Hierarchical Evolutionary Multi-Agent Collaboration for Task Offloading Optimization in Computing Power Networks

Qunjian Chen, Chen Yang, *Member, IEEE*, Shulin Lan, *Member, IEEE*, Zexuan Zhu, *Senior Member, IEEE*, and Liehuang Zhu, *Senior Member, IEEE*

***Abstract*—Existing decision-making methods for task offloading optimization in computing power networks are typically implemented in a centralized scheduling framework. However, as the scale of the offloading scheduling problem increases, their search efficiency may quickly degrade due to the high-dimensional search space. The offloading scheduling system overly relies on a single control center, which makes it difficult to flexibly develop scheduling plans. Deploying autonomous agents in computing power networks to automatically perceive the network environment and collaboratively make intelligent decisions can effectively optimize computational task offloading and computing resource scheduling. In this paper, we present an SDN-based hierarchical scheduling framework for computing power networks to improve the scheduling performance of the system through the collaboration of local and global agents. To jointly optimize computational task offloading and computing resource scheduling, we propose a distributed multi-objective optimization algorithm with a two-stage search, designed to fit seamlessly into the hierarchical scheduling framework. The first phase enables rapid local optimization for different subproblems using local agents, while the second stage uses global agent to sample and reuse promising candidate solutions found by local agents to achieve collaborative global optimization. Experimental results demonstrate that the proposed approach effectively optimizes processing delay and energy consumption, improving the scalability of the system.**

***Index Terms*—Computing power networks, task offloading, resource scheduling, collaborative control, multi-objective optimization.**

I. Introduction

EMERGING mobile applications, such as speech recognition, image processing, online gaming, and assisted driving, have urgent needs to be deployed in end devices. Typically, resource-limited end devices cannot support the rapid response and effective execution of compute-intensive mobile applications [1], [2]. One of the promising ways is task offloading enabled by mobile edge computing (MEC), which delivers a faster response time and a better user experience [3], [4]. Hou et al. [5] considered mobile vehicles and fixed roadside units as offloading nodes in an SDN-enabled Internet of Vehicles architecture. Zhai et al. [6] proposed a fog computing-based Internet of Vehicles to investigate energy-aware task offloading. Lee et al. [7] designed a practical framework for task offloading between mobile core networks and broadband IP networks. However, the explosive growth of end devices and compute-intensive mobile applications renders MEC prone to computing resource scarcity and inefficient resource scheduling.

By integrating networking and computing, computing power networks (CPNs) have emerged as a transformative network model, enabling elastic integration and flexible scheduling of computing power for task offloading. Lei et al. [8] proposed an IP extension-based network and computing inter-working architecture, which combines user needs and network context to generate optimal resource allocation in CPNs. Considering the tidal characteristics of computing power nodes in CPNs, Pang et al. [9] proposed a classification method for tidal and non-tidal nodes to reduce overall network energy consumption. Sun et al. [10] suggested coordinating networking and computing resources of heterogeneous nodes for specific computational tasks in wireless CPNs by jointly optimizing the computing and learning policies of individual nodes to minimize total energy consumption. However, existing decision-making methods in CPNs are typically implemented in a centralized scheduling framework. As the scale of the offloading scheduling problem increases, their scheduling performance may quickly degrade due to the high-dimensional search space. The offloading scheduling system relies heavily on a single control center, which often struggle with scalability.

Enhancing scheduling performance through agentic AI has emerged as one of the most effective solutions. Deploying autonomous agents capable of perceiving the network environment and collaborating to make intelligent decisions in the offloading scheduling system enables the automatic search for computation task offloading and computing resource scheduling schemes, which can effectively improve resource utilization and reduce overall overhead. Leveraging these strengths,

This work was supported by the National Natural Science Foundation of China under Grant 62472035, U24B20148, 72192843, and 72192844; in part by the Fundamental Research Funds for the Central Universities under Grant 2023CX01020 and State Key Laboratory of Intelligent Manufacturing Equipment and Technology IMETKF2026008. *(Corresponding author: Chen Yang and Shulin Lan).*

Qunjian Chen is with the School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China (Email: ChenQJ@outlook.com).

Chen Yang and Liehuang Zhu is with the School of Cyberspace Science and Technology, Beijing Institute of Technology, Beijing 100081, China (Email: yangchen666@bit.edu.cn; liehuangz@bit.edu.cn).

Shulin Lan is with the School of Economics and Management, University of Chinese Academy of Sciences and National Key Laboratory of Intelligent Manufacturing Equipment and Technology, Beijing 100081, China (Email:lanshulin@ucas.ac.cn).

Zexuan Zhu is with the National Engineering Laboratory for Big Data System Computing Technology, Shenzhen University, Shenzhen 518060, China (e-mail: zhuzx@szu.edu.cn).

related methods have achieved notable progress across diverse application scenarios. Seid et al. [11] proposed a dynamic resource allocation framework that employs multiple agents to allocate resources to user equipment with optimal costs in a multi-UAV-enabled 5G radio access network. Yang et al. [12] introduced a framework that combines the advantages of large language models and classical control algorithms, enabling efficient task coordination and execution in heterogeneous multi-agent systems. Wang et al. [13] designed a UAV-aided MEC framework, in which the trajectory of each UAV is independently managed by a multi-agent based trajectory control algorithm. Although decentralized decision-making methods can achieve shorter response times through smaller decision spaces, they often result in suboptimal scheduling plans. In addition, Li et al. [14] proposed a distributed multi-objective optimization algorithm for network resource allocation of multi-agent systems, which focuses on searching for the local optimal solutions that satisfy the global constraints depending on the preference index, without considering the allocation of computing resources among the multiple agents. Zhou et al. [15] proposed a distributed multi-objective evolutionary algorithm to search the task allocation for multi-agent systems, where the evaluation operation is performed by segmenting and reorganizing the chromosomes, resulting in frequent communication between master and slave nodes. Therefore, it is crucial to introduce both local and global agents in CPNs while exploring a hierarchical scheduling framework to flexibly make decisions through the collaboration of local and global agents.

Most offloading scheduling problems typically involve searching for multiple continuous and discrete variables, whose complexity increases with the dimensions. This kind of problem is essentially a multi-objective optimization problem, subject to many constraints, and has been proven to be NP-hard. Evolutionary algorithm is an prominent optimization approach for dealing with the above problems. Song et al. [16] introduced task precedence as a new constraint for multi-objective computation offloading problems and used a problem-specific population initialization scheme and a dynamic voltage and frequency scaling-based energy conservation scheme to design an improved multi-objective evolutionary algorithm. Peng et al. [17] modeled the computation offloading process in cooperative edge and cloud computing networks as a constrained multi-objective problem and introduced a push-and-pull search framework into three multi-objective evolutionary algorithms to enhance the global search ability. Wang et al. [18] proposed an improved decomposition-based multi-objective evolutionary algorithm for computation offloading in MEC networks, introducing the cost of mobile users purchasing computing resources as a new optimization objective. However, these algorithms lack direct applicability to the hierarchical scheduling framework. Although Jian et al. [19] considered the computation offloading problem in a hierarchical MEC system, they proposed a centralized scheduling algorithm. Zhang et al. [20] proposed a hierarchical scheduling algorithm to solve the caching and resource allocation problem in a cooperative MEC system, but both the upper and lower layers of the algorithm are deployed in a single control center. Therefore, it is necessary to redesign an evolutionary algorithm deployed on local and global agents within the hierarchical scheduling framework, leveraging the collaboration between local and global agents for two-stage search, thereby efficiently solving offloading scheduling problems in CPNs.

Unlike previous studies, we combine a hierarchical scheduling framework with coordinated local-global agents and a distributed multi-objective optimization algorithm featuring an efficient two-stage search to provide an effective solution for task offloading and resource scheduling in CPNs. The main contributions of this paper can be summarized as follows.

1) We propose an SDN-based hierarchical scheduling framework in CPNs, which uses a global agent and multiple local agents to collaboratively develop scheduling plans, improving the scheduling performance of the system.
2) We design a distributed multi-objective evolutionary algorithm (DMOEA) for the hierarchical scheduling framework, which uses a two-stage search to transfer optimization information, avoiding frequent interactions between local and global agents.
3) The DMOEA fits seamlessly into the hierarchical scheduling framework, balancing the search capabilities of global and local agents by reasonably allocating fitness evaluation resources and enhancing the search efficiency of global agents through reuse of optimization information from local agents.

The rest of this article is organized as follows. Section II briefly introduces the concept of multi-objective optimization. Section III provides a comprehensive description of the system model. In Section IV, the proposed algorithm is introduced in detail. The experimental results and performance evaluation are presented in Section V. Section VI presents the conclusion of this article.

II. Preliminary

Many real-world problems not only involve multiple conflicting optimization objectives but also need to meet various constraints, which can usually be defined as constrained multi-objective optimization problems (CMOPs) [21]. Without loss of generality, suppose there is a minimization problem with n decision variables and m objective functions, a typical CMOP can be formalized as follows:

$$\min \quad F(x) = (f_1(x), f_2(x), ..., f_m(x))^T \tag{1}$$

$$s.t. \quad x = (x_1, x_2, ..., x_n) \in X \tag{2}$$

$$g_i(x) \leq 0, i = 1, ..., p \tag{3}$$

$$h_j(x) = 0, j = 1, ..., q \tag{4}$$

where x is a solution in the decision space X, $F(x) \subseteq \mathbb{R}^m$ is the corresponding objective vector. $g_i(x) \leq 0$ represents the inequality constraint and $h_j(x) = 0$ represents the equality constraint. The number of inequality constraints and equality constraints are denoted as p and q, respectively.

The set of all solutions in the decision space includes feasible solutions and infeasible solutions. To evaluate whether

a solution satisfies the constraints, it is usually necessary to calculate its value of total constraint violation.

First, the equality constraints need to be converted into inequality constraints through a relaxation operation with a small positive value δ, which can be expressed as:

$$|h_j(x)| - \delta \leq 0, j = 1, ..., q \tag{5}$$

Next, the value of total constraint violation of a solution x can be calculated as follows:

$$CV(x) = \sum_{i=1}^{p} \max\{0, g_i(x)\} + \sum_{j=1}^{q} \max\{0, |h_j(x)| - \delta\} \tag{6}$$

where $CV(x) = 0$ denotes that x is a feasible solution.

For two given candidate solutions x_1 and x_2, the candidate solution x_1 is said to be Pareto dominance candidate solution x_2, denoted by $x_1 \prec x_2$, if $F(x_1)$ is not greater than $F(x_2)$ in any objective value and $F(x_1)$ has at least one objective value less than $F(x_2)$. A candidate feasible solution x^* is said to be Pareto optimal solution if there are no other candidate feasible solutions in the decision space X dominating solution x^*. The set of all Pareto optimal solutions is called the Pareto optimal set (PS) [22], and their corresponding objective vectors in the objective space constitute the Pareto front (PF).

III. Task Offloading and Resource Scheduling

In this section, we elaborate on the proposed scheduling framework and system model in detail, as shown in Fig. 1. The offloading scheduling system comprises a device layer and an edge layer, where local agents are deployed at the device layer and global agents at the edge layer. Task offloading and resource scheduling are achieved through the coordination between local and global agents.

A. Scheduling Framework

In the device layer, user devices with different computing capacities are wirelessly connected to small eNodeBs (SeNBs). Each SeNB forms a small cell and serves a set of user devices as an access point. In the edge layer, there is a macro eNodeB (MeNB) that connects all the SeNBs via wired links. A computing power pool composed of multiple MEC servers capable of executing multiple computational tasks in parallel is deployed at the MeNB. User devices in the system can either process their computational tasks locally or offload their computational tasks to the MEC servers for execution. SDN is introduced into CPNs to coordinate and orchestrate heterogeneous network resources [23], which separates the system into control and data planes, in which the control plane is in charge of the collection of status information and the execution of optimization algorithms.

In contrast to the scheduling framework using centralized management reported in [24]–[26], we introduce a hierarchical scheduling framework. In this framework, each local agent manages a sub-region of the network, while the global agent manages the entire network. To achieve a best scheduling plan for the offloading scheduling problem, the collaboration between local and global agents is critical.

First, user devices in the coverage of a SeNB, report their real-time status information (generated computational tasks and available computing resources, etc.) to the corresponding local agent. Meanwhile, each local agent can transmit the collected status information to the global agent. Then, each local agent generates a preliminary scheduling plan for the served user devices using a multi-objective solver. In particular, these preliminary scheduling plans can be integrated as a temporary solution to the offloading scheduling problems or can be further optimized in the global agent. Finally, the global agent generates an advanced scheduling plan for all user devices in CPNs based on the optimization information from local agents.

Through the coordination of local and global agents, the hierarchical scheduling framework improves the scalability of the system. As the size of the offloading scheduling problem increases, the local agents with different search capabilities can quickly process the corresponding sub-problems with relatively small problem sizes in parallel. Moreover, the global agent can perform collaborative optimization by further exploiting the optimization information from local agents.

B. System Model

The system consists of one MeNB, multiple SeNBs, and multiple user devices. The MeNB communicates with user devices through SeNBs and provides available computing resources for mobile applications on the user devices. We denote the number of SeNBs as S, the i-th SeNB is represented as S_i. In SeNB S_i, there are U_i user devices that are covered. Also, the j-th user device in i-th SeNB is denoted $U_{i,j}$. User device $U_{i,j}$ has $C_{i,j}$ latency-sensitive and compute-intensive tasks that need to be executed. The k-th task of user device $U_{i,j}$ is denoted as $C_{i,j,k}$. Each computational task can be represented by a two-tuple $C_{i,j,k} = \{f_{i,j,k}, d_{i,j,k}\}$, where $f_{i,j,k}$ is the total number of CPU cycles required to complete the computational task and the $d_{i,j,k}$ represents the input data size of the computational task.

Since user devices have limited computing capabilities and energy resources, uploading some of their computational tasks to the MEC servers for execution can be considered to achieve performance improvement. Therefore, the computational tasks on a user device can be executed by the user device itself or by the nearby MEC servers. We denote the task offloading decision of task $C_{i,j,k}$ as $o_{i,j,k} \in \{0, 1\}$. Specifically, $o_{i,j,k} = 1$ indicates that task $C_{i,j,k}$ is determined to be dealt with by the MEC servers, otherwise, $o_{i,j,k} = 0$. It is noted that each user device needs to serially execute its own single or multiple computational tasks, while the MEC servers can execute multiple computational tasks in parallel. The following is a detailed description of the computation model and communication model.

If task $C_{i,j,k}$ is executed locally, there is no additional communication overhead. Considering that each user device may have different computing capability, we denote the computing capability in terms of CPU frequency of user device $U_{i,j}$ as $F_{i,j}^{loc}$. Noted that there may be multiple computational tasks to be performed serially, the computing delay required to execute

the computational tasks of user device $U_{i,j}$ can be expressed as:

$$T_{i,j}^{ud} = \sum_{k=1}^{C_{i,j}} [(1 - o_{i,j,k}) \times f_{i,j,k} / F_{i,j}^{loc}] \quad (7)$$

Let $e_{i,j}$ be the energy coefficient [27] of user device $U_{i,j}$, the corresponding energy consumption required to execute the computational tasks of user device $U_{i,j}$ can be represented as:

$$E_{i,j}^{ud} = \sum_{k=1}^{C_{i,j}} [(1 - o_{i,j,k}) \times f_{i,j,k} \times (F_{i,j}^{loc})^2 \times e_{i,j}] \quad (8)$$

If user device $U_{i,j}$ chooses to offload task $C_{i,j,k}$ to the MEC servers for execution, additional communication costs need to be considered, including transmission delay and corresponding energy consumption. The transmission delay includes transmitting the input data needed to complete the computational task on the uplink and transmitting the output data back to the user device on the downlink. Similar to [28], we neglect the impact of output data returns, which are generally small in size. Following [29], orthogonal frequency division multiple access (OFDMA) is employed as the multiple access scheme. Let B be the bandwidth of an wireless channel, the wireless transmission rate for user device $U_{i,j}$ to transmit the input data of task $C_{i,j,k}$ can be calculated as:

$$r_{i,j,k}^{SeNB} = B \times log_2[1 + p_{i,j,k} \times g_{i,j,k} / (\gamma \times \sigma^2 + I_{i,j,k})] \quad (9)$$

where $p_{i,j,k}$ is the transmission power, $g_{i,j,k}$ is the channel gain, and σ^2 is the noise power. $I_{i,j,k}$ denotes the interference from other user devices in neighboring cells and γ denotes the gap from channel capacity due to a practical coding and modulation scheme [30].

Thus, the upload delay required to transmit the input data of task $C_{i,j,k}$ can be quantified as:

$$T_{i,j,k}^{up} = o_{i,j,k} \times (d_{i,j,k} / r_{i,j,k}^{SeNB} + d_{i,j,k} / r_{i,j,k}^{MeNB}) \quad (10)$$

where $r_{i,j,k}^{MeNB}$ denotes the wired transmission rate for task $C_{i,j,k}$ to transmit the input data from a SeNB to the MeNB.

We focus on the energy consumption of user devices, the corresponding energy consumption required to transmit the input data of task $C_{i,j,k}$ can be calculated as:

$$E_{i,j,k}^{up} = o_{i,j,k} \times d_{i,j,k} / r_{i,j,k}^{SeNB} \times p_{i,j,k} \quad (11)$$

When the process of data transmission of computational tasks is completed, the MEC servers immediately start to execute the offloaded computational tasks. According to the widely adopted computing model [31], we suppose that the MEC servers have multiple processing cores and are able to perform multiple computational tasks in parallel. However, there is intense competition for computing resources between offloaded computational tasks. We denote the computing resources assigned to the task $C_{i,j,k}$ as $F_{i,j,k}^{c} \in (0, F^{mec}]$, where F^{mec} represents the CPU frequency of the MEC servers. For all the computational tasks executed in the MEC servers, the total amount of computing resources assigned does not exceed F^{mec}. In such a setting, the computing delay required to execute the tasks $C_{i,j,k}$ of user device $U_{i,j}$ can be represented as:

$$T_{i,j,k}^{mec} = o_{i,j,k} \times f_{i,j,k} / F_{i,j,k}^{c} \quad (12)$$

For each user device, the total processing delay required to complete the computational tasks depends on the maximum of the task offloading delay and local execution delay, where the task offloading delay is divided into communication delay and computing delay. Therefore, the total delay of executing $C_{i,j}$ tasks of user device $U_{i,j}$ can be obtained by:

$$T_{i,j}^{total} = \max\{T_{i,j}^{ud}, \max_{k \in C_{i,j}} (T_{i,j,k}^{up} + T_{i,j,k}^{mec})\} \quad (13)$$

The total energy consumption of each user device involves executing computational tasks locally and transmitting the input data for computational tasks to the MEC servers. Thereby,

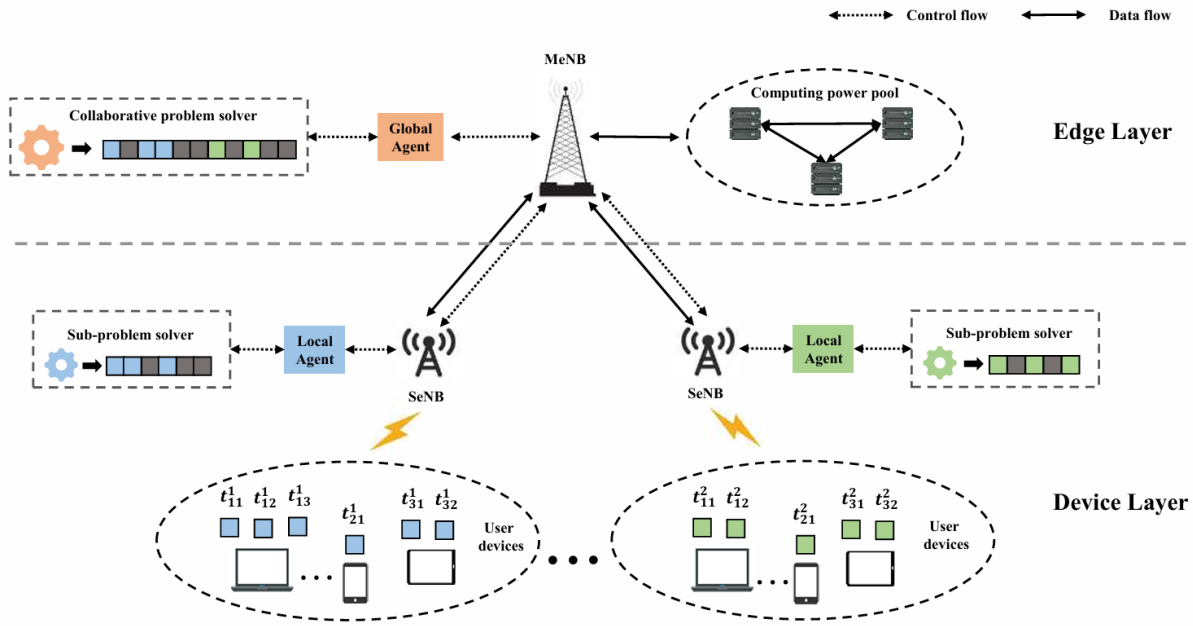

Fig. 1. SDN-based hierarchical scheduling framework.

the total energy consumption of completing $C_{i,j}$ tasks of user device $U_{i,j}$ can be obtained by:

$$E_{i,j}^{total} = E_{i,j}^{ud} + \sum_{k=1}^{C_{i,j}} E_{i,j,k}^{up} \tag{14}$$

C. Problem Formulation

In CPNs, the average utility of processing delay and energy consumption is an important criterion to measure scheduling performance, which can be expressed as:

$$T^{AVG} = \sum_{i=1}^{S}\sum_{j=1}^{U_i} T_{i,j}^{total} / |\sum_{i=1}^{S} U_i| \tag{15}$$

$$E^{AVG} = \sum_{i=1}^{S}\sum_{j=1}^{U_i} E_{i,j}^{total} / |\sum_{i=1}^{S} U_i| \tag{16}$$

According to the above modeling of processing delay and energy consumption of user devices in CPNs, we can formulate an offloading scheduling problem. Due to the conflict between processing delay and energy consumption, we consider to establish a constrained multi-objective optimization problem with the goal of searching for a set of optimal compromise solutions, which can be formulated as:

$$\min_{o_{i,j,k}, F_{i,j,k}^{c}} \{T^{AVG}, E^{AVG}\} \tag{17}$$

$$s.t. \quad o_{i,j,k} \in \{0, 1\} \tag{18}$$

$$F_{i,j,k}^{c} \in (0, F^{mec}] \tag{19}$$

$$\sum_{i=1}^{S}\sum_{j=1}^{U_i}\sum_{k=1}^{C_{i,j}} o_{i,j,k} \times F_{i,j,k}^{c} \leq F^{mec} \tag{20}$$

where the first constraint indicates that there are only two processing modes for the computational tasks, and the second and third constraints represent the computing resource assignment limits of offloaded computational tasks.

Since there are two kinds of optimization variables coupled in the offloading scheduling problem, i.e., task offloading decision variables and computing resource assignment variables, the formulated constrained multi-objective problem is non-convex and NP-hard [17]. In the next section, we propose a distributed multi-objective evolutionary algorithm that seamlessly fits into the hierarchical scheduling framework to solve the offloading scheduling problem.

IV. Proposed Algorithm

In this section, we present a detailed description of the proposed algorithm, as shown in Fig. 2. In DMOEA, the search process is divided into two phases. In the first phase, local agents run in parallel, each of which searches a set of local solutions for a sub-problem. Note that the relatively small problem sizes of the sub-problems allow for fast convergence of the local MOEAs. In the second phase, the global agent searches for a global solution set for the offloading scheduling problem based on optimization information from local agents.

A. Algorithm Flow

At the beginning, the local agents and the global agent are allocated different amounts of fitness evaluation resources. The evaluation counts of local MOEAs and global MOEA are used as termination conditions for the search process. Subsequently, the local MOEAs need to initialize the population of candidate individuals, evaluate the fitness of each member, generate corresponding offspring individuals, and select appropriate candidate individuals to form the next generation to complete an evolutionary iteration. When the termination conditions are reached, the outputs of local MOEAs form the local solution sets for the corresponding sub-problems that can be used in the global agent in the second phase. Specifically, we construct the population of the global MOEA by making a sampling of the local solution sets. Then, the corresponding evolutionary operators are conducted in the global MOEA. This process iterates until the termination condition is satisfied. Ultimately, the output of the global MOEA forms the global solution set of the offloading scheduling problem. The evolutionary operators used are briefly described below.

1) **Initialize population**:
   In the local MOEAs, the population of candidate individuals is randomly initialized for the corresponding sub-problem. Each local MOEA is given the same population size but may have a different individual dimension. Note that the individual dimension of each local MOEA depends on the size of the corresponding sub-problem. Different from local MOEAs, the population of the global MOEA is constructed by sampling the sets of local solutions obtained by the local MOEAs, rather than randomly generated.
2) **Evaluate fitness**:
   For a given candidate individual, its corresponding chromosome can be decoded into a task offloading decision vector and a computing resource assignment vector. In this setting, the fitness value of this candidate individual can be calculated according to these two vectors, in which two objective functions in terms of processing delay and energy consumption are evaluated separately.
3) **Generate offspring**:
   To generate offspring individuals, the parent population is first applied a binary tournament selection operator [32] to build a mating pool, where two parent individuals are selected at random from the parent population and the better parent individual is put into the mating pool. Second, a simulated binary crossover operator and a polynomial mutation operator are applied to the mating pool to generate the offspring population [33], which is of the same size as the parent population.
4) **Select individual**:
   The selection of candidate individuals follows an elite policy that ensures the promising candidate individuals survive through the generations. The non-dominated front numbers and crowding distances of candidate individuals are calculated for comparing population members [34]. In order to keep the population size unchanged, half of the candidate individuals in a combined

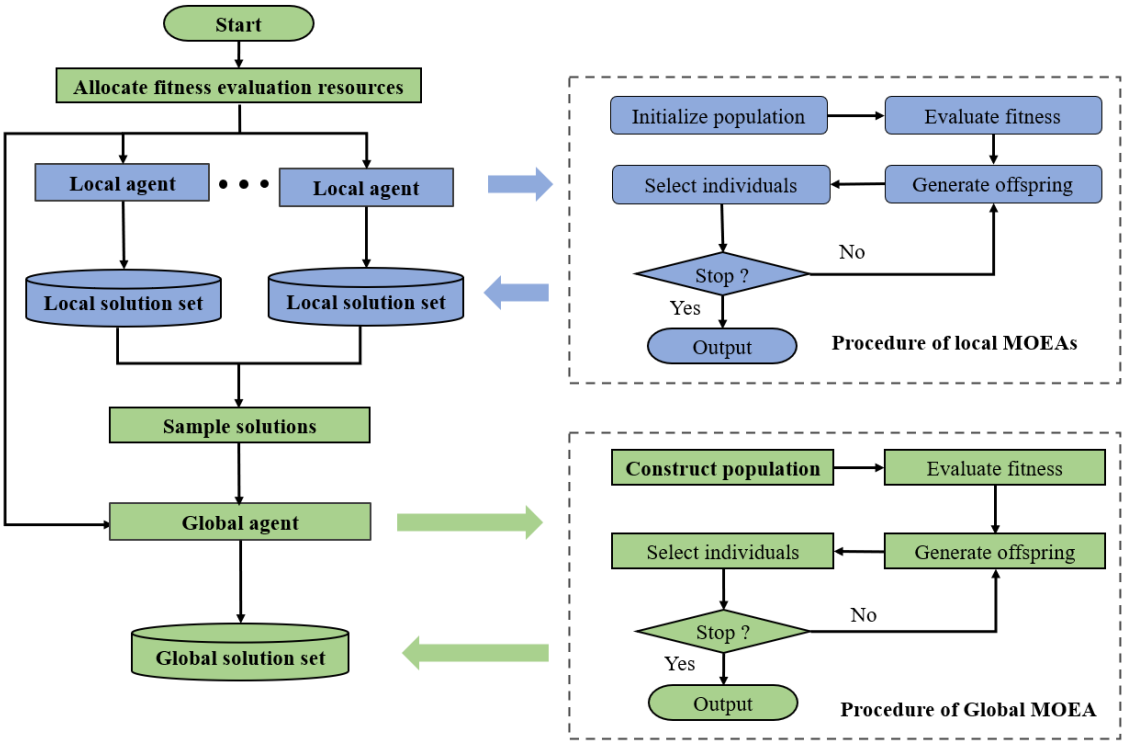

Fig. 2. Flowchart of the proposed algorithm.

population consisting of parent and offspring individuals are selected to the next generation.

B. Encoding and Decoding

As stated in Section III, each computational task involves two kinds of optimization variables, i.e., the task offloading decision variable and the computing resource assignment variable. The former determines the execution location of the computational task, while the latter indicates the amount of computing resources assigned to the computational task. In DMOEA, each chromosome corresponds to a feasible solution. The dimension of a chromosome is equal to the number of computational tasks, and each gene in the chromosome is encoded in the range [-1, 1]. Note that each chromosome can be decoded into two parts. For each gene in a chromosome, the value is greater than 0, indicating that the referred computational task is determined to be executed at the MEC servers. Otherwise, the referred computational task is determined to be executed locally. There is intense competition for computing resources among computational tasks executed at the MEC servers. The amount of computing resources for each of them is the percentage of the gene value it corresponds to over the sum of gene values they correspond to. Fig. 3 shows an example of decoding a chromosome into a feasible solution.

There are two advantages of the proposed encoding and decoding method. First, this method can obtain a task offloading decision vector and a computing resource assignment vector for a feasible solution based on a chromosome. Second, this method can directly obtain feasible solutions that satisfy the constraints of the offloading scheduling problem.

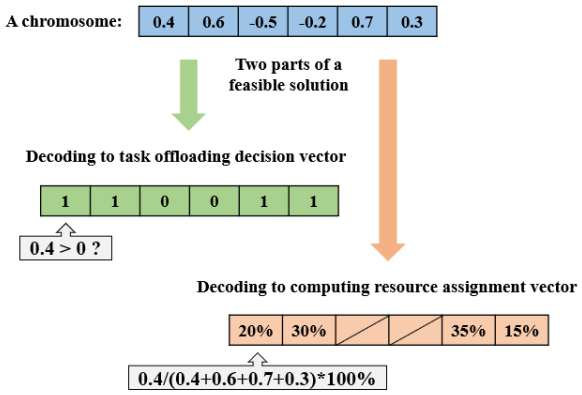

Fig. 3. Example of decoding a chromosome into a feasible solution.

C. Allocate Fitness Evaluation Resources

In DMOEA, both local and global agents require fitness evaluation resources to search for feasible solutions. Fitness evaluation resources refer to the computing resources needed to evaluate candidate solutions. More fitness evaluation resources facilitate the discovery of high-quality solutions [35]. To improve search efficiency of the proposed DMOEA, the fitness evaluation resources should be rationally allocated to local and global agents to balance the search capabilities of global MOEA and local MOEAs.

The following is used to account for the number of function evaluations consumed by the local MOEAs and the global

Algorithm 1 Allocate fitness evaluation resources

Input:
Total number of function evaluations: $Maxeval$
Allocation ratio: α
Individual dimension of global MOEA: D
Individual dimensions of local MOEAs: $d_1, d_2, ..., d_S$
Output:
Number of function evaluations for local agents and global agent
1. **for** $i = 1$ to S **do**
2. Calculate $Leval_i \leftarrow Maxeval \times \alpha \times (d_i/D)$
3. Let $Leval_i$ be the termination condition of the i-th local MOEA
4. **end for**
5. Calculate $Geval \leftarrow Maxeval \times (1 - \alpha)$
3. Let $Geval$ be the termination condition of the global MOEA

MOEA that are involved in DMOEA.

$$Maxeval = Geval + \sum_{i}^{S} Leval_i \tag{21}$$

where $Maxeval$ is the total number of function evaluations set in DMOEA, $Geval$ is the number of function evaluations consumed by the global MOEA, and $\sum_i^S Leval_i$ is the number of function evaluations consumed by the local MOEAs.

As shown in Algorithm 1, we allocate the different numbers of function evaluations to the local agents and the global agent according to the sizes of the problems to be solved, which are considered as the termination conditions for the evolutionary search of local MOEAs and global MOEA. The individual dimension of global MOEA is denoted by D, and the individual dimensions of local MOEAs are denoted by $d_1, d_2, ..., d_S$. Note that local MOEAs have a relatively small search space compared to the global MOEA, enabling faster local optimization in the first stage of DMOEA. We use α to denote the allocation ratio, with a larger value of α indicating that more fitness evaluation resources should be allocated to the local MOEAs. For a local MOEA, the number of allocated function evaluations is proportional to its individual dimension. This suggests that allocating more resources to fitness evaluation for sub-problems that are not easy to be solved helps to improve the evolutionary search for local MOEAs. By allocating different computing resources to distinct local agents, subproblems with varying sizes can be handled quickly and efficiently, thereby enhancing the system's scalability.

D. Sample Solutions and Construct Population

The optimization information from local agents can be reused to enhance the search in global agent. In other words, local MOEAs can reduce the search difficulty of global MOEA as the local solutions found by local MOEAs help global MOEA to explore the areas of the search space with potential high-quality global solutions. Based on this insight, the outputs of the local MOEAs, i.e., the local solution sets of the sub-problems, are first sampled, and then integrated to construct the population of the global MOEA, as shown in Algorithm 2.

For each local solution set, we use a sample criterion (SC) to determine which local solution to be chosen in a loop, as shown below:

$$\begin{aligned} \mathrm{SC}(LS_i^j) = w \times [\frac{f_T(LS_i^{\max}) - f_T(LS_i^j)}{f_T(LS_i^{\max}) - f_T(LS_i^{\min})}] \\ +(1-w) \times [\frac{f_E(LS_i^{\max}) - f_E(LS_i^j)}{f_E(LS_i^{\max}) - f_E(LS_i^{\min})}] \end{aligned} \tag{22}$$

where LS_i^j denotes a candidate solution in local solution set LS_i, $f_T(LS_i^{\max})$, $f_T(LS_i^{\min})$, $f_E(LS_i^{\max})$, and $f_E(LS_i^{\min})$ are the maximum and minimum objective function values found in the LS_i in terms of processing delay and energy consumption, and w is an adjustable weight value.

Depending on a sampling rate β, $N \times \beta$ candidate solutions for global MOEA are firstly generated based on the local solutions. In each of the cycles, each generated candidate solution is composed of local solutions with the best SC. By selecting local solutions through SC, their performance across different optimization objectives can be evaluated, thereby ensuring the constructed population exhibits uniform distribution. After that, the remaining candidate solutions of the global MOEA are generated by random sampling of the local solutions. For a random integer γ in the range [1, S], it suggests that a random local solution LS_γ^{rand} in LS_γ is sampled. The rest of the candidate solutions generated based on LS_γ^{rand} are randomly filled. In this way, the generated population of the global MOEA is able to reuse the optimization information from the local MOEAs and is endowed with good diversity.

Algorithm 2 Sample solutions and construct population

Input:
Local solution set: $LS_1, LS_2, ..., LS_S$
Population size: N
Sampling rate: β
Output:
Population for global MOEA
1. **for** $i = 1$ to $(N \times \beta)$ **do**
2. Calculate $w \leftarrow 1/(N \times \beta)$
3. **for** $j = 1$ to S **do**
4. Sample the best solution LS_j^{best} from LS_j according to Eq. 22
5. **end for**
6. Generate a candidate solution $GS^i \leftarrow [LS_1^{best}, LS_2^{best}, ..., LS_S^{best}]$ for global MOEA
7. Calculate $w \leftarrow w + 1/(N \times \beta)$
8. **end for**
9. **for** $i = (N \times \beta + 1)$ to N **do**
10. Calculate $\gamma \leftarrow randi[1 \sim S]$
11. Sample a random solution LS_γ^{rand} from LS_γ
12. Generate a candidate solution $GS^i \leftarrow [rand, LS_\gamma^{rand}, ..., rand]$ for global MOEA
13. **end for**

E. Complexity Analysis

For the proposed algorithm, the local MOEAs are performed in parallel in the first phase and the global MOEA is performed in the second phase. At the beginning, the complexity of fitness evaluation resources allocation in Algorithm 1 is denoted as $O(S)$. In each iteration of a local MOEA, the crossover operator with a complexity of $O(N * d_i)$ and the mutation operator with a complexity of $O(N*d_i)$ are used to generate N offspring individuals of dimension d_i. The evaluation operator with a complexity of $O(m * N * d_i)$ is used to calculate the fitness values for N offspring individuals of dimension d_i on m objectives. The complexity of fast non-dominated sort on candidate individuals in the current population can be denoted as $O(m * N^2)$. The complexity of crowding distance assignment on candidate individuals in the current population can be denoted as $O(m * NlogN)$. The selection operator is used to create a new population with N candidate individuals with a complexity of $O(NlogN)$. Since the total number of iterations of the local MOEA is $Leval_i/N$, the time complexity of the local MOEA is $O(Leval_i * m * d_i + Leval_i * m * N)$. After that, the complexity of solutions sampling and population construction in Algorithm 2 is denoted as $O(N * S)$. As the global MOEA uses the same evolutionary operators as the local MOEAs, the time complexity of the global MOEA is $O(Geval * m * D + Geval * m * N)$, where $D = \sum_i^S d_i$ and $Geval = Maxeval - \sum_i^S Leval_i$.

By performing Algorithm 1 to allocate fitness evaluation resources and executing Algorithm 2 to sample solutions and construct population, the search efficiency of the proposed algorithm is improved. The increase in problem size leads to a high-dimensional search space, which significantly affects the search efficiency of centralized scheduling algorithms. Based on a hierarchical scheduling framework, the proposed algorithm employs a two-stage search leveraging collaboration between local and global agents. When the size of the offloading scheduling problem increases, the corresponding sub-problems may still have relatively small problem sizes, allowing for quick searches in local MOEAs. Parallel-running local MOEAs with different search capabilities are able to simultaneously deal with the corresponding sub-problems, reducing the running time of the proposed algorithm. Moreover, the global MOEA is able to perform an efficient search by further exploiting the optimization information from local MOEAs.

V. Performance Evaluation

A. Experimental Setup

The performance evaluation is conducted based on the system model and problem formulation presented in Section III. We consider a computing power pool consisting of multiple MEC servers deployed in the MeNB. There are several SeNBs connected to the MeNB, where each SeNB serves a group of user devices randomly located in the coverage area. Each user device has different computational tasks to be executed. The number of computational tasks submitted by each user device is at most 10. The input data size and the required number of CPU cycles of each computational task are randomly generated between [400, 800] KB and $[0.5 * 10^9, 1.0 * 10^9]$ cycles. These heterogeneous user devices may have different computing capabilities and transmission power [17]. The computing capability of each user device is uniformly generated between $[0.5 * 10^9, 1 * 10^9]$ GHz and the transmission power of each user device is randomly distributed in the range of [100, 200] mW. The bandwidth of uplink channel is set to 2 MHz. In addition, γ is set to the constant 1 and σ^2 is set to -100 dBm. I is affected by the transmission power of the mobile devices in the test instances.

As shown in Table I, six test instances of offloading scheduling problem are given. Different instances involve different numbers of user devices and computational tasks, which indicates the different sizes of offloading scheduling problem [36]. In addition, different instances are configured with different computing power of MEC servers, which implies the different degrees of computing resource competition between computational tasks. In the simulation, the local MOEAs are performed in the local agents configured in the SeNBs and the global MOEA is performed in the global agent configured in the MeNB. Note that the dimension of the search space of each local MOEA is related to the number of computational tasks generated from the user devices within the coverage area of the corresponding SeNB.

To demonstrate the performance of the proposed algorithm, we first compare the proposed algorithm with several popular scheduling strategies as follows:

1) AUD: It indicates that all computational tasks are performed on the user devices, where each user device performs its own computational tasks independently.
2) AMS: It indicates that all computational tasks are offloaded to the MEC servers for execution, where each computational task is allocated equal computing resources.
3) RDM: For each computational task, it can be offloaded to the MEC servers for execution or performed on the user device, and this process is random.

Second, the proposed algorithm is compared with various well-known scheduling algorithms based on multi-objective optimization, including NSGAIII-TOMEC [37], MOHGP [38], MOWOA [39], MOGCCA [40], and PPS-MOEA [17]. For a fair comparison, the number of independent runs of all scheduling algorithms is set to 20 and the population size of all scheduling algorithms is set to 50. Considering the multiple stochastic runs of the scheduling algorithms, the Wilcoxon rank sum test [41] with a significance level of 0.05 is conducted to examine the statistical significance of the numerical results.

B. Performance Indicator

To illustrate the advantages of the proposed algorithm over AUD, AMS, and RDM, we introduce a reference point rf. The reference point rf is the product of 1.1 and the worst values of objective functions found by the scheduling strategies in terms of processing delay and energy consumption. The performance improvements of DMOEA, AUD, AMS, and

RDM with respect to the reference point rf can be expressed as:

$$\mathrm{PI}(T) = \sum_{NS^i \in NS} [rf_T - f_T(NS^i)]/rf_T/|NS| \quad (23)$$

$$\mathrm{PI}(E) = \sum_{NS^i \in NS} [rf_E - f_E(NS^i)]/rf_E/|NS| \quad (24)$$

$$\mathrm{PI}(C) = \sum_{NS^i \in NS} [rf_T + rf_E - f_T(NS^i) - f_E(NS^i)]/(rf_T + rf_E)/|NS| \quad (25)$$

where NS^i represents a solution of a non-dominated set NS. In addition, PI(T), PI(E), and PI(C) denote the performance improvements of a scheduling strategy in terms of processing delay, energy consumption, and comprehensive performance, respectively.

For the test instances involved, their true PF is not known. Hypervolume [42] is a suitable indicator to compare the sets of non-dominated solutions obtained by DMOEA, NSGAIII-TOMEC, MOHGP, MOWOA, MOGCCA, and PPS-MOEA. Let Λ denote the Lebesgue measure, the hypervolume indicator of a non-dominated set NS can be defined as the hypervolume of the space that is dominated by the set NS and is bounded by a reference point rf:

$$\mathrm{HV}(NS, rf) = \Lambda(\cup_{NS^i \in NS}[f_T(NS^i), rf_T] \times [f_E(NS^i), rf_E]) \quad (26)$$

where a larger HV value suggests a better quality of a non-dominated set. To make a fair comparison, all comparison algorithms use the same reference point to calculate the HV metrics for each instance.

In addition, the inverted generational distance is a common indicator that can be used to evaluate the convergence and uniformity of a non-dominated set. Let AS denote a set of uniformly distributed objective vectors over the true PF, the inverted generational distance indicator of a non-dominated set NS can be defined as the Euclidean distance between the AS and NS:

$$\mathrm{IGD}(AS, NS) = \sum_{AS^i \in AS} \min_{NS^j \in NS} d(AS^i, NS^j)/|AS| \quad (27)$$

where a smaller IGD value suggests a better quality of a non-dominated set. Since the true PF is not known, we use an approximate PF obtained through a large number of iterations of scheduling algorithms as a substitution.

C. Performance of Different Scheduling Strategies

As shown in Fig. 4, the performance improvement of different scheduling strategies is presented. As expected, the comprehensive performance of DMOEA outperforms all compared scheduling strategies in six test instances. Compared with DMOEA, AUD, and RDM, AMS benefits in terms of energy consumption. As all computational tasks are offloaded to the MEC servers for execution, user devices can achieve significant energy savings. On the contrary, AUD performs the worst in terms of energy consumption because the computing resources provided by the MEC servers are not utilized. Although RDM is worse than DMOEA, it obtains better comprehensive performance than AUD and AMS.

It can be observed from Table I that instance 1 and instance 2 are configured with the same computing resources of MEC servers, but instance 2 has a larger number of computational tasks and user devices than instance 1. In Fig. 4(a) and Fig. 4(b), there is a significant change in the processing delay obtained by AUD and AMS. This is because computational tasks can be offloaded for execution to fully utilize the computing resources of the MEC servers when the problem size is small. When the problem size increases, the competition of a large number of computational tasks for the limited computing resources of the MEC servers intensifies, thus inevitably leading to an increase in processing delay. Similar

TABLE I
SIX TEST INSTANCES OF OFFLOADING SCHEDULING PROBLEM.

Test Instance	Number of SeNBs	Number of User Devices	Number of Computational Tasks per User Device	Number of Computational Tasks	Computing Power of MEC Servers
1	2	10 (in 1st SeNB) 10 (in 2nd SeNB) 20 (in total)	1∼3	19 (in 1st SeNB) 22 (in 2nd SeNB) 41 (in total)	20 GHz
2	2	20 (in 1st SeNB) 20 (in 2nd SeNB) 40 (in total)	1∼5	65 (in 1st SeNB) 60 (in 2nd SeNB) 125 (in total)	20 GHz
3	2	20 (in 1st SeNB) 20 (in 2nd SeNB) 40 (in total)	1∼10	140 (in 1st SeNB) 116 (in 2nd SeNB) 256 (in total)	20 GHz
4	3	10 (in 1st SeNB) 10 (in 2nd SeNB) 10 (in 3rd SeNB) 30 (in total)	1∼3	20 (in 1st SeNB) 25 (in 2nd SeNB) 19 (in 3rd SeNB) 64 (in total)	30 GHz
5	3	20 (in 1st SeNB) 20 (in 2nd SeNB) 20 (in 3rd SeNB) 60 (in total)	1∼5	51 (in 1st SeNB) 58 (in 2nd SeNB) 61 (in 3rd SeNB) 170 (in total)	30 GHz
6	3	20 (in 1st SeNB) 20 (in 2nd SeNB) 20 (in 3rd SeNB) 60 (in total)	1∼10	103 (in 1st SeNB) 99 (in 2nd SeNB) 108 (in 3rd SeNB) 310 (in total)	30 GHz

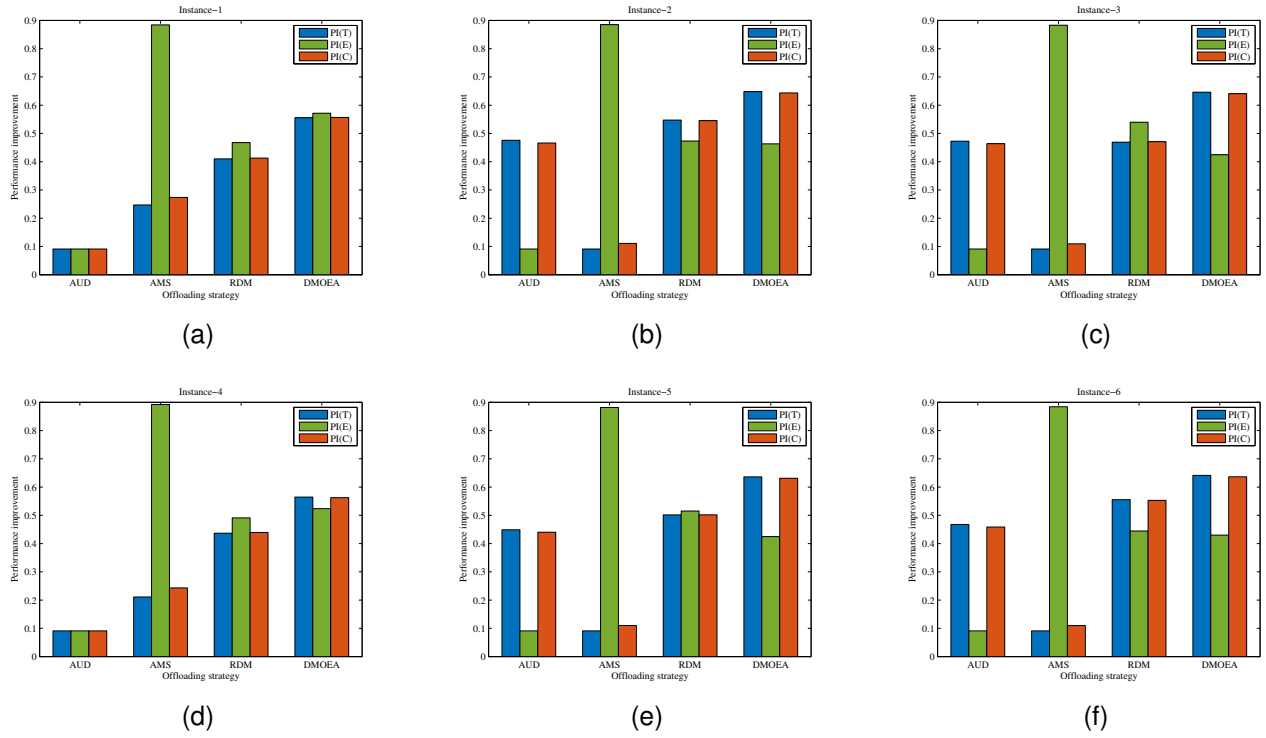

Fig. 4. Performance improvement of different scheduling strategies.

results of AUD and AMS on instance 4 and instance 5 can be observed in Fig. 4(d) and Fig. 4(e). In contrast, the performance of DMOEA is relatively stable across six instances, which indicates that DMOEA is able to maintain a good trade-off between processing delay and energy consumption.

It is also noted that instance 5 and instance 6 have the same number of user devices, while instance 6 has more intensive computational tasks. As shown in Fig. 4(e) and Fig. 4(f), AUD shows the worst energy consumption in instances 5 and AMS shows the worst processing latency in instances 6. When the number of computational tasks per user device increases, user devices lack sufficient computing capability to perform all computational tasks. Excessive offloading of computational tasks to the MEC servers can lead to intense competition for computational resources among user devices. DMOEA is able to optimize both task offloading decision and computing resource assignment for computational tasks, thus achieving good processing delay and energy consumption.

D. Performance of Different Scheduling Algorithms

As shown in Table II, the mean and standard deviation of HV values obtained by different scheduling algorithms on six instances are given, where the best results are marked in bold. It can be seen that DMOEA achieves the best HV values in all instances, which indicates that DMOEA performs good in terms of convergence and distribution. The main difference is that DMOEA implements a collaborative search for the offloading scheduling problem based on local and global agents, while the searches in compared algorithms are based on a centralized scheduling framework for the offloading scheduling problem. It can be observed in instances 1 to 3, MOHGP outperforms MOWOA in instance 1 but performs worse than MOWOA in instances 2 and 3. It is shown that MOHGP struggles to achieve good search efficiency when the size of the problem increases. In instances 1 to 3, there are two local agents configured. The search in DMOEA is first performed by the local agents, and then the local solution sets found are sampled to facilitate the search of the global agent. When the size of the offloading scheduling problem increases, corresponding sub-problems may still have relatively small problem sizes, resulting in efficient searches in the local agents. This is also verified by the results in instances 4 to 6.

As shown in Table III, the mean and standard deviation of IGD values obtained by different scheduling algorithms on six instances are presented, where the best results are marked in bold. We can see that DMOEA obtains the best IGD values in all instances, which indicates that DMOEA is able to search a high-quality set of non-dominated solutions. To demonstrate this clearly, Fig. 5 depicts the PF approximation obtained by different scheduling algorithms. From Fig. 5(a) to Fig. 5(f), we can observe a noticeable margin between the convergence characteristics of DMOEA and compared algorithms. This means that DMOEA enables better utilization of limited function evaluation calls to obtain an effective search for offloading scheduling problem. In Fig. 5(e), compared algorithms seem to prefer optimizing the objective function of processing delay, resulting in a disadvantageous distribution of non-dominated solutions. Instead, DMOEA maintains a good diversity of the obtained non-dominated solutions, which benefits from the uniform sampling of local solution sets. As

TABLE II
THE AVERAGE AND STANDARD DEVIATION OF HV VALUES OBTAINED BY DIFFERENT SCHEDULING ALGORITHMS ON SIX TEST INSTANCES.

Test Instance	Function Evaluation	DMOEA	NSGAIII-TOMEC	MOHGP	MOWOA	MOGCCA	PPS-MOEA
1	5000	**1.0199e-01** (3.2836e-03)	9.5956e-02 - (3.5743e-03)	9.5195e-02 - (4.9146e-03)	9.4173e-02 - (5.2457e-03)	9.6329e-02 - (2.3095e-03)	9.6783e-02 - (4.8940e-03)
2	5000	**3.8290e-01** (5.6987e-03)	3.5893e-01 - (5.4442e-03)	3.4516e-01 - (1.0923e-02)	3.6349e-01 ≈ (3.9710e-02)	3.5901e-01 - (6.5148e-03)	3.6027e-01 - (9.4572e-03)
3	5000	**1.5693e+00** (2.6980e-02)	1.4724e+00 - (2.7461e-02)	1.4331e+00 - (2.0938e-02)	1.4401e+00 - (1.0499e-01)	1.4577e+00 - (3.0013e-02)	1.4605e+00 - (1.3741e-02)
4	5000	**1.0848e-01** (3.1256e-03)	9.4797e-02 - (2.6867e-03)	9.7886e-02 - (5.0015e-03)	9.5108e-02 - (5.0474e-03)	9.3867e-02 - (2.9895e-03)	9.3735e-02 - (3.0156e-03)
5	5000	**2.9846e-01** (3.3013e-03)	2.5472e-01 - (4.3959e-03)	2.5117e-01 - (5.0893e-03)	2.6696e-01 - (1.2213e-02)	2.5737e-01 - (4.8256e-03)	2.5688e-01 - (6.4814e-03)
6	5000	**1.0339e+00** (1.9318e-02)	9.2998e-01 - (1.3793e-02)	9.0655e-01 - (1.3465e-02)	8.8708e-01 - (4.7969e-02)	9.2541e-01 - (1.4902e-02)	9.3039e-01 - (1.8279e-02)

TABLE III
THE AVERAGE AND STANDARD DEVIATION OF IGD VALUES OBTAINED BY DIFFERENT SCHEDULING ALGORITHMS ON SIX TEST INSTANCES.

Test Instance	Function Evaluation	DMOEA	NSGAIII-TOMEC	MOHGP	MOWOA	MOGCCA	PPS-MOEA
1	5000	**3.4229e-04** (1.1710e-04)	4.6821e-04 ≈ (1.5959e-04)	5.8330e-04 - (1.2273e-04)	6.3301e-04 - (1.3420e-04)	5.9871e-04 - (8.0998e-05)	5.2009e-04 - (8.4668e-05)
2	5000	**2.0531e-03** (3.6235e-04)	2.6162e-03 - (3.4857e-04)	2.9712e-03 - (7.2660e-04)	3.6117e-03 - (4.2761e-04)	2.5819e-03 - (4.1983e-04)	2.6824e-03 - (5.5383e-04)
3	5000	**2.5162e-03** (5.9269e-04)	3.6576e-03 - (6.0017e-04)	3.8348e-03 - (5.6312e-04)	4.5872e-03 - (7.7003e-04)	3.8745e-03 - (7.2591e-04)	3.8652e-03 - (9.3879e-04)
4	5000	**5.7520e-04** (1.6525e-04)	9.4803e-04 - (1.3596e-04)	9.7915e-04 - (1.2296e-04)	8.4342e-04 - (2.2784e-04)	1.0060e-03 - (1.3646e-04)	9.8876e-04 - (1.8270e-04)
5	5000	**1.6547e-03** (2.2801e-04)	2.8344e-03 - (2.2060e-04)	2.6648e-03 - (3.4350e-04)	2.9606e-03 - (3.9122e-04)	2.7299e-03 - (3.9486e-04)	2.6449e-03 - (4.4388e-04)
6	5000	**3.5156e-03** (5.0083e-04)	5.2075e-03 - (5.1594e-04)	5.2827e-03 - (5.8114e-04)	6.0110e-03 - (7.4672e-04)	5.0035e-03 - (2.7566e-04)	5.1223e-03 - (3.7168e-04)

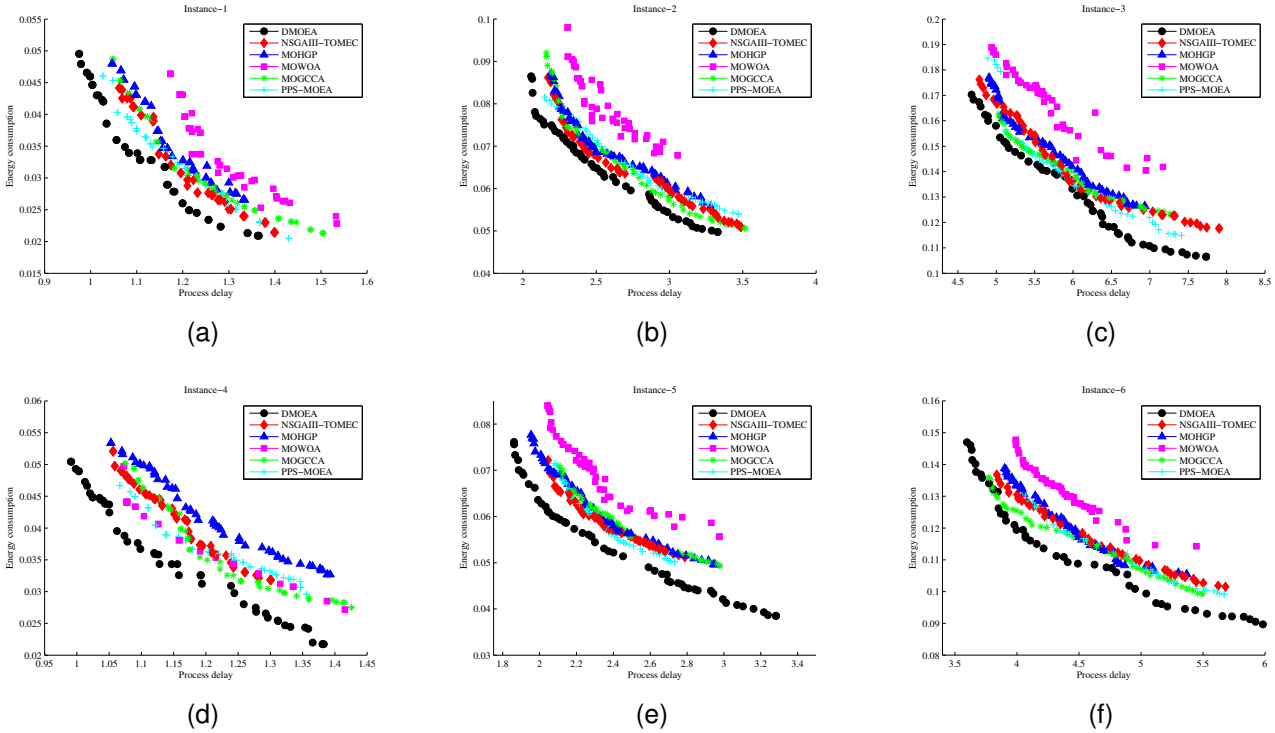

Fig. 5. PF approximation obtained by different scheduling algorithms.

shown in Fig. 6, the convergence trend of different scheduling algorithms on instances 6 is given. It can be seen that DMOEA achieves the best convergence performance, indicating that the collaboration between local agents and global agent can

effectively enhance the scheduling performance of the system.

As shown in Fig. 7, the running time of different scheduling algorithms is presented. It can be seen that DMOEA has better performance in terms of running time than different scheduling algorithms in instances with different numbers of computational tasks. As the size of the offloading scheduling problem increases, the centralized scheduling framework struggles to improve the search efficiency of the scheduling algorithm that relies on a single control center. Depending on a hierarchical scheduling framework, DMOEA utilizes the collaboration between local and global agents for decision-making. Parallel-running local agents with different problem-solving capabilities are able to simultaneously perform quick searches for corresponding sub-problems with relatively small problem sizes, bringing good scalability when the size of the offloading scheduling problem increases.

E. Sensitivity Analysis of Key Parameter

The allocation ratio α and the sampling rate β are two key parameters of the proposed algorithm. We analyze the effect of different parameters on the performance of DMOEA using instance 6. As shown in Fig. 8, there are 9x9 combinations of α and β. The results suggest that the DMOEA obtains best performance with $\alpha = 0.5$ and $\beta = 0.6$. For the proposed algorithm, the allocation ratio affects the search ability of local and global agents and the sampling rate affects the diversity of non-dominated solutions. To effectively address the offloading scheduling problem, the allocation rate can be appropriately increased when the number of local agents rises, so as to facilitate local optimization. When the size of the offloading scheduling problem increases, the sampling rate can be reasonably increased to take full advantage of the local optimization information.

VI. Conclution

In this paper, We propose an SDN-based hierarchical scheduling framework in CPNs, which flexibly formulates scheduling plans through collaborative local and global agents, thereby improving the scalability of the offloading scheduling system. on this basis, we design a distributed evolutionary algorithm (DMOEA) with an efficient two-stage search for the proposed hierarchical scheduling framework. The sub-problems are solved by the local MOEAs in the first phase. Based on the local solution sets obtained by the local MOEAs, the global MOEA is able to achieve an efficient search for the offloading scheduling problem in the second phase.

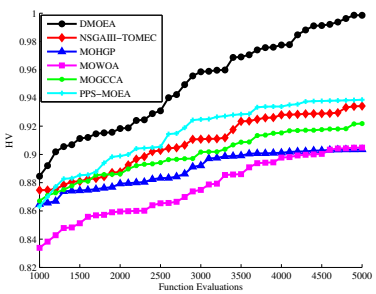

Fig. 6. Convergence trend of different scheduling algorithms.

References

[1] M. Tao, L. Liao, R. Xie, S. Chen, D. Lan, L. Liu, Y. Zhang, D. Li, and C. Wu, "Bidding-enabled resource pricing for computation offloading in 6g vehicle-to-edge networks," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–13, 2025.

[2] M. Tao, X. Li, J. Feng, D. Lan, J. Du, and C. Wu, "Multi-agent cooperation for computing power scheduling in uavs empowered aerial computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 42, no. 12, pp. 3521–3535, 2024.

[3] J. Huang, F. Liu, and J. Zhang, "Multi-dimensional qos evaluation and optimization of mobile edge computing for iot: A survey," *Chinese Journal of Electronics*, vol. 33, no. 4, pp. 859–874, 2024.

[4] J. Huang, M. Zhang, J. Wan, Y. Chen, and N. Zhang, "Joint data caching and computation offloading in uav-assisted internet of vehicles via federated deep reinforcement learning," *IEEE Transactions on Vehicular Technology*, vol. 73, no. 11, pp. 17 644–17 656, 2024.

[5] X. Hou, Z. Ren, J. Wang, W. Cheng, Y. Ren, K.-C. Chen, and H. Zhang, "Reliable computation offloading for edge-computing-enabled software-defined iov," *IEEE Internet of Things Journal*, vol. 7, no. 8, pp. 7097–7111, 2020.

[6] Y. Zhai, W. Sun, J. Wu, L. Zhu, J. Shen, X. Du, and M. Guizani, "An energy aware offloading scheme for interdependent applications in software-defined iov with fog computing architecture," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3813–3823, 2021.

[7] H. Lee, H. Kim, and Y. Kim, "A practical sdn-based data offloading framework," in *2017 International Conference on Information Networking (ICOIN)*, 2017, pp. 604–607.

[8] B. Lei, Q. Zhao, and J. Mei, "Computing power network: An interworking architecture of computing and network based on ip extension," in *2021 IEEE 22nd International Conference on High Performance Switching and Routing (HPSR)*, 2021, pp. 1–6.

[9] R. Pang, H. Li, Y. Ji, G. Wang, and C. Cao, "Energy-saving mechanism based on tidal characteristic in computing power network," in *2021 International Conference on Networking and Network Applications (NaNA)*, 2021, pp. 150–154.

[10] W. Sun, Z. Li, Q. Wang, and Y. Zhang, "Fedtar: Task and resource-aware federated learning for wireless computing power networks," *IEEE Internet of Things Journal*, vol. 10, no. 5, pp. 4257–4270, 2023.

[11] A. Mohammed Seid, A. Erbad, H. N. Abishu, A. Albaseer, M. Abdallah, and M. Guizani, "Blockchain-empowered resource allocation in multi-uav-enabled 5g-ran: A multi-agent deep reinforcement learning approach," *IEEE Transactions on Cognitive Communications and Networking*, vol. 9, no. 4, pp. 991–1011, 2023.

[12] T. Yang, P. Feng, Q. Guo, J. Zhang, X. Zhang, J. Ning, X. Wang, and Z. Mao, "Autohma-llm: Efficient task coordination and execution in heterogeneous multi-agent systems using hybrid large language models," *IEEE Transactions on Cognitive Communications and Networking*, vol. 11, no. 2, pp. 987–998, 2025.

[13] L. Wang, K. Wang, C. Pan, W. Xu, N. Aslam, and L. Hanzo, "Multi-agent deep reinforcement learning-based trajectory planning for multi-uav assisted mobile edge computing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 73–84, 2021.

[14] Z. Li and Z. Ding, "Distributed multiobjective optimization for network resource allocation of multiagent systems," *IEEE Transactions on Cybernetics*, vol. 51, no. 12, pp. 5800–5810, 2021.

[15] J. Zhou, X. Zhao, X. Zhang, D. Zhao, and H. Li, "Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm," *IEEE Access*, vol. 8, pp. 19 306–19 318, 2020.

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

[17] 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 of Things Journal*, vol. 8, no. 17, pp. 13 723–13 736, 2021.

[18] P. Wang, K. Li, B. Xiao, and K. Li, "Multiobjective optimization for joint task offloading, power assignment, and resource allocation in mobile edge computing," *IEEE Internet of Things Journal*, vol. 9, no. 14, pp. 11 737–11 748, 2022.

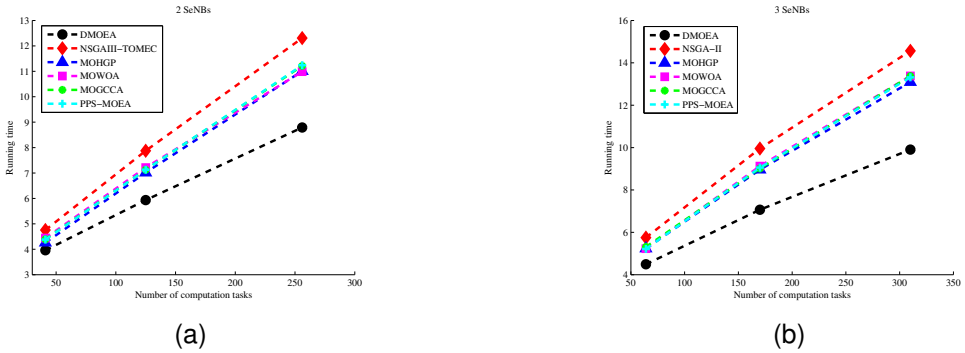

Fig. 7. Running time of different scheduling algorithms.

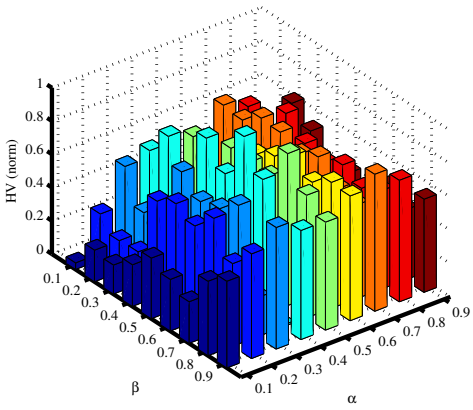

Fig. 8. HV values obtained by DMOEA under different parameters.

[19] C. Jian and J. Gong, "Learning-based computation offloading in hierarchical mec system with energy harvesting," in *2024 IEEE 99th Vehicular Technology Conference (VTC2024-Spring)*, 2024, pp. 1–5.

[20] W. Zhang, G. Zhang, and S. Mao, "Deep-reinforcement-learning-based joint caching and resources allocation for cooperative mec," *IEEE Internet of Things Journal*, vol. 11, no. 7, pp. 12 203–12 215, 2024.

[21] J. Liang, X. Ban, K. Yu, B. Qu, K. Qiao, C. Yue, K. Chen, and K. C. Tan, "A survey on evolutionary constrained multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 27, no. 2, pp. 201–221, 2023.

[22] J. Luo, Y. Dong, Z. Zhu, W. Cao, and X. Li, "Expensive multiobjective optimization based on information transfer surrogate," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 53, no. 3, pp. 1684–1696, 2023.

[23] J. Jiang, "Sdn technology analysis and application research," in *2021 7th Annual International Conference on Network and Information Systems for Computers (ICNISC)*, 2021, pp. 35–39.

[24] Y. Wang, Z.-Y. Ru, K. Wang, and P.-Q. Huang, "Joint deployment and task scheduling optimization for large-scale mobile users in multi-uav-enabled mobile edge computing," *IEEE Transactions on Cybernetics*, vol. 50, no. 9, pp. 3984–3997, 2020.

[25] P.-Q. Huang, Y. Wang, K. Wang, and Z.-Z. Liu, "A bilevel optimization approach for joint offloading decision and resource allocation in cooperative mobile edge computing," *IEEE Transactions on Cybernetics*, vol. 50, no. 10, pp. 4228–4241, 2020.

[26] T. Huang, W. Lin, C. Xiong, R. Pan, and J. Huang, "An ant colony optimization-based multiobjective service replicas placement strategy for fog computing," *IEEE Transactions on Cybernetics*, vol. 51, no. 11, pp. 5595–5608, 2021.

[27] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Transactions on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, 2013.

[28] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, no. 5, pp. 2795–2808, 2016.

[29] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint computation offloading and user association in multi-task mobile edge computing," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 12, pp. 12 313–12 325, 2018.

[30] A. Younis, T. X. Tran, and D. Pompili, "Energy-latency-aware task offloading and approximate computing at the mobile edge," in *2019 IEEE 16th International Conference on Mobile Ad Hoc and Sensor Systems (MASS)*, 2019, pp. 299–307.

[31] Y. Qu, H. Dai, F. Wu, D. Lu, C. Dong, S. Tang, and G. Chen, "Robust offloading scheduling for mobile edge computing," *IEEE Transactions on Mobile Computing*, vol. 21, no. 7, pp. 2581–2595, 2022.

[32] D. Zhang, F. Haider, M. St-Hilaire, and C. Makaya, "Model and algorithms for the planning of fog computing networks," *IEEE Internet of Things Journal*, vol. 6, no. 2, pp. 3873–3884, 2019.

[33] X. Ma, F. Liu, Y. Qi, X. Wang, L. Li, L. Jiao, M. Yin, and M. Gong, "A multiobjective evolutionary algorithm based on decision variable analyses for multiobjective optimization problems with large-scale variables," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 2, pp. 275–298, 2016.

[34] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[35] M. Gong, Z. Tang, H. Li, and J. Zhang, "Evolutionary multitasking with dynamic resource allocating strategy," *IEEE Transactions on Evolutionary Computation*, vol. 23, no. 5, pp. 858–869, 2019.

[36] Z. Gao, L. Yang, and Y. Dai, "Large-scale cooperative task offloading and resource allocation in heterogeneous mec systems via multiagent reinforcement learning," *IEEE Internet of Things Journal*, vol. 11, no. 2, pp. 2303–2321, 2024.

[37] Z.-y. Chai, D. Yuan, and Y.-l. Li, "Multiobjective optimization-based task offloading combined with power and resource allocation in mobile edge computing," *IEEE Systems Journal*, vol. 17, no. 4, pp. 5738–5749, 2023.

[38] F. Guo, H. Zhang, H. Ji, X. Li, and V. C. M. Leung, "An efficient computation offloading management scheme in the densely deployed small cell networks with mobile edge computing," *IEEE/ACM Transactions on Networking*, vol. 26, no. 6, pp. 2651–2664, 2018.

[39] H. Peng, W.-S. Wen, M. Tseng, and L. ling Li, "Joint optimization method for task scheduling time and energy consumption in mobile cloud computing environment," *Appl. Soft Comput.*, vol. 80, pp. 534–545, 2019.

[40] X. Chen, Y. Mao, H. Wang, Y. Xu, D. Li, S. Liu, and X. Zhao, "Data-driven task offloading method for resource-constrained terminals via unified resource model," *IEEE Internet of Things Journal*, vol. 10, no. 11, pp. 9703–9715, 2023.

[41] L. Feng, L. Zhou, A. Gupta, J. Zhong, Z. Zhu, K.-C. Tan, and K. Qin, "Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking," *IEEE Transactions on Cybernetics*, vol. 51, no. 6, pp. 3171–3184, 2021.

[42] K. Li, J. Zheng, M. Li, C. Zhou, and H. Lv, "A novel algorithm for non-dominated hypervolume-based multiobjective optimization," in *2009 IEEE International Conference on Systems, Man and Cybernetics*, 2009, pp. 5220–5226.