/Jun. 2023.

[2] H. Wu et al., "Container scheduling strategy based on image layer reuse and sequential arrangement in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 24, no. 9, pp. 8700–8713, Sep. 2025.

[3] Y. Wang, J. Zhu, H. Huang, and F. Xiao, "Bi-objective ant colony optimization for trajectory planning and task offloading in UAV-assisted MEC systems," *IEEE Trans. Mobile Comput.*, vol. 23, no. 12, pp. 12360–12377, Dec. 2024.

[4] J. Zhou, X. Hou, Y. Zeng, P. Cong, W. Jiang, and S. Guo, "Quality of experience and reliability-aware task offloading and scheduling for multi-user mobile-edge computing systems," *IEEE Trans. Services Comput.*, vol. 18, no. 3, pp. 1683–1696, May/Jun. 2025.

[5] W. Shen, W. Lin, W. Wu, H. Wu, and K. Li, "Reinforcement learning-based task scheduling for heterogeneous computing in end-edge-cloud environment," *Cluster Comput.*, vol. 28, no. 3, 2025, Art. no. 179.

[6] Y. Shi, S. Chen, and X. Xu, "MAGA: A mobility-aware computation offloading decision for distributed mobile cloud computing," *IEEE Internet Things J.*, vol. 5, no. 1, pp. 164–174, Feb. 2018.

[7] U. Saleem, Y. Liu, S. Jangsher, Y. Li, and T. Jiang, "Mobility-aware joint task scheduling and resource allocation for cooperative mobile edge computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 360–374, Jan. 2021.

[8] F. Tang, C. Liu, K. Li, Z. Tang, and K. Li, "Task migration optimization for guaranteeing delay deadline with mobility consideration in mobile edge computing," *J. Syst. Archit.*, vol. 112, 2021, Art. no. 101849.

[9] M. Zhao, R. Zhang, Z. He, and K. Li, "Joint optimization of trajectory, offloading, caching, and migration for UAV-assisted MEC," *IEEE Trans. Mobile Comput.*, vol. 24, no. 3, pp. 1981–1998, Mar. 2025.

[10] Z. Luo, J. Zhang, J. Wei, L. Zhou, K. Cao, and H. Zhao, "Trajectory design and task scheduling for multi-UAV aided mobile edge computing networks," in *Proc. 2025 IEEE Wireless Commun. Netw. Conf.*, 2025, pp. 1–6.

[11] Y. Zhang, Y. Liu, J. Zhou, J. Sun, and K. Li, "Slow-movement particle swarm optimization algorithms for scheduling security-critical tasks in resource-limited mobile edge computing," *Future Gener. Comput. Syst.*, vol. 112, pp. 148–161, 2020.

[12] G. Zhao, H. Xu, Y. Zhao, C. Qiao, and L. Huang, "Offloading tasks with dependency and service caching in mobile edge computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 11, pp. 2777–2792, Nov. 2021.

[13] S. Li, W. Sun, Y. Sun, and Y. Huo, "Energy-efficient task offloading using dynamic voltage scaling in mobile edge computing," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 1, pp. 588–598, First Quarter, 2021.

[14] W. Zhan, C. Luo, G. Min, C. Wang, Q. Zhu, and H. Duan, "Mobility-aware multi-user offloading optimization for mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 69, no. 3, pp. 3341–3356, Mar. 2020.

[15] Y. Ding, C. Liu, K. Li, Z. Tang, and K. Li, "Task offloading and service migration strategies for user equipments with mobility consideration in mobile edge computing," in *Proc. 2019 IEEE Int. Conf. Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw.*, 2019, pp. 176–183.

[16] C. Wang, X. Yu, L. Xu, and W. Wang, "Energy-efficient task scheduling based on traffic mapping in heterogeneous mobile-edge computing: A green IoT perspective," *IEEE Trans. Green Commun. Netw.*, vol. 7, no. 2, pp. 972–982, Jun. 2023.

[17] K. Li, "Scheduling independent tasks on multiple cloud-assisted edge servers with energy constraint," *J. Parallel Distrib. Comput.*, vol. 184, 2024, Art. no. 104781.

[18] Z. Lian, J. Shu, Y. Zhang, and J. Sun, "Convergent grey wolf optimizer metaheuristics for scheduling crowdsourcing applications in mobile edge computing," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 1866–1879, Jan. 2024.

[19] B. Huang et al., "Security modeling and efficient computation offloading for service workflow in mobile edge computing," *Future Gener. Comput. Syst.*, vol. 97, pp. 755–774, 2019.

[20] B. Lin et al., "A time-driven data placement strategy for a scientific workflow combining edge computing and cloud computing," *IEEE Trans. Ind. Informat.*, vol. 15, no. 7, pp. 4254–4265, Jul. 2019.

[21] B. Lin, Y. Huang, J. Zhang, J. Hu, X. Chen, and J. Li, "Cost-driven off-loading for DNN-based applications over cloud, edge, and end devices," *IEEE Trans. Ind. Informat.*, vol. 16, no. 8, pp. 5456–5466, Aug. 2020.

[22] W. Hua, P. Liu, and L. Huang, "Energy-efficient resource allocation for heterogeneous edge–cloud computing," *IEEE Internet Things J.*, vol. 11, no. 2, pp. 2808–2818, Jan. 2024.

[23] T. Zhu, T. Shi, J. Li, Z. Cai, and X. Zhou, "Task scheduling in deadline-aware mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4854–4866, Jun. 2019.

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

[24] Z. Wang, Z. Zhao, G. Min, X. Huang, Q. Ni, and R. Wang, "User mobility aware task assignment for mobile edge computing," *Future Gener. Comput. Syst.*, vol. 85, pp. 1–8, 2018.
[25] L. Niu et al., "Multiagent meta-reinforcement learning for optimized task scheduling in heterogeneous edge computing systems," *IEEE Internet Things J.*, vol. 10, no. 12, pp. 10519–10531, Jun. 2023.
[26] J. Wang, G. Liang, and J. Zhang, "Cooperative differential evolution framework for constrained multiobjective optimization," *IEEE Trans. Cybern.*, vol. 49, no. 6, pp. 2060–2072, Jun. 2019.
[27] J.-H. Zhong et al., "A differential evolution algorithm with dual populations for solving periodic railway timetable scheduling problem," *IEEE Trans. Evol. Comput.*, vol. 17, no. 4, pp. 512–527, Aug. 2013.
[28] Y. Fu, M. Zhou, X. Guo, and L. Qi, "Scheduling dual-objective stochastic hybrid flow shop with deteriorating jobs via bi-population evolutionary algorithm," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 12, pp. 5037–5048, Dec. 2020.
[29] T. Park and K. R. Ryu, "A dual-population genetic algorithm for adaptive diversity control," *IEEE Trans. Evol. Comput.*, vol. 14, no. 6, pp. 865–884, Dec. 2010.
[30] X. Chen, "Novel dual-population adaptive differential evolution algorithm for large-scale multi-fuel economic dispatch with valve-point effects," *Energy*, vol. 203, 2020, Art. no. 117874.
[31] J. Zou, R. Sun, S. Yang, and J. Zheng, "A dual-population algorithm based on alternative evolution and degeneration for solving constrained multi-objective optimization problems," *Inf. Sci.*, vol. 579, pp. 89–102, 2021.
[32] A. Turky, S. Abdullah, and A. Dawod, "A dual-population multi operators harmony search algorithm for dynamic optimization problems," *Comput. Ind. Eng.*, vol. 117, pp. 19–28, 2018.
[33] M. Ming, R. Wang, H. Ishibuchi, and T. Zhang, "A novel dual-stage dual-population evolutionary algorithm for constrained multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 26, no. 5, pp. 1129–1143, Oct. 2022.
[34] M. Ming, A. Trivedi, R. Wang, D. Srinivasan, and T. Zhang, "A dual-population-based evolutionary algorithm for constrained multiobjective optimization," *IEEE Trans. Evol. Comput.*, vol. 25, no. 4, pp. 739–753, Aug. 2021.
[35] M. S. S. Raju, S. Dutta, R. Mallipeddi, and K. N. Das, "A dual-population and multi-stage based constrained multi-objective evolutionary," *Inf. Sci.*, vol. 615, pp. 557–577, 2022.
[36] Q. Bao, M. Wang, G. Dai, X. Chen, Z. Song, and S. Li, "A dual-population based bidirectional coevolution algorithm for constrained multi-objective optimization problems," *Expert Syst. Appl.*, vol. 215, 2023, Art. no. 119258.
[37] C. Wang, Z. Wang, Y. Tian, X. Zhang, and J. Xiao, "A dual-population based evolutionary algorithm for multi-objective location problem under uncertainty of facilities," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 7692–7707, Jul. 2022.
[38] J. Zhou, L. Gao, X. Yao, C. Zhang, F. T. Chan, and Y. Lin, "An adaptive dual-population evolutionary paradigm with adversarial search: Case study on many-objective service consolidation," *Appl. Soft Comput.*, vol. 90, 2020, Art. no. 106160.
[39] W.-F. Gao, G. G. Yen, and S.-Y. Liu, "A dual-population differential evolution with coevolution for constrained optimization," *IEEE Trans. Cybern.*, vol. 45, no. 5, pp. 1108–1121, May 2015.
[40] V. T. Vu, L. T. Bui, and T. T. Nguyen, "A modified dual-population approach for solving multi-objective problems," in *Proc. 21st Asia Pacific Symp. Intell. Evol. Syst.*, 2017, pp. 89–94.
[41] Y.-H. Zhang, Y.-J. Gong, J. Zhang, and Y.-B. Ling, "A hybrid evolutionary algorithm with dual populations for many-objective optimization," in *Proc. 2016 IEEE Congr. Evol. Comput.*, 2016, pp. 1610–1617.
[42] D. Kimovski, N. Mehran, C. E. Kerth, and R. Prodan, "Mobility-aware IoT applications placement in the cloud edge continuum," *IEEE Trans. Services Comput.*, vol. 15, no. 6, pp. 3358–3371, Nov./Dec. 2022.
[43] Q. Chen, Z. Kuang, and L. Zhao, "Multi-user computation offloading and resource allocation for cloud-edge heterogeneous network," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3799–3811, Mar. 2022.
[44] R. Tian, Y. Zhang, and Y. Cao, "A mutation particle swarm optimization method for task scheduling in seismic edge networks," *IEEE Internet Things J.*, vol. 12, no. 11, pp. 15667–15680, Jun. 2025.
[45] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 301–305, Apr. 2014.
[46] G. Rudolph, "Convergence analysis of canonical genetic algorithms," *IEEE Trans. Neural Netw.*, vol. 5, no. 1, pp. 96–101, Jan. 1994.
[47] Z. Kuang, L. Li, J. Gao, L. Zhao, and A. Liu, "Partial offloading scheduling and power allocation for mobile edge computing systems," *IEEE Internet Things J.*, vol. 6, no. 4, pp. 6774–6785, Aug. 2019.
[48] J. Zhao, L. Deng, Y. Liu, and J. Sun, "Energy-efficient partial offloading in mobile edge computing under a deadline constraint," in *Proc. Int. Conf. Intell. Technol. Embedded Syst.*, 2021, pp. 14–21.
[49] L. Yin, J. Sun, J. Zhou, Z. Gu, and K. Li, "ECFA: An efficient convergent firefly algorithm for solving task scheduling problems in cloud-edge computing," *IEEE Trans. Services Comput.*, vol. 16, no. 5, pp. 3280–3293, Sep./Oct. 2023.
[50] Q. Yan, L. Ma, and J. Sun, "Novel bat algorithms for scheduling independent tasks in collaborative Internet-of-Things," in *Proc. IEEE 22nd Int. Conf. High Perform. Comput. Commun.; IEEE 18th Int. Conf. Smart City; IEEE 6th Int. Conf. Data Sci. Syst.*, 2020, pp. 674–681.
[51] H. Yuan, J. Bi, and M. Zhou, "Spatial task scheduling for cost minimization in distributed green cloud data centers," *IEEE Trans. Automat. Sci. Eng.*, vol. 16, no. 2, pp. 729–740, Apr. 2019.
[52] S. Meng et al., "Security-aware dynamic scheduling for real-time optimization in cloud-based industrial applications," *IEEE Trans. Ind. Informat.*, vol. 17, no. 6, pp. 4219–4228, Jun. 2021.
[53] S. Tian, X. Li, J. Wan, and Y. Zhang, "A novel cuckoo search algorithm for solving permutation flowshop scheduling problems," in *Proc. Int. Conf. Recent Adv. Syst. Sci. Eng.*, 2021, pp. 1–8.
[54] Z. Deng, Z. Yan, H. Huang, and H. Shen, "Energy-aware task scheduling on heterogeneous computing systems with time constraint," *IEEE Access*, vol. 8, pp. 23936–23950, 2020.

Lu Yin received the BS degree in network engineering and PhD degree in computer science and technology from the Nanjing University of Science and Technology, Nanjing, China, in 2018 and 2024, respectively. She is now with the School of Computer Engineering, Nanjing Institute of Technology, Nanjing, Jiangsu, China. Her research interests include cloud computing, mobile edge computing, and task scheduling.

Jin Sun (Member, IEEE) received the BS and MS degrees in computer science from the Nanjing University of Science and Technology, Nanjing, China, in 2004 and 2006, respectively, and the PhD degree in electrical and computer engineering from the University of Arizona, in 2011. From January 2012 to September 2014, he was with Orora Design Technologies, Inc. as a member of technical staff. He is currently an associate professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology. His research interests include high-performance computing, integrated circuit modeling and analysis, and computer-aided design.

Junlong Zhou (Member, IEEE) received the PhD degree in computer science from East China Normal University, Shanghai, China, in 2017. He was a visiting scholar with the University of Notre Dame, Notre Dame, Indiana, during 2014–2015. He is currently an assistant professor with the School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China. His research interests include embedded systems, cloud-edge-IoT, and cyber physical systems, where he has published more than 60 refereed papers, including more than 20 IEEE/ACM Transactions. He has been program vice-chairs, section chairs, publicity chairs, publication chairs, and TPC members for numerous conferences. He serves as an associate editor for the *Journal of Circuits, Systems, and Computers* and the *IET Cyber-Physical Systems: Theory & Applications*, a subject area editor for the *Journal of Systems Architecture: Embedded Software Design*, and a guest editor for 5 ACM/IET/Elsevier/Wiley Journals such as *ACM Transactions on Cyber-Physical Systems*.

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.

Zhihui Wei (Member, IEEE) received the BSc and MSc degrees in applied mathematics and the PhD degree in communication and information system from South East University, Nanjing, China, in 1983, 1986, and 2003, respectively. He is currently a professor and a doctoral supervisor with the Nanjing University of Science and Technology (NJUST), Nanjing. His research interests include partial differential equations, mathematical image processing, multiscale analysis, sparse representation, and compressive sensing.

Keqin Li (Fellow, IEEE) received the BS degree in computer science from Tsinghua University, in 1985 and the PhD degree in computer science from the University of Houston, in 1990. He is a SUNY distinguished professor with the State University of New York and a National distinguished professor with Hunan University (China). He has authored or co-authored more than 1270 journal articles, book chapters, and refereed conference papers. He holds nearly 80 patents announced or authorized by the Chinese National Intellectual Property Administration. He is among the world's top few most influential scientists in parallel and distributed computing, regarding single-year impact (ranked #2) and career-long impact (ranked #3) based on a composite indicator of the Scopus citation database. He is listed in Scilit Top Cited Scholars (2023–2025) and is among the top 0.02% out of more than 20 million scholars worldwide based on top-cited publications in the last ten years. He is listed in ScholarGPS Highly Ranked Scholars (2022–2025) and is among the top 0.002% out of more than 30 million scholars worldwide based on a composite score of three ranking metrics for research productivity, impact, and quality in the recent five years. He received the IEEE TCCLD Research Impact Award from the IEEE CS Technical Committee on Cloud Computing, in 2022 and the IEEE TCSVC Research Innovation Award from the IEEE CS Technical Community on Services Computing, in 2023. He won the IEEE Region 1 Technological Innovation Award (Academic), in 2023. He was a recipient of the 2022–2023 International Science and Technology Cooperation Award and the 2023 Xiaoxiang Friendship Award of Hunan Province, China. He is a member of the SUNY Distinguished Academy. He is an AAAS fellow, an AAIS fellow, an ACIS fellow, an AIIA fellow, and a WAET fellow. He is a member of the National Academy of Artificial Intelligence, a member of the European Academy of Sciences and Arts, and a member of Academia Europaea (Academician of the Academy of Europe).

Authorized licensed use limited to: NANJING UNIVERSITY OF SCIENCE AND TECHNOLOGY. Downloaded on August 14,2026 at 05:20:39 UTC from IEEE Xplore. Restrictions apply.