Integrated CPU-GPU Task Scheduling for Energy Efficiency and Low Latency in Heterogeneous Industrial IoT Systems

Jiahui Zhai[1], Jing Bi[1], Haitao Yuan[2], Ziqi Wang[1], Jia Zhang[3], and Rajkumar Buyya[4]

[1]College of Computer Science, Beijing University of Technology, Beijing, 100124, China
[2]School of Automation Science and Electrical Engineering, Beihang University, Beijing, 100191, China
[3]Dept. of Computer Science, Southern Methodist University, Dallas, TX 75275, USA
[4]School of Computing and Information Systems, University of Melbourne, Melbourne, VIC 3010, Australia

***Abstract*—The unprecedented prosperity of the Industrial Internet of Things has significantly driven the transition from traditional manufacturing to intelligence. In industrial environments, resource-constrained industrial equipment (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 three-stage heterogeneous computing architecture that accurately models the multi-task processing of both scientific and concurrent workflows in real industrial environments. We formulate a joint optimization problem to minimize task completion time and energy consumption for IEs simultaneously. To solve this problem, we design an Improved Two-stage Multi-Objective Evolutionary Algorithm (IT-MOEA). IT-MOEA employs a novel multi-objective 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 multi-objective algorithms by 39.45% in terms of delay and energy consumption.**

***Index Terms*—Industrial Internet of Things, mobile edge computing, task offloading, multi-objective optimization, evolutionary algorithms.**

I. Introduction

With the rapid development of wireless communication and the Internet of Things (IoT), Industrial IoT (IIoT) is transforming manufacturing by enhancing efficiency through various types of industrial equipment (IE) such as robots, vehicles, and sensors [1]. However, limited computational power and battery capacity often prevent IEs from running compute-intensive, latency-critical tasks independently. Offloading these tasks to resource-rich computing nodes can reduce both latency and energy consumption [2]. Traditionally, cloud data centers (CDCs) handle such workloads, while distance-induced delays are increasingly prohibitive for large-scale IIoT. Mobile edge computing (MEC) addresses this issue by placing resources closer to IEs, thereby mitigating latency and energy overhead [3]. Nonetheless, the complexity and diversity of industrial tasks demand an offloading strategy that effectively balances delay and energy metrics, ensuring optimal performance in evolving industrial environments. The increasing complexity of IIoT tasks necessitates advanced heterogeneous computing architectures, as traditional CPU-centric methods struggle with parallel processing demands [4]. Modern systems increasingly integrate hybrid platforms like CPU+GPU or CPU+ASIC configurations to optimize subtask execution. Although GPUs demonstrate significant parallel acceleration capabilities [5], prevailing offloading research predominantly targets isolated CPU or GPU resources [2], [3], overlooking synergistic heterogeneous resource coordination.

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 multi-objective task offloading problem in industrial environments is a typical mixed-integer nonlinear program (MINLP) [6], which is NP-hard and complicates the development of efficient and scalable algorithms. This work proposes an improved multi-objective evolutionary algorithm (MOEA) that integrates various advanced optimization techniques to address this challenge. Primary contributions can be summarized as:

1) We propose a novel hybrid equipment-edge-cloud architecture with multiple IEs, access points (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

This work was supported by the National Natural Science Foundation of China under Grants 62173013 and 62473014; in part by the Beijing Natural Science Foundation under Grants L233005 and 4232049; in part by 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).

solve the MINLP problem. It jointly optimizes task allocation across IEs, APs, and a CDC, the association between IEs and APs, 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, extensive experiments demonstrate that IT-MOEA outperforms several state-of-the-art peers regarding convergence and distribution performance.

II. Problem Formulation

A. System Model

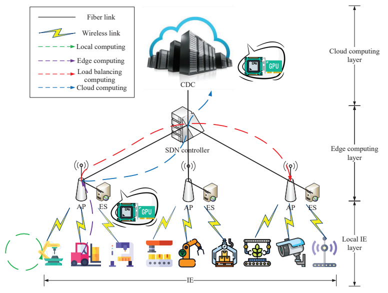


Fig. 1. Framework of an industrial environment.

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.

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 edge servers (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 in IE, ESs, and CDC, respectively. Thus, $\acute{\lambda}_m^k+\dot{\lambda}_m^k+\grave{\lambda}_m^k=1$. 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 $\sum_{j=1}^{J}x_{m,j}=1$.

B. Task Model

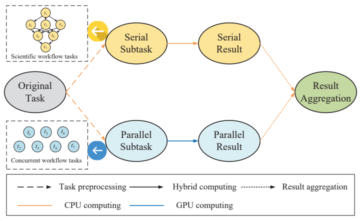


Fig. 2. Three-stage heterogeneous computing model of tasks.

This work focuses on both serial and parallel tasks in IIoT applications. CPUs are more efficient for serial tasks, whereas GPUs excel at parallel workloads [5]. Accordingly, we propose a three-stage task model, *i.e.,* task preprocessing, hybrid computing, and results aggregation depicted in Fig. 2. Building on typical GPGPU frameworks (*e.g.*, CUDA and OpenCL [5]), the CPU first preprocesses tasks and dispatches parallelizable components to GPUs. In the hybrid computing stage, CPUs and GPUs handle their respective subtasks concurrently, producing separate outputs. Finally, CPU aggregates these outputs to generate the final result. Although serial and parallel subtasks run simultaneously, the three stages themselves proceed in sequence. Let the task $t_m^k\triangleq\{I_m^k,\varpi_m^k,q_m^k,o_m^k\}$ generated by IE m is characterized by: data size I_m^k (bits), required computation ϖ_m^k (FLOPs), hybrid computation ratio q_m^k, and parallel workload ratio o_m^k.

C. Communication Model

This work employs an orthogonal frequency-division multiple access communication model between various IEs and their associated APs. According to [3], the path loss between IE m and its corresponding AP j ($1\leq j\leq J$) is given by $(d_{m,j})^{-\varsigma}$, where $d_{m,j}$ represents the distance from IE m to AP j, and ς represents the path loss exponent. The uplink rate between IE m and AP j is denoted by $R_{m,j}$. Hence, we have $R_{m,j}=W_{m,j}\log_2(1+\frac{p_m^o\varrho(d_{m,j})^{-\varsigma}|h|^2\zeta}{\sigma^2})$, 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. The uplink channel fading coefficient is h. σ^2 represents the power parameter of Gaussian white noise. ϱ and ζ are the path-loss coefficients and log-normal shadowing.

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 $\acute{\lambda}_m^k=1$, 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 t_m^k in the local computing, respectively. Thus, $\acute{\tau}_m^{k,1}$ is expressed as $\acute{\tau}_m^{k,1}=\frac{(1-q_m^k)\varpi_m^k}{\omega_m^{C,k}\acute{f}^C}$.

Let $\acute{\tau}_m^{k,C}$ and $\acute{\tau}_m^{k,G}$ denote the delays for the hybrid computation stage of t_m^k, respectively, which are expressed as $\acute{\tau}_m^{k,C}=\frac{q_m^k(1-o_m^k)\varpi_m^k}{\omega_m^{C,k}\acute{f}^C}$ and $\acute{\tau}_m^{k,G}=\frac{q_m^k o_m^k\varpi_m^k}{\omega_m^{G,k}\acute{f}^G}$. Thus, the total delay consumed during the hybrid computation stage, $\acute{\tau}_m^{k,2}$, is given as $\acute{\tau}_m^{k,2}=\max(\acute{\tau}_m^{k,C},\acute{\tau}_m^{k,G})$. Let $\acute{\tau}_m^k$ denote the total delay for local computing of t_m^k, which is obtained as $\acute{\tau}_m^k=\acute{\tau}_m^{k,1}+\acute{\tau}_m^{k,2}$. Let $\acute{e}_m^k$ denote the total energy consumption of local computing of t_m^k, which is expressed by $\acute{e}_m^k=\acute{\tau}_m^k(\acute{p}^{C,i}+\acute{p}^{G,i})+(\acute{\tau}_m^{k,1}+\acute{\tau}_m^{k,C})\acute{p}^{C,d}+\acute{\tau}_m^{k,G}\acute{p}^{G,d}$.

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

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

The computing 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 $\dot{\tau}_{m,j}^{k,1}=\frac{(1-q_m^k)\varpi_m^k}{\omega_{m,j}^{C,k}\dot{f}^C}$ and $\dot{\tau}_{m,j}^{k,2}=max(\frac{q_m^k(1-o_m^k)\varpi_m^k}{\omega_{m,j}^{C,k}\dot{f}^C},\frac{q_m^k o_m^k\varpi_m^k}{\omega_{m,j}^{G,k}\dot{f}^G})$, where $\omega_{m,j}^{C,k}$ and $\omega_{m,j}^{G,k}$ are the numbers of CPU and GPU resources allocated to 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 $\dot{\tau}_{m,j}^k=\dot{\tau}_{m,j}^{k,u}+\dot{\tau}_{m,j}^{k,1}+\dot{\tau}_{m,j}^{k,2}$.

Meanwhile, $\dot{e}_{m,j}^k$ denotes overall energy consumed by task t_m^k associated with AP j, which is computed as $\dot{e}_{m,j}^k=p_m^o\dot{\tau}_{m,j}^{k,u}+\dot{\tau}_{m,j}^k(\acute{p}^{C,i}+\acute{p}^{G,i})$.

3) Cloud computing: If the computational 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 $\grave{\tau}_{m,j}^{k,c}=\frac{I_m^k}{W_c}$, where W_c denotes the SDN controller to CDC transmission rate.

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

According to $\dot{\tau}_{m,j}^{k,u}$ and $\grave{\tau}_{m,j}^{k,c}$, the total delay of t_m^k and energy consumption of the IE during the offloading process at CDC are computed as $\grave{\tau}_{m,j}^{k,2}=max(\frac{q_m^k(1-o_m^k)\varpi_m^k}{\omega_{m,j}^{C,k}\grave{f}^C},\frac{q_m^k o_m^k\varpi_m^k}{\omega_{m,j}^{G,k}\grave{f}^G})$ and $\grave{e}_{m,j}^k=p_m^o\grave{\tau}_{m,j}^{k,u}+\grave{\tau}_{m,j}^k(\acute{p}^{C,i}+\acute{p}^{G,i})$.

In summary, the average delay of K tasks and the energy consumption of M IEs in the equipment-edge-cloud system are obtained as $\boldsymbol{T}=\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\{\acute{\lambda}_m^k\acute{\tau}_m^k+\sum_{j=1}^{J}(\dot{\lambda}_m^k\dot{\tau}_{m,j}^k+\grave{\lambda}_m^k\grave{\tau}_{m,j}^k)\}$ and $\boldsymbol{E}=\frac{1}{M}\sum_{m=1}^{M}\sum_{k=1}^{K}\{\acute{\lambda}_m^k\acute{e}_m^k+\sum_{j=1}^{J}(\dot{\lambda}_m^k\dot{e}_{m,j}^k+\grave{\lambda}_m^k\grave{e}_{m,j}^k)\}$.

E. Problem Formulation

The formulated problem is given as follows. In (1), $\boldsymbol{\hbar}$ 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.*,

$$\arg\min_{\hbar}\{\boldsymbol{T},\boldsymbol{E}\} \tag{1}$$

III. Proposed Framework

This section proposes IT-MOEA to solve the problem in (1). The framework of IT-MOEA and details are presented.

A. Framework of IT-MOEA

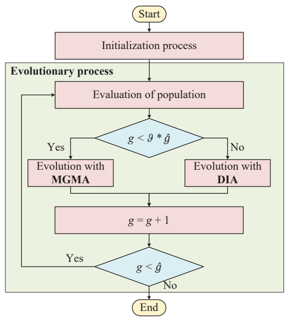


Fig. 3. Main flow of IT-MOEA.

IT-MOEA framework in Fig. 3 consists of initialization and evolutionary phases. Following initialization as described in Section III-B, the evolutionary phase is divided into two stages with the threshold ϑ to enhance convergence and diversity. The first stage utilizes the novel MGMA method from Section III-C to accelerate convergence and improve optimization, while the second stage employs DIA from Section III-D to enhance population distribution. The evolutionary loop concludes after $\hat{g}$ iterations, resulting in the final IT-MOEA population.

B. Population Initialization

Typically, individuals are randomly initialized without prior information, resulting in uneven distributions and slow convergence. The tent map offers more uniform coverage and faster searches, but has small cycles and unstable periodic points. To overcome these limitations, we propose an improved tent chaotic map $x_{g+1}=\begin{cases} 2x_g+\mathrm{rand}(0,1)\times\frac{1}{N}, 0\leq x\leq\frac{1}{2} \\ 2(1-x_g)+\mathrm{rand}(0,1)\times\frac{1}{N}, \frac{1}{2}<x\leq 1 \end{cases}$. The transformed expression is given as $x_{g+1}=(2x_g) \bmod 1+\mathrm{rand}(0,1)\times\frac{1}{N}$, 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.

C. Proposed MGMA

Multi-objective grey wolf optimizer (MOGWO) preserves GWO's population updating, mimicking grey wolves' hierarchy and hunting behavior for fast convergence, high efficiency, and precision. However, MOGWO is prone to premature convergence and local optima [7]. To address these issues, we propose an improved MOGWO with the position update mechanism of the three best wolves, the population update mechanism, and the archive update mechanism.

1) 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. To address this issue, separate update strategies are proposed for α, β, and δ. δ accepts leadership from α and β. Its update method are given as $\boldsymbol{X}_\delta(g+1)=D_\delta-\boldsymbol{X}_\delta(g)$ and $D_\delta=\rho\boldsymbol{X}_\delta(g)+(1-\rho)\boldsymbol{X}_\alpha(g)+(1-\rho)\boldsymbol{X}_\beta(g)$, 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) to approach α in a spiral motion. Its update method is given as $\boldsymbol{X}_\beta(g+1) = |\boldsymbol{X}_\alpha(g)-\boldsymbol{X}_\beta(g)|\, e^{b\dot{l}}\cos(2\pi\dot{l}) + \rho\boldsymbol{X}_\alpha(g)$, 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.

To update α, we introduce random walks using the Lévy flight mechanism, which balances local exploration with occasional long jumps. Short-distance exploration enhances optimization speed and accuracy by effectively searching around α's current position, while long jumps expand the search area. To address the stochastic nature of Lévy flights, we incorporate a greedy selection strategy for a survival-of-the-fittest approach. The position update of α using the Lévy flight mechanism is given as $\boldsymbol{X}_\alpha^{'}(g+1)=\boldsymbol{X}_\alpha(g)+\xi\oplus\text{Lévy}(\hbar)$, 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 Lévy$(\hbar)=\frac{\mu}{|\nu|^{\frac{1}{\kappa}}}$, where κ=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 $\sigma_\mu^2=\{\frac{\Gamma(1+\kappa)}{\frac{\kappa\Gamma(1+\kappa)}{2}}\frac{\sin(\frac{\pi\kappa}{2})}{2^{\frac{\kappa-1}{2}}}^{\frac{2}{\kappa}}\}$ and σ_ν^2=1, where $\Gamma(\cdot)$ represents the Gamma function. The position update of α is expressed as $\boldsymbol{X}_\alpha(g+1)=\begin{cases} \boldsymbol{X}_\alpha^{'}(g+1),\ \text{other} \\ \boldsymbol{X}_\alpha(g), f(\boldsymbol{X}_\alpha^{'}(g+1)>\boldsymbol{X}_\alpha(g))\ \text{and rand}<\dot{p} \end{cases}$, 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.

2) 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). The first half of (2) provides rapid convergence to the optimal solution, while the latter provides higher population diversity to prevent premature convergence. Thus,

$$\begin{aligned}\boldsymbol{X}(g+1)=&\frac{w(\boldsymbol{X}_1(g)+\boldsymbol{X}_2(g)+\boldsymbol{X}_3(g))}{3}+a(1-w)\\ &(r(\boldsymbol{X}_r(g)-\boldsymbol{X}(g))+a\dot{u}r(\boldsymbol{X}_\alpha(g)-\boldsymbol{X}(g)))\end{aligned} \tag{2}$$

where $w=\frac{(\hat{w}-\check{w})g}{\hat{g}}+\check{w}$, $\dot{u}=2e^{r\iota}\sin(2\pi r)$, and $\iota=\frac{\hat{g}-g+1}{\hat{g}}$. 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.

3) Archive update mechanism: 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. Hence, this work introduces associative learning to update some individuals in the archive. Thus,

$$\begin{aligned}\boldsymbol{X}(g+1)=&\boldsymbol{X}(g)+0.001G(\boldsymbol{X}(g)-\check{\varkappa},\hat{\varkappa}-\boldsymbol{X}(g))\\ &+b_0S_1r_1(\boldsymbol{X}_r(g)-\boldsymbol{X}(g))+b_0S_2r_2(\boldsymbol{X}_\alpha(g)-\boldsymbol{X}(g))\end{aligned} \tag{3}$$

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 $S_1=(1-\frac{g}{\hat{g}})$ and $S_2=\frac{2g}{\hat{g}}$.

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 [8] between the individual and its corresponding weight vector, *i.e.*, $V_i=\left\|F(\boldsymbol{X}_i)-(Z^*+d(\boldsymbol{X}_i,\boldsymbol{\lambda}_i,Z^*)\frac{\boldsymbol{\lambda}_i}{\|\boldsymbol{\lambda}_i\|})\right\|$

and $d(\boldsymbol{X}_i, \boldsymbol{\lambda}_i, Z^*) = \frac{\|(F(\boldsymbol{X}_i) - Z^*)\boldsymbol{\lambda}_i\|}{\|\boldsymbol{\lambda}_i\|}$, where $d(\boldsymbol{X}_i, \boldsymbol{\lambda}_i, Z^*)$ denotes the projection of vector $F(\boldsymbol{X}_i) - Z^*$ on the weight vector $\boldsymbol{\lambda}_i$ $(1 \leq i \leq N)$. $F(\cdot)$ 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 $c_i = \left\lceil \frac{N(1-V_i)}{\sum_i^N (1-V_i)} \right\rceil$.

Note that smaller vertical distance values imply that the individual is closer to its weight vector and performs better about diversity. Then, each individual performs the proportional cloning operator, defined as $\tilde{\boldsymbol{X}} = \bigcup_{i=1}^{N} \{c_i \otimes \boldsymbol{X}_i\}$, 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.

Algorithm 1: IT-MOEA

```
   Input: Maximum iteration number ($\hat{g}$), population size ($N$),
          objective function ($F$)
   Output: Final population ($\mathbb{P}$)
   /* Initialization process                                        */
1  Initialize parameters of MGMA and DIA;
2  Initialize positions of individuals to obtain $\mathbb{P}$;
   /* Evolutionary process                                          */
   /* MGMA                                                          */
3  Initialize $a$, $A$, and $C$;
4  Evaluate $F(\boldsymbol{X}_i)$ of each $\boldsymbol{X}_i$ to perform non-dominated sorting on
     $\mathbb{P}$ to establish the archive;
5  for $g \leftarrow 1$ to $\vartheta \times \hat{g}$ do
6      for $i \leftarrow 1$ to $N$ do
7          Select $\alpha$, $\beta$, and $\delta$ with roulette wheel selection in the
             archive;
8          Update $\alpha$, $\beta$, and $\delta$;
9          Update positions of $\boldsymbol{X}_i$ in the $\mathbb{P}$ with (2);
10     end
11     Update $a$, $A$, and $C$;
12     Evaluate $F(\boldsymbol{X}_i)$ of each $\boldsymbol{X}_i$ to perform non-dominated sorting
         on $\mathbb{P}$ to update the archive;
13     Perform (3) for updating the archive;
14     if the archive is full then
15         Calculate the crowding density and remove individuals
             from the most crowded regions;
16     end
17 end
   /* DIA                                                           */
18 for $g \leftarrow \vartheta \times \hat{g}$ to $\hat{g}$ do
19     for $i \leftarrow 1$ to $N$ do
20         Calculate the vertical distance $V_i$ for $\boldsymbol{X}_i$;
21         Calculate the cloning number $c_i$ for $\boldsymbol{X}_i$;
22     end
23     Perform cloning operator on $\mathbb{P}$ to generate offspring $\tilde{\boldsymbol{X}}$;
24     Add all cloning offspring $\tilde{\boldsymbol{X}}$ into $\mathbb{P}$;
25 end
```

IV. Performance Evaluation

This work simulates the experiments in a 1000 m$\times$1000 m area with MATLAB, where a cloud server, IEs, and APs are deployed in a grid network. Three experimental scales with different M, J, and K are given in Table I.

A. Experimental Setup

1) Parameter settings: 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$\times$10^9 FLOPS, $\dot{f}^C$=1$\times$10^{10} FLOPS, $\grave{f}^C$=1$\times$10^{11} FLOPS, $\acute{f}^G$=2$\times$10^7 FLOPS, $\dot{f}^G$=5$\times$10^8 FLOPS, and $\grave{f}^G$=3$\times$10^9 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 t_m^k, I_m^k=$[25, 50]$ MB, ϖ_m^k=$[2.5\times10^8, 5\times10^8]$ FLOPS, 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, ς=4, h=0.98, σ=1.6$\times$10^{-11}, $W_{m,j}$=$[2, 4]$ MHz, ϱ=1, ζ=1, W_c=256 Mbps, and δ=1. 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.

TABLE I
SIX TEST INSTANCES.

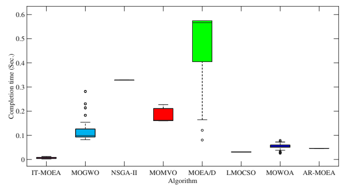
Instances	M	J	K
1	1–5	1–3	1–4
2	6–10	4–6	5–8
3	11–15	7–9	9–12

2) Compared algorithms: To comprehensively evaluate the performance of IT-MOEA, several state-of-the-art (SOTA) algorithms are employed for comparisons, including seven MOEAs (MOGWO, NSGA-II, MOMVO, MOEA/D, LMOCSO [9], MOWOA, and AR-MOEA [10]) with three test instances.

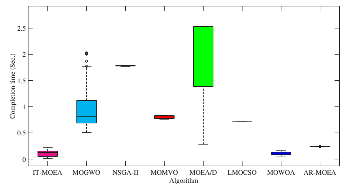
B. Experimental Results

This section discusses the experimental results by comparing IT-MOEA with several SOTA MOEAs on three scales. 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 three test instances. Figs. 4 and 5 present the box plots of eight algorithms concerning completion time and energy consumption, respectively. In Figs. 4–5, 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 position update, the population update, and the archive update mechanisms.

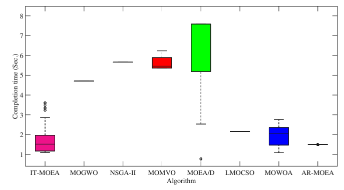
Fig. 6 shows the Pareto fronts of eight algorithms, with IT-MOEA outperforming others in three test instances. Although AR-MOEA and LMOCSO achieve comparable Pareto sets, their convergence and distribution are weaker. IT-MOEA effectively balances completion time and energy consumption while maintaining superior diversity across all instances. In larger-scale experiments, other algorithms struggle with convergence and distribution, but IT-MOEA remains robust, demonstrating clear convergence advantages. Its stability and performance in realistic industrial environments are due to features such as a high-quality initial population, avoidance of local optima, global exploration, and distributed enhancement.



(a) Instance 1

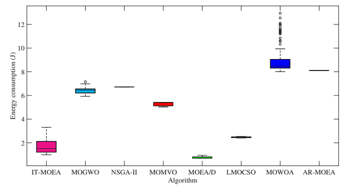


(b) Instance 2

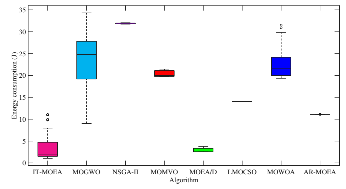


(c) Instance 3

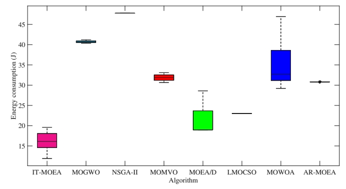
Fig. 4. Box plots of eight algorithms regarding completion time.



(a) Instance 1

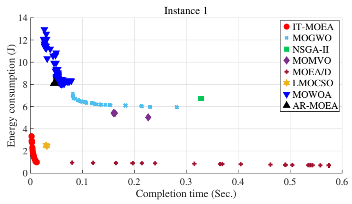


(b) Instance 2

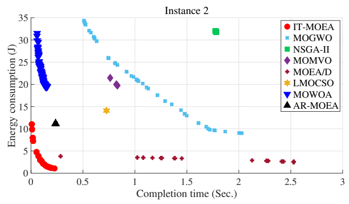


(c) Instance 3

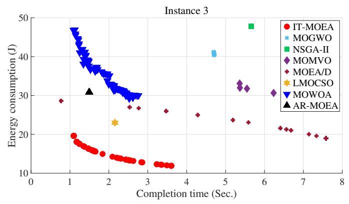
Fig. 5. Box plots of eight algorithms regarding energy consumption.



(a) Instance 1



(b) Instance 2



(c) Instance 3

Fig. 6. Pareto-optimal fronts of eight algorithms for three test instances.

V. Conclusion

In the industrial Internet, joint optimization of application completion time and equipment energy consumption remains challenging due to heterogeneous task requirements and multiple access-point (AP) coordination. Existing task offloading methods often prioritize single objectives or neglect task heterogeneity in hybrid CPU-GPU environments. We propose a three-stage computing framework for scientific and concurrent workflows, incorporating offloading decisions and task priorities. An Improved Two-stage Multi-Objective Evolutionary Algorithm (IT-MOEA) simultaneously minimizes time and energy consumption, demonstrating superior convergence and Pareto front distribution in experiments.

References

[1] A. Jayanetti, S. Halgamuge, and R. Buyya, "Multi-Agent Deep Reinforcement Learning Framework for Renewable Energy-Aware Workflow Scheduling on Distributed Cloud Data Centers," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 4, pp. 604–615, Apr. 2024.

[2] J. Bi, Z. Wang, H. Yuan, J. Zhang, and M. Zhou, "Cost-Minimized Computation Offloading and User Association in Hybrid Cloud and Edge Computing," *IEEE Internet of Things Journal*, vol. 11, no. 9, pp. 16672–16683, May 2024.

[3] J. Zhai, J. Bi, H. Yuan, M. Wang, J. Zhang, Y. Wang, and M. Zhou, "Cost-Minimized Microservice Migration With Autoencoder-Assisted Evolution in Hybrid Cloud and Edge Computing Systems," *IEEE Internet of Things Journal*, vol. 11, no. 24, pp. 40951–40967, 15 Dec.15, 2024.

[4] Q. Zeng, Y. Du, K. Huang, and K. K. Leung, "Energy-Efficient Resource Management for Federated Edge Learning With CPU-GPU Heterogeneous Computing," *IEEE Transactions on Wireless Communications*, vol. 20, no. 12, pp. 7947–7962, Dec. 2021.

[5] 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 Transactions on Parallel and Distributed Systems*, vol. 29, no. 6, pp. 1275–1288, Jun. 2018.

[6] H. Yuan, M. Wang, J. Bi, S. Shi, J.Yang, Jia Zhang, M. Zhou, and R. Buyya "Cost-Efficient Task Offloading in Mobile Edge Computing With Layered Unmanned Aerial Vehicles," *IEEE Internet of Things Journal*, vol. 11, no. 19, pp. 30496-30509, 1 Oct.1, 2024.

[7] J. Bi, J. Zhai, H. Yuan, Z. Wang, J. Qiao, J. Zhang, and M. Zhou, "Multi-swarm Genetic Gray Wolf Optimizer with Embedded Autoencoders for High-dimensional Expensive Problems," *2023 IEEE International Conference on Robotics and Automation*, 2023, London, United Kingdom, pp. 7265–7271.

[8] L. Li, Q. Lin, K. Li, and Z. Ming, "Vertical Distance-based Clonal Selection Mechanism for the Multiobjective Immune Algorithm," *Swarm and Evolutionary Computation*, vol. 63, no. 2021, pp. 100886–100903, Apr. 2021.

[9] Y. Tian, X. Zheng, X. Zhang, and Y. Jin, "Efficient Large-Scale Multiobjective Optimization Based on a Competitive Swarm Optimizer," *IEEE Transactions on Cybernetics*, vol. 50, no. 8, pp. 3696–3708, Aug. 2020.

[10] 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 Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 609–622, Aug. 2018.