Energy-Efficient and Latency-Aware Task Offloading for Industrial Cloud-Edge Systems With Heterogeneous CPUs and GPUs

Jiahui Zhai, *Student Member, IEEE*, Jing Bi, *Senior Member, IEEE*, Haitao Yuan, *Senior Member, IEEE*, Jia Zhang, *Senior Member, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

***Abstract*—The unprecedented prosperity of the Industrial Internet of Things (IIoT) has significantly driven the transition from traditional manufacturing to intelligent one. In industrial environments, resource-constrained industrial equipments (IEs) often fail to meet the diverse demands of numerous compute-intensive and latency-sensitive tasks. Mobile edge computing has emerged as an innovative paradigm to reduce latency and energy consumption for IEs. However, the increasing number of IEs in industrial settings relies on heterogeneous platforms integrated with different processing units, i.e., CPUs and GPUs. To address this challenge, we propose a software-defined networking-based equipment-edge-cloud architecture with three-stage heterogeneous computing. This architecture accurately models the multitask processing of both scientific and concurrent workflows in real industrial environments. We formulate a joint optimization problem to simultaneously minimize task completion time and energy consumption for IEs. To solve this problem, we design an improved two-stage multiobjective evolutionary algorithm (IT-MOEA). IT-MOEA employs a novel multiobjective grey wolf optimizer based on manta ray foraging and associative learning to accelerate convergence in the early evolution stages and adopts a diversity-enhancing immune algorithm to enhance diversity in the later stages. Simulation results with various benchmarks demonstrate that IT-MOEA outperforms several state-of-the-art single-objective optimization algorithms by an average of 24.7% and multiobjective algorithms by 41.0% in terms of delay and energy consumption.**

***Index Terms*—Evolutionary algorithms, Industrial Internet of Things (IIoT), mobile-edge computing (MEC), multiobjective optimization, task offloading.**

Received 7 January 2025; revised 22 March 2025; accepted 7 April 2025. Date of publication 10 April 2025; date of current version 27 June 2025. This work was supported in part by the National Natural Science Foundation of China under Grant 62173013 and Grant 62473014; in part by the Beijing Natural Science Foundation under Grant L233005 and Grant 4232049; in part by the Beihang World Top University Cooperation Program; and in part by the 2023 International Cooperation Training Program for Innovative Talents ("Double First-class" Construction Special Program-"Artificial Intelligence + Internet of Things") of the China Scholarship Council (CSC). *(Corresponding author: Haitao Yuan.)*

Jiahui Zhai and Jing Bi are with the College of Computer Science, Beijing University of Technology, Beijing 100124, China (e-mail: zhaijiahui@emails.bjut.edu.cn; bijing@bjut.edu.cn).

Haitao Yuan is with the School of Automation Science and Electrical Engineering, Beihang University, Beijing 100191, China (e-mail: yuan@buaa.edu.cn).

Jia Zhang is with the Department of Computer Science, Southern Methodist University, Dallas, TX 75206 USA (e-mail: jiazhang@smu.edu).

Rajkumar Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Lab, School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia (e-mail: rbuyya@unimelb.edu.au).

This article has supplementary downloadable material available at https://doi.org/10.1109/JIOT.2025.3559690, provided by the authors.

Digital Object Identifier 10.1109/JIOT.2025.3559690

I. Introduction

With the continuous advancement of wireless communication technologies and the Internet of Things (IoT), Industrial IoT (IIoT) technologies have emerged as a torchbearer in driving the industrial revolution and enhancing traditional industrial efficiency [1]. In IIoTs, various industrial equipments (IEs), e.g., industrial robots, vehicles, and sensors are required to play crucial roles across different production stages, aiming to enhance production efficiency while reducing resource consumption [2]. However, compute-intensive and latency-sensitive industrial applications cannot autonomously be operated in IEs constrained by their limited computational capabilities and battery. To address this challenge, offloading tasks from industrial applications to resource-sufficient computing nodes to reduce computation time and energy consumption [3]. Given the massive data transmission and computation demands of IEs in industrial environments, traditional approaches involve processing data in cloud data centers (CDCs) [4]. However, due to the long geographical distances between CDCs and IEs, this solution fails to meet the latency requirements of rapidly expanding IE scales and industrial applications [4]. Mobile edge computing (MEC) deploys cloud resources in access points (APs) closer to IEs, enabling low-latency and energy-efficient IE tasks [5]. Additionally, delay and energy consumption in MEC serve as primary metrics for evaluating the effectiveness of offloading decisions [6]. Given the complexity and diversity of task types in industrial applications, it is essential to consider the diverse requirements of multiple tasks during the offloading process. Therefore, it is crucial to design a task offloading approach that balances delay and energy consumption for complex and diverse tasks in industrial environments.

Moreover, the complexity of tasks in the IIoTs imposes higher demands on heterogeneous computing resources. Traditional general-purpose CPU computing methods are insufficient to effectively address the requirements of parallel computing. General-purpose GPUs (GPGPUs) have become increasingly prevalent in cloud processing and high-performance embedded systems, accelerating parallel computing applications [7]. Many tasks require coordinating multiple computing resources, such as CPU+GPU and CPU+ASIC. For instance, when processing scene generation tasks of virtual reality (VR), large scenes are decomposed into multiple miniature scenes by CPUs. These smaller scenes are then subjected to high-precision parameter calculations and real-time graphic rendering with both CPUs and GPUs, generating multiple smaller scenes. CPUs further process intermediate data fragments to achieve visual effects of the final VR scene. High-performance GPUs, exemplified by products from NVIDIA and AMD, have proven capable of meeting the demands of extensive parallel computing tasks, leading to the increasing adoption of heterogeneous resource integration frameworks across various computing devices [8]. Existing task offloading studies have predominantly focused on either CPU or GPU resources, often neglecting the coordinated utilization of heterogeneous computing platforms [3], [4], [5]. This oversight limits the efficiency of task execution, particularly for complex and data-intensive applications that require high levels of parallelism and computational diversity. To address this gap, this work designs a task offloading mechanism that intelligently distributes computing tasks across heterogeneous platforms with CPUs and GPUs while considering delay and energy constraints of IIoT applications. In addition, IIoT tasks exhibit significant heterogeneity in terms of computational requirements, energy consumption, and latency constraints. Current studies often assume uniform task characteristics, failing to adequately address the diverse and dynamic nature of real-world industrial applications [6], [7], [8]. Consequently, this work establishes a practical heterogeneous computing process model for systematically analyzing delay and energy consumption of heterogeneous computing tasks. Within the broader context of cloud-edge computing, we leverage software-defined networking (SDN) to provide a centralized and programmable network control plane that flexibly coordinates offloading decisions among IEs, APs, and CDCs. Thus, an SDN-enabled equipment-edge-cloud framework holds great promise for efficient task offloading and resource allocation in heterogeneous systems containing both CPUs and GPUs.

Considering the heterogeneous task offloading with CPUs and GPUs in IIoT, we aim to minimize the completion time for industrial applications and the energy consumption of IEs. This multiobjective task offloading problem in industrial environments is a typical mixed-integer nonlinear programming (MINLP) [9], which complicates the development of efficient and scalable algorithms. This work proposes an improved multiobjective evolutionary algorithm (MOEA) that integrates various advanced optimization techniques to address this challenge. Primary contributions can be summarized as follows.

1) We propose an SDN-enabled equipment-edge-cloud architecture for multitask offloading in a hybrid heterogeneous system, encompassing multiple IEs, APs, and a CDC. It considers the three-stage heterogeneous computing processes of tasks with CPUs and GPUs. On this basis, we propose a large-scale constrained bi-objective optimization problem that simultaneously minimizes task completion time and energy consumption for IEs.
2) To address the bi-objective optimization problem, we design an Improved Two-stage MOEA (IT-MOEA) to solve the MINLP problem. It jointly optimizes task allocation across IEs, APs, and CDC, the association between IEs and APs, the load balancing of edge servers (ESs), the transmission power of IEs, and the heterogeneous computing capabilities with CPUs and GPUs.
3) IT-MOEA adopts a Multi-objective Grey wolf optimizer based on Manta ray foraging and Associative learning (MGMA) in the first evolutionary stage, aiming to accelerate the convergence of population, and applies a Diversity-enhanced Immune Algorithm (DIA) in the second evolutionary stage to improve the distribution of the final population.

In addition, this work evaluates the performance of IT-MOEA with three types of benchmark methods regarding single-objective, multiobjective, and offloading schemes. Extensive experiments demonstrate that IT-MOEA outperforms several state-of-the-art (SOTA) peers regarding convergence and distribution performance.

The remainder of this article is organized as follows. Section II provides an overview and comparison of relevant studies. Section III elucidates the architecture of a hybrid equipment-edge-cloud system and formulates the system optimization problem. Section IV outlines the details of IT-MOEA. Section V discusses the performance evaluation results. Section VI concludes this work. Section III of the supplementary file shows the limitations and future work.

II. Related Work

This section provides a comprehensive overview of task offloading and the tradeoffs between delay and energy consumption in industrial environments, focusing on approaches involving IIoT, MEC, and edge-cloud collaboration systems.

A. Task Offloading for Industrial Environments

Task offloading is a highly efficient method for enhancing network performance. Peng et al. [10] introduce an end-edge-cloud collaboration optimization method for IIoTs, addressing multiobjective issues in task offloading and resource allocation, enhancing energy efficiency, time consumption, resource utilization, and load balancing. However, it does not consider the optimization objectives for multiple tasks with complex constraints. Deng et al. [11] provide an autonomous partial offloading system for multiuser IIoT systems, utilizing reinforcement learning strategies to optimize delay performance. However, it does not consider the issue of load balancing among multiple ESs. The workflow application is a complex task paradigm involving stringent task limitations. Sun et al. [12] employ a Bayesian network-based evolutionary algorithm for optimizing task allocation in MEC-enabled IIoT architectures, improving response time by considering task priority constraints. However, its task priority is limited to the constraints of execution order, and it ignores the constraints of task offloading sequence. Lin et al. [13] design a self-adaptive discrete particle swarm optimization algorithm to optimize data transmission time in scientific workflows by combining edge and cloud computing, enhancing efficiency through improved data placement strategies. However, it does not consider concurrent workflow applications in edge-cloud systems. Given the constrained resources of ESs, employing edge-cloud collaboration for task offloading is a more optimal approach for IEs. Wu et al. [14] propose a blockchain-enabled IoT-edge-cloud architecture, optimizing energy consumption and task response time by dynamically selecting computing locations with the Lyapunov optimization. Nevertheless, it neglects the load balancing of MEC servers and the constraints on CPUs, GPUs, and memory. Xu et al. [15] present an edge-cloud collaboration method for the distributed COVID-19 detection model training on chest X-ray images, optimizing training efficiency, model accuracy, time cost, and energy consumption. However, it does not consider the transmission latency between APs and ESs. Chouikhi et al. [16] propose a multiagent deep reinforcement learning approach for optimizing computation offloading in IIoT systems, training models centrally in CDC while executing decentralized decisions at ESs to minimize long-term energy consumption. However, it neglects an SDN-enabled hierarchical architecture with load balancing, optimized resource allocation across APs via SDN control, and hybrid computational hardware (CPUs/GPUs) at both ESs and CDC.

Different from these methods, this work considers the computational offloading of multiple IEs, computation-intensive, and latency-sensitive tasks in a large-scale hybrid system that includes equipment-edge-cloud infrastructure. Specifically, this work considers the heterogeneous computing resources of various devices in the equipment-edge-cloud framework with CPUs and GPUs. Besides, we devise a three-stage heterogeneous computing task model to accurately describe the latency and energy consumption processes of scientific and concurrent workflow tasks to minimize task completion time and energy consumption for IEs.

B. *Tradeoff of Delay and Energy Consumption*

Many recent studies have generated a single-objective function by weighting two related objective functions to strike a tradeoff between delay and energy consumption. Peng et al. [17] propose an online resource coordinating and allocating scheme, optimizing multiuser cooperative partial offloading for device-to-device underlay MEC. Unlike it, we focus on minimizing the total task completion time and energy consumption of IEs. Ding et al. [18] introduce and evaluate two architectures for end-edge-cloud computing, optimizing offloading strategies with potential game-based algorithms, enhancing resource utilization and quality of experience based on real-world data experiments. Unlike this, we optimize the quality of service for the hybrid system while ensuring resource utilization and allocation efficiency. Wang et al. [19] present a distributed deep learning-based task offloading and resource allocation algorithm for a software-defined framework of MEC and IoTs, optimizing task offloading and power allocation with deep learning. However, it does not consider the transmission latency and energy consumption between the SDN controller and APs. Shakarami et al. [20] develop an autonomous task offloading framework for MEC, addressing challenges in time and resource-intensive applications with a hybrid model. However, it fails to consider a real-world scenario where ESs cannot handle tasks due to excessive load. The studies mentioned above transform the multiobjective problem into a single-objective one by assigning weights to the two objectives. Existing studies also address the simultaneous optimization of dual objectives. Keshavarznejad et al. [21] present a multiobjective optimization approach for task offloading in fog computing to reduce system power consumption and task execution delay, demonstrating robustness in achieving a balance between offloading probability and power efficiency. Unlike this, we conduct detailed experimental setups to optimize the task completion time and energy consumption of IEs with different evaluation metrics. Peng et al. [22] introduce three constrained MOEAs for IoT-enabled task offloading, optimizing time and energy consumption. In contrast, our method thoroughly considers the features of large-scale cloud and edge systems, making it suitable for more specialized and sophisticated optimization of problems to handle tasks among IEs, APs, and CDC.

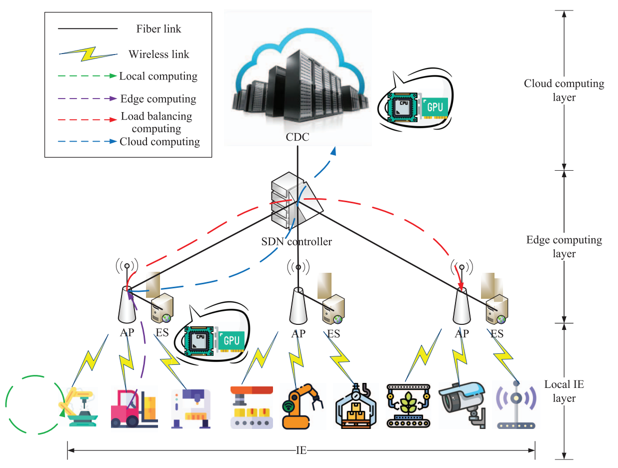


Fig. 1. Framework of an industrial environment.

Unlike existing studies, this work simultaneously minimizes the task completion time and energy consumption of IEs in large-scale equipment-edge-cloud systems. This architecture primarily encompasses the processes of task uploading, transmission, and computation. Our method jointly optimizes task allocation across the IEs, edge, and cloud, the association between IEs and APs, the transmission power of IEs, and the heterogeneous computing capabilities with CPUs and GPUs. Additionally, it considers constraints, such as task workflow and load balancing of ESs.

III. Problem Formulation

A. *System Model*

This work illustrates an SDN-enabled equipment-edge-cloud architecture for multiple task offloading in a hybrid heterogeneous system, including M IEs, J APs, an SDN controller, and a single CDC. Fig. 1 illustrates the proposed IIoT application architecture, which comprises three distinct layers: a local IE layer, an edge computing layer, and a cloud computing layer.

1) Local IE Layer: This layer incorporates multiple types of IEs, e.g., instruments, vehicles, machines, monitors, and robots. These IEs periodically collect environmental data and generate latency-sensitive and computation-intensive tasks. Each IE can connect to its local AP via a wireless network. Each AP covers M IEs, represented by $\mathcal{M} = \{1, 2, \ldots, M\}$. IEs perform initial sensing and local computing before offloading heavier tasks.

2) Edge Computing Layer: This layer consists of J APs and an SDN controller. As illustrated in Fig. 1, J APs are evenly distributed in the industrial environment, each covering distinct types of IEs without mutual interference and each paired with an ES. Each ES is equipped with CPUs and GPUs to handle complex computations and reduce latency at the network edge. A set of APs is denoted as $\mathcal{J} = \{1, 2, \ldots, J\}$. APs are interconnected via optical cables. Each AP is connected to an SDN controller *via* an optical fiber link. The SDN controller enables APs to connect to CDC through the core backbone network and implements load balancing strategies to optimize task distribution across APs. Furthermore, the SDN controller can centrally manage all APs and IEs, i.e., making computational offloading decisions.

3) Cloud Computing Layer: This layer contains a single CDC with extensive computational resources, supporting applications with higher computational demands. CDC integrates high-performance computing clusters with CPUs and GPUs. It is connected to the SDN controller via a redundant core switch, guaranteeing that computational traffic is not retransmitted across the Internet, enhancing transmission stability. The dynamic task offloading process is given in Fig. S1 in the supplementary file, which extends the three-layer architecture of Fig. 1 by explicitly modeling offloading decisions.

For clarity, Table S1 in the supplementary file lists main notations and decision variables in this section. IEs, ESs, and CDC are equipped with CPUs and GPUs. Each IE runs K delay-sensitive and computation-intensive tasks, represented by $\mathcal{K} = \{1, 2, \ldots, K\}$. K tasks need to be completed before a given deadline. The CPU parameter of IE m is represented by the tuple $\{\acute{\omega}^C, \acute{f}^C, \acute{p}^{C,d}, \acute{p}^{C,i}\}$. $\acute{\omega}^C$ is the number of CPUs, $\acute{f}^C$ is the computational capability of each CPU (in FLOPS), $\acute{p}^{C,d}$ is the dynamic power of the CPU (in W), and $\acute{p}^{C,i}$ is the static power of the CPU (in W). Similar to CPU, IE m's GPU parameters are represented as the tuple $\{\acute{\omega}^G, \acute{f}^G, \acute{p}^{G,d}, \acute{p}^{G,i}\}$. Moreover, $\{\dot{\omega}^C, \dot{f}^C, \dot{\omega}^G, \dot{f}^G\}$ and $\{\grave{\omega}^C, \grave{f}^C, \grave{\omega}^G, \grave{f}^G\}$ denote the computing resources of ESs and CDC, respectively. When a task is generated by an IE, a computation request is sent to the SDN controller, which makes the optimal execution decisions. Ultimately, the SDN controller schedules the pending tasks to the designated parts for computation.

This work examines a binary offloading strategy where tasks are either processed locally in IEs or entirely offloaded to ESs or CDC. Each task requires at least one CPU, and CPUs and GPUs are dedicated to a single task at any given time. The utilization of both CPUs and GPUs can be up to 100%. Let $\acute{\lambda}_m^k$ ($\acute{\lambda}_m^k \in \{0, 1\}$), $\dot{\lambda}_m^k$ ($\dot{\lambda}_m^k \in \{0, 1\}$), and $\grave{\lambda}_m^k$ ($\grave{\lambda}_m^k \in \{0, 1\}$) denote the offload factors of task k ($1 \leq k \leq K$) of IE m ($1 \leq m \leq M$) executed

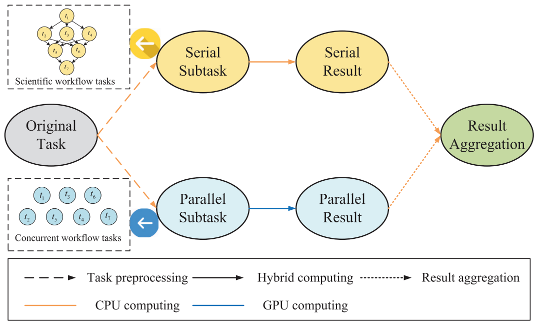


Fig. 2. Three-stage heterogeneous computing model of tasks.

in IE, ESs, and CDC, respectively. Thus

$$\acute{\lambda}_m^k + \dot{\lambda}_m^k + \grave{\lambda}_m^k = 1. \tag{1}$$

Each IE is exclusively linked to a solitary AP. Then, if tasks from IE m are linked to AP j, $x_{m,j} = 1$; otherwise, $x_{m,j} = 0$. Thus, for each IE m, we have

$$\sum_{j=1}^{J} x_{m,j} = 1. \tag{2}$$

B. Task Model

This work considers both serial and parallel tasks generated by IIoT applications, e.g., deep learning and reinforcement learning tasks. CPUs and GPUs can handle these tasks but with varying efficiencies. It is shown that CPUs are more efficient for serial tasks, while GPUs are superior for parallel tasks [8]. We focus on scenarios where CPUs handle serial tasks and GPUs manage parallel tasks. For instance, in a smart manufacturing environment, CPUs execute time-critical scheduling algorithms, coordinate sensor data acquisition, or manage logical decision-making (e.g., controlling operational states of robotic arms). Meanwhile, GPUs can concurrently perform large-scale parallel computations for tasks, such as real-time defect detection in production lines, where convolutional neural networks analyze high-resolution images at scale. Likewise, in a predictive maintenance scenario, CPUs execute sequential logic to decide when to query system health, while GPUs process large amounts of sensor data in parallel to detect early warning signs of potential failures. Therefore, we propose a three-stage task model, including task preprocessing, hybrid computing, and results aggregation shown in Fig. 2. According to typical GPGPU models, e.g., CUDA and OpenCL [23], the CPU preprocesses tasks and sends them to CPUs and GPUs for serial or parallel computation. Developers may split tasks into serial and parallel subtasks for optimal resource utilization, scheduling them to CPUs and GPUs, respectively. During the hybrid computing stage, these heterogeneous resources are used concurrently. CPU handles logic-intensive or sequential subtasks, while GPU processes large-scale parallel subtasks. It improves scalability by separating the original tasks from CPU-limited operations, meaning that additional GPU resources can be seamlessly added for more demanding parallel subtasks without changing CPU-centric subtasks. The final stage involves CPUs aggregating these results to produce the final output. Importantly, serial and parallel subtasks are independent and can be processed concurrently, and each stage is executed sequentially. This model helps balance the workload and reduces data-transfer costs. It also integrates both CPU and GPU, giving high speed on parallel parts and precise control over tasks that need a clear sequence.

Let $t_m^k = \{I_m^k, \varpi_m^k, \hat{T}_m^k, \hat{E}_m^k, q_m^k, o_m^k\}$ denote the task k generated by IE m. I_m^k denotes the data size of the t_m^k (in bits), ϖ_m^k denotes the floating-point number (in FLOPs) required to complete the t_m^k, $\hat{T}_m^k$ denotes the maximum allowable delay (in sec) of the t_m^k, $\hat{E}_m^k$ denotes the maximum energy consumption (in J) of the t_m^k, q_m^k denotes the ratio of the computing load during the hybrid computation stage compared to the overall t_m^k, and o_m^k denotes the ratio of parallel computational load during the hybrid computation stage compared to the overall hybrid computation stage in the t_m^k. Given that serial subtasks must be executed sequentially while parallel subtasks can be executed concurrently, we categorize tasks accordingly. Sequentially executed tasks are modeled as scientific workflow tasks, whereas tasks with concurrent execution are modeled as concurrent workflow tasks in Fig. 2. $\mathcal{C}$ denotes a set of priority constraints for scientific workflow tasks in $\mathcal{K}$. $\mathcal{U}_m^k$ and $\mathcal{V}_m^k$ denote the sets of predecessor and successor tasks of t_m^k, respectively. t_m^s and t_m^e denote the start and end tasks of t_m^k.

C. Communication Model

This work employs an orthogonal frequency-division multiple access communication model between various IEs and their associated APs. It is assumed that the spectrum between an IE and its associated AP is orthogonally allocated. Similarly, the spectrum between APs and the CDC is also orthogonal. Let $(d_{m,j})^{-\varsigma}$ denote the path loss between IE m and its corresponding AP j $(1 \le j \le J)$. $d_{m,j}$ represents the distance from IE m to AP j and ς represents the path loss exponent. According to [5], the uplink rate between IE m and AP j is denoted by $R_{m,j}$. Hence, we have

$$R_{m,j} = W_{m,j} \log_2\left(1+\frac{p_m^o \varrho (d_{m,j})^{-\varsigma} |h|^2 \zeta}{\sigma^2}\right) \tag{3}$$

where $W_{m,j}$ is the bandwidth of uplink channel allocated to IE m for AP j, p_m^o is the transmission power of IE m, h denotes the uplink channel fading coefficient, σ^2 represents the power parameter of Gaussian white noise, and ϱ and ζ are the path-loss coefficient and log-normal shadowing.

When task t_m^k is offloaded to an ES not directly linked to AP j, additional transmission delay is incurred from AP j to the target ES. W_v is the transmission rate between APs. The limited computational resources may create a bottleneck when many tasks are offloaded to the ES. To fully utilize the computational resources of other idle ESs and achieve load balancing, AP j can offload tasks to the idle ES directly connected to AP n $(n \neq j)$. Let W_s denote the data transmission rate between the SDN controller and each AP. Additionally, when the computational resources of all ESs are insufficient, IE needs to offload tasks to CDC to leverage the massive computational resources available at CDC. Let W_c denote the data transmission rate between the SDN controller and CDC.

The communication overhead for task offloading comprises three main components: 1) control signaling between SDN controller and APs for path selection, which is proportional to $1/W_s$; 2) inter-AP coordination overhead for redirecting tasks among ESs, which scales with J^2/W_v; and 3) meta-data exchange for resource discovery in CDC offloading, which is inversely proportional to W_c. In large-scale networks $(J \to \infty)$, the quadratic growth of inter-AP coordination overhead becomes significant, requiring dynamic topology-aware scheduling as discussed in [24]. The total overhead Ω can be expressed as

$$\Omega = \frac{\eta_1}{W_s} + \frac{\eta_2 J^2}{W_v} + \frac{\eta_3}{W_c} \tag{4}$$

where η_1, η_2, and η_3 denote protocol-specific coefficients that capture control message sizes. This formulation matches the multitier overhead analysis in [25], suggesting that edge-to-edge coordination dominates in dense deployments, supporting the contract-theoretic analysis in [26].

D. Delay and Energy Consumption Models

The delay and energy consumption models of three computing models, including local computing, edge computing, and cloud computing, are discussed below.

1) Local Computing: When $\acute{\lambda}_m^k = 1$, task t_m^k is computed at IE m. Fig. 2 shows that CPUs execute both the task preprocessing and result aggregation stages. Let $\acute{\tau}_m^{k,1}$ denote the sum of the delays for these two stages of t_m^k. $\omega_m^{C,k}$ and $\omega_m^{G,k}$ denote CPU and GPU resources allocated to task t_m^k in the local computing, respectively. Thus, the delay $\acute{\tau}_m^{k,1}$ is expressed as

$$\acute{\tau}_m^{k,1} = \frac{\left(1-q_m^k\right)\varpi_m^k}{\omega_m^{C,k} \acute{f}^C}. \tag{5}$$

Let $\acute{\tau}_m^{k,C}$ and $\acute{\tau}_m^{k,G}$ denote the delays for the hybrid computation stage of scientific and concurrent workflow tasks of t_m^k, respectively, which are expressed as

$$\acute{\tau}_m^{k,C} = \frac{q_m^k\left(1-o_m^k\right)\varpi_m^k}{\omega_m^{C,k} \acute{f}^C} \tag{6}$$

$$\acute{\tau}_m^{k,G} = \frac{q_m^k o_m^k \varpi_m^k}{\omega_m^{G,k} \acute{f}^G}. \tag{7}$$

Thus, the total delay consumed during the hybrid computation stage, $\acute{\tau}_m^{k,2}$, is given as

$$\acute{\tau}_m^{k,2} = \max\left(\acute{\tau}_m^{k,C}, \acute{\tau}_m^{k,G}\right). \tag{8}$$

Let $\acute{\tau}_m^k$ denote the total delay for local computing of t_m^k, which is obtained as

$$\acute{\tau}_m^k = \acute{\tau}_m^{k,1} + \acute{\tau}_m^{k,2}. \tag{9}$$

To address the energy consumption of IEs under both dynamic and static power conditions of CPUs and GPUs. Let

$\acute{e}_m^k$ denote the total energy consumption of local computing of t_m^k, which is expressed by [27]

$$\acute{e}_m^k = \acute{\tau}_m^k\left(\acute{p}^{C,i}+\acute{p}^{G,i}\right)+\left(\acute{\tau}_m^{k,1}+\acute{\tau}_m^{k,C}\right)\acute{p}^{C,d}+\acute{\tau}_m^{k,G}\acute{p}^{G,d} \quad (10)$$

where $\acute{p}^{C,d}$ is the dynamic power of CPU (in W), and $\acute{p}^{C,i}$ is the static power of CPU (in W).

2) Edge Computing: The computing capability of IE is limited, and it has to offload its excessive task t_m^k to its ES, i.e., $\dot{\lambda}_m^k = 1$. Then, the ES processes the computational task t_m^k on behalf of the IE. The task offloading process consists of three phases: uploading, transfer, and computation.

The computing task t_m^k of IE needs to be uploaded to its neighboring AP j. According to (3), $\dot{\tau}_{m,j}^{k,u}$ denotes the data transmission delay from IE m to AP j, which is given as

$$\dot{\tau}_{m,j}^{k,u} = \frac{I_m^k}{R_{m,j}}. \quad (11)$$

The computing task t_m^k is transferred from the source AP j to the target ES associated with AP n that completes the task. $\dot{\tau}_{m,j}^{k,v}$ is the transmission delay for t_m^k, which is computed as

$$\dot{\tau}_{m,j}^{k,v} = \nVdash(n\neq j)\frac{\delta I_m^k}{W_v} \quad (12)$$

where $\nVdash$(condition) denotes an indicator function that returns 1 if the condition is true and 0 otherwise. δ is the number of hops between APs. Note that $\delta = 0$ if $n = j$. In additional, $\dot{\tau}_{m,j}^{k,s}$ is the transmission delay of t_m^k from AP j to the SDN controller, which is given as

$$\dot{\tau}_{m,j}^{k,s} = \frac{I_m^k}{W_s}. \quad (13)$$

The computing task t_m^k is processed by ES directly associated with AP j. Let $\dot{\tau}_{m,j}^{k,1}$ denote the sum of the delays for the task preprocessing stage and result aggregation stage. Let $\dot{\tau}_{m,j}^{k,2}$ denote the total delay of the hybrid computing stage. $\dot{\tau}_{m,j}^{k,1}$ and $\dot{\tau}_{m,j}^{k,2}$ are calculated as

$$\dot{\tau}_{m,j}^{k,1} = \frac{\left(1-q_m^k\right)\varpi_m^k}{\omega_{m,j}^{C,k}\dot{f}^C} \quad (14)$$

$$\dot{\tau}_{m,j}^{k,2} = \max\left(\frac{q_m^k\left(1-o_m^k\right)\varpi_m^k}{\omega_{m,j}^{C,k}\dot{f}^C}, \frac{q_m^k o_m^k \varpi_m^k}{\omega_{m,j}^{G,k}\dot{f}^G}\right). \quad (15)$$

where $\omega_{m,j}^{C,k}$ and $\omega_{m,j}^{G,k}$ are the numbers of CPU and GPU resources allocated to task t_m^k by AP j in the offloading computing process.

Considering the computational results of the task are significantly smaller than the original task data, the delay from the ES returning to the IE can be neglected. The total delay of the offloading process at the ES directly associated with AP j for task t_m^k, $\dot{\tau}_{m,j}^k$, is calculated as

$$\begin{aligned}\dot{\tau}_{m,j}^k &= \phi_{m,j}^k\left(\dot{\tau}_{m,j}^{k,u}+\dot{\tau}_{m,j}^{k,v}\right)\\ &\quad+\left(1-\phi_{m,j}^k\right)\left(\dot{\tau}_{m,j}^{k,u}+2\dot{\tau}_{m,j}^{k,s}+\dot{\tau}_{m,j}^{k,v}\right)+\dot{\tau}_{m,j}^{k,1}+\dot{\tau}_{m,j}^{k,2} \quad (16)\end{aligned}$$

where $\phi_{m,j}^k$ is a load balancing decision variable for task t_m^k by AP j. When the ES directly connected with AP j has sufficient processing capacity to handle task t_m^k, $\phi_{m,j}^k = 1$; otherwise, $\phi_{m,j}^k = 0$. To achieve load balancing, task t_m^k needs to be sent to an idle ES directly associated with AP n ($n\neq j$) for processing.

Meanwhile, $\dot{e}_{m,j}^k$ denotes overall energy consumed by task t_m^k associated with AP j, which is computed as

$$\dot{e}_{m,j}^k = p_m^o\dot{\tau}_{m,j}^{k,u}+\dot{\tau}_{m,j}^k\left(\acute{p}^{C,i}+\acute{p}^{G,i}\right). \quad (17)$$

3) Cloud Computing: If the computational task t_m^k is offloaded to CDC for processing, i.e., $\grave{\lambda}_m^k = 1$, the IE m first transmits it to AP j that provides transmission services via a wireless link. Subsequently, AP j forwards it to the SDN controller, which then transmits it to the CDC for processing over a wired link. Therefore, we consider the data uploading, transmission, and computation delays associated with the task t_m^k. $\grave{\tau}_{m,j}^{k,c}$ is the transmission delay associated with AP j from the SDN controller to CDC, which is given as

$$\grave{\tau}_{m,j}^{k,c} = \frac{I_m^k}{W_c}. \quad (18)$$

Let $\grave{\tau}_{m,j}^{k,1}$ denote the sum of the delays for the task preprocessing stage and result aggregation stage of t_m^k associated with AP j. Let $\grave{\tau}_{m,j}^{k,2}$ denote the total delay of the hybrid computing stage of t_m^k associated with AP j. They are obtained as

$$\grave{\tau}_{m,j}^{k,1} = \frac{\left(1-q_m^k\right)\varpi_m^k}{\omega_{m,j}^{C,k}\grave{f}^C} \quad (19)$$

$$\grave{\tau}_{m,j}^{k,2} = \max\left(\frac{q_m^k\left(1-o_m^k\right)\varpi_m^k}{\omega_{m,j}^{C,k}\grave{f}^C}, \frac{q_m^k o_m^k \varpi_m^k}{\omega_{m,j}^{G,k}\grave{f}^G}\right). \quad (20)$$

According to (11), (13), and (18), the total delay of t_m^k and energy consumption of the IE during the offloading process at CDC are computed as

$$\grave{\tau}_{m,j}^k = \dot{\tau}_{m,j}^{k,u}+\dot{\tau}_{m,j}^{k,s}+\grave{\tau}_{m,j}^{k,c}+\grave{\tau}_{m,j}^{k,1}+\grave{\tau}_{m,j}^{k,2} \quad (21)$$

$$\grave{e}_{m,j}^k = p_m^o\dot{\tau}_{m,j}^{k,u}+\grave{\tau}_{m,j}^k\left(\acute{p}^{C,i}+\acute{p}^{G,i}\right). \quad (22)$$

In summary, the average delay of K tasks and the energy consumption of M IEs in the equipment-edge-cloud system is obtained as

$$\boldsymbol{T} = \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\left\{\acute{\lambda}_m^k\acute{\tau}_m^k+\sum_{j=1}^{J}\left(\dot{\lambda}_m^k\dot{\tau}_{m,j}^k+\grave{\lambda}_m^k\grave{\tau}_{m,j}^k\right)\right\} \quad (23)$$

$$\boldsymbol{E} = \frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\left\{\acute{\lambda}_m^k\acute{e}_m^k+\sum_{j=1}^{J}\left(\dot{\lambda}_m^k\dot{e}_{m,j}^k+\grave{\lambda}_m^k\grave{e}_{m,j}^k\right)\right\}. \quad (24)$$

E. Load Variance Model

To enhance the task-processing efficiency of IEs, it is crucial to utilize ESs for task processing effectively. Load balancing is a key metric for the rational management of ES computational capacity. One constraint is the load balancing among multiple

ESs. U_j is the resource utilization rate of each ES, which is directly connected with AP j and given as

$$U_j = \sum_{m=1}^{M}\sum_{k=1}^{K}\left(\frac{\omega_{m,j}^{C,k}}{\dot{\omega}^C}+\frac{\omega_{m,j}^{G,k}}{\dot{\omega}^G}\right). \tag{25}$$

U is the average resource utilization rate of all ESs, which is computed as

$$U = \frac{1}{J}\sum_{j=1}^{J} U_j. \tag{26}$$

l_j denotes the average load variance of ESs directly connected to AP j, which is obtained as

$$l_j = \left(U_j - U\right)^2. \tag{27}$$

L is the average load variance of all ESs, computed as

$$L = \frac{1}{J}\sum_{j=1}^{J} l_j. \tag{28}$$

F. CPU, GPU, and Memory Models

The CPU and GPU resources allocated to IEs, ESs, and CDC cannot exceed their maximum limits. Thus

$$1 \le \omega_m^{C,k} \le \acute{\lambda}_m^k \acute{\omega}^C \tag{29}$$

$$1 \le \omega_m^{G,k} \le \acute{\lambda}_m^k \acute{\omega}^G \tag{30}$$

$$1 \le \omega_{m,j}^{C,k} \le \dot{\lambda}_m^k \dot{\omega}^C + \grave{\lambda}_m^k \grave{\omega}^C \tag{31}$$

$$1 \le \omega_{m,j}^{G,k} \le \dot{\lambda}_m^k \dot{\omega}^G + \grave{\lambda}_m^k \grave{\omega}^G. \tag{32}$$

Let $\hat{G}_j$ denote the maximum amount of memory in ES directly connected with AP j. The total memory consumption of all IE tasks executed in an ES cannot exceed its limit, i.e.,

$$\sum_{m=1}^{M}\sum_{k=1}^{K}\left(\dot{\lambda}_m^k I_m^k \psi_m^k\right) \le \hat{G}_j \tag{33}$$

where ψ_m^k denotes the memory amount per bit of data for task t_m^k generated by IE m.

G. Problem Formulation

The formulated problem is given as follows. In (34), $\hbar$ denotes a set of decision variables, including $\acute{\lambda}_m^k$, $\dot{\lambda}_m^k$, $\grave{\lambda}_m^k$, $\phi_{m,j}^k$, $\omega_m^{C,k}$, $\omega_m^{G,k}$, $\omega_{m,j}^{C,k}$, $\omega_{m,j}^{G,k}$, p_m^o, and $x_{m,j}$. Our objective is to jointly minimize $\boldsymbol{T}$ and $\boldsymbol{E}$, i.e.,

$$\arg\ \mathbf{min}_{\hbar}\{\boldsymbol{T}, \boldsymbol{E}\} \tag{34}$$

subject to (1), (2), (29), (30), (31), (32), (33)

$$\sum_{m=1}^{M}\sum_{j=1}^{J} W_{m,j} \le \hat{W} \tag{35}$$

$$\boldsymbol{T} \le \sum_{m=1}^{M}\sum_{ky1}^{K} \hat{T}_m^k \tag{36}$$

$$\boldsymbol{E} \le \sum_{m=1}^{M}\sum_{k=1}^{K} \hat{E}_m^k \tag{37}$$

$$\{t_m^l, t_m^k\} \in \mathcal{C} \tag{38}$$

$$\begin{aligned}
&\chi\left(\dot{\tau}_{m,j}^{e,k,t}\right) \le \chi\left(\dot{\tau}_{m,j}^{s,k,1}\right)\\
&\chi\left(\grave{\tau}_{m,j}^{e,k,t}\right) \le \chi\left(\grave{\tau}_{m,j}^{s,k,1}\right)\\
&\chi\left(\acute{\lambda}_m^l(\acute{\tau}_{m,j}^{e,l,1}+\acute{\tau}_{m,j}^{e,l,C})\right) \le \chi\left(\acute{\lambda}_m^k(\acute{\tau}_{m,j}^{s,k,1}+\acute{\tau}_{m,j}^{s,k,C})\right)\\
&\chi\left(\dot{\lambda}_m^l(\dot{\tau}_{m,j}^{e,l,1}+\dot{\tau}_{m,j}^{e,l,C})\right) \le \chi\left(\dot{\lambda}_m^k(\dot{\tau}_{m,j}^{s,k,1}+\dot{\tau}_{m,j}^{s,k,C})\right)\\
&\chi\left(\grave{\lambda}_m^l(\grave{\tau}_{m,j}^{e,l,1}+\grave{\tau}_{m,j}^{e,l,C})\right) \le \chi\left(\grave{\lambda}_m^k(\grave{\tau}_{m,j}^{s,k,1}+\grave{\tau}_{m,j}^{s,k,C})\right)
\end{aligned} \tag{39}$$

$$\acute{\lambda}_m^l+\dot{\lambda}_m^l+\grave{\lambda}_m^l = 1,\ \ \acute{\lambda}_m^k+\dot{\lambda}_m^k+\grave{\lambda}_m^k = 1\ \ \forall t_m^l \in \mathcal{U}_m^k \tag{40}$$

$$L \le \hat{L} \tag{41}$$

$$0 \le p_m^o \le \hat{p}_m^o. \tag{42}$$

Equation (34) represent a classic Pareto optimization problem setting, including $\boldsymbol{T}$ and $\boldsymbol{E}$ [28]. By jointly considering latency and energy, we inherently formulate a multiobjective optimization task where no single solution simultaneously minimizes both objectives to a global optimum without tradeoffs. We adopt the Pareto dominance principle to evaluate feasible solutions. Equation (39) represents that the total bandwidth allocated by J APs to M IEs cannot exceed the total bandwidth $\hat{W}$. Equations (36) and (37) denote the total execution time and energy consumption of K tasks generated by M IEs cannot exceed their maximum limits. Equation (38) shows the constraint of the execution order among scientific workflow tasks t_m^l and t_m^k, i.e., if $\{t_m^l, t_m^k\} \in \mathcal{C}$, task t_m^l cannot be executed before the completion of t_m^k. (39) requires that task t_m^k cannot be executed before it is completely offloaded to AP j or CDC, where $\chi(\dot{\tau}_{m,j}^{e,k,t})$ is the end time of task t_m^k completely offloaded to AP j and $\chi(\dot{\tau}_{m,j}^{s,k,1})$ is the start time of task t_m^k at AP j ready to execute in the task preprocessing and result aggregation stages. Equation (40) represents the constraint on executing scientific workflow tasks, which ensures that task t_m^k cannot be executed until all its preceding tasks are completed. Equation (41) is the constraint of the average load variance of J APs. Equation (42) represents the power constraint for uploading data from IE m.

Due to the integer constraint (e.g., binary task offloading decisions) and other nonlinear constraints (e.g., bandwidth usage, energy consumption, and load variance), the multiobjective task offloading problem is a typical constrained MINLP problem. This formulation is common in complex IIoT scheduling and resource allocation settings because both discrete (selection of AP or CDC) and continuous (bandwidth and power) decisions need to be optimized simultaneously. Such a problem is generally NP-hard [29]. Nonetheless, this formulation faithfully models industrial environments. These environments support the intricacy of real-world industrial workflows, which typically involve heterogeneous tasks, strict deadlines, and limited resources. To solve this problem, we design an improved two-stage MOEA introduced next.

IV. Proposed Framework

This section proposes IT-MOEA to solve the problem. The framework of IT-MOEA and details are presented.

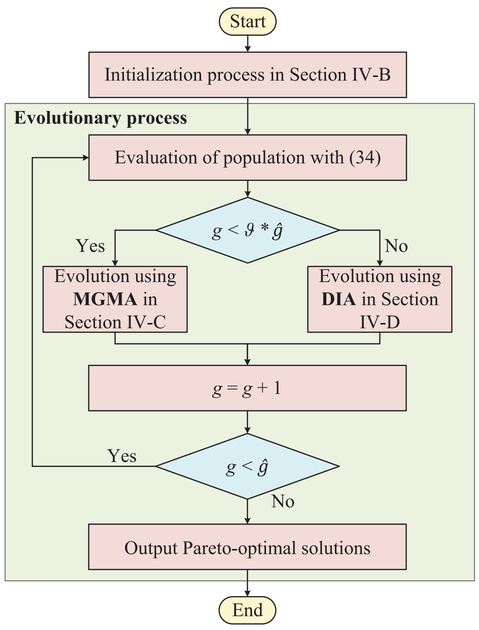

Start
Initialization process in Section IV-B
Evolutionary process
Evaluation of population with (34)
$g < \vartheta * \hat{g}$
Yes
No
Evolution using MGMA in Section IV-C
Evolution using DIA in Section IV-D
$g = g + 1$
Yes
$g < \hat{g}$
No
Output Pareto-optimal solutions
End

Fig. 3. Main flow of IT-MOEA.

A. Framework of IT-MOEA

The framework of IT-MOEA is shown in Fig. 3. IT-MOEA mainly includes the initialization and evolutionary processes. Specifically, IT-MOEA starts with the initialization process in Section IV-B. Then, IT-MOEA enters the evolutionary process. First, the objective function and decision variables for task offloading with (34) are utilized to evaluate the Pareto-optimal solutions of the initial population. To improve the convergence and diversity of MOEA, IT-MOEA divides the evolutionary process into two stages according to a predefined threshold ϑ. IT-MOEA adopts a novel MGMA to evolve the population in the first evolutionary stage, aiming to accelerate the convergence speed and improve optimization performance introduced in Section IV-C. Then, IT-MOEA adopts DIA to evolve the population during the second evolutionary stage, aiming to enhance the population distribution, detailed in Section IV-D. Finally, the entire evolutionary loop terminates upon reaching the maximum number of iterations $\hat{g}$. Subsequently, we can derive the Pareto-optimal solutions and the corresponding decision variables of task offloading with (34) from the final population obtained by IT-MOEA.

B. Population Initialization

Typically, individuals are randomly initialized in the absence of prior information. It often leads to uneven distribution within the search domain, causing individuals to be distant from the global optimum and slowing convergence. Chaos offers a better alternative with properties like ergodicity, randomness, and regularity. Standard chaotic perturbation equations include logistic and tent maps. The logistic map has higher probabilities at the extremes, which is disadvantageous if the global optimum is not at the extremes. The tent map provides better ergodic uniformity and faster search speed but contains small cycles and unstable periodic points. To address

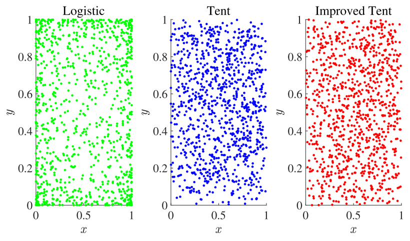


Fig. 4. Comparison of initial distributions of three chaotic sequences.

this, the improved tent chaotic map is expressed as

$$x_{g+1} = \begin{cases} 2x_g + \mathrm{rand}(0,1) \times \frac{1}{N}, & 0 \le x \le \frac{1}{2} \\ 2\left(1 - x_g\right) + \mathrm{rand}(0,1) \times \frac{1}{N}, & \frac{1}{2} < x \le 1 \end{cases}. \quad (43)$$

The transformed expression is given as

$$x_{g+1} = \left(2x_g\right) \bmod 1 + \mathrm{rand}(0,1) \times \frac{1}{N} \quad (44)$$

where N is the population size and $\mathrm{rand}(0, 1)$ is a random number in [0, 1]. IT-MOEA preserves randomness while controlling the random value within a certain range, ensuring the regularity of the Tent chaos.

Fig. 4 shows the initial distribution of chaotic sequences in a 2-D region generated by the logistic, tent, and improved chaotic tent maps. The improved chaotic tent map exhibits better distribution uniformity. This method aims to enhance and improve the distribution quality of the initial population in the search space, strengthen its global search capability, and consequently improve the solution accuracy.

C. Proposed MGMA

Multiobjective grey wolf optimizer (MOGWO) [30] retains the population updating mechanism of GWO, which iteratively simulates the strict hierarchical structure of grey wolves and their hunting and predation behaviors in nature. Therefore, it possesses advantages, such as fast convergence, high efficiency, and high precision. However, MOGWO suffers from premature convergence and local optima trapping issues [31]. Hence, this work proposes an improved MOGWO in the convergence factor update, the position update mechanism of the three best wolves, the population update mechanism, and the archive update mechanism. Fig. 5 shows the main steps of MGMA.

1) Convergence Factor Update: In vanilla MOGWO, A is a crucial parameter controlling the hunting behavior of the wolf population. The variation in the convergence factor a directly influences A, where a linearly decreases from 2 to 0. However, a linearly decreasing strategy often leads to insufficient search space exploration for complex problems. Therefore, we propose a nonlinear convergence factor a, which is updated as

$$a = 1 + \cos\left(\frac{\pi g}{\hat{g}}\right) \quad (45)$$

where $\hat{g}$ represents the maximum iteration number, and g denotes the current iteration number.

Fig. 6 compares the nonlinear function in this work with the linear function in MOGWO. In early iterations, the nonlinear

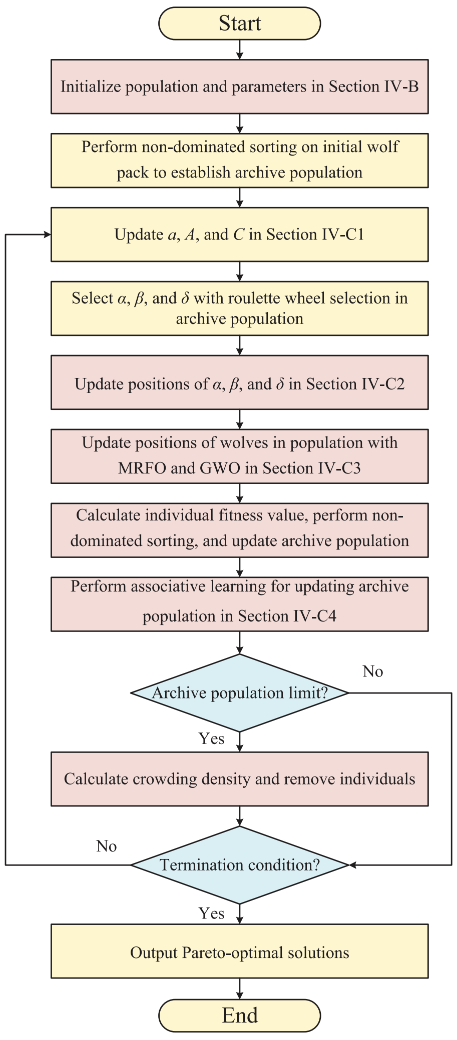

Start
Initialize population and parameters in Section IV-B
Perform non-dominated sorting on initial wolf pack to establish archive population
Update a, A, and C in Section IV-C1
Select α, β, and δ with roulette wheel selection in archive population
Update positions of α, β, and δ in Section IV-C2
Update positions of wolves in population with MRFO and GWO in Section IV-C3
Calculate individual fitness value, perform non-dominated sorting, and update archive population
Perform associative learning for updating archive population in Section IV-C4
Archive population limit?
No
Yes
Calculate crowding density and remove individuals
Termination condition?
No
Yes
Output Pareto-optimal solutions
End

Fig. 5. Main steps of MGMA.

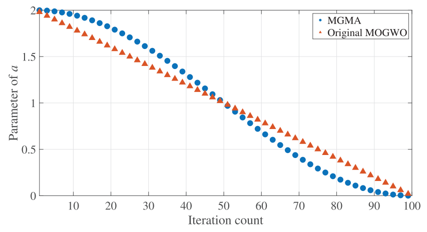


Fig. 6. Comparison of convergence factors between MGMA and MOGWO.

function has a wider range of a, which expands its exploration capability. In later iterations, a becomes smaller, facilitating local exploitation and accelerating the convergence speed. In addition, C denotes the random value in $[0, 2]$ to emphasize exploration [30].

2) Position Update Mechanism of Three Best Wolves: α, β, and δ represent the evolutionary directions of the population in MOGWO, playing crucial roles in guiding the search process. However, their position updates rely on the same mechanism, disregarding the unique status of α in the traditional MOGWO. In practice, individuals typically follow guidance primarily from higher ranked wolves. Yet, α, β, and δ also receive leadership from wolves of lower ranks, which is not rational. The performance is limited because when all wolves are attracted to α, population diversity deteriorates rapidly, leading to premature convergence.

To address this issue, separate update strategies are proposed for α, β, and δ. δ accepts leadership from α and β. Its update method is given as

$$\boldsymbol{X}_{\delta}(g+1) = D_{\delta-}\boldsymbol{X}_{\delta}(g) \tag{46}$$

$$D_{\delta} = \rho\boldsymbol{X}_{\delta}(g)+(1-\rho)\boldsymbol{X}_{\alpha}(g)+(1-\rho)\boldsymbol{X}_{\beta}(g) \tag{47}$$

where ρ is a random number uniformly distributed in [0, 1]. The random number introduces variability throughout the optimization process, facilitating global exploration.

β accepts leadership from α, incorporating a spiral updating mechanism inspired by the whale optimization algorithm (WOA) [32] to approach α in a spiral motion. Its update method is given as

$$\boldsymbol{X}_{\beta}(g+1) = \left|\boldsymbol{X}_{\alpha}(g) - \boldsymbol{X}_{\beta}(g)\right| e^{b\dot{l}} \cos\left(2\pi \dot{l}\right) + \rho \boldsymbol{X}_{\alpha}(g) \tag{48}$$

where b is a constant for defining the shape of the logarithmic spiral, $\dot{l}$ is a random number in [-1, 1], and ρ is a random number uniformly distributed in [0, 1]. Randomness similarly enhances the exploration capability of β.

α holds the highest rank in the wolf population and is not guided by other wolves. Therefore, random walks are introduced to update α. The Lévy flight mechanism is chosen for its ability to explore short distances and occasionally make long jumps, providing a balance between local exploration and occasional long-range exploration. Short-distance exploration ensures that α searches effectively around its current position, enhancing the speed and accuracy of optimization. Occasional long jumps extend the search area of α, enabling broader exploration. Given the stochastic nature of the Lévy mechanism, we incorporate a greedy selection strategy to achieve a survival-of-the-fittest mechanism. The position update of α with the Lévy flight mechanism is given as

$$\boldsymbol{X}'_{\alpha}(g+1) = \boldsymbol{X}_{\alpha}(g) + \xi \oplus \text{Lévy}(\hbar) \tag{49}$$

where ξ is the step size control factor. $\oplus$ is the dot product operation. Lévy$(\hbar)$ represents the random search path, which follows a Lévy distribution. For computational convenience, the Mantegna algorithm is commonly used to simulate its flight trajectory. Lévy$(\hbar)$ is obtained as

$$\text{Lévy}(\hbar) = \frac{\mu}{|\nu|^{\frac{1}{\kappa}}} \tag{50}$$

where $\kappa = 1.5$, and μ and ν follow normal distributions, i.e., $\mu \sim N(0, \sigma_{\mu}^2)$ and $\nu \sim N(0, \sigma_{\nu}^2)$. The variances are expressed as

$$\sigma_{\mu}^2 = \left\{ \frac{\Gamma(1+\kappa)}{\frac{\kappa\Gamma(1+\kappa)}{2}} \frac{\sin\left(\frac{\pi\kappa}{2}\right)}{2^{\frac{\kappa-1}{2}}} \right\}^{\frac{2}{\kappa}}$$
$$\sigma_{\nu}^2 = 1 \tag{51}$$

where $\Gamma(\cdot)$ represents the Gamma function. The position update of α is expressed as

$$\boldsymbol{X}_{\alpha}(g+1) = \begin{cases} \boldsymbol{X}'_{\alpha}(g+1), & \text{other} \\ \boldsymbol{X}_{\alpha}(g), & f\left(\boldsymbol{X}'_{\alpha}(g+1) > \boldsymbol{X}_{\alpha}(g)\right) \text{ and rand} < \dot{p} \end{cases} \tag{52}$$

where rand represents a random variable in [0, 1], $\dot{p}$ denotes the probability of survival-of-the-fittest selection, and $f(\cdot)$ represents the fitness value. This mechanism guides the population toward optimal evolution while effectively enhancing search efficiency.

3) Population Update Mechanism: We design a novel position updating mechanism to enhance information exchange among the grey wolf population, inspired by the manta rays foraging optimization (MRFO) [33]. The first half of (53) provides rapid convergence to the optimal solution, while the latter provides higher population diversity to prevent premature convergence. Thus

$$\begin{aligned}\boldsymbol{X}(g{+}1) \ &= \frac{w(\boldsymbol{X}_1(g){+}\boldsymbol{X}_2(g){+}\boldsymbol{X}_3(g))}{3}{+}a(1{-}w)\\ &\quad (r(\boldsymbol{X}_r(g){-}\boldsymbol{X}(g)){+}a\dot{u}r(\boldsymbol{X}_\alpha(g){-}\boldsymbol{X}(g)))\end{aligned} \tag{53}$$

where $w = [(\hat{w}{-}\check{w})g/\hat{g}]{+}\check{w}$, $\dot{u} = 2e^{r\iota}\sin(2\pi r)$, and $\iota = (\hat{g}{-}g{+}1/\hat{g})$. w denotes an inertia weight in iteration g. $\check{w}$ and $\hat{w}$ are the minimum and maximum values of w. r is a random number in (0, 1). $\boldsymbol{X}_r(g)$ denotes a randomly selected individual. In the initial stage, individuals exhibit higher social learning capabilities, ensuring exploration of the globally optimal position and enhancing search space coverage. The later stage focuses on searching around α to accelerate convergence.

4) Archive Update Mechanism: In MOGWO, an external archive stores nondominated solutions. This mechanism effectively preserves elite-level information during early iterations. However, as iterations progress, the number of nondominated solutions increases sharply. Although crowding distance-based deletion partially ensures solution quality, it may still result in loss of solution information. Additionally, the influx of similar solutions in later iterations can lead the population to local optima. To enhance solution set diversity, this work perturbs solutions in the crowded regions of the archive. Associative learning, a recently proposed update strategy, is employed to improve exploratory performance [34]. Hence, this work introduces associative learning to update some individuals in the archive. Thus

$$\begin{aligned}\boldsymbol{X}(g{+}1) &= \boldsymbol{X}(g){+}0.001G\big(\boldsymbol{X}(g){-}\check{\varkappa},\, \hat{\varkappa}{-}\boldsymbol{X}(g)\big)\\ &\quad {+}b_0S_1r_1(\boldsymbol{X}_r(g){-}\boldsymbol{X}(g)){+}b_0S_2r_2(\boldsymbol{X}_\alpha(g){-}\boldsymbol{X}(g))\end{aligned} \tag{54}$$

where $G(\cdot)$ denotes a Gaussian distribution function, $\check{\varkappa}$ and $\hat{\varkappa}$ represent the upper and lower bounds of the search space dimensions, respectively. r_1 and r_2 are random numbers in (0, 1), b_0 is a constant, and S_1 and S_2 are adaptive cognitive and social factors, respectively. S_1 and S_2 are updated as

$$\begin{aligned}S_1 &= \left(1{-}\frac{g}{\hat{g}}\right)\\ S_2 &= \frac{2g}{\hat{g}}.\end{aligned} \tag{55}$$

D. Diversity-Enhanced Immune Algorithm

In the second evolutionary stage, IT-MOEA uses DIA to improve the population distribution. Specifically, DIA allocates the cloning resources for each individual according to the vertical distance [35] between the individual and its corresponding weight vector, i.e.,

$$V_i = \left\| F(\boldsymbol{X}_i){-}\left(Z^*{+}d(\boldsymbol{X}_i, \boldsymbol{\lambda}_i, Z^*)\frac{\boldsymbol{\lambda}_i}{\|\boldsymbol{\lambda}_i\|}\right)\right\| \tag{56}$$

$$d\big(\boldsymbol{X}_i, \boldsymbol{\lambda}_i, Z^*\big) = \frac{\|(F(\boldsymbol{X}_i){-}Z^*)\boldsymbol{\lambda}_i\|}{\|\boldsymbol{\lambda}_i\|} \tag{57}$$

where $d(\boldsymbol{X}_i, \boldsymbol{\lambda}_i, Z^*)$ denotes the projection of vector $F(\boldsymbol{X}_i){-}Z^*$ on the weight vector $\boldsymbol{\lambda}_i$ $(1{\le}i{\le}N)$. $F(\cdot)$ denotes the objective function, and Z^* denotes an ideal approximated point. Based on the above vertical distance, the cloning number c_i of individual $\boldsymbol{X}_i$ is calculated as

$$c_i = \left\lceil \frac{N(1{-}V_i)}{\sum_i^N(1{-}V_i)} \right\rceil. \tag{58}$$

Note that smaller vertical distance values imply that the individual is closer to its weight vector and performs better about diversity. As a result, these individuals with smaller vertical distance values obtain more opportunities for cloning. Then, each individual performs the proportional cloning operator [36], defined as

$$\tilde{\boldsymbol{X}} = \bigcup_{i=1}^{N}\{c_i{\otimes}\boldsymbol{X}_i\} \tag{59}$$

where $\otimes$ indicates the cloning operator, $\tilde{\boldsymbol{X}}$ indicates the cloning population consisting of all cloning offspring. According to the principle of DIA, these individuals with better diversity receive more computing resources to generate more promising offspring, thus significantly improving the distribution of the whole population [35]. The computational complexity analysis of IT-MOEA is provided in Section I-A of the supplementary file.

V. Performance Evaluation

In this section, we first introduce the experimental setup. Then, we investigate the effect of related parameters on the IT-MOEA. Finally, we discuss the experimental results.

A. Experimental Setup

1) Dataset: CPU, GPU, and memory requirements of the task and the expected computing time are derived from two datasets, i.e., ClusterData 2011 traces [37] and Alibaba Cluster Data V2018 [38]. Furthermore, the ESs and CDC specifications in the simulation environment are obtained from the SPECpower_ssj2008 dataset.

2) Simulation Environment: In this work, the simulation environment is implemented in MATLAB, and runs on a personal computer with Intel Xeon Gold 6152 10-Core 2.1 GHz, 32.0-GB RAM, and NVIDIA GeForce RTX 3090. Moreover, this work simulates the experiments in a 1000 m × 1000 m area with MATLAB [39], where a CDC, IEs, and APs are deployed in a grid network. Specifically, the service range of each AP is 100 m, and each AP is randomly deployed following a uniform distribution. Considering the assistance of multiple APs, the maximum communication distance between APs is 150 m. IEs are uniformly deployed within the area of

Algorithm 1: IT-MOEA

```
Input: Maximum iteration number ($\hat{g}$), population size
       ($N$), objective function ($F$)
Output: Final population ($\mathbb{P}$)
/* Initialization process */
Initialize parameters of MGMA and DIA;
Initialize positions of individuals to obtain $\mathbb{P}$ with (44);
/* Evolutionary process */
/* MGMA */
Initialize $a$ with (45), $A$, and $C$;
Evaluate $F(\boldsymbol{X}_i)$ of each $\boldsymbol{X}_i$ with (34);
Perform nondominated sorting on $\mathbb{P}$ to establish the
archive $\mathbb{R}$;
for $g \leftarrow 1$ to $\vartheta \times \hat{g}$ do
    for $i \leftarrow 1$ to $N$ do
        Select $\alpha$, $\beta$, and $\delta$ in $\mathbb{R}$;
        Update $\alpha$, $\beta$, and $\delta$ with (46)–(52);
        Update positions of $\boldsymbol{X}_i$ in $\mathbb{P}$ with (53);
    end
    Update $a$ with (45), $A$, and $C$;
    Evaluate $F(\boldsymbol{X}_i)$ of each $\boldsymbol{X}_i$ with (34);
    Perform nondominated sorting on $\mathbb{P}$;
    Update $\mathbb{R}$ with (54) and (55);
    if $|\mathbb{R}| > N$ then
        Update $\mathbb{R}$ with crowding density;
    end
end
/* DIA */
for $g \leftarrow \vartheta \times \hat{g}$ to $\hat{g}$ do
    for $i \leftarrow 1$ to $N$ do
        Calculate $V_i$ for $\boldsymbol{X}_i$ with (56) and (57);
        Calculate $c_i$ for $\boldsymbol{X}_i$ with (58);
    end
    Generate $\tilde{\boldsymbol{X}}$ on $\mathbb{P}$ with (59);
    $\mathbb{P} \leftarrow \mathbb{P} \cup \tilde{\boldsymbol{X}}$;
end
```

TABLE I
SIX TEST INSTANCES

Instances	M	J	K
1	1–5	1–3	1–4
2	6–10	4–6	5–8
3	11–15	7–9	9–12
4	16–20	10–12	13–16
5	21–25	13–15	17–20
6	26–30	16–18	21–24

the APs. Six experimental scales with different M, J, and K are given in Table I.

3) Parameter Settings: According to [40], [41], parameters of IEs, APs, CDC, and tasks are set as follows. For the heterogeneous computing process with CPUs and GPUs, $\acute{\omega}^C = 4$, $\dot{\omega}^C = 16$, $\grave{\omega}^C = 32$, $\acute{\omega}^G = 128$, $\dot{\omega}^G = 1024$, and $\grave{\omega}^G = 6192$. Besides, $\acute{f}^C = 2\times10^9$ FLOPS, $\dot{f}^C = 1\times10^{10}$ FLOPS, $\grave{f}^C y 1\times10^{11}$ FLOPS, $\acute{f}^G = 2\times10^7$ FLOPS, $\dot{f}^G = 5\times10^8$ FLOPS, and $\grave{f}^G = 3\times10^9$ FLOPS. $\acute{p}^{C,d} = 65$ W, $\acute{p}^{G,d} = 160$ W, $\acute{p}^{C,i} = 15$ W, and $\acute{p}^{G,i} = 50$ W. For the computation task t_m^k, $I_m^k = [25, 50]$ MB, $\varpi_m^k = [2.5\times10^8, 5\times10^8]$ FLOPS, $\hat{T}_m^k = [1, 5]$ sec, $\hat{E}_m^k = [3, 6]$ J, $q_m^k = [0.5, 0.9]$, and $o_m^k = [0.5, 0.9]$. For the communication process, $d_{m,j} = [50, 200]$ m, $\varsigma = 4$, $h = 0.98$, $\sigma = 1.6\times10^{-11}$, $W_{m,j} = [2, 4]$ MHz, $\varrho = 1$, $\zeta = 1$, $W_v = 125$ Mb/s, $W_s = 200$ Mb/s, $W_c = 256$ Mb/s, and $\delta = 1$. For the constraints, $\hat{G}_j = 2$ GB, $\hat{W} = 20$ MHz, $\hat{L} = 0.5$, and $\hat{p}_m^o = 0.5$ W. For MGMA, $b_0 = 1.4172$. The threshold ϑ in IT-MOEA is 0.6. The overall population size (N) is 30, and the maximum iteration number ($\hat{g}$) is 1000.

4) Compared Algorithms: To comprehensively evaluate the performance of IT-MOEA, several SOTA algorithms are employed for comparisons, including four single-objective optimization evolutionary algorithms (PGL [5], HGGWO [42], SBAGO [43], SAPSO [44]), and seven MOEAs (MOGWO, NSGA-II [45], MOMVO [46], MOEA/D [47], LMOCSO [48], MOWOA [32], and AR-MOEA [49]) with six test instances. The reasons for choosing them for comparison are given in Section I-B of the supplementary file.

5) Performance Metrics: We consider not only the two metrics of completion time and energy consumption separately but also their weighted sum, which is obtained as

$$\varpi = \varphi \times \bar{\boldsymbol{T}} + (1-\varphi)\bar{\boldsymbol{E}} \tag{60}$$

where $\bar{\boldsymbol{T}}$ and $\bar{\boldsymbol{E}}$ are the normalized completion time and the normalized energy consumption, respectively. φ is a control factor and is 0.5. It maintains equal weighting between completion time and energy consumption. This balanced weighting prevents inherent bias toward either objective while enabling fair performance comparison across different optimization scenarios. Particularly, two normalized objective values $\bar{\boldsymbol{T}}$ and $\bar{\boldsymbol{E}}$ are computed as

$$\bar{\boldsymbol{T}} = \frac{\boldsymbol{T}}{\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\acute{\tau}_m^k} \tag{61}$$

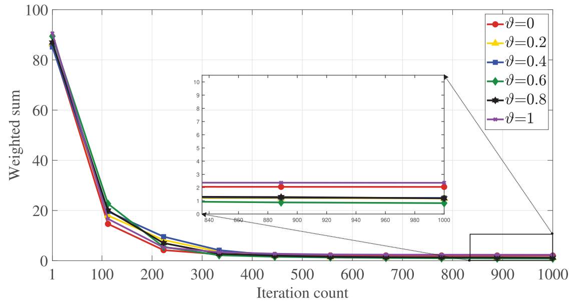
$$\bar{\boldsymbol{E}} = \frac{\boldsymbol{E}}{\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\sum_{j=1}^{J}\left(\dot{\lambda}_m^k \dot{e}_{m,j}^k + \grave{\lambda}_m^k \grave{e}_{m,j}^k\right)}. \tag{62}$$

By normalizing these two objective values to the same order of magnitude, the weighted sum can effectively represent the MOEA's overall optimization performance concerning both completion time and energy consumption.

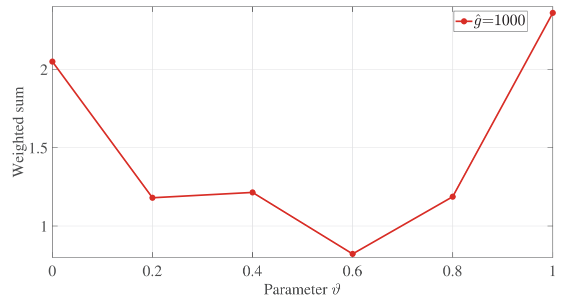
For the comparison of MOEAs, two other evaluation metrics are used to evaluate the performance of the algorithms, including average inverse generation distance (AIGD) and average spacing (ASP) [30], [50]. AIGD assesses the convergence and distribution performance of MOEAs, where a smaller AIGD signifies superior overall performance. ASP measures the distribution degree of nondominated solutions, with a smaller ASP indicating better distribution and diversity.

B. Parameter Analysis

To discuss the effect of ϑ on the performance of IT-MOEA, the parameter analysis is conducted with different ϑ (i.e., $\vartheta \in [0, 1]$) in this section. According to Algorithm 1, the evolutionary preference of IT-MOEA for two components, i.e., MGMA and DIA, is adjusted by ϑ. The relevant parameters



(a)



(b)

Fig. 7. Result of IT-MOEA with different ϑ for instance 4. (a) Convergence result over iterations (b) The final weighted sum after 1000 iterations

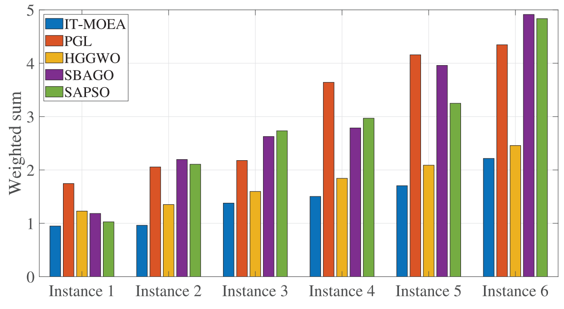
of all variants are kept consistent except for ϑ. The average results of ten independent runs at instance 4 for all variations are employed for comparative analysis.

Fig. 7(a) provides the convergence results of all variants within 1000 iterations at instance 4. Specifically, IT-MOEA exhibits good convergence performance when $\vartheta > 0$, which implies that MGMA can accelerate the convergence speed in the early stage of IT-MOEA. Moreover, using only MGMA or DIA, i.e., $\vartheta = 1$ or 0, for IT-MOEA leads to premature convergence due to falling into local optimum. Therefore, IT-MOEA utilizes MGMA and DIA in two evolutionary stages during the evolution process to obtain higher quality solutions. Moreover, Fig. 7(b) shows the final results obtained by IT-MOEA after 1000 iterations for different ϑ. As ϑ increases, the weighted sum obtained by IT-MOEA exhibits a decreasing-then-increasing trend, with the minimum value occurring when $\vartheta = 0.6$. Thus, $\vartheta = 0.6$ for IT-MOEA.

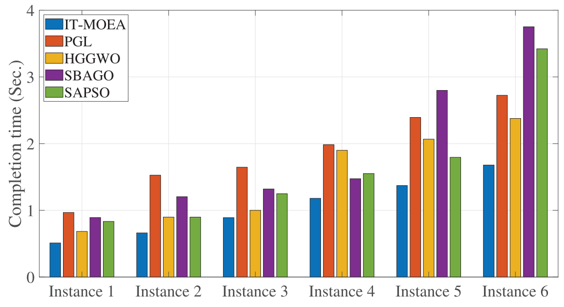
C. Experimental Results

This section discusses the experimental results by comparing IT-MOEA with several SOTA single-objective algorithms, MOEAs, and offloading schemes on six scales. The remaining experimental results are presented in the supplementary file. Each peer is independently executed ten times, and the average results are recorded to assess overall performance.

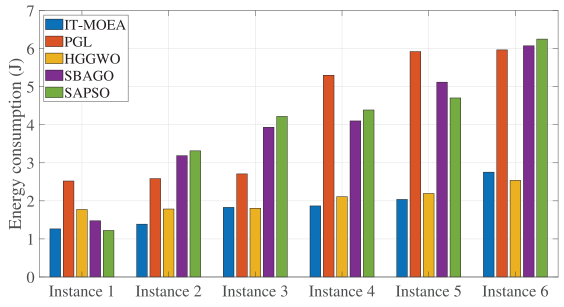
1) Comparison of Single Objective Results: Fig. 8 exhibits the experimental results of IT-MOEA compared to the other single-objective peers concerning weighted sum, completion time, and energy consumption, respectively. In Fig. 8(a), it is evident that SBAGO and SAPSO consistently yield larger weighted sums than IT-MOEA in most instances, indicating



(a)



(b)



(c)

Fig. 8. Single-objective comparison results regarding weighted sum, completion time, and energy consumption, respectively. (a) Weighted sum. (b) Completion time. (c) Energy consumption.

that they fail to converge to high-quality solutions. The reason is that these algorithms require careful tuning of key parameters and are prone to local optima, hindering their global exploration capabilities. Furthermore, although PGL occasionally achieves lower weighted sums than SBAGO and SAPSO, it still cannot outperform IT-MOEA. The reason is that PGL only finds a locally optimal solution compared with IT-MOEA. In Fig. 8(c), IT-MOEA demonstrates superior performance in most test scenarios except HGGWO in instance 6. This is because of HGGWO's specialized local exploitation strategy for high-dimensional energy optimization. Regarding weighted sum, completion time, and energy consumption, IT-MOEA outperforms other SOTA single-objective peers with the average reductions of 24.7%, 14.9%, and 34.5%, respectively. Notably, IT-MOEA achieves its most substantial advantages in large-scale tasks, i.e., instances 4–6. The main reason is that IT-MOEA employs a two-stage evolutionary process, where the first stage uses MGMA to accelerate the convergence speed and improve optimization performance, and the second stage adopts DIA to enhance the diversity.

2) Comparison of Multiple Objective Results: We conduct a comparative analysis of the performance of eight MOEAs, focusing on the completion time of industrial applications and the energy consumption of IEs for six test instances. Figs. S2 and S3 of the supplementary file present the box plots of eight algorithms concerning completion time and energy consumption, respectively. In Figs. S2 and S3 of the supplementary file, IT-MOEA performs better than others in all test instances. This is because IT-MOEA utilizes MGMA to optimize completion time and energy consumption by enhancing MOGWO. Improvements include the convergence factor update, the position update, the population update, and the archive update mechanisms.

Fig. 9 shows the Pareto fronts obtained by eight algorithms, demonstrating that IT-MOEA performs best in six test instances. While the Pareto-optimal solution sets of AR-MOEA and LMOCSO are close to those of IT-MOEAs, their convergence performance and distribution are weaker than those of IT-MOEAs. Notably, IT-MOEA effectively strikes the optimal balance between completion time and energy consumption. IT-MOEA exhibits superior diversity and distribution across six test instances. As the scale of experiments increases, the comparison algorithms face significant challenges regarding convergence and distribution, but IT-MOEA still performs well. IT-MOEA shows a clear advantage in convergence in large-scale scenarios. IT-MOEA shows exceptional stability and convergence capabilities in addressing the problem in a realistic industrial environment. This advantage can be attributed to several key merits inherent to IT-MOEA, including a high-quality initial population, avoiding local optima, global exploration capability, and distributed enhancement.

Tables II and III illustrate the average values of AIGD and ASP for all algorithms. The best results for AIGD and ASP are highlighted in bold. The results in Tables II and III demonstrate that for all six test instances, IT-MOEA consistently outperforms other MOEAs. IT-MOEA indicates a better balance between global exploration and local exploitation to achieve the best AIGD and ASP with average reductions of 38.3% and 43.7%, respectively, over existing algorithms. Specifically, IT-MOEA reduces AIGD by 11.5% compared to the second-best AR-MOEA across six instances. Similarly, it reduces ASP by 28.5% on average compared to AR-MOEA, particularly excelling in low-dimensional cases. In Instance 6 with the highest dimensionality, IT-MOEA reduces the advantage AIGD by 4.6% compared to AR-MOEA while improving ASP by 13.8%, demonstrating superior scalability and robustness. The reason is that NSGA-II, MOGWO, MOMVO, and MOWOA only rely on the Pareto-dominance and are trapped in local optima. Furthermore, MOEA/D is sensitive to decomposition methods and weight vector settings, potentially leading to poor diversity and convergence on complex Pareto fronts. In addition, LMOCSO has potential stagnation risks and Pareto front degradation in large-scale multiobjective optimization scenarios. AR-MOEA uses adaptive reference points and an enhanced indicator to speed up convergence, but these features weaken its ability to differentiate solutions, especially on unevenly distributed Pareto fronts. IT-MOEA incorporates several more outstanding strategies to enhance its performance. First, IT-MOEA leverages improved initialization to generate a high-quality initial population, facilitating effective search space exploration. Additionally, IT-MOEA conducts an improved strategy of updating the positions of the three best wolves to enhance the speed and accuracy of the optimization process, thereby broadening the search and facilitating global exploration. Inspired by MRFO, IT-MOEA promotes information exchange among populations, enhancing its exploration capabilities. IT-MOEA employs associative learning to perturb and mutate solutions in the archive, improving overall exploration performance. Finally, IT-MOEA uses the vertical distance value during the cloning process to maintain the diversity of the whole population, effectively improving the performance of MOEAs in solving complicated industrial problems. Overall, IT-MOEA combines advanced initialization, population interaction, solution perturbation, and distribution enhancement strategies, yielding better AIGD and ASP.

TABLE II
AIGD of Eight Algorithms

Algorithms	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6
NSGA-II	18.784	18.553	19.689	22.018	25.553	27.411
MOGWO	10.540	17.346	18.329	20.864	22.726	24.162
MOMVO	8.678	9.721	10.265	12.344	14.129	17.127
MOEA/D	7.350	8.166	8.782	10.182	12.226	14.827
LMOCSO	6.716	7.419	8.543	9.825	10.439	13.699
MOWOA	7.017	8.288	8.691	9.781	11.165	13.876
AR-MOEA	4.182	6.203	7.619	8.161	10.235	12.197
IT-MOEA	**3.572**	**5.146**	**6.873**	**7.203**	**9.077**	**11.635**

TABLE III
ASP of Eight Algorithms

Algorithms	Instance 1	Instance 2	Instance 3	Instance 4	Instance 5	Instance 6
NSGA-II	2.602	3.491	4.396	4.419	6.063	9.452
MOGWO	2.022	2.726	3.925	3.642	5.152	7.821
MOMVO	1.748	2.624	2.665	3.345	5.139	7.593
MOEA/D	1.394	2.153	2.376	4.381	4.984	6.252
LMOCSO	1.020	1.514	1.774	2.985	4.316	6.147
MOWOA	1.031	1.625	1.847	2.892	4.574	6.038
AR-MOEA	0.847	1.329	1.706	2.752	4.138	5.945
IT-MOEA	**0.252**	**0.846**	**1.024**	**2.579**	**3.963**	**5.126**

3) Comparison of Different Offloading Schemes: The above experimental results have shown that IT-MOEA exhibits excellent and reliable performance concerning both convergence and distribution for reducing completion time and energy consumption. To further evaluate the performance of IT-MOEA on different offloading problems, we compare IT-MOEA with other three offloading schemes, which are edge offloading scheme (EOS) [51], cloud offloading scheme (COS) [52], and partial offloading scheme (POS) [53]. EOS and COS represent that all tasks are offloaded to ESs and CDC. POS refers to the allocation of all tasks based on predetermined and practical user association tactics. Each task is offloaded to its corresponding AP, and if the resources of ESs are inadequate, some tasks are offloaded to a CDC.

To more effectively assess the efficacy of various schemes, we compare the weighted sum of completion time and energy consumption. Fig. S4 of the supplementary file shows the convergence plots of all comparison offloading schemes at six

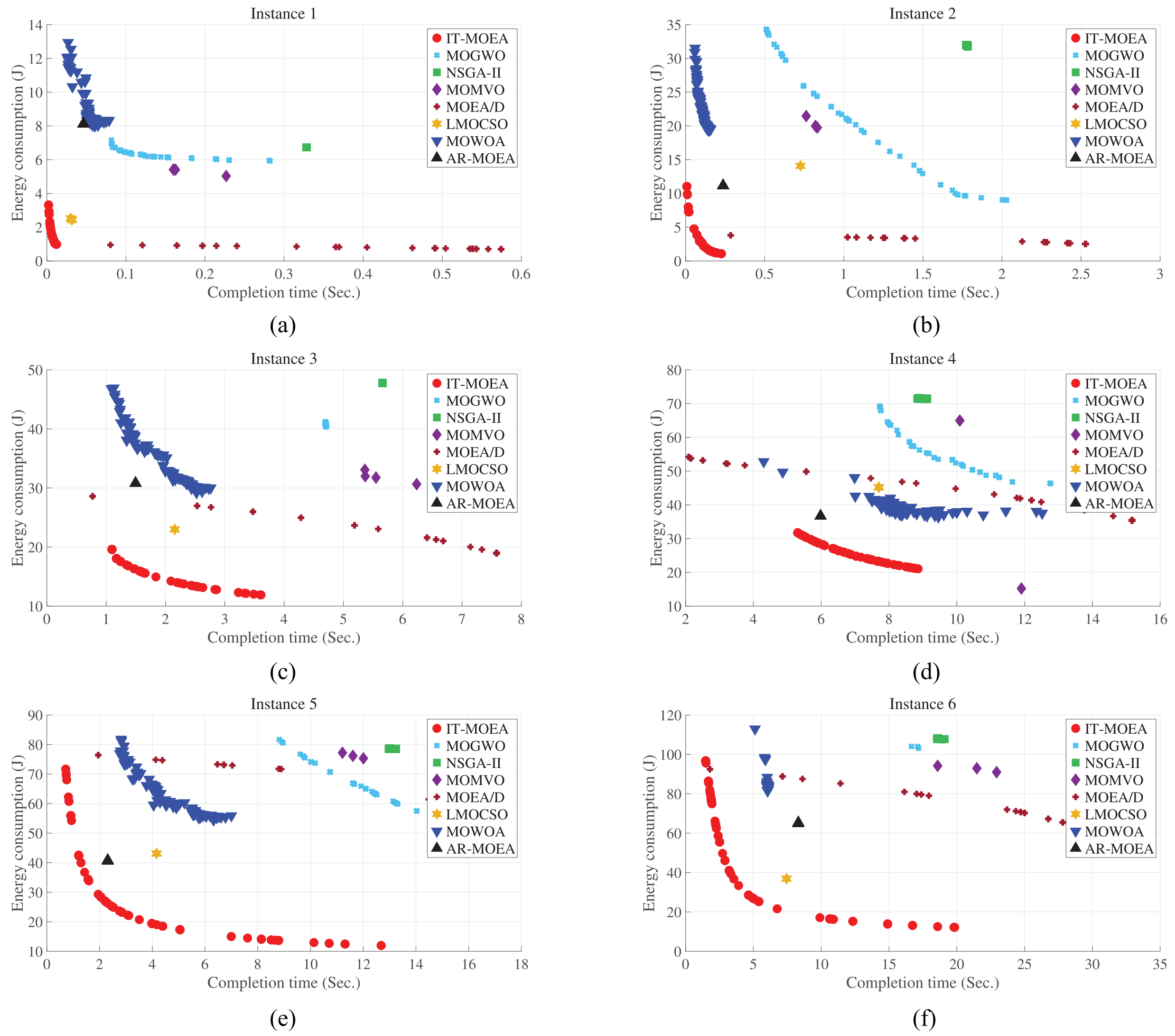


Fig. 9. Pareto-optimal fronts of eight algorithms for six test instances. (a) Instance 1. (b) Instance 2. (c) Instance 3. (d) Instance 4. (e) Instance 5. (f) Instance 6.

different experimental scales, where the *y*-axis indicates the weighted sum values calculated by (60) and the *x*-axis indicates the iteration count. In six different experimental scales, IT-MOEA shows fast convergence speed as its weighted sum decreases rapidly, which is mainly attributed to MGMA in the early evolutionary stage of IT-MOEA because it provides strong convergence performance to accelerate the convergence speed significantly. Moreover, the weighted sum obtained by IT-MOEA is lower than comparison algorithms after 1000 iterations. In addition, Fig. S5 of the supplementary file shows experimental results of IT-MOEA compared to the other offloading schemes concerning completion time and energy consumption. Experimental results demonstrate that IT-MOEA consistently outperforms EOS, COS, and POS across six task scales in terms of both completion time and energy consumption. This is because IT-MOEA uses a two-stage evolutionary optimization framework, i.e., MGMA accelerates convergence by efficiently approximating Pareto-optimal solutions, while DIA enhances solution diversity to avoid local optima. In contrast, EOS faces high energy cost due to limited edge resources, COS suffers from slow task completion and high energy from long-distance cloud communication, and POS cannot adapt flexibly when the number of tasks grows, leading to inefficient energy usage. The above results demonstrate that the computing offloading decision scheme obtained by IT-MOEA outperforms its peers in real-world scenarios.

VI. Conclusion

In the industrial Internet, both the completion time of industrial applications and the energy consumption of IEs need to be optimized. Considering the diverse requirements of heterogeneous tasks in industrial environments, coordinating and task offloading multiple IEs with diverse task types and multiple APs poses challenges. Existing studies on task offloading either consider only one objective or do not consider the impact of heterogeneous tasks in hybrid computing with CPUs and GPUs. This work proposes a three-stage heterogeneous computing architecture that accurately describes the multitask processing of scientific and concurrent workflows, considering constraints, such as offloading decisions, task priorities, and load balancing in real industrial environments. We design an IT-MOEA to simultaneously minimize task completion time and energy consumption for IEs. Comprehensive experimental results indicate that IT-MOEA outperforms other SOTA algorithms regarding convergence and distribution performance.

REFERENCES

[1] Z. Cao, P. Zhou, R. Li, S. Huang, and D. Wu, "Multiagent deep reinforcement learning for joint multichannel access and task offloading of mobile-edge computing in industry 4.0," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6201–6213, Jul. 2020.

[2] P. Zhang, C. Wang, C. Jiang, and Z. Han, "Deep reinforcement learning assisted federated learning algorithm for data management of IIoT," *IEEE Trans. Ind. Informat.*, vol. 17, no. 12, pp. 8475–8484, Dec. 2021.

[3] J. Bi, H. Yuan, S. Duanmu, M. Zhou, and A. Abusorrah, "Energy-optimized partial computation offloading in mobile-edge computing with genetic simulated-annealing-based particle swarm optimization," *IEEE Internet Things J.*, vol. 8, no. 5, pp. 3774–3785, Mar. 2021.

[4] M. Chen, Z. Yang, W. Saad, C. Yin, H. V. Poor, and S. Cui, "A joint learning and communications framework for federated learning over wireless networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 1, pp. 269–283, Jan. 2021.

[5] J. Bi, H. Yuan, K. Zhang, and M. Zhou, "Energy-minimized partial computation offloading for delay-sensitive applications in heterogeneous edge networks," *IEEE Trans. Emerg. Topics Comput.*, vol. 10, no. 4, pp. 1941–1954, Oct.–Dec. 2022.

[6] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial Internet of Things: Challenges, opportunities, and directions," *IEEE Trans. Ind. Informat.*, vol. 14, no. 11, pp. 4724–4734, Nov. 2018.

[7] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing," *IEEE Trans. Wireless Commun.*, vol. 20, no. 12, pp. 7947–7962, Dec. 2021.

[8] C. Chen, K. Li, A. Ouyang, Z. Zeng, and K. Li, "GFlink: An in-memory computing architecture on heterogeneous CPU-GPU clusters for big data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 6, pp. 1275–1288, Jun. 2018.

[9] H. Yuan, J. Bi, and M. Zhou, "Energy-efficient and QoS-optimized adaptive task scheduling and management in clouds," *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 2, pp. 1233–1244, Apr. 2022.

[10] K. Peng, H. Huang, B. Zhao, A. Jolfaei, X. Xu, and M. Bilal, "Intelligent computation offloading and resource allocation in IIoT with end-edge-cloud computing using NSGA-III," *IEEE Trans. Netw. Sci. Eng.*, vol. 10, no. 5, pp. 3032–3046, Sep.-Oct. 2023.

[11] X. Deng, J. Yin, P. Guan, N. N. Xiong, L. Zhang, and S. Mumtaz, "Intelligent delay-aware partial computing task offloading for multiuser Industrial Internet of Things through edge computing," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 2954–2966, Feb. 2023.

[12] L. Sun, J. Wang, and B. Lin, "Task allocation strategy for MEC-enabled IIoTs via Bayesian network based evolutionary computation," *IEEE Trans. Ind. Informat.*, vol. 17, no. 5, pp. 3441–3449, May 2021.

[13] 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.

[14] H. Wu, K. Wolter, P. Jiao, Y. Deng, Y. Zhao, and M. Xu, "EEDTO: An energy-efficient dynamic task offloading algorithm for blockchain-enabled IoT-edge-cloud orchestrated computing," *IEEE Internet Things J.*, vol. 8, no. 4, pp. 2163–2176, Feb. 2021.

[15] X. Xu, H. Tian, X. Zhang, L. Qi, Q. He, and W. Dou, "DisCOV: Distributed COVID-19 detection on X-ray images with edge-cloud collaboration," *IEEE Trans. Services Comput.*, vol. 15, no. 3, pp. 1206–1219, May/Jun. 2022.

[16] S. Chouikhi, M. Esseghir, and L. Merghem-Boulahia, "Energy-efficient computation offloading based on multiagent deep reinforcement learning for Industrial Internet of Things systems," *IEEE Internet Things J.*, vol. 11, no. 7, pp. 12228–12239, Apr. 2024.

[17] J. Peng, H. Qiu, J. Cai, W. Xu, and J. Wang, "D2D-assisted multi-user cooperative partial offloading, transmission scheduling and computation allocating for MEC," *IEEE Trans. Wireless Commun.*, vol. 20, no. 8, pp. 4858–4873, Aug. 2021.

[18] Y. Ding, K. Li, C. Liu, and K. Li, "A potential game theoretic approach to computation offloading strategy optimization in end-edge-cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 6, pp. 1503–1519, Jun. 2022.

[19] Z. Wang, T. Lv, and Z. Chang, "Computation offloading and resource allocation based on distributed deep learning and software defined mobile edge computing," *Comput. Netw.*, vol. 205, no. 2022, pp. 108732–108744, Jan. 2022.

[20] A. Shakarami, A. Shahidinejad, and M. Ghobaei-Arani, "An autonomous computation offloading strategy in mobile edge computing: A deep learning-based hybrid approach," *J. Netw. Comput. Appl.*, vol. 178, no. 2021, pp. 102974–102992, Jan. 2021.

[21] M. Keshavarznejad, M. Rezvani, and S. Adabi, "Delay-aware optimization of energy consumption for task offloading in fog environments using metaheuristic algorithms," *Clust. Comput.*, vol. 24, no. 2021, pp. 1825–1853, Jan. 2021.

[22] G. Peng, H. Wu, H. Wu, and K. Wolter, "Constrained multiobjective optimization for IoT-enabled computation offloading in collaborative edge and cloud computing," *IEEE Internet Things J.*, vol. 8, no. 17, pp. 13723–13736, Sep. 2021.

[23] Q. Lu, J. Yao, Z. Qi, B. He, and H. Guan, "Fairness-efficiency allocation of CPU-GPU heterogeneous resources," *IEEE Trans. Services Comput.*, vol. 12, no. 3, pp. 474–488, May/Jun. 2019.

[24] M. Karakus, and A. Durresi, "A survey: Control plane scalability issues and approaches in software-defined networking (SDN)," *Comput. Netw.*, vol. 112, pp. 279–293, Jan. 2017.

[25] Y. Gu, C. Yin, Y. Guo, B. Xia, and Z. Chen, "Communication-computation-aware user association in MEC HetNets: A meta-analysis," *IEEE Trans. Wireless Commun.*, vol. 22, no. 12, pp. 8919–8933, Dec. 2023.

[26] Y. Zheng, L. Zou, W. Zhang, J. Yang, L. Yang, and Z. Lin, "Contract-based cooperative computation and communication resources sharing in mobile edge computing," *J. Grid Comput.*, vol. 21, no. 14, pp. 1–19, Feb. 2023.

[27] R. Zhu, N. Lin, V. Dinavahi, and G. Liang, "An accurate and fast method for conducted EMI modeling and simulation of MMC-based HVdc converter station," *IEEE Trans. Power Electron.*, vol. 35, no. 5, pp. 4689–4702, May 2020.

[28] Z. -Y. Chai, Y. -J. Zhao, and Y.-L. Li, "Multitask computation offloading based on evolutionary multiobjective optimization in Industrial Internet of Things," *IEEE Internet Things J.*, vol. 11, no. 9, pp. 15894–15908, May 2024.

[29] H. Yuan, J. Bi, J. Zhang, and M. Zhou, "Energy consumption and performance optimized task scheduling in distributed data centers," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 52, no. 9, pp. 5506–5517, Sep. 2022.

[30] S. Mirjalili, S. Seyedali, S. Mirjalili, and L. Coelho. "Multi-objective grey wolf optimizer: A novel algorithm for multi-criterion optimization," *Expert Syst. Appl.*, vol. 47, no. 2016, pp. 106–119, Apr. 2016.

[31] J. Bi et al., "Multi-swarm genetic gray wolf optimizer with embedded autoencoders for high-dimensional expensive problems," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2023, pp. 7265–7271.

[32] H. Peng, W. Wen, M. Tseng, and L. Li, "Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment," *Appl. Soft Comput.*, vol. 80, no. 2019, pp. 534–545, May 2019.

[33] I. Attiya, M. A. Elaziz, L. Abualigah, T. N. Nguyen, and A. A. A. El-Latif, "An improved hybrid swarm intelligence for scheduling IoT application tasks in the cloud," *IEEE Trans. Ind. Informat.*, vol. 18, no. 9, pp. 6264–6272, Sep. 2022.

[34] A. A. Heidari, I. Aljarah, H. Faris, H. Chen, J. Luo, and S. Mirjalili, "An enhanced associative learning-based exploratory whale optimizer for global optimization," *Neural Comput. Appl.*, vol. 32, no. 9, pp. 5185–5211, May 2020.

[35] L. Li, Q. Lin, K. Li, and Z. Ming, "Vertical distance-based clonal selection mechanism for the multiobjective immune algorithm," *Swarm Evol. Comput.*, vol. 63, pp. 100886–100903, Apr. 2021.

[36] M. Gong, L. Jiao, H. Du, and L. Bo, "Multiobjective immune algorithm with nondominated neighbor-based selection," *Evol. Comput.*, vol. 16, no. 2, pp. 225–255, Jun. 2008.

[37] Y. Lin, A. Barker, and S. Ceesay, "Exploring characteristics of inter-cluster machines and cloud applications on Google clusters," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Atlanta, GA, USA, 2020, pp. 2785–2794.

[38] J. Guo et al., "Who limits the resource efficiency of my datacenter: An analysis of Alibaba datacenter traces," in *Proc. IEEE/ACM 27th Int. Symp. Qual. Service (IWQoS)*, Phoenix, AZ, USA, 2019, pp. 1–10.

[39] J. Zhou and X. Zhang, "Fairness-aware task offloading and resource allocation in cooperative mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 5, pp. 3812–3824, Mar. 2022.

[40] F. Song, H. Xing, S. Luo, D. Zhan, P. Dai, and R. Qu, "A multiobjective computation offloading algorithm for mobile-edge computing," *IEEE Internet Things J.*, vol. 7, no. 9, pp. 8780–8799, Sep. 2020.

[41] T. Yang et al., "Two-stage offloading optimization for energy-latency tradeoff with mobile edge computing in maritime Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 5954–5963, Jul. 2020.

[42] E. Daniel, "Optimum wavelet-based homomorphic medical image fusion using hybrid genetic-grey wolf optimization algorithm," *IEEE Sensors J.*, vol. 18, no. 16, pp. 6804–6811, Aug. 2018.

[43] J. Bi, H. Yuan, J. Zhai, M. Zhou, and H. V. Poor, "Self-adaptive bat algorithm with genetic operations," *IEEE/CAA J. Automatica Sinica*, vol. 9, no. 7, pp. 1284–1294, Jul. 2022.

[44] F. Javidrad and M. Nazari, "A new hybrid particle swarm and simulated annealing stochastic optimization method," *Appl. Soft Comput.*, vol. 60, pp. 634–654, Nov. 2017.

[45] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, Apr. 2002.

[46] S. Mirjalili, P. Jangir, S. Mirjalili, S. Saremi, and I. Trivedi, "Optimization of problems with multiple objectives using the multi-verse optimization algorithm," *Knowl. Based Syst.*, vol. 134, pp. 50–71, Oct. 2017.

[47] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Trans. Evol. Comput.*, vol. 11, no. 6, pp. 712–731, Dec. 2007.

[48] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient large-scale multiobjective optimization based on a competitive swarm optimizer," *IEEE Trans. Cybern.*, vol. 50, no. 8, pp. 3696–3708, Aug. 2020.

[49] Y. Tian, R. Cheng, X. Zhang, F. Cheng, and Y. Jin, "An indicator-based multiobjective evolutionary algorithm with reference point adaptation for better versatility," *IEEE Trans. Evol. Comput.*, vol. 22, no. 4, pp. 609–622, Aug. 2018.

[50] L. Pan, C. He, Y. Tian, H. Wang, X. Zhang, and Y. Jin, "A classification-based surrogate-assisted evolutionary algorithm for expensive many-objective optimization," *IEEE Trans. Evol. Comput.*, vol. 23, no. 1, pp. 74–88, Feb. 2019.

[51] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Trans. Veh. Technol.*, vol. 67, no. 12, pp. 12313–12325, Dec. 2018.

[52] 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.

[53] Y. Wang, M. Sheng, X. Wang, L. Wang, and J. Li, "Mobile-edge computing: Partial computation offloading using dynamic voltage scaling," *IEEE Trans. Commun.*, vol. 64, no. 10, pp. 4268–4282, Oct. 2016.

Jing Bi (Senior Member, IEEE) received the B.S., and Ph.D. degrees in computer science from Northeastern University, Shenyang, China, in 2003 and 2011, respectively.

She is currently a Professor with the Faculty of Information Technology, Beijing University of Technology, Beijing, China. She has over 170 publications in international journals and conference proceedings. Her research interests include distributed computing, cloud and edge computing, large-scale data analytics, machine learning, industrial internet, and performance optimization.

Prof. Bi is now an Associate Editor of IEEE TRANSACTIONS ON SYSTEMS MAN AND CYBERNETICS: SYSTEMS.

Haitao Yuan (Senior Member, IEEE) received the Ph.D. degree in computer engineering from New Jersey Institute of Technology (NJIT), Newark, NJ, USA in 2020.

He is currently the Deputy Director of the Department of Science and Technology Innovation, Wenchang International Aerospace City, Hainan, China. He is currently an Associate Professor with the School of Automation Science and Electrical Engineering, Beihang University, Beijing, China, and he is named in the world's top 2% of Scientists List. His research interests include the Internet of Things, edge computing, deep learning, data-driven optimization, and computational intelligence algorithms.

Dr. Yuan received the Chinese Government Award for Outstanding Self-Financed Students Abroad, the 2021 Hashimoto Prize from NJIT, the Best Paper Award in the 17th ICNSC, and the Best Student Paper Award Nominees in 2024 IEEE SMC. He is an Associate Editor for IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS: SYSTEMS, IEEE INTERNET OF THINGS JOURNAL, and *Expert Systems With Applications*.

Jia Zhang (Senior Member, IEEE) received the Ph.D. degree in computer science from the University of Illinois at Chicago, Chicago, IL, USA, in 2000.

She is currently the Cruse C. and Marjorie F. Calahan Centennial Chair in Engineering, a Professor with the Department of Computer Science in the Lyle School of Engineering, Southern Methodist University, Dallas, TX, USA. Her research interests emphasize the application of machine learning and information retrieval methods to tackle data science infrastructure problems, with a recent focus on scientific workflows, provenance mining, software discovery, knowledge graphs, and interdisciplinary applications of all of these interests in earth science.

Jiahui Zhai (Student Member, IEEE) received the B.E. degree in software engineering from Zhengzhou University, Zhengzhou, China, in 2019, and the M.E. degree in software engineering from Beijing University of Technology, Beijing, China, in 2022. He is currently pursuing the Ph.D. degree with the Faculty of Information Technology, School of Software Engineering, Beijing University of Technology.

He is now a visiting Ph.D. student with the School of Computer Science, University College Dublin, Dublin, Ireland. His research interests include cloud/edge computing, computation offloading, intelligent optimization algorithms, machine learning, and reinforcement learning.

Mr. Zhai was the recipient of the Best Paper Award-Finalist in the 18th IEEE International Conference on Networking, Sensing and Control.

Rajkumar Buyya (Fellow, IEEE) received the B.E and M.E in computer science and engineering from Mysore and Bangalore Universities in 1992 and 1995, respectively, and the Ph.D. in computer science and software engineering from Monash University, Melbourne, VIC, Australia, in 2002.

He is a Redmond Barry Distinguished Professor and the Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Parkville, VIC, Australia. He was a Future Fellow of the Australian Research Council from 2012 to 2016. He has authored over 800 publications and seven textbooks. He is one of the highly cited authors in computer science and software engineering worldwide, with over 156 600 citations and an h-index of 170. He was recognized as a "Web of Science Highly Cited Researcher" from 2016 to 2021 by Thomson Reuters.