This article has been accepted for publication in IEEE Transactions on Computers. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TC.2023.3326977

Hybrid Edge-cloud Collaborator Resource Scheduling Approach Based on Deep Reinforcement Learning and Multi-Objective Optimization

Jiangjiang Zhang, Zhenhu Ning, Muhammad Waqas, *Senior Member, IEEE,* Hisham Alasmary, *Member, IEEE,* Shanshan Tu, *Member, IEEE,* Sheng Chen, *Fellow, IEEE*

.

***Abstract*—Collaborative resource scheduling between edge terminals and cloud centers is regarded as a promising means of effectively completing computing tasks and enhancing quality of service. In this paper, to further improve the achievable performance, the edge cloud resource scheduling (ECRS) problem is transformed into a multi-objective Markov decision process based on task dependency and features extraction. A multi-objective ECRS model is proposed by considering the task completion time, cost, energy consumption and system reliability as the four objectives. Furthermore, a hybrid approach based on deep reinforcement learning (DRL) and multi-objective optimization are employed in our work. Specifically, DRL preprocesses the workflow, and a multi-objective optimization method strives to find the Pareto-optimal workflow scheduling decision. Various experiments are performed on three real data sets with different numbers of tasks. The results obtained demonstrate that the proposed hybrid DRL and multi-objective optimization design outperforms existing design approaches.**

***Index Terms*—Edge cloud resource scheduling, deep reinforcement learning, multi-objective optimization, Markov decision process.**

I. Introduction

WITH the rapid development of information technology, traditional cloud computing mode needs to gather a large amount of data to be processed in the cloud data center [1]. This not only imposes a long delay and wastes existing resources, but also places a huge burden on the cross-domain link bandwidth [2]. The devices of mobile edge computing (MEC) are usually deployed at the edge of the cloud and close to end-users with the characteristics of distributed, low-delay and continuous operation. Limited by the computing ability of MEC devices, the quality of service (QoS) often cannot meet the expectations [3]. To improve the QoS for users, the cooperation between MEC terminal and cloud to effectively complete the computing resource scheduling will become particularly urgent [4].

Recently, researchers have carried out extensive investigation on the edge cloud resource scheduling (ECRS) problem. For the risk-aware energy scheduling problem in microgrid edge computing, Munir *et al.* [5] designed a deep reinforcement learning (DRL) method for the multi-agent random game based on Nash equilibrium to minimize the expected residual energy of the MEC network. Ning *et al.* [6] constructed a three-layer DRL architecture for Internet of vehicles to minimize the overall power consumption. Mao *et al.* [7] described a joint unmanned aerial vehicle position optimization and resource scheduling method in space-air-ground integrated networks with mixed cloud-edge computing to minimize the maximum computing delay between edge devices. Rui *et al.* [8] showed an emergency task allocation mechanism based on the comprehensive reputation and regional prediction model in the ECRS environment to solve the emergency task allocation problem in intelligent network maintenance. It can be seen that machine learning method is regarded as an important tool for dealing with various ECRS problems.

However, due to poor robustness and design constraints of scheduling rules, it is very challenging for these machine learning methods to obtain the global or local optimal solutions under the large-scale ECRS environment [9]. The performance of these methods are often insufficiently good, particularly in dynamic ECRS environments [10]. Meanwhile, meta-heuristic algorithms have gradually been used to address various ECRS problems, because of their fast convergence, high accuracy and low complexity [11]. Qin *et al.* [12] applied a hybrid collaborative multi-objective fruit-fly optimization algorithm to optimize both the execution time and cost for scheduling workflow in cloud environment. Li *et al.* [13] proposed a multi-swarm co-evolution-based hybrid intelligent optimization algorithm for multiple-workflow scheduling to minimize the total cost while meeting the deadline constraint of each workflow in the cloud. Xiao *et al.* [14] designed a co-evolutionary hyper-heuristic framework to minimize the completion time of the workflow scheduling problem, where the

This work was supported by the National Key Research and Development Project of China (2019YFB2102300) and National Natural Science Foundation of China (61971014). The authors also extend their appreciation at King Khalid University for funding this work through Large Group Project under grant number RGP.2/312/44

J. Zhang, Z. Ning, and S. Tu are with Faculty of Information Technology, Beijing University of Technology, 100124 Beijing, China (e-mails: zhangjiangjiang@emails.bjut.edu.cn, ningzhenhu@bjut.edu.cn, sstu@bjut.edu.cn).

M. Waqas is with School of Computing and Mathematical Science, Faculty of Engineering and Science, University of Greenwich, SE10 9LS London, UK and also with the School of Engineering, Edith Cowan University, Perth WA 6027, Australia (e-mail: engr.waqas2079@gmail.com).

H. Alasmary is with the Department of Computer Science, College of Computer Science, King Khalid University, Abha 61421, Saudi Arabia (e-mail: alasmary@kku.edu.sa).

S. Chen is with School of Electronics and Computer Science, University of Southampton, Southampton SO17 1BJ, U.K. (e-mail: sqc@ecs.soton.ac.uk).

task and resource selection rules are automatically learned by a co-evolutionary genetic algorithm. Therefore, meta-heuristic algorithm is another important tool for researcher to deal with challenging ECRS problems.

For any ECRS problem, there are various factors to be considered, and moreover these different factors interact with each other. During resource scheduling, the virtualization characteristic nature of the ECRS environment means that user may not be able to identify whether the provided computing service is a malicious service or not, which affects the reliability of the whole scheduling system [15]. On the other hand, the total execution time of tasks directly affects the energy consumption and the operation cost of edge cloud service provider. Edge cloud service provider will try to avoid idle machines and reduce the cost and energy waste [5]. Therefore, the factors affecting the ECRS problem, including the task makespan time, cost, energy consumption and system reliability, need to be considered jointly in order to achieve a satisfactory result. Thus the ECRS problem is inherently a multi-objective optimization problem (MoOP).

Traditional computing mode transfers the data to the cloud center for centralized processing by the cloud data center, which can no longer meet the needs of today's enterprises and organizations [16]. By placing storage, computing, intelligent data analysis and other work on the edge, an ECRS system can reduce response delay and bandwidth cost, alleviate the pressure on cloud, and provide cloud services, such as whole network scheduling and distributing computing power [17]. With the deepening of the research on the ECRS problem, various ECRS applications have also appeared in practice [18]. For example, in intelligent transportation, edge clusters are deployed in various regions to collect vehicle and road information [19]. The data are recorded in real time and transmitted to the cloud computing center. The cloud computing center conducts analysis and processing on the data transmitted from the regional edge clusters, perceives the overall road traffic situation, and forms an overall scheduling mechanism through big data integration. Then, the cloud data center issues instructions to each edge cluster for overall regulation and control to provide road information to users in advance, avoid road congestion, reduce the probability of accidents, and provide more convenient services for travel [6], [20]. In smart home, MEC nodes can collect the operation and energy consumption information of various home terminal devices and focus on the cloud data center control records, so that users can conveniently understand the operation of real-time terminal equipment and access previous data [21].

In addition, each task in an ECRS system can be split into multiple dependent subtasks [22]. These subtasks should be assigned to appropriate edge computing nodes. The goal of ECRS is to fully consider the dependencies between tasks in the scheduling decision-making process and build a multi-task and multi-user collaborative scheduling service, which requires comprehensive consideration of task completion time, cost, energy consumption and system reliability [23]. Executing tasks under a reliable ECRS system can complete the collaborative scheduling of tasks faster with better QoS, and avoids link interruption. By contrast, lack of available service capacity or low task execution of collaborative tasks may be caused by unreliable virtual machines, which cannot meet the needs of tasks, and further lead to additional overhead caused by link reconstruction transmission [24].

In this paper, edge-cloud collaborative computing mode is adopted for task computing. By combining the advantages of the both computing modes, it can enhance the QoS and experience of service (EoS) for users as well as improve the efficiency of resource scheduling. We aim to reduce the resource waste caused by the uneven allocation of computing resources, and avoid the situation of the backlog in some edge servers due to intensive task requests while other edge servers being relatively idle. Hence, we tackle the task processing timeliness demand lag caused by scheduling tasks to the cloud computing center, in order to obtain a reliable energy-saving and efficient scheduling decision-making scheme that can effectively weigh the factors affecting the ECRS problem. The contributions of our work are given as follows.

- We transform the ECRS problem into a multi-objective Markov decision process (MDP) based on task dependency and ECRS feature extraction. To describe this process, we establish a multi-objective ECRS optimization model that comprehensively considers the makespan time, cost, energy consumption and system reliability of the ECRS system.
- In order to effectively solve the constructed ECRS model, we design a hybrid optimization method by combining DRL and meta heuristic algorithm. Specifically, the deep Q network (DQN) [9], [20] is employed for workflow preprocessing. We add dynamic factors into the MDP to help improving the robustness and efficiency of the algorithm. Moreover, the improved multi-objective evolutionary algorithm (MoEA) [12] is used for optimization. For the preprocessed workflow, the MoEA strives to find excellent solutions that balance convergence and diversity (CaD), and pushes the final solution set to the Pareto-optimal front (PF) [25], [26].
- The experiments are performed on three real data sets with different task numbers. First, the performance comparison of different workflow preprocessing methods is investigated for the MoEA-ECRS. Then the experiments are performed to compare different algorithms with the same pre-processing workflow mode. In addition, the performance boxes of different algorithms are located on different objectives for the specific workflows that have undergone DQN preprocessing operations. The experimental results show that the proposed MoEA-ECRS outperforms other MoEA approaches.

The rest of the paper is organized as follows. Section II introduces the problem description and defines the system objective function. The proposed hybrid optimization approach for solving the ECRS design is detailed in Section III. The

This article has been accepted for publication in IEEE Transactions on Computers. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TC.2023.3326977

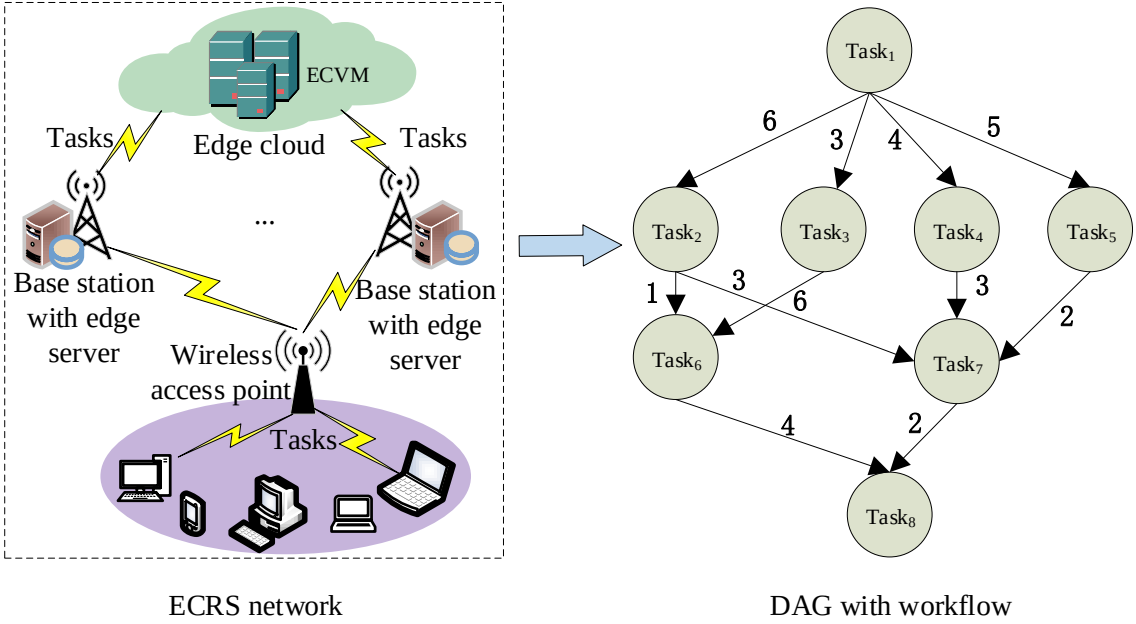


Fig. 1. ECRS network and directed acyclic graph (DAG) with workflow, where ECVM stands for edge cloud virtual machine

simulation experiments are presented in Section IV. The paper is concluded in Section V.

II. Problem Formulation and Objective Function

In this section, we first discuss the problem description, and then define the system's objective function, which includes the makespan time, cost, energy consumption and system reliability. The ECRS network model is illustrated in Fig. 1.

A. Problem Description

When an edge server faces a surge of requests, the computing tasks may be aggregated to cause access congestion because of the limited bandwidth and computing resources. Consequently, the data link is overloaded to form a routing hotspot. Once the load exceeds the critical value, the MEC service distribution capacity will decline exponentially, eventually triggering the avalanche effect that leads to the network disaster [24]. Based on the cooperation between the terminal and the edge cloud, the ECRS system can effectively complete the computing resource scheduling, to alleviate the task aggregation caused access congestion and improve the network QoS [11].

As shown in Fig. 1, wireless access points, edge servers, cloud servers and edge devices jointly organize and form an ECRS network [27]. Edge devices are computing nodes in the edge-cloud domain. To enhance the computing power of the ECRS system, the ECRS center can schedule the computing tasks by matching the needs of each computing task with the computing power of each edge computing node. To intuitively understand the resource scheduling process of edge cloud, the scheduling process can be regarded as a directed acyclic graph (DAG) workflow composed of edge node set $E = \{e_{i,j} \mid i \neq j, i, j \in \{1, 2, \cdots, N_E\}\}$ and edge node task set $Task = \{Task_1, Task_2, \cdots, Task_n\}$, where, $e_{i,j}$ denotes the output data size of the computing task from node $Task_i$ to $Task_j$, N_E is the number of edge nodes, and n is the number of tasks. In particular, the notation $e_{i,j}$ means that task $Task_i$ is the parent task of task $Task_j$, and $Task_j$ is the child task of task $Task_i$. It is necessary to consider the interdependence between tasks when making resource scheduling decisions, and $e_{i,j} = 0$ indicates that the two tasks, $Task_i$ and $Task_j$, are independent.

Users expect to receive cost-effective service from the ECRS, i.e., users favour high QoS with the least cost. For the ECRS provider, only by meeting the needs of users can it get more benefits. Shorter task execution time and better task processing performance are obviously preferred by both users and ECRS providers, which however is usually accompanied by higher costs per unit time [28]. Therefore, the cost is another common concern of users and service providers. Additionally, to ensure the sustainable operation of the system, energy consumption needs to be considered. Moreover, when the equipment fails, the edge terminal may not be able to process the computing task in time, resulting in computing failure and serious reliability problem. Hence, the system reliability factor of the edge device node and/or cloud node should also be taken into account during the construction of the ECRS model, so as to ensure the high-quality resource scheduling.

B. Objective Function

As discussed previously, the performance of an ECRS network depends on multiple factors, and the main objectives in ECRS design include the makespan time, cost consumption, energy consumption and system reliability.

1) Makespan time (Obj_1): The server is usually deployed near the task to be calculated, and the end time of each task is determined by the size of the task and the processing speed of the server to which the task is assigned. The makespan time

of all the tasks in an ECRS system is determined by the end time of the task with the longest end time completed by the server in the process of resource scheduling [12]. Hence, the makespan time $R_{t,l}$ of the server in the l-th edge cloud virtual machine (ECVM) of the ECRS is defined as

$$Obj_1 \triangleq R_{t,l} = \max_{i \in Task} TEnd\,(Task_i, ES_l), \tag{1}$$

where $TEnd\,(Task_i, ES_l)$ denotes the end time of the i-th task $Task_i$ on the l-th server ES_l.

2) Cost (Obj_2): The cost of the ECRS is jointly determined by the rental server cost per unit time and the resources for the scheduling and execution times [9]. The resource of scheduling time for a pair of tasks is calculated as

$$TS_{i,j} = \begin{cases} 0, & \text{if two tasks in same server,} \\ \frac{e_{i,j}}{bw}, & \text{otherwise,} \end{cases} \tag{2}$$

where bw indicates the network bandwidth. Then the corresponding scheduling cost for the ECRS is expressed as

$$\text{Cost}_1 = \sum_{i,j \in Task}^{n} TS_{i,j} \cdot Mony_1, \tag{3}$$

where $Mony_1$ is the price of data transfer. The execution time of task $Task_i$ on server ES_l is defined by

$$\Delta T_l = TEnd\,(Task_i, ES_l) - TStart\,(Task_i, ES_l), \tag{4}$$

where $TStart\,(Task_i, ES_l)$ is the start time of task $Task_i$ on server ES_l. The execution cost of the ECRS is given by

$$\text{Cost}_2 = \sum_{l=1}^{N_{ES}} \sum_{i \in Task}^{n} \Delta T_l \cdot X_{i,l} \cdot Mony_{2,l}, \tag{5}$$

$$X_{i,l} = \begin{cases} 1, & \text{if } Task_i \text{ on } ES_l, \\ 0, & \text{otherwise,} \end{cases} \tag{6}$$

where N_{ES} is the number of servers, and $Mony_{2,l}$ is the price of server ES_l. Thus the cost of the whole ECRS is given by

$$Obj_2 \triangleq R_C = \text{Cost}_1 + \text{Cost}_2. \tag{7}$$

3) Energy consumption (Obj_3): In the running process of the ECRS, edge cloud service providers consume energy to provide services. Similar to [29], we will not consider the energy consumption when the server is idle. The energy consumption of the whole ECRS system in the working state can be described as follows

$$Obj_3 \triangleq R_E = \sum_{l=1}^{N_{ES}} \delta \cdot \Delta T_l, \tag{8}$$

where δ is the energy consumption coefficient [29].

4) System reliability (Obj_4): Server's computing services are usually hidden, and the user will not participate in the specific process of task processing after sending the computing task request. Therefore, it may be unclear whether the server provides users with reliable computing resources to meet the required QoS. This uncertain computing process may also lead to longer makespan time and more energy consumption. Also it is impossible to avoid that the server provider will provide users with the services not meeting users' needs or to rule out malicious services. Therefore, the reliability evaluation of the ECRS system plays an important role in ensuring the tasks being properly and efficiently executed.

In general, the occurrence of failure can be modeled as a Poisson distribution [30]. Therefore, the probability that task $Task_i$ is performed correctly in server ES_l is calculated using the exponential distribution. Assuming that the failures of servers are independent, the reliability performance of the whole ECRS system can be calculated according to

$$Obj_4 \triangleq R_S = \prod_{l=1}^{N_{ES}} e^{-\lambda_l \cdot \Delta T_l}, \tag{9}$$

where $\lambda_l > 0$ is the failure coefficient of server ES_l.

III. Proposed Design Algorithm

Our proposed hybrid optimization framework includes the pretreatment procedure and the MoEA-ECRS optimization. The pretreatment procedure involves DQN, MDP and pretreatment operations, while the MoEA-ECRS optimization is specified by the mating selection, environment selection and overall optimization process. We end this section by analyze the complexity of this hybrid optimization framework.

A. Pretreatment Procedure

1) DQN: DQN combines deep learning and reinforcement learning to realize an end-to-end revolutionary algorithm from perception to action, which mainly includes function value approximation (FVA) and neural network, and it adopts the method of target network and empirical feedback for training. To alleviate the instability and other problems associated with nonlinear feedback networks, DQN makes three improvements to the traditional Q-learning algorithm [9], [27].

1.a) DQN uses experience replay in the training process to process the transferred samples online. At each time t, it stores the transferred samples obtained by the agent interacting with the environment in the playback memory unit. During training, a small batch of transfer samples are randomly selected each time, and the stochastic gradient descent (SGD) algorithm is used to update the network parameters. When training the deep network, independent samples by random sampling are used to reduce the problem caused by correlated samples, thereby improving the stability of the algorithm [9].

1.b) In addition to use an evaluation network for the current FVA, another target network is employed to generate the target Q value. The evaluation network and the target network have the same network structure but different parameters. These network parameters are updated by minimizing the mean square error between the current and target Q values. After the target value network, the target Q value remains unchanged for a period of time, reducing the correlation between the current Q value and the target Q value to a certain extent and improving the algorithm's stability.

1.c) By limiting the reward value and error term to within certain interval, DQN ensures that the Q value and the gradient value are in a reasonable range, and hence improves the stability of the algorithm.

2) MDP: MDP is a mathematical model that simulates the randomness strategy and return of agent action in the environment [22]. It is expressed as a five tuple $\{S, A, Q, R, \gamma\}$. Here, S and A are the state and action sets, respectively. Q is the state transition function, with $Q(s_t, a_t)$ denoting the Q-value function obtained when the state changes from s_t to s_{t+1} at time $t+1$ after the action a_t is executed at time t. R is the immediate return function, with $R(s_t, a_t)$ denoting the return value obtained when the action a_t is executed in the state s_t. The discount factor $\gamma \in [0,\ 1]$ is used to weigh the importance of long-term and immediate returns. It is worth noting that $\gamma = 0$ means that only the immediate return is considered, while $\gamma = 1$ indicates that the long-term return is as important as the immediate return [27]. For the ECRS problem, each state in S is usually defined as a task matching matrix. In the action space A, each action describes the task sequence number of the corresponding state in S. The MDP model of the ECRS problem is defined as follows [19], [22], [31].

2.a) The state space is composed of different states s, which is a dynamic matrix. The state s is represented by a one-dimensional array. The subscript of s represents the task sequence number, and the value of s represents the virtual machine sequence number. At each decision step, the state of workflow scheduling is defined as the input for training. And, the definition of state space can reflect the execution of tasks and the state of servers. At the current time, each server load and the load standard deviation are included in the state space.

2.b) The action is defined as an integer variable. For each task, we need to determine which server is used to execute it. All servers consist of action space. When the action of assigning the i-th task to the j-th virtual machine is executed, the integer variable j is assigned to the i-th value in the state s array.

2.c) Immediate return is usually defined as sparse function, i.e., non-zero return is obtained only in a small number of cases. This return function should reflect the makespan time, cost, energy consumption and system reliability factors. And it should reflect the quality of the solution that transitions from the current state to the next state after taking actions.

The return function r_t is defined as

$$r_t = \sum_{m=1}^{4} \frac{Obj_m(s_t) - Obj_m(s_{t+1})}{Obj_m(s_{t+1})}, \tag{10}$$

where $Obj_m(s_t)$ denotes the value of objective function Obj_m at s_t. The target value function F_t is calculated according to

$$F_t = r_t + \gamma \max_{a \in A} Q(s_{t+1}, a). \tag{11}$$

$F_t - Q(s_t, a)$ is the deviation between the target value and the Q value at decision step t.

3) Pretreatment Operations: Recall that the evaluation network and the target network have the same network structure.

Algorithm 1: DQN Pretreatment Operations

Input: Workflow with number of tasks n;
Output: Workflow after pretreatment with optimized ω_1 and ω_2;
Initialization: Set maximum episodes N_{episode}, set training set size B, set number of steps that ω_2 remains unchanged h;
Begin:
  Initialize workflow state space and related parameters;
  While number of episodes < N_{episode} **do**
    Updating the ranking task list;
    For each task in the task list **do**
      Generate random variable $pro \in [0,\ 1]$;
      $a_t = \begin{cases} \arg\max_{a \in A} Q(s_t, a|\omega_1), & \text{if } pro \in [0,\ \xi]; \\ \text{randomly selected}, & \text{if } pro \in (\xi,\ 1]; \end{cases}$
      Obtain the next state s_{t+1} and reward r_t;
      $q_t = Q(s_t, a_t|\omega_1)$;
      Store $\{s_t, a_t, q_t, r_t, s_{t+1}\}$ in experience reply pool;
      Randomly select B data from the pool for training;
      If $Task_i$ is last task
        $F_t = r_t$;
      Else
        $F_t = r_t + \gamma \max_{a \in A} Q(s_{t+1}, a|\omega_2)$;
      End If;
      Calculate loss function $L(\omega_1) = E\left[(F_t - q_t)^2\right]$;
      Employ the SGD method to update ω_1;
      Copy the value of ω_1 to ω_2 every h step;
    End For;
  End While;
End.

Let ω_1 and ω_2 be the Q-value parameters of the evaluation network and the objective function parameters of the target network, respectively [9]. For a workflow, the updating ranking is used to generate a task list [32]. At the current state, tasks should be selected according to the order of the tasks list. Then an action is determined for this task. When the action is completed, the corresponding reward is obtained and the environmental information is updated to the next state. The pseudo code of the DQN pretreatment operations is listed in Algorithm 1, where an episode refers to a complete circle of allocating all the tasks in a workflow, ξ is a probability threshold, and $E[\cdot]$ denotes the average operator.

B. MoEA-ECRS

Owing to their ability to deal with multiple conflicting objectives, MoEAs are widely applied to various MoOPs [26], [33], [34]. In particular, hpaEA [35] is proved to obtain good performance and provide excellent decision-making schemes. However, hpaEA still has room for improvement in obtaining solutions with better CaD. To obtain a better decision solution, an improved MoEA based on hpaEA is described in this paper. In our algorithm, two important selection operations are included, mating selection and environment selection. In

this subsection, we specify these two selection processes and then summarize the operations of our proposed MoEA-ECRS.

1) Mating Selection: The main purpose of mating selection is to generate more individuals with balanced CaD. According to [35], a good evolutionary direction needs to be guided by the excellent solutions in the population. For the problem with m objectives, a solution is referred to as an excellent solution if there exist m different elements in its adjacent solution set and its objective vector is positioned below the hyperplane formed by its adjacent solution set [36], [37]. The steps of identifying the excellent solution are as follows.

1.a) An empty set is initialized to record the excellent solution. With the help of nondominated sorting mechanism, the population is sorted hierarchically, and all the solutions of the first layer are normalized.

1.b) An empty set is initialized to record the objective vector of the adjacent solutions of each solution in the first layer.

1.c) Find the adjacent solution of each solution and record the corresponding objective vector.

1.d) For each solution, find its adjacent solutions set with m different elements.

Then, we judge whether a solution is below the hyperplane formed by its adjacent solution set. If so, the solution is regarded as an excellent solution and it is added to the population. If not, the solution is not an excellent solution and it is discarded. After all the excellent solutions are found and added to the population, the environment selection mechanism is triggered to select the other solutions.

2) Environment selection: The main purpose of environment selection is also to maintain good CaD as well as alleviate the decline in the selection pressure due to the increase of nondominated solutions in the later stage of iteration. Based on the mating selection principle and definition of excellent solutions, some 'temporary' excellent solutions have been selected as the final decision solutions. There are two situations regarding the results of the mating selection.

First, if the number of the solutions has met or even exceeded the required number, the environment selection can be used as an auxiliary pruning mechanism to effectively delete redundant solutions, while retaining the remaining excellent solutions. On the other hand, if the number of the solution has not met the required number, the environment selection operation plays an important role in selecting more solutions with good CaD conditions.

To ensure that the remaining solutions with better CaD characteristics are selected, an evaluation metric is naturally required. Hypervolume (HV) is widely employed to measure the CaD of solution x [38]. Let $Z^r = \left(Z_1^r, \cdots, Z_m^r\right)$ be the reference point set in the objective space, where m is the number of objectives. Let $f_i(x)$ denote the i-th fitness value of solution x, which is dominated by all the Pareto-optimal objective vectors, and $\left[f_i(x), Z_i^r\right]$ denote the hypercube, which can be constructed with the reference point Z_i^r and the solution value $f_i(x)$ as two diagonal corners of the hypercube. Further let $PF^\star$ denote the approximation set of solutions x [38]. Then, the HV metric of the approximate front-surface solutions in $PF^\star$ and the reference point set Z^r can be computed as follows

$$HV\left(PF^\star\right) = vol\left(\bigcup_{x \in PF^\star} \left[f_1(x), Z_1^r\right] \times \cdots \times \left[f_m(x), Z_m^r\right]\right), \tag{12}$$

where $vol(\cdot)$ is the Lebesgue measure, which calculates the hypervolume of all the objectives' hypercubes. The larger the HV value is, the more favourable the approximation set is. The iteration procedure of the environment selection continues until the number of the selected solutions reaches the population size.

3) MoEA-ECRS Optimization Operation: The main function of MOEA-ECRS optimization operations is to coordinate mating selection and environment selection, so as to obtain solutions with good CaD. For the workflow after pretreatment, the MOEA-ECRS strives to allocate more evolutionary efforts on the excellent solutions in the current population by mating selection, enabling better solutions in terms of convergence being generated. Once the excellent solutions are determined, the environment selection mechanism is triggered to balance CaD. The excellent solutions are selected preferentially to strengthen the selection pressure.

More specifically, the excellent solution set $\mathcal{D}$ contains K individuals, which is initially set to $\mathcal{D} = \emptyset$. To build a mating pool, the $N - K$ solutions are randomly selected from the current population $\mathcal{P}$ of N individuals, which are combined with all the K excellent solutions of the previous generation. The new offspring are produced in the combined population by using simulated binary crossover (SBX) and polynomial mutation (PM) operations [39], [40]. The dominating solutions according to the dominating relationship are eliminated from the combined parent and offspring population. Then the mating selection and environment selection are triggered in turn. Under the coordination of the two selection operations, the final solution set is pushed to the Pareto-optimal front (PF). The evolution cycle continues until the stopping condition is reached. The pseudo code of the MOEA-ECRS optimization operations are described in Algorithm 2.

4) Summary of Proposed Design: Our MoEA-ECRS design is presented in Algorithm 3. The DQN pre-processing of the ECRS problem is a sequential decision problem that maps the tasks to the corresponding servers, i.e., an MDP [9]. In each decision step, the agent selects the action in the current state and obtains a reward value, which is used to evaluate the quality of the action. After completing a decision step, the agent enters the next state. Through the continuous interaction between the agent and the environment as well as the continuous learning on feedback information, a series of optimal strategies are obtained, to optimally pre-process the workflow. Then as discussed in this subsection, the MoEA-ECRS undertakes the next optimization. The mating selection and environment selection mechanisms ensure that the final decision solutions with good CaD are obtained. In particular, the

Algorithm 2: MoEA-ECRS optimization operations

Input: Workflow after pretreatment;
Output: Selected population $\mathcal{P}$;
Initialization: Set maximum number of iterations $I_{\max}$;
Begin:
Initialize populations $\mathcal{P}$ with N individuals, excellent solution set $\mathcal{D}$ with K individuals, reference vectors and related parameters;
While number of iterations $< I_{\max}$ **do**
Updating ranking task list;
For Each task in task list **do**
Randomly select $N-K$ solutions from current population $\mathcal{P}$ to form $\mathcal{D}^*$;
$\mathcal{P} = \mathcal{D} \cup \mathcal{D}^*$;
Generate offspring Q from parent $\mathcal{P}$ by employing SBX and PM;
$\mathcal{P}^* = \mathcal{P} \cup Q$;
Remove dominated solutions in $\mathcal{P}^*$ to update combined population $\mathcal{P}^*$;
$\mathcal{G}, \mathcal{V} = \emptyset$; /* Mating selection */
$[F1, F2, \cdots]$ = Nondominated sorting($\mathcal{P}^*$);
Take F1 layer solution in $\mathcal{P}^*$ and store it in $\mathcal{G}$: $\mathcal{G} = \mathcal{P}^*_{(F1)}$;
Find and store objective vector of adjacent solutions of $\mathcal{G}$ to $\mathcal{V}$;
Determine in turn whether solution in $\mathcal{G}$ is excellent solution;
Add excellent solutions to excellent solution set $\mathcal{D}$ to update $\mathcal{D}$ to K individuals;
If $K < N$ /* Environment selection */
Computing HV values of residual population individuals;
Select $N-K$ candidate solutions $\mathcal{D}^*$ with better HV values;
$\mathcal{P} = \mathcal{D} \cup \mathcal{D}^*$;
Else
Computing HV values of all individuals in $\mathcal{D}$;
Remove $K-N$ solutions of $\mathcal{D}^*$ with lowest HV values;
$\mathcal{P} = \mathcal{D} - \mathcal{D}*$;
End If;
End While;
End.

Algorithm 3: MoEA-ECRS algorithm design

Input: Workflow;
Output: Selected population $\mathcal{P}$;
Begin:
Initialize related parameters;
Call Algorithm 1 to perform DQN pretreatment;
Call Algorithm 2 to do MoEA-ECRS optimization;
Obtain selected population $\mathcal{P}$;
End.

proposed MoEA-ECRS optimization has obvious advantages in overcoming the decline of selection pressure in the later stage of population evolution, which ensures that the decision solution obtained is as close as possible to the PF.

C. Complexity Analysis

The computational cost of our design comes from two sources, the DQN preprocessing and the MoEA-ECRS optimization. To provide a more intuitive description, the computational complexity is listed in Table I.

1) For a workflow with n tasks, the DQN generates a solution in N_{episode} episodes, and the computational complexity of the DQN preprocessing operations can be shown to be on the order of nN^2_{episode}, denoted as $\mathrm{O}\left(nN^2_{\text{episode}}\right)$ [9].

2) As for the MoEA-ECRS optimization operations, the solution set is generated in $I_{\max}$ evolutionary iterations, and its computational cost occurs in generating offspring operation, mating selection and environment selection for each task.

2.1) Since we have population size N and m objectives, the computational complexity of employing SBX and PM operation to generate offspring is on the order of $\mathrm{O}\left(mN^2\right)$, and the complexity for the nondominated sorting of the mating selection is $\mathrm{O}\left(mN^2\right)$ [40], while the complexity of normalizing the solutions is $\mathrm{O}(mN)$. Thus the computational complexity of generating offspring is on the order of $\mathrm{O}\left(mN + 2mN^2\right)$.

2.2) In the process of selecting excellent solutions, finding objective vectors of the adjacent solutions for all solutions imposes the complexity $\mathrm{O}(mN \log N)$, and since it takes $\mathrm{O}\left(m^3\right)$ to obtain the parameters of a hyperplane, the complexity of identifying the adjacent solutions is $\mathrm{O}\left(m^3 N\right)$ [35]. In addition, it costs $\mathrm{O}\left(mN^2\right)$ to select and add excellent solutions to the excellent solution set. Hence the complexity of mating selection is on the order of $\mathrm{O}\left(m^3 N + mN \log N + mN^2\right)$.

2.3) In environment selection, it costs $\mathrm{O}\left(mN^2\right)$ to calculate the HV value, and the computational complexity of selecting a solution from the ordered HV values is $\mathrm{O}\left(mN^2\right)$ [35]. Thus the complexity of environment selection is $\mathrm{O}\left(2mN^2\right)$.

Therefore, for each task and one evolutionary iteration, the computational complexity of generating offspring operation, mating selection and environment selection is on the order of $\mathrm{O}\left(mN\left(1 + m^2 + \log N + 5N\right)\right)$. Consequently, the computational complexity of the MoEA-ECRS optimization is on the order of $\mathrm{O}\left(mnI_{\max}N\left(1 + m^2 + \log N + 5N\right)\right)$.

3) The total computational complexity of our proposed DQN-MoEA-ECRS design is therefore on the order of $\mathrm{O}\left(nN^2_{\text{episode}} + mnI_{\max}N\left(1 + m^2 + \log N + 5N\right)\right)$.

IV. Simulation Experiments

It is complicated and cost-high to evaluate the algorithm performance in the real edge-cloud environment. We perform extensive simulation experiments to evaluate our proposed

TABLE I
COMPUTATIONAL COMPLEXITY

Operations Name		Complexity
DQN preprocessing		$\mathrm{O}\left(nN_{\text{episode}}^2\right)$
MoEA-ECRS Optimization Operations	Generating Offspring Operation	$\mathrm{O}\left(mN + 2mN^2\right)$
	Mating Selection	$\mathrm{O}\left(m^3N + mN\log N + mN^2\right)$
	Environment Selection	$\mathrm{O}\left(2mN^2\right)$
MoEA-ECRS Optimization		$\mathrm{O}\left(mnI_{\max}N(1 + m^2 + \log N + 5N)\right)$

DQN-aided MoEA-ECRS design. WorkflowSim is the simulation toolkit extended from CloudSim, which can be utilized to simulate an edge-cloud workflow scheduling environment accurately and used in our work.

A. Simulation System

1) ECRS system specifications: We construct an edge–cloud system consisting of six edge servers and four cloud servers. The simulated ECRS system's key parameters, including the task attributes and the infrastructure parameters [9], [10], [29], are listed in Table II.

TABLE II
ECRS SYSTEM PARAMETERS

Description	Value
Task size	60000 to 120000
Number of tasks	50 to 100
Number of cloud server	4
Number of edge servers	6
Calculation speed of cloud server	$[1.5,\ 3.5] \times 10^6$
Calculation speed of edge server	$[0.8,\ 1.5] \times 10^6$
Energy consumption factor	10^{-20}
Failure coefficient	[0.3, 1]
Price of data transfer	0.01
Price of server	[0.06, 0.24]
Network bandwidth	10000

2) Data sets: To create a realistic simulation environment, three real-world workflow datasets, Montage, Cybershake and Inspiral [9], are used in our simulation experiments. These workflows have different numbers of tasks and similar task structures. Different tasks are set in different types of workflows in the experiments. Each workflow is specified by task length, task input and output data size, and DAG structure.

3) Benchmarks and algorithmic parameters: To demonstrate the effectiveness of the DQN preprocessing in our DQN-MoEA-ECRS design, we test two other preprocessing methods, the heterogeneous earliest-finish-time (HEFT) algorithm [41], [42] and random method by combining them with the proposed MoEA-ECRS optimization to form the HEFT-MoEA-ECRS and random-MoEA-ECRS designs. Moreover, three advanced designs, the dynamic constrained NSGA-III (DCNSGA-III) [43], Pareto front shape estimation based evolutionary algorithm (PeEA) [44] and adaptive reference vector-guided evolutionary algorithm (RVEA) with improved growing neural gas (RVEA-iGNG) [37], are adopted as the benchmarks for the comparison with our proposed MoEA-ECRS design.

For a fair comparison, all the original parameter settings are employed for the involved algorithms, as these algorithmic parameters have been shown to achieve the best performance. The default algorithmic parameters used in the simulation experiments are listed in Table III.

TABLE III
DEFAULT ALGORITHMIC PARAMETERS

Description	Value
Discount factor	$\gamma = 0.9$
Network parameter update steps	$h = 10$
Threshold value	$\xi = 0.9$
Replay memory size	10^6
Minebatch size	$B = 128$
Number of episodes	$N_{\text{episode}} = 10000$
Crossover probability	1
Mutation probability	0.1
Population size	$N = 240$
Number of evolutions	$I_{\max} = 10000$
Number of independent runs	20

B. Simulation Results

The characteristics of the true PF are not known a priori, and they are hard to capture. Also, the effective areas may be disconnected, irregular or more sophisticated, and it is hard to fit them in a specific model. To tackle this problem, the PF which results from the union of the PFs of all the methods is considered as the true PF [45].

1) Comparison of different workflow preprocessing methods: To investigate the impact of workflow preprocessing on the achievable performance of the MoEA-ECRS design, we compare three workflow preprocessing methods, the HEFT, the random method and the proposed DQN based preprocessing. Table IV compares the HV metric performance achieved by the HEFT-MoEA-ECRS, Random-MoEA-ECRS and DQN-MoEA-ECRS on Montage, Cybershake and Inspiral test workflows, where the results are obtained by averaging over 20 independent simulations and presented as mean (standard deviation). The boldfaced value in each row of Table IV indicates the best performance, i.e., the highest HV value. Furthermore, we apply Friedman statistical test [35] to test the statistical significance of the different results obtained by different methods. Specifically, we use the symbols '+', '−' and '≈' to indicate whether the result obtained by a different preprocessing method is significantly 'superior', 'inferior', or 'similar' to that obtained by the DQN-MoEA-ECRS using Friedman statistical test with a significance level 0.05.

It can be seen from Table IV that the DQN-MoEA-ECRS achieves the best results in the 5 cases out of the 9 cases, and these 5 results are validated by Friedman statistical test as significantly better than corresponding results of the other two methods. Additionally, for the 4 cases that the DQN-MoEA-ECRS is not the best, Friedman statistical test shows that the

This article has been accepted for publication in IEEE Transactions on Computers. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TC.2023.3326977

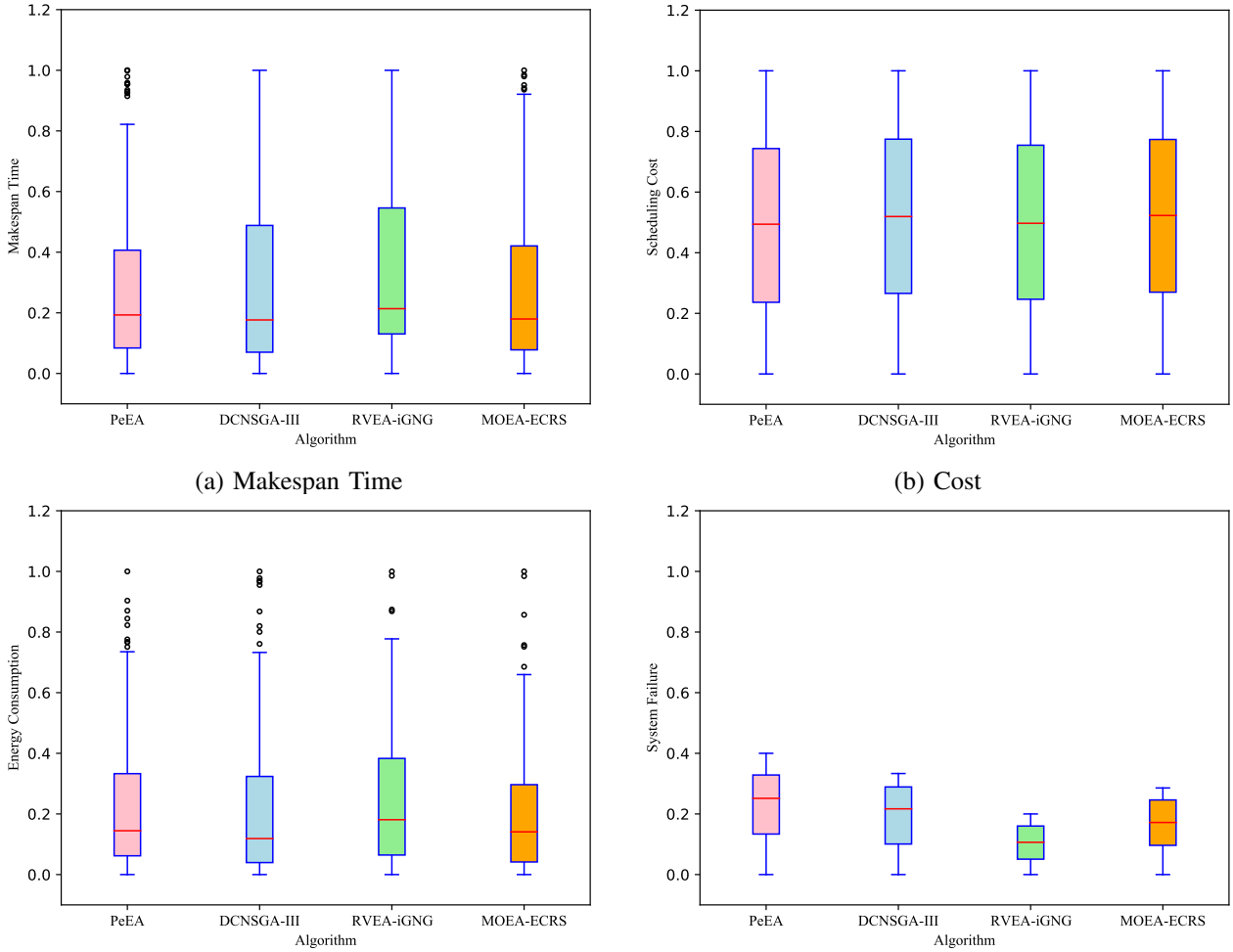
TABLE IV
PERFORMANCE COMPARISON OF MOEA-ECRS DESIGN WITH THREE DIFFERENT WORKFLOW PREPROCESSING METHODS, IN TERMS OF HV METRIC

Workflow	Number of tasks	HEFT-MoEA-ECRS	Random-MoEA-ECRS	DQN-MoEA-ECRS
Montage	50	9.9971e-1 (3.24e-5) –	9.9971e-1 (2.73e-5) –	**9.9972e-1 (2.12e-5)**
	75	9.9984e-1 (1.89e-5) –	9.9987e-1 (1.21e-5) –	**9.9988e-1 (1.15e-5)**
	100	9.9996e-1 (7.42e-5) ≈	**9.9997e-1 (6.45e-6)** ≈	9.9996e-1 (6.45e-6)
Cybershake	50	9.9975e-1 (9.95e-6) –	9.9979e-1 (1.53e-5) –	**9.9981e-1 (1.35e-5)**
	75	9.9991e-1 (2.33e-5) –	**9.9993e-1 (1.16e-5)** ≈	9.9992e-1 (1.37e-5)
	100	9.9994e-1 (2.76e-5) –	9.9994e-1 (6.51e-5) –	**9.9996e-1 (4.52e-6)**
Inspiral	50	9.9991e-1 (4.65e-5) –	**9.9992e-1 (2.45e-5)** ≈	9.9992e-1 (1.02e-5)
	72	**9.9997e-1 (6.12e-6)** ≈	9.9995e-1 (6.89e-6) –	9.9996e-1 (6.30e-6)
	100	9.9997e-1 (3.54e-5) –	9.9997e-1 (1.50e-5) –	**9.9998e-1 (3.82e-6)**

results are statistically similar to the best HV values attained the other two methods. Moreover, no case that the other two methods achieve superior performance than the DQN preprocessing. The results of Table IV therefore indicates that the proposed DQN based preprocessing outperforms the other two methods.

2) Comparison of different design algorithms: Next we investigate the achievable performance of different design algorithms in the same DQN-based workflow preprocessing mode. Table V compares the HV metric values attained by the proposed MoEA-ECRS and the three benchmark designs for Montage, Cybershake and Inspiral test workflows, in terms of mean (standard deviation) averaging over 20 independent experiments, where again the symbols '+', '−' and '≈ indicate whether the results obtained by the benchmark algorithms are significantly 'superior', 'inferior', or 'similar' to those obtained by the MoEA-ECRS using Friedman statistical test with a significance level 0.05.

It can be seen that in terms of HV metric, our MoEA-ECRS obtains the best results in 8 out of the 9 cases. RVEA-iGNG is the second best, as it is inferior to the MoEA-ECRS in one case while its performance in the other 8 cases are judged



(a) Makespan Time

(b) Cost

(c) Energy Consumption

(d) System Failure

Fig. 2. Performance comparison of different objectives by four design algorithms on Montage-DQN-100

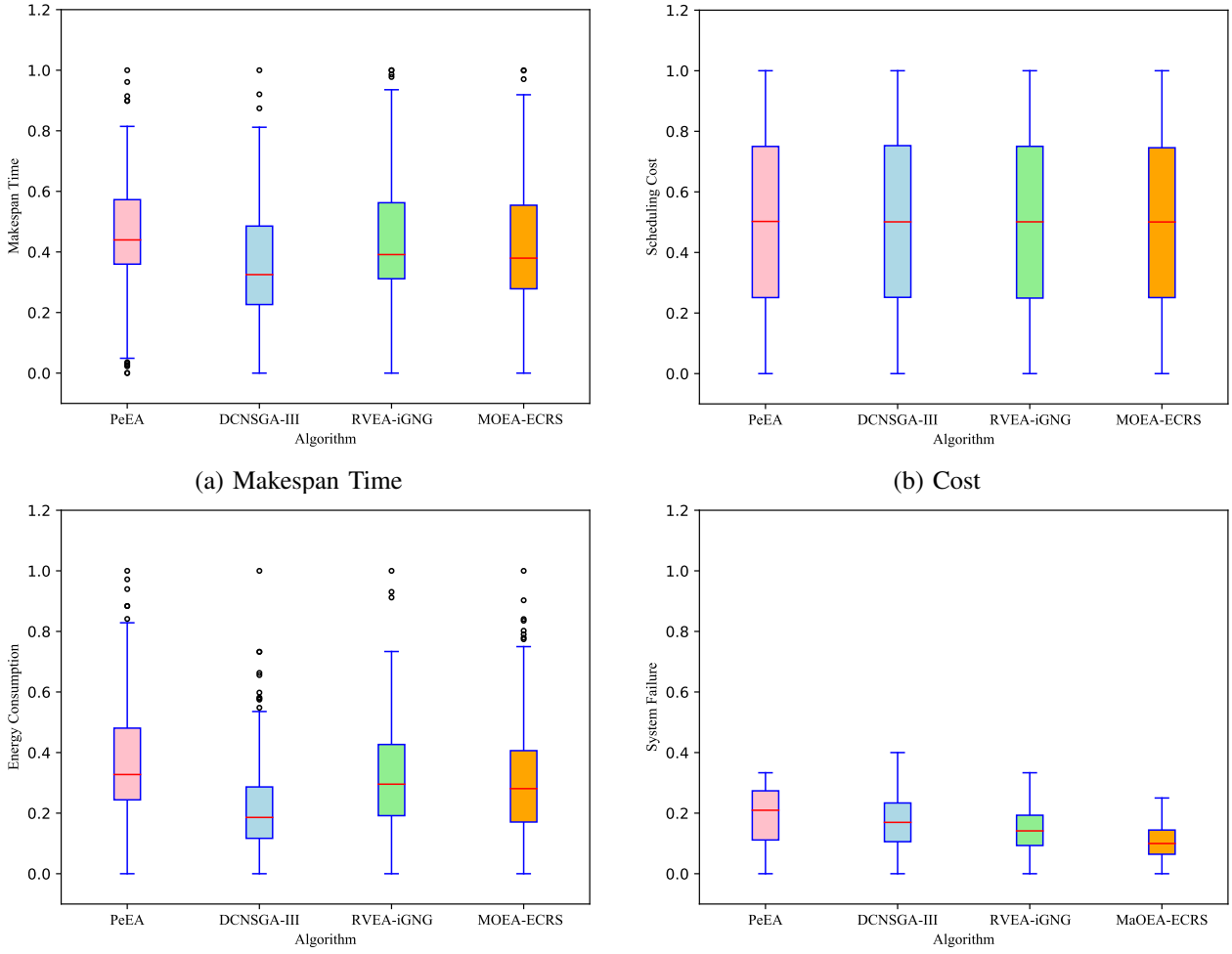
This article has been accepted for publication in IEEE Transactions on Computers. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TC.2023.3326977

TABLE V
PERFORMANCE COMPARISON OF DIFFERENT DESIGN ALGORITHMS WITH SAME DQN PREPROCESSING IN TERMS OF HV METRIC

Workflow	Tasks	PeEA	DCNSGA-III	RVEA-iGNG	MoEA-ECRS
Montage-DQN	50	9.9972e-1 (2.21e-5) ≈	9.9970e-1 (2.86e-5) –	9.9971e-1 (2.92e-5) ≈	**9.9972e-1 (2.12e-5)**
	75	9.9987e-1 (1.68e-5) ≈	9.9986e-1 (1.60e-5) –	9.9987e-1 (1.81e-5) ≈	**9.9988e-1 (1.15e-5)**
	100	9.9989e-1 (8.52e-5) –	9.9995e-1 (2.47e-5) ≈	**9.9996e-1 (7.68e-6)** ≈	9.9996e-1 (6.45e-6)
Cybershake-DQN	50	9.9978e-1 (5.71e-5) –	9.9979e-1 (3.95e-5) ≈	9.9980e-1 (1.48e-5) ≈	**9.9981e-1 (1.35e-5)**
	75	9.9991e-1 (9.83e-6) –	9.9989e-1 (2.98e-5) –	9.9992e-1 (1.02e-5) ≈	**9.9992e-1 (1.37e-5)**
	100	9.9992e-1 (7.58e-5) –	9.9996e-1 (1.42e-5) ≈	9.9995e-1 (5.49e-6) –	**9.9996e-1 (4.52e-6)**
Inspiral-DQN	50	9.9975e-1 (2.12e-4) –	9.9989e-1 (4.65e-5) –	9.9991e-1 (2.71e-5) ≈	**9.9992e-1 (1.02e-5)**
	72	9.9984e-1 (1.35e-4) –	9.9995e-1 (3.12e-5) –	9.9996e-1 (2.35e-5) ≈	**9.9996e-1 (6.30e-6)**
	100	9.9991e-1 (7.93e-5) –	9.9997e-1 (2.25e-5) –	9.9997e-1 (1.42e-5) ≈	**9.9998e-1 (3.82e-6)**

to be similar to the MoEA-ECRS by Friedman test. PeEA is the worst, as its performance are inferior to the MoEA-ECRS in 7 cases and only similar to the MoEA-ECRS in 2 cases. The reason for the superior performance of our MoEA-ECRS algorithm is that it can make the whole population evolve in a good direction driven by the coordinated operation of mating and environment selection. Thus the MoEA-ECRS optimization operation can slow down the decline of selection pressure in the later stage of population evolution and balance well the CaD of solutions in the process of population evolution, so as to obtain a better decision-making solution for ECRS.

Figs. 2 to 4 compare the scheduling performance of different algorithms, in terms of the four objectives, for the three workflows that have undergone the DQN preprocessing with 100 tasks, respectively. We define the system failure metric as: system failure = 1 − system reliability. Since minimizing the system failure is equivalent to maximizing the system reliability, we can use the system failure as the fourth objective. The upper and lower quartile values in the box plot, which measure the solution distribution obtained on each objective, can be used together with the median value to evaluate the performance of an algorithm. Specifically, the smaller the



(a) Makespan Time

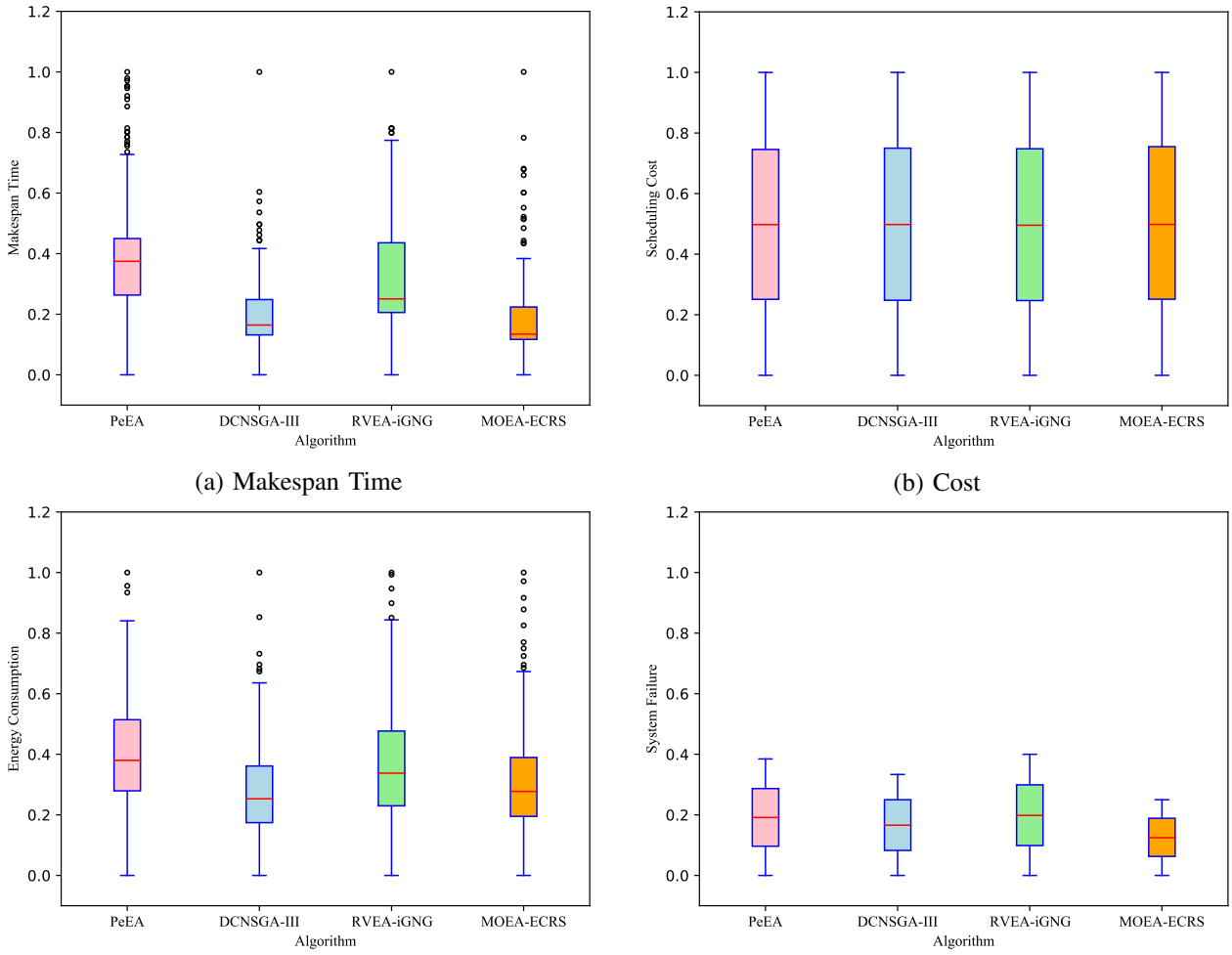
(b) Cost

(c) Energy Consumption

(d) System Failure

Fig. 3. Performance comparison of different objectives by four design algorithms on Cybershake-DQN-100

This article has been accepted for publication in IEEE Transactions on Computers. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TC.2023.3326977



(a) Makespan Time

(b) Cost

(c) Energy Consumption

(d) System Failure

Fig. 4. Performance comparison of different objectives by four design algorithms on Inspiral-DQN-100

upper and lower quartiles and the smaller the median value, the better the performance of the algorithm is.

a) Makespan time objective. Fig. 2 (a), Fig. 3 (a) and Fig. 4 (a) depict the solution distributions of the four algorithms for the three workflows, respectively.

a.1) Montage-DQN-100 workflow. Fig. 2 (a) shows that the median values of the four algorithms are not significantly different, with the median value of RVEA-iGNG slightly higher. In terms of solution distribution, PeEA and MoEA-ECRS are similar, which are tighter and more concentrated than DCNSGA-III and RVEA-iGNG. Hence the performance in terms of makespan time can be sorted as MoEA-ECRS ≈ PeEA > DCNSGA-III > RVEA-iGNG, where '≈' and '>' indicate 'similar to' and 'better than', respectively.

a.2) Cybershake-DQN-100 workflow. As can be seen from Fig. 3 (a), DCNSGA-III, RVEA-iGNG and MoEA-ECRS have similar quartile solution distributions but DCNSGA-III has smaller median value. The reason from relaxed dominant relationships of DCNSGA-III maintain a high proportion of non-dominant solutions in the population, which is particularly reflected in the completion time and energy consumption objectives. Although the quartile solution distribution of PeEA is tighter, it has higher median value. Therefore, we can sort the performance in terms of makespan time as DCNSGA-III > RVEA-iGNG ≈ MoEA-ECRS > PeEA.

a.3) Inspiral-DQN-100 workflow. By examining the distributions of upper and lower quartile solutions and comparing the median values of the four algorithm shown in Fig. 4 (a), their performance can be sorted according to MoEA-ECRS > DCNSGA-III > RVEA-iGNG > PeEA.

b) Cost objective. Fig. 2 (b), Fig. 3 (b) and Fig. 4 (b) depict the solution distributions, in terms of cost objective, for Montage-DQN-100, Cybershake-DQN-100 and Inspiral-DQN-100, respectively. It can be seen that the four algorithms have the same performance impact on cost consumption for the three workflows, based on their similar median values and the similar distributions of upper and lower quartile solutions. This may be attributed to the reference point strategy [26], [37] adopted by these algorithms. This strategy has certain preference in dealing with multi-modal cost function, which may be the reason leading to this cost solution landscape.

c) Energy consumption objective. There are some outliers in all the algorithms of Fig. 2 (c), Fig. 3 (c) and Fig. 4 (c), which are caused by some solutions not in the upper and lower

quartile value range. However, we can clearly observe that the algorithms have different performance from their overall distributions of upper and lower quartile solutions as well as the median values.

c.1) Montage-DQN-100 workflow. Clearly, from Fig. 2 (c), the performance of RVEA-iGNG is the poorest. The distributions of PeEA and DCNSGA-III are similar. By carefully observing the upper and lower quartiles values, it can be seen that MoEA-ECRS has a slightly tighter interval than PeEA and DCNSGA-III, which means that the solution obtained by our MoEA-ECRS has a higher probability of obtaining a small energy consumption. Overall the performance can be sorted according to MoEA-ECRS ≈ PeEA ≈ DCNSGA-III > RVEA-iGNG.

c.2) Cybershake-DQN-100 workflow. According to Fig. 3 (c), the ranking performance can be sorted according to DCNSGA-III > MoEA-ECRS > RVEA-iGNG > PeEA.

c.3) Inspiral-DQN-100 workflow. The landscape of Fig. 4 (c) is similar to that of Fig. 3 (c), and the performance can be sorted according to DCNSGA-III > MoEA-ECRS > RVEA-iGNG > PeEA.

d) System failure. Fig. 2 (d), Fig. 3 (d) and Fig. 4 (d) depict the solution distributions, in terms of the fourth objective, for the three workflows, respectively.

d.1) Montage-DQN-100 workflow. It can be seen clearly from Fig. 2 (d) that the ranking performance can be sorted as RVEA-iGNG > MoEA-ECRS > DCNSGA-III > PeEA based on the median values.

d.2) Cybershake-DQN-100 workflow. Clearly, Fig. 3 (d) shows that our MoEA-ECRS achieves the best performance, and the ranking performance can be sorted according to MoEA-ECRS > RVEA-iGNG > DCNSGA-III > PeEA.

d.3) Inspiral-DQN-100 workflow. It can be seen from Fig. 4 (d) that our MoEA-ECRS has clearly smaller upper and lower quartiles and median value than the other algorithms. Also PeEA and RVEA-iGNG have similar upper and lower quartile distribution lengths. The ranking performance can be sorted as MoEA-ECRS > DCNSGA-III > PeEA ≈ RVEA-iGNG.

Based on the above discussion, we conclude that the results of Figs. 2 to 4 again demonstrate the overall superior performance of our MoEA-ECRS over the other benchmarks.

V. Conclusions

In this paper, the ECRS problem has been transformed into a multi-objective MDP based on task dependency and ECRS feature extraction. We have built a multi-objective ECRS model by considering the makespan time, cost, energy consumption and system reliability. To solve this multi-objective ECRS problem, a hybrid approach based on DQN and MoEA-ECRS has been designed, which uses the DQN to preprocess workflow and then applies MoEA-ECRS to optimize the workflow scheduling decision. Our MoEA-ECRS strives to allocate more evolutionary efforts to the excellent solutions in the current population for mating selection, enabling better solution in terms of convergence. Once the excellent solution is determined, the environment selection mechanism is triggered to balance CaD. To verify the performance of our approach in solving the multi-objective ECRS problem, experiments have been performed on three real data sets with different task numbers. The simulation results have shown that the DQN preprocessing is superior than other common preprocessing methods. More significantly, the extensive simulation results have verified that our proposed MoEA-ECRS algorithm outperforms the existing benchmark algorithms.

References

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

[2] Q. Luo, S. Hu, C. Li, G. Li, and W. Shi, "Resource scheduling in edge computing: A Survey," *IEEE Communi. Surveys Tutorials*, vol. 23, no. 4, pp. 2131–2165, Fourthquarter 2021.

[3] F. Zhao, Y. Chen, Y. Zhang, Z. Liu, and X. Chen, "Dynamic offloading and resource scheduling for mobile-edge computing with energy harvesting devices," *IEEE Trans. Network and Service Management*, vol. 18, no. 2, pp. 2154–2165, Jun. 2021.

[4] J. Tang, *et al.*, "Slicing-based reliable resource orchestration for secure software-defined edge-cloud computing systems," *IEEE Internet of Things J.*, vol. 9, no. 4, pp. 2637–2648, Feb. 2022.

[5] M. S. Munir, *et al.*, "Risk-aware energy scheduling for edge computing with microgrid: A multi-agent deep reinforcement learning approach," *IEEE Trans. Network and Service Management*, vol. 18, no. 3, pp. 3476–3497, Jan. 2021.

[6] Z. Ning, *et al.*, "Deep reinforcement learning for intelligent Internet of vehicles: An energy-efficient computational offloading scheme," *IEEE Trans. Cognitive Communi. and Networking*, vol. 5, no. 4, pp. 1060–1072, Jul. 2019.

[7] S. Mao, S. He, and J. Wu, "Joint UAV position optimization and resource scheduling in space-air-ground integrated networks with mixed cloud-edge computing," *IEEE Systems J.*, vol. 15, no. 3, pp. 3992–4002, Sep. 2021.

[8] L. Rui, *et al.*, "Smart network maintenance in edge cloud computing environment: An allocation mechanism based on comprehensive reputation and regional prediction model," *J. Network and Computer Applications*, vol. 198, pp. 1–16, Feb. 2022.

[9] T. Dong, F. Xue, C. Xiao, and J. Zhang, "Workflow scheduling based on deep reinforcement learning in the cloud environment," *J. Ambient Intelligence and Humanized Computing*, vol. 12, no. 12, pp. 10823–10835, 2021.

[10] T. Dong, F. Xue, C. Xiao, and J. Zhang, "Deep reinforcement learning for dynamic workflow scheduling in cloud environment," in *Proc. CSC 2021* (Chicago, IL, USA), Sep. 5-10, 2021, pp. 107–115.

[11] M. Alaei, R. Khorsand, and M. Ramezanpour, "An adaptive fault detector strategy for scientific workflow scheduling based on improved differential evolution algorithm in cloud," *Applied Soft Computing*, vol. 99, 106895, pp. 1–16, Feb. 2021.

[12] S. Qin, D. Pi, Z. Shao, and Y. Xu, "Hybrid collaborative multi-objective fruit fly optimization algorithm for scheduling workflow in cloud environment," *Swarm and Evolutionary Computation*, vol. 68, 101008, pp. 1–14, Feb. 2022.

[13] H. Li, D. Wang, M. Zhou, Y. Fan, and Y. Xia, "Multi-swarm co-evolution based hybrid intelligent optimization for bi-objective multi-workflow scheduling in the cloud," *IEEE Trans. Parallel and Distributed Systems*, vol. 33, no. 9, pp. 2183–2197, Sep. 2022.

[14] Q.-Z. Xiao, J. Zhong, L. Feng, L. Luo, and J. Lv, "A cooperative coevolution hyper-heuristic framework for workflow scheduling problem," *IEEE Trans. Services Computing*, vol. 15, no. 1, pp. 150–163, Jan.-Feb. 2022.

[15] Y. Miao, *et al.*, "Intelligent task prediction and computation offloading based on mobile-edge cloud computing," *Future Generation Computer Systems*, vol. 102, pp. 925–931, Jan. 2020.

[16] X. Zhao, C. Lin, and J. Zhang, "Cloudlet deployment for workflow applications in a mobile edge computing-wireless metropolitan area network," *Peer-to-Peer Networking and Applications*, vol. 15, no. 1, pp. 739–750, Jan. 2022.

[17] K. Cao, L. Li, Y. Cui, T. Wei, and S. Hu, "Exploring placement of heterogeneous edge servers for response time minimization in mobile edge-cloud computing," *IEEE Trans. Industrial Informatics*, vol. 17, no. 1, pp. 494–503, Jan. 2021.

[18] A. Krishnakumar *et al.*, "Runtime Task Scheduling Using Imitation Learning for Heterogeneous Many-Core Systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 11, pp. 4064-4077, Oct. 2020.

[19] W. Zhan, *et al.*, "Deep-reinforcement-learning-based offloading scheduling for vehicular edge computing," *IEEE Internet of Things J.*, vol. 7, no. 6, pp. 5449–5465, Jun. 2020.

[20] Q. Luo, C. Li, T. H. Luan, and W. Shi, "Collaborative data scheduling for vehicular edge computing via deep reinforcement learning," *IEEE Internet of Things J.*, vol. 7, no. 10, pp. 9637–9650, Oct. 2020.

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

[22] S. Tuli, S. Ilager, K. Ramamohanarao, and R. Buyya, "Dynamic scheduling for stochastic edge-cloud computing environments using A3C learning and residual recurrent neural networks," *IEEE Trans. Mobile Computing*, vol. 21, no. 3, pp. 940–954, Mar. 2022.

[23] P. Zhang, Y. Zhang, H. Dong, and H. Jin, "Mobility and dependence-aware QoS monitoring in mobile edge computing," *IEEE Trans. Cloud Computing*, vol. 9, no. 3, pp. 1143–1157, Jul.-Sep. 2021.

[24] C. Li, J. Bai, Y. Ge, and Y. Luo, "Heterogeneity-aware elastic provisioning in cloud-assisted edge computing systems," *Future Generation Computer Systems*, vol. 112, pp. 1106–1121, Nov. 2020.

[25] Z. Liang, T. Luo, K. Hu, X. Ma, and Z. Zhu, "An indicator-based many-objective evolutionary algorithm with boundary protection," *IEEE Trans. Cybernetics*, vol. 51, no. 9, pp. 4553–4566, Sep. 2021.

[26] H. Ge, *et al.*, "A many-objective evolutionary algorithm with two interacting processes: Cascade clustering and reference point incremental learning," *IEEE Trans. Evolutionary Computation*, vol. 23, no. 4, pp. 572–586, Aug. 2019.

[27] G. Qu, H. Wu, R. Li, and P. Jiao, "DMRO: A deep meta reinforcement learning-based task offloading framework for edge-cloud computing," *IEEE Trans. Network and Service Management*, vol. 18, no. 3, pp. 3448–3459, Sep. 2021.

[28] Y. Ma, Y. Han, J. Wang, and Q. Zhao:, "A constrained static scheduling strategy in edge computing for industrial cloud systems," *Int. J. Information Technologies and Systems Approach*, vol. 14, no. 1, pp. 33–61, 2021.

[29] B. Cao, *et al.*, "Edge-cloud resource scheduling in space-air-ground-integrated networks for Internet of vehicles," *IEEE Internet of Things J.*, vol. 9, no. 8, pp. 5765–5772, Apr. 2022.

[30] L. Zhang, K. Li, C. Li, and K. Li, "Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems," *Information Sciences*, vol. 379, pp. 241–256, Feb. 2017.

[31] Y. Xiao, L. Xiao, K. Wan, H. Yang, Y. Zhang, Y. Wu, and Y. Zhang, "Reinforcement Learning Based Energy-Efficient Collaborative Inference for Mobile Edge Computing," *IEEE Transactions on Communications*, vol. 71, no. 2, pp. 864-876, Feb. 2023.

[32] K. J. Naik, M. Pedagandam, and A. Mishra, "Workflow scheduling optimisation for distributed environment using artificial neural networks and reinforcement learning," *Int. J. Computational Science and Engineering*, vol. 24, no. 6, pp. 653–670, Dec. 2021.

[33] P. Paknejad, R. Khorsand, and M. Ramezanpour, "Chaotic improved PICEA-g-based multi-objective optimization for workflow scheduling in cloud environment," *Future Generation Computer Systems*, vol. 117, pp. 12–28, 2021.

[34] X. Cai, J. Zhang, H. Liang, L. Wang, and Q. Wu, "An ensemble bat algorithm for large-scale optimization," *Int. J. Machine Learning and Cybernetics*, vol. 10, pp. 3099–3113, 2019.

[35] H. Chen, *et al.*, "Hyperplane assisted evolutionary algorithm for many-objective optimization problems," *IEEE Trans. Cybernetics*, vol. 50, no. 7, pp. 3367–3380, Jul. 2020.

[36] R. Cheng, Y. Jin, M. Olhofer, and B. Sendhoff, "A reference vector guided evolutionary algorithm for many-objective optimization," *IEEE Trans. Evolutionary Computation*, vol. 20, no. 5, pp. 773–791, Oct. 2016.

[37] Q. Liu, Y. Jin, M. Heiderich, T. Rodemann, and G. Yu, "An adaptive reference vector-guided evolutionary algorithm using growing neural gas for many-objective optimization of irregular problems," *IEEE Trans. Cybernetics*, Early Access, Oct. 2020, DOI:10.1109/TCYB.2020.3020630.

[38] Z. Cui, *et al.*, "Hybrid many-objective particle swarm optimization algorithm for green coal production problem," *Information Sciences*, vol. 518, pp. 256–271, May 2020.

[39] J. Zhang, *et al.*, "Privacy protection based on many-objective optimization algorithm," *Concurrency and Computation Practice and Experience*, vol. 31, no. 20, pp. 1-14, May 2019.

[40] X. Cai, J. Zhang, Z. Ning, Z. Cui, and J. Chen, "A many-objective multistage optimization-based fuzzy decision-making model for coal production prediction," *IEEE Trans. Fuzzy Systems*, vol. 29, no. 12, pp. 3665–3675, Dec. 2021.

[41] A. Kaur, P. Singh, R. S. Batth, and C. P. Lim, "Deep-Q learning-based heterogeneous earliest finish time scheduling algorithm for scientific workflows in cloud," *Software – Practice and Experience*, vol. 52, no. 3, pp. 689–709, Mar. 2022.

[42] J. Liang, K. Li, C. Liu, and K. Li, "Are task mappings with the highest frequency of servers so good? A case study on heterogeneous earliest finish time (HEFT) algorithm," *J. Systems Architecture*, vol. 121, no. 102311, pp. 1–13, Dec. 2021.

[43] R. Jiao, S. Zeng, C. Li, S. Yang, and Y.-S. Ong, "Handling constrained many-objective optimization problems via problem transformation," *IEEE Trans. Cybernetics*, vol. 51, no. 10, pp. 4834–4847, Oct. 2021.

[44] L. Li, G. G Yen, A. Sahoo, L. Chang, and T. Gu, "On the estimation of Pareto front and dimensional similarity in many-objective evolutionary algorithm," *Information Sciences*, vol. 563, pp. 375–400, Jul. 2021.

[45] M. Abd Elaziz and S. Lu, "Many-objectives multilevel thresholding image segmentation using Knee evolutionary algorithm," *Expert Systems with Applications*, vol. 125, pp. 305–316, Jul. 2019.

Jiangjiang Zhang is pursuing his Doctor degree in Beijing University of Technology, China. His research interest includes data security, privacy protection, computational intelligence, combinatorial optimization and modelling.

This article has been accepted for publication in IEEE Transactions on Computers. This is the author's version which has not been fully edited and content may change prior to final publication. Citation information: DOI 10.1109/TC.2023.3326977

Zhenhu Ning received the Ph.D. degree in Faculty of Information Technology, Beijing University of Technology, Beijing, in 2016, where he is currently with the Faculty of Information Technology. His research interests include the field of security, including wireless sensor networks, sensing data transmission and computing environment of the sensing node, cloud storage security and privacy, machine learning, system optimization and control, and practical partial differential equations.

MUHAMMAD WAQAS (M'18, SM'22) received his PhD degree (Sept. 2015 – Jun. 2019) with the Department of Electronic Engineering, Tsinghua University, Beijing, China. From Aug. 2019 to Mar.. 2022, he served Faculty of Computer Science and Engineering, GIK Insitute of Engineering Sciences and Technology, Pakistan as an Assistant Professor. He was also associated with the Faculty of Information Technology, Beijing University of Technology, Beijing, China as Research Associate from Oct. 2019 to Sept. 2022. Currently, he is an Senior Lecturer at the School of Computing and Mathematical Sciences, Faculty of Engineering and Science, University of Greenwich, London, UK. He is also an Adjunct Senior Lecturer at the School of Engineering, Edith Cowan University, Perth, Australia. His current research interests are in the areas of physical layer security, vehicular networks, mobile edge computing and the internet of things.

Hisham Alasmary is an Assistant Professor at King Khalid University. He obtained his Ph.D. from the Department of Computer Science at the University of Central Florida in 2020, and his M.Sc. degree in Computer Science from The George Washington University, in Washington, D.C., USA, in 2016. His research interests include Software Security, IoT Security and Privacy, ML/DL Applications in Information Security, and Adversarial Machine Learning.

Shanshan Tu received his PhD degree from Computer Science Department at Beijing University of Posts and Telecommunications in 2014. From 2013 to 2014, he visited University of Essex for National Joint Doctoral Training. He worked in the Department of Electronic Engineering at Tsinghua University as a postdoctoral researcher from 2014 to 2016. He is currently an Associate Professor and Deputy Dean at Faculty of Information Technology, Beijing University of Technology, China. His research interests are in the areas of cloud computing, MEC and information security techniques.

Sheng Chen received his PhD degree from the City University, London, in 1986, both in control engineering. In 2005, he was awarded the higher doctoral degree, Doctor of Sciences (DSc), from the University of Southampton, Southampton, UK. Since 1999, he has been with the School of Electronics and Computer Science, the University of Southampton, UK, where he holds the post of Professor in Intelligent Systems and Signal Processing. Dr Chen's research interests include neural network and machine learning, adaptive signal processing, wireless communications, modelling and identification of nonlinear systems, evolutionary computation methods and optimization. He has published over 700 research papers. Dr. Chen is a Fellow of the United Kingdom Royal Academy of Engineering, a Fellow of IEEE, and a fellow of IET.