The Application of Optimization Algorithms for Workflow Scheduling Based on Cloud Computing IaaS Environment in Industry Multi-Cloud Scenarios

Cunbing Li
Inspur Software Technology Co., Ltd., Ji'nan, 250101, China

***Abstract*—The advancement of cloud computing has enabled workflow scheduling to provide users with more network resources. However, there are some scheduling issues between resource allocation and user needs in workflows in IaaS environments. Based on this, this study adopts a heuristic scheduling model based on deadline and list and constructs a single objective workflow scheduling model based on deadline. Based on fuzzy-dominated sorting, traditional non-dominated sorting is improved to construct a time-cost dual objective workflow scheduling model. Introducing evolutionary algorithms with a reliability index as the scheduling objective, a time-cost reliability three-objective workflow scheduling model is constructed. The results showed that the total execution time of the single objective workflow scheduling model in four standard workflows was 92s, 106s, 113s, and 105s, respectively. The throughput was 144b/s, 138b/s, 140b/s, and 142b/s, respectively, all of which were superior to other models. Compared with other comparative models, the dual objective workflow scheduling model and the three objective workflow scheduling model had higher HV values, less execution time, and better Pareto frontier solutions. This study solves the three objective scheduling problem of time cost reliability in IaaS environment, which has a certain reference value in resource scheduling on cloud platforms.**

***Keywords*—Cloud computing; IaaS; scheduling model; evolutionary algorithms; heuristic model**

I. Introduction

Cloud computing (CC) is an internet-based computing model that enables on-demand access and utilization of computing resources by centralizing and providing computing resources to users. In the infrastructure as service (IaaS) environment, workflow scheduling refers to allocating a workflow composed of multiple tasks to available virtual machines for execution, which can optimize the execution time and resource utilization of the workflow [1]. Workflow scheduling needs to consider issues such as resource management, task scheduling, task allocation, and optimization objectives. Tasks are scheduled based on the dependencies and scheduling constraints between tasks by allocating and managing resources reasonably. Then, tasks are assigned to appropriate virtual machines. Workflow execution efficiency and resource utilization are improved through optimization algorithms [2]. However, in the IaaS environment, the application of scientific workflows faces an imbalance between user needs and resource allocation, making it impossible to achieve reasonable scheduling of network resources. Based on this, this study constructs a single objective workflow scheduling model based on deadline. It is based on fuzzy dominance sorting and constructs a time-cost dual objective workflow scheduling model. It introduces evolutionary algorithms to construct a time-cost reliability three objective workflow scheduling model, achieving efficient scheduling of workflows in the IaaS environment.

II. Related Works

CC is a new type of internet computing model. Some scholars have conducted relevant research on CC environments. B Mukhopadhyay et al. found that there were some security issues in CC infrastructure. Traditional secure socket layers and related security models had certain flaws. Based on this, this study proposed a new security problem solving framework. The experimental results showed that the framework promises to provide high availability, redundancy, load balancing, and secure data channels simultaneously [3]. Suryateja believed that the OpenStack management platform in CC was complex and had an impact on the underlying hardware utilization of the host system. Based on this, the paper analyzed the impact of OpenStack on host hardware and addressed the resource waste of OpenStack on hardware. This experiment demonstrated the effectiveness of this method by comparing various indicator data [4]. Sharma believed that CC led a new trend in IT information computing and storage, changing traditional IT businesses. Based on this, the paper introduced the development and deployment models of CC, including private cloud, public cloud, hybrid cloud, and community cloud. The paper discussed various service models. Then, the paper made improvements to address the information storage and data allocation in CC [5]. Modisane et al. found that small and medium-sized enterprises generally lacked resource capabilities. CC provided them with the ability to access high-level ICT services. Small and medium-sized enterprises could achieve many benefits through reducing capital expenditure, improving access to network systems, improving data security, and reducing agile development costs among the numerous advantages of CC.

Workflow scheduling in the IaaS environment can meet users' network resource requirements. Some scholars have conducted relevant research on workflow scheduling models. P Chen et al. found that there were some limitations in traditional multi-objective workflow scheduling in CC environments. When scheduling in real-time on dynamic cloud infrastructure, invalid issues might arise [6]. Based on this, it proposed a new IaaS multi-workflow scheduling algorithm based on

reinforcement learning, aiming to optimize manufacturing span and dwell time, which achieved a unique set of related equilibrium solutions. The simulation experiment results showed that compared with current advanced models, the performance of this model was superior to other models [7]. Ramadhan et al. found that meeting the quality of service needs of users was urgent to be solved in CC environments. Based on this, three particle swarm optimization algorithms were compared in terms of time span and cost, which were tested using the same number of virtual machines and workflows. The experimental results aimed to select the optimal model for CC platforms to achieve workflow scheduling [8]. Li et al. believed that in CC environments, there were issues of network congestion and uneven resource allocation. Based on this, a new multi-workflow scheduling model using edge computing resources for location awareness and proximity constraints was proposed. The proposed method could minimize monetary costs while meeting the user's required workflow completion deadline. This method used evolutionary algorithms to generate nearly optimal scheduling decisions. The experimental results showed that the performance of this model was superior to other existing advanced algorithms [9]. Ghafouri et al. believed that traditional workflow scheduling models that constrain time and reduce costs did not take into account the variability of virtual machine performance. Therefore, a workflow scheduling model based on constraint time and cost reduction was proposed, which was called an adaptive constraint time and cost reduction scheduling algorithm. The experimental results showed that the algorithm took into account the performance changes of virtual machines in the cloud environment and could complete tasks within a limited time [10].

In summary, scientific workflow scheduling in the CC environment achieves the scheduling problem between network resources and user needs. However, many studies only consider a single optimization objective and lack scheduling research on the three objectives of time cost reliability. Therefore, this study introduces evolutionary algorithms and constructs a three objective workflow scheduling model of time cost reliability by considering the target scheduling of time and cost. This provides certain reference value for multi-objective efficient scheduling of workflows in IaaS environments.

III. Methods

To achieve efficient scheduling of workflows in the IaaS environment, this chapter is divided into two parts to construct a target workflow scheduling model. The first part constructs a heuristic scheduling model based on deadline to achieve single objective workflow scheduling with time. The second part constructs a time-cost dual objective and time-cost reliability three objective workflow scheduling model based on fuzzy dominated sorting and evolutionary algorithms.

A. Construction of a single objective workflow scheduling model Based on deadline

IaaS is one of the CC service models, where users can deploy and run their applications using virtual machines, storage, networks, and other infrastructure resources provided by cloud service providers. The schematic diagram of the cloud server is shown in Fig. 1.

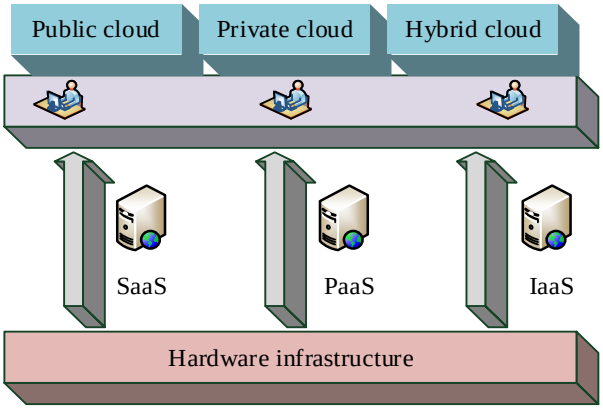
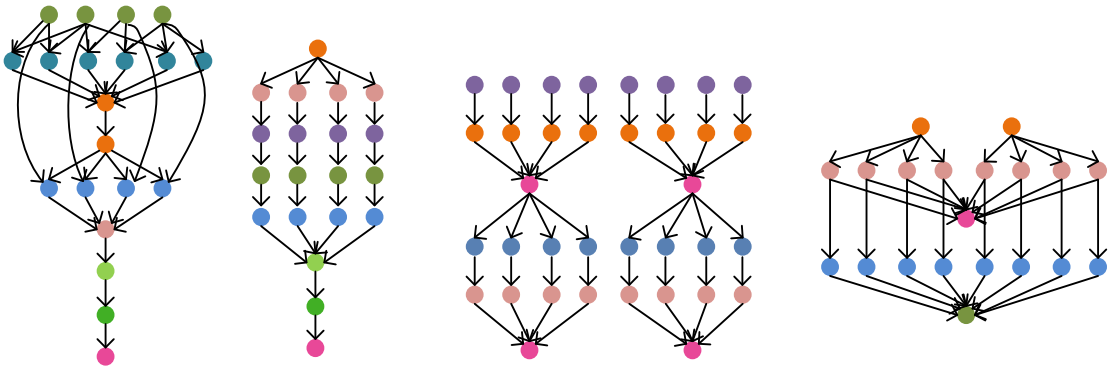


Fig. 1. Schematic diagram of cloud server.

In traditional network environments, completion time is the most important optimization goal in workflow scheduling models. However, in CC environments, cost has also become an optimization goal within the scope of user needs consideration. The common workflow structure diagram is shown in Fig. 2.



(a) Montage (b) Epigenomics (c) LIGO (d) CyberShake

Fig. 2. Common workflow structure diagram.

Common standard workflows include Montage, Epigenomics, LIGO, and CyberShake. Workflows decompose computing tasks into a series of processing steps, each of which generates output data based on input data, forming an executable computing process [11-12]. Based on this, an effective scheduling scheme is shown in Eq. (1).

$$s = (V, o, T2V, V2P) \tag{1}$$

In Eq. (1), s represents an effective scheduling scheme represented by quads. V represents a collection of virtual machines. o represents the task sequence of the target scheduling order. $T2V$ represents a mapping function from executing tasks to virtual machines. $V2P$ represents the mapping function between virtual machine instances and virtual machine configurations. It is necessary to calculate the total completion time of the scheduling scheme to complete the calculation within a limited time. The start time of the task is shown in Eq. (2).

$$St(\tau_i) = \max\{avail(v_a), \max_{\tau_j \in P_{\tau_i}}(Ft(\tau_j) + Act(\tau_j, \tau_i))\} \tag{2}$$

In Eq. (2), τ_i represents the task to be executed. $St(\tau_i)$ represents the start time of the task. $Ft(\tau_j)$ represents the end time of the task. v_a represents the virtual machine instance where task τ_i is located. $avail(v_a)$ represents the earliest available time of the virtual machine instance. $Act(\tau_j, \tau_i)$ represents the communication time between tasks. The formula for calculating the end time of a task is shown in Eq. (3).

$$Ft(\tau_j) = St(\tau_i) + Aet(\tau_i) \tag{3}$$

In Eq. (3), $Aet(\tau_i) s = (V, o, T2V, V2P)$ represents the time the task runs on the virtual machine. Therefore, the calculation of the end time of a task is equivalent to the start time of the task plus the time the task runs on the virtual machine. In addition to the completion time, cost is also an optimization objective that needs to be considered. The usage cost of a virtual machine is related to the configuration type and usage time of the virtual machine. The usage time should include the total time of all virtual machines from the beginning of the first task to the end of all tasks, as shown in Eq. (4).

$$Et(v) = \max_{T2V(\tau)=v} Ft(\tau) - \min_{T2V(\tau)=v} St(\tau) \tag{4}$$

In Eq. (4), $Et(v)$ represents the usage time of virtual machine v. The execution cost of virtual machine v is related to the duration. Therefore, the total execution cost of the workflow is shown in Eq. (5).

$$\mathrm{Cost}(s) = \sum_{v \in V} Price(v) \cdot Et(v) \tag{5}$$

In Eq. (5), $Price(v)$ represents the cost per unit time of virtual machine v. Therefore, by combining time and cost, the optimization goals for the IaaS platform's time and cost can be obtained, as shown in Eq. (6).

$$\begin{cases} Minimize\ \mathrm{Cost} \\ Subject\ to\ Makespan \le Deadline \end{cases} \tag{6}$$

In Eq. (6), $Deadline$ represents the deadline given by the user. Therefore, the optimization goal of time and cost is to minimize the cost and complete it within the specified time. Based on the deadline and list, a heuristic scheduling algorithm (deadline constrained workflow scheduling for IaaS, DCWS) is constructed. DCWS is divided into task priority sorting stage and service selection stage, with each task assigned a sub-deadline period to select the most suitable virtual machine instance [13]. In the service selection phase, the average running time of a task τ on virtual machine instances of different configuration types is shown in Eq. (7).

$$Aet(\tau) = \frac{1}{m} \sum_{i=1}^{m} Et(\tau_i, p_i) \tag{7}$$

In Eq. (7), $Et(\tau_i, p_i)$ represents the running time of the task. p_i represents the configuration type. The calculation formula for the sub-deadline period is shown in equation (8).

$$\begin{cases} D(\tau_i) = Deadline;\ if\ \tau_i = \tau_{exit} \\ D(\tau_i) = min(D(\tau_i) - (Act(\tau_j, \tau_i) + Aet(\tau_i))); \\ if\ \tau_i \ne \tau_{exit}, \tau_j \in S_{\tau_i} \end{cases} \tag{8}$$

In Eq. (8), τ_{exit} represents the sub-deadline period of the task. $Act(\tau_j, \tau_i)$ represents the communication time for transmitting data between tasks. S_{τ_i} represents a collection of tasks. Therefore, the total workflow deadline can be calculated based on the sub-deadline period of each task. Virtual machines that meet the deadline are retained. In the task priority sorting stage, a rank value is given based on the priority of the task and arranged in descending order. The DCWS algorithm finds the time spent on each task, as shown in Eq. (9).

$$\begin{cases} Est(\tau_i) = 0;\ if\ \tau_i = \tau_{entry} \\ Est(\tau_i) = \max(Et(\tau_j) + Act(\tau_j, \tau_i)); \\ if\ \tau_i \ne \tau_{entry}, \tau_j \in S_{\tau_i} \\ Lft(\tau_i) = Ct;\ if\ \tau_i = \tau_{exit} \\ Lft(\tau_i) = \min(Ct(\tau_j) - Et(\tau_j) + Act(\tau_j, \tau_i))); \\ if\ \tau_i \ne \tau_{exit}, \tau_j \in P_{\tau_i} \end{cases} \tag{9}$$

In Eq. (9), $Est(\tau_i)$ represents the earliest start time of the task. $Et(\tau_j)$ represents the latest completion time. Ct represents the end time of the workflow. $Ct(\tau_j)$ represents the completion time of non-end tasks. $Lft(\tau_i)$ represents the latest end time of the task. The rank value of the task can be obtained by adding the earliest start time and the latest end time of the task. Sorting based on the rank value, the optimal scheduling scheme for deploying the task to the virtual machine can be obtained.

B. Construction of Multi-Objective Workflow Scheduling Model Based on Fuzzy Dominated Sorting and Evolutionary Algorithm

Due to the conflict between multiple objectives in the workflow scheduling problem that minimizes time and cost, it is necessary to optimize the multi-objective problem [14]. The traditional algorithm for solving time and cost optimization problems is the heterogeneous early best time (HEFT) algorithm. However, this algorithm requires complete traversal of all tasks to select the optimal scheduling solution, which has high complexity. Compared with traditional non-dominated sorting methods, fuzzy dominated sorting can easily compare one solution with another and evaluate them based on their fuzzy dominated relationship, with fast convergence properties [15]. Based on this, the concept of fuzzy dominance sort is introduced for improvement, and the fuzzy dominance sort-based heterogeneous early perfect time (FDHEFT) algorithm is constructed to solve the dual objective optimization problem [16]. There are three definitions in fuzzy dominance ranking, namely the dimensional fuzzy dominance of the solution, the fuzzy dominance of the solution, and the fuzzy dominance of the solution and the set. The formula for calculating the dimensional fuzzy dominance of the solution is shown in Eq. (10).

$$\mu_r^{dom}(f_r(v)-f_r(u))=\mu_r^{dom}(u\succ {}^{F}_{r}r) \tag{10}$$

In Eq. (10), $f_r(\cdot)$ represents the objective function to be optimized. μ_r^{dom} represents a monotonic non-decreasing function with a value of [0,1]. u and v represent two different solutions. When $f_r(v)>f_r(u)$, solution u can be seen as the r-dimensional fuzzy dominated solution of solution v, which can be represented as v. Fuzzy domination can also be achieved through the probability of fuzzy intersection operation. The formula for calculating the fuzzy domination of the solution is shown in Eq. (11).

$$\mu^{dom}(u\succ {}^{F}r)=\bigcap_{r=1}^{M}\mu_r^{dom}(u\succ {}^{F}_{r}v) \tag{11}$$

In Eq. (11), $\bigcap_{r=1}^{M}\mu_r^{dom}$ represents the fuzzy intersection operation. It sets a solution set S, where solution set $v\in S$. When fuzzy dominated by any solution u in the solution set, v is fuzzy dominated in the solution set. Therefore, based on the μ_r^{dom} function, it can be determined that each solution has a fuzzy dominant value. When the fuzzy dominance value is lower, it indicates that the solution is better. Based on this method, better solutions can be selected [17]. As the FDHEFT algorithm is improved based on the HEFT algorithm, there are two stages for selecting the optimal solution, namely task priority sorting and virtual machine selection. Firstly, it inputs the task set, calculates the node set and the communication cost between tasks, and selects a rank value for each task as the initial scheduling scheme [18]. According to the fuzzy dominance sorting method, tasks are sorted and their priority is determined. The calculation time, communication time, and other indicators of each task can be considered. The superiority and inferiority of each task can be calculated through fuzzy dominance relationships [19]. It schedules tasks in descending order based on their priority. For each task, it is necessary to find the node among the available computing nodes that can complete the task the earliest as the scheduling goal, taking into account the calculation time and communication time of the task. Meanwhile, it considers the occupancy of system resources and the dependencies between tasks. It assigns tasks to corresponding computing nodes for execution and updates the execution status and available resources of the computing nodes [20]. In the virtual machine selection stage, the FDHEFT algorithm sorts all scheduling schemes generated based on fuzzy dominance values to select the optimal virtual machine configuration type. When the fuzzy dominated solutions are the same, a diversity fitness function is used to compare the solutions. The value solved by the diversity function is equivalent to the boundary value of the maximum objective space, which represents the sparsity of the solution [21]. The formula for calculating the boundary value of solution S_l is shown in Eq. (12).

$$P(S_l)=\sum_{r=1}^{M}\frac{f_r(u)-f_r(v)}{\max f_r-\min f_r} \tag{12}$$

In Eq. (12), M represents the number of objective functions. u and v represent solutions with the same fuzzy dominance value. $f_r(u)$ and $f_r(v)$ represent the r-th target values of solutions u and v, respectively. $\max f_r$ and $\min f_r$ represent the maximum and minimum values of the r-th objective of all solutions. When the fuzzy dominance values are the same, choosing a solution with a large boundary value is beneficial for ensuring the diversity of solutions. Due to the inevitable occurrence of various errors during software operation, malfunctions may occur [22]. Therefore, in the workflow scheduling model, in addition to time and cost issues, it is also necessary to consider the reliability of operation. It sets a workflow scheduling scheme as s, and the calculation formula for reliability is shown in Eq. (13).

$$R(s)=\prod_{j=1}^{k}R_j(s)=e^{-\sum_{j=1}^{k}\lambda_j\cdot Et(v_j)} \tag{13}$$

In Eq. (13), v_j represents a virtual machine. λ_j. represents the failure coefficient of virtual machine v_j. $R(s)$ represents the reliability of the workflow scheduling scheme. $\prod_{j=1}^{k}R_j(s)$ represents the product of the probability of successfully completing the task. $Et(v)$ represents the usage duration of virtual machine v_j. Due to using the maximum reliability value as the scheduling goal, the other two goals will become the minimum values, so the reliability index (RI) is used as the scheduling goal. The calculation formula for RI is shown in Eq. (14).

$$RI(s)=\sum_{j=1}^{k}\lambda_j\cdot Et(v_j) \tag{14}$$

In Eq. (14), $RI(s)$ represents the reliability index. Due to the fact that time cost reliability is a multi-objective optimization problem, the MEAC algorithm is a multi-objective evolutionary algorithm on the cloud (MEAC)-based workflow scheduling optimization algorithm specifically designed for IaaS environments. It uses a fitness function to measure the quality of the three objective optimization solutions [23], as shown in Eq. (15).

$$\begin{cases} F_{RI(s)} = \sum_{j=1}^{k} \lambda_j \cdot Et(v_j) \\ F_{makespan}(s) = Ft(\tau_{exit}) \\ F_{cost}(s) = \sum_{j=1}^{k} \Pr ice(v_j) \cdot Et(v_j) \end{cases} \tag{15}$$

In Eq. (15), $F_{makespan}(s)$ represents the fitness function of the total completion time. $F_{cost}(s)$ represents the fitness function of the total execution cost. $F_{RI(s)}$ represents the fitness function of RI. In evolutionary algorithms, a group of individuals is initialized randomly or according to the characteristics of the problem as a population. Each individual is evaluated according to the objective function of the problem to obtain fitness values [24]. A portion of individuals is selected as parents based on their fitness values to generate offspring. The selected parent individuals are cross operated to generate a certain number of offspring individuals. The cross operation is shown in Fig. 3.

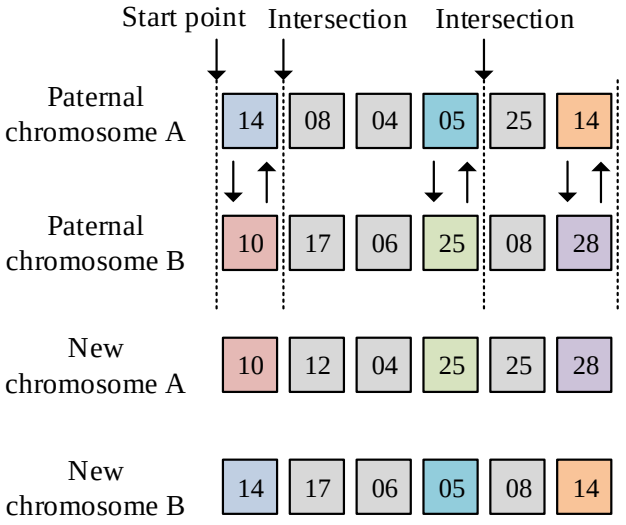


Fig. 3. Cross operation.

Mutation operations are performed on the generated offspring individuals, introducing random perturbations. It evaluates the fitness of the generated offspring individuals and updates the individuals in the population through selection and replacement operations [25]. Finally, it determines whether the termination condition is me. If so, the optimal solution found is outputted. The flowchart for constructing single objective and multi-objective workflow optimization scheduling models is shown in Fig. 4.

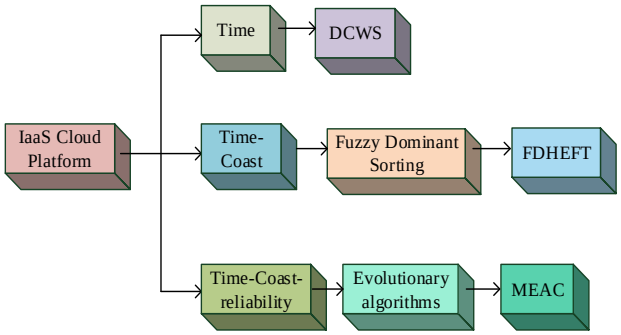


Fig. 4. Model construction flow chart.

IV. Model Experiments and Analysis

To verify the performance of the constructed model, this chapter is divided into three parts for model testing. The first part tests and analyzes the DCWS model. The second part tests and analyzes the FDHEFT model. The third part tests and analyzes the MEAC model.

A. DCWS Model Experiment and Analysis

To test the performance of the DCWS model, the experimental environment for this experiment is a 3.5GHz Intel Core i5 CPU and 16GB of running memory, with a universal computing framework of JMetal. The configuration information of the virtual machine is shown in Table I.

It compares the execution time of the DCWS model with the IC-PCP (IC-PCP) model, JIT (Just in time) model, and PDC (Plan Do Check Act) model in four standard workflows. The total execution time comparison results are shown in Fig. 5.

Fig. 5 shows that the DCWS model has the lowest total execution time in Montage, Epigenomics, LIGO, and CyberShake standard workflows, with 92s, 106s, 113s, and 105s, respectively. In Montage, the DCWS model reduces time by 30.8%, 24.2%, and 14.8% compared to the IC-PCP, JIT, and PDC models, respectively. In Epigenomics, the DCWS model reduces time by 24.3%, 22.1%, and 13.1% compared to the IC-PCP, JIT, and PDC models, respectively. In LIGO, the DCWS model reduces time by 18.1%, 14.5%, and 12.4% compared to the IC-PCP, JIT, and PDC models, respectively. In CyberShake, the DCWS model reduces time by 23.3%, 19.2%, and 4.2% compared to the IC-PCP, JIT, and PDC models, respectively. It compares the throughput of the four models, and the results are shown in Fig. 6.

Fig. 6 shows that the throughput of DCWS in Montage, Epigenomics, LIGO, and CyberShake standard workflows is 144b/s, 138b/s, 140b/s, and 142b/s, respectively. Compared to the other three models, it has the highest throughput, so the DCWS model can complete more tasks.

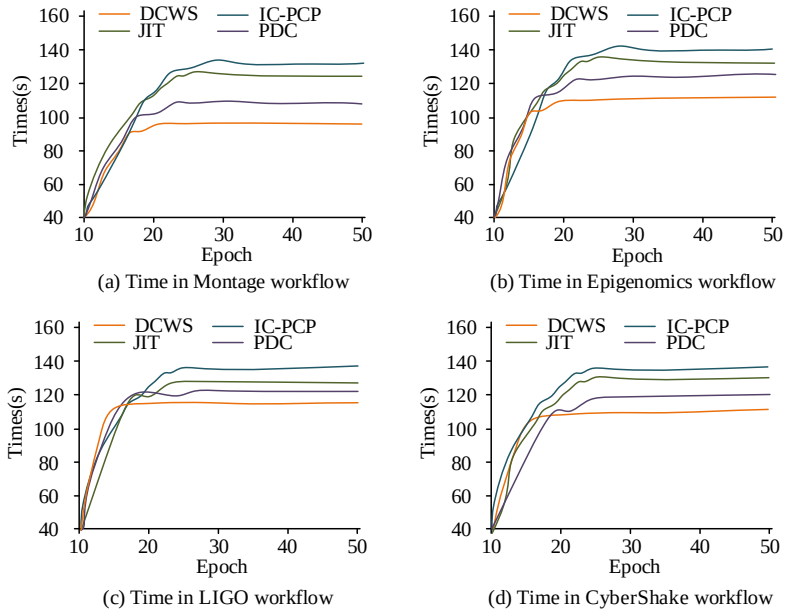
CyberShake, Inspiral, Montage, and Sipht are all derived from the scientific workflow management system Pegasus, which are widely used in the workflow scheduling to evaluate the performance of different scheduling algorithms. For the composite workflow diagram, the use of a utility program called Workflow Generator is investigated to generate the workflow to extend the test on a larger scale workflow. The composite workflow is consistent with the standard workflow

types described above, that is, the composite workflow has a similar structure to the standard workflow. The workflow is synthesized using workflow generator range in number from 20 to 100 tasks, and the main directed acyclic graph (DAG) features of the synthesized workflow are shown in Table II.

According to the synthesized workflow shown in Table II, the throughput and task completion rate of the research extraction method are analyzed, and the results are shown in Fig. 7. The throughput of a server is defined as the number of tasks completed by the server in a time interval. The experimental results are shown in Fig. 7(a). Existing algorithms do not find the most appropriate virtual machine instance for the task to minimize execution time. However, the DCWS algorithm selects the best virtual machine instance for the task based on the task's subcut-off time, which reduces the task's execution time and increases throughput. The throughput of DCWS algorithm is 18%, 13% and 11% higher than that of IC-PCP, JIT, and PDC, respectively. The comparison results of task completion rate are shown in Fig. 7(b). The DCWS algorithm deploys the task to the most suitable virtual machine based on the task requirements. The percentage of the research algorithm is higher than other algorithms, which can indicate that DCWS algorithm is more reliable than other existing algorithms. The task completion rate of DCWS algorithm is 22%, 17%, and 14% higher than that of IC-PCP, JIT, and PDC, respectively.

TABLE I. VIRTUAL MACHINE PARAMETER CONFIGURATION

Configuration information	m4.large	m4.xlarge	m4.2xlarge	m4.4xlarge	m4.10xlarge	m3.medium	m3.large	m3.xlarge	m3.2xlarge
CPU	2	4	8	16	40	1	2	4	8
ECU	6.5	13	26	53.5	124.5	3	6.5	13	26
Bandwidth	56.25	93.75	125	250	500	56.25	56.25	62.5	125
Price	0.120	0.239	0.479	0.958	2.394	0.067	0.133	0.266	0.532



(a) Time in Montage workflow (b) Time in Epigenomics workflow

(c) Time in LIGO workflow (d) Time in CyberShake workflow

Fig. 5. Time of different models in different workflows.

TABLE II. THE DAG CHARACTERISTIC TABLE OF SYNTHETIC WORKFLOW BASED ON WORKFLOW GENERATOR TOOL PROGRAM

Workflow	Number of nodes	Edge number	Average data size (MB)	Mean time to completion (s)
CyberShake	30	114	746.92	23.74
	50	188	861.49	29.85
	100	384	847.21	31.54
Inspiral	30	94	9.01	206.78
	50	169	9.17	227.51
	100	321	8.96	206.46
Montage	30	97	3.47	8.47
	50	210	3.68	9.72
	100	436	3.27	10.63
Sipht	30	91	7.78	176.85
	50	197	6.91	194.36
	100	329	6.34	176.63

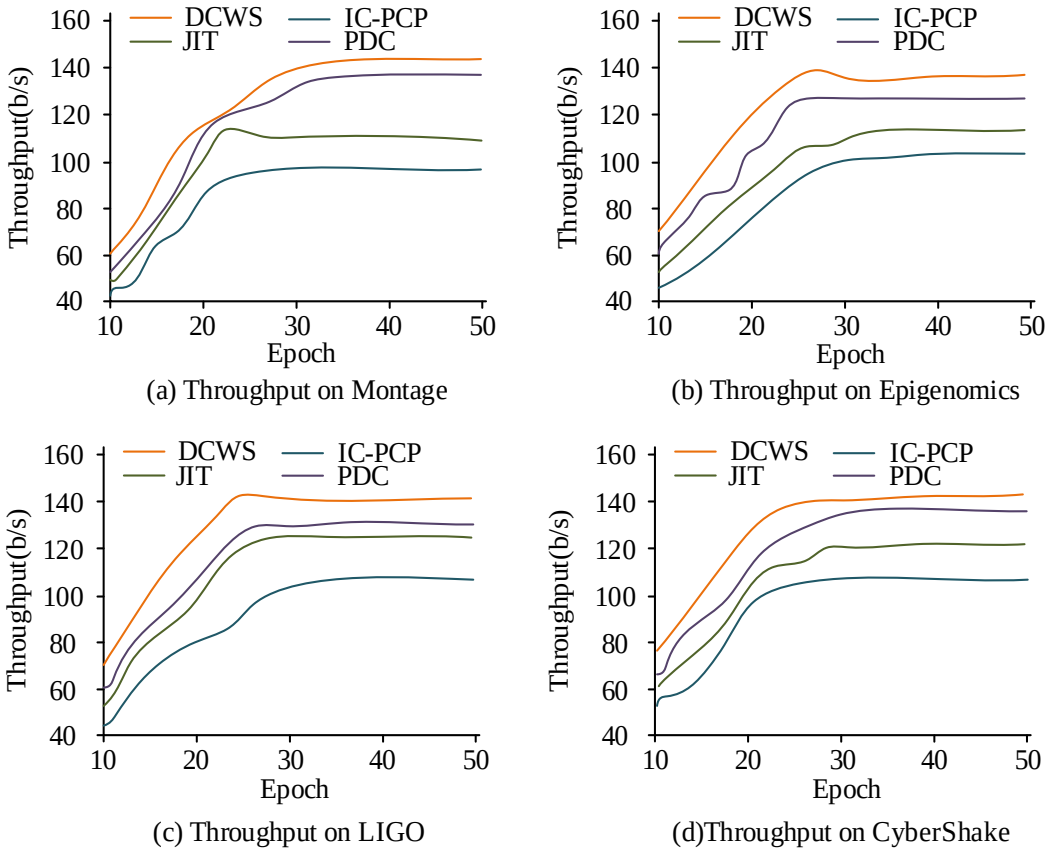


Fig. 6. Comparison of throughput in different workflows.

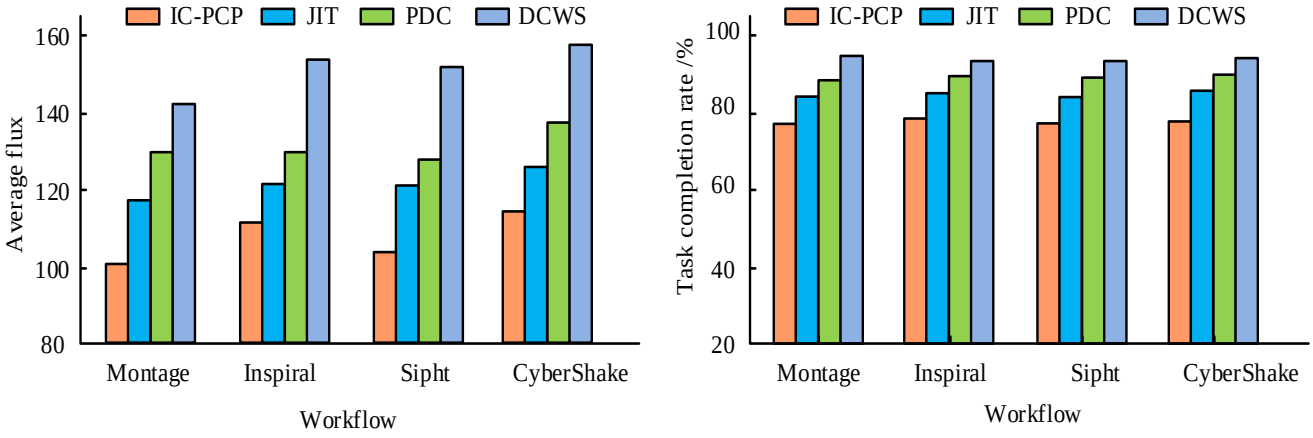


Fig. 7. Comparison of algorithm optimization performance.

B. FDHEFT Model Testing and Analysis

To test the performance of the FDHEFT model, the FDHEFT model is combined with the Multi-objective heterogeneous earliest completion time algorithm (MOHEFT) A multi-objective PSO algorithm based on crowding distance and mutation dominance (NSPSO) and an improved non-dominated sorting genetic algorithm (strength pareto evolutionary algorithm2 *, SPEA2*) are compared and tested. The population size is set to 100, and the probabilities of crossover and mutation are 0.9 and 0.2, respectively. The Pareto frontier solutions for time and cost are shown in Fig. 8.

In Fig. 8, each point represents a scheduling scheme. This indicates that the frontier solution obtained by the FDHEFT model is superior to the other three models and can provide better time-cost solutions. In a randomly generated workflow, the ratio of HV value to time is used as an indicator to measure the distribution of the model's solution set. The higher the HV value, the better the model's performance. The HV indicators and time results of the FDHEFT model and other models are shown in Fig. 9.

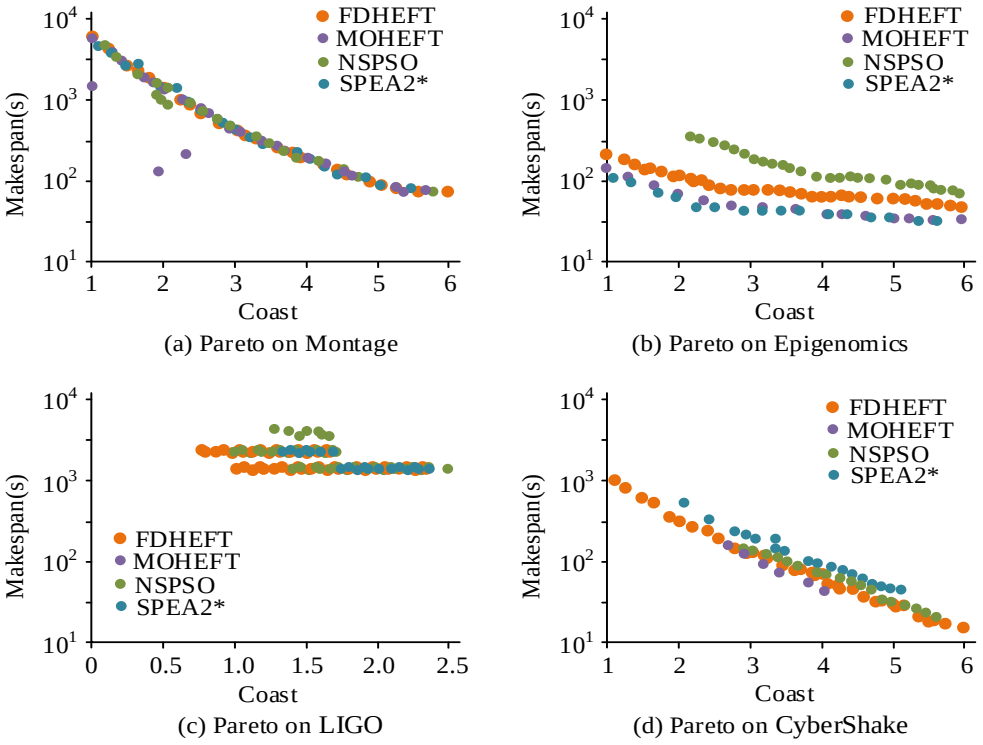


Fig. 8. Pareto frontier solution for time cost in different workflows.

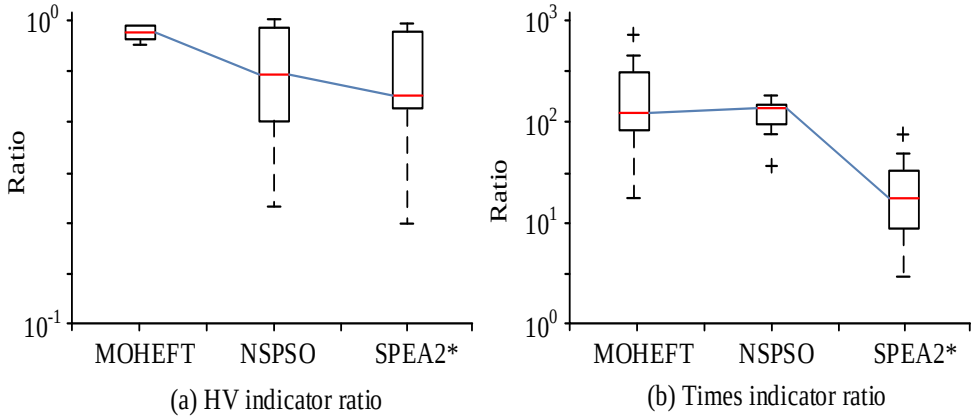


Fig. 9. HV metrics and time results.

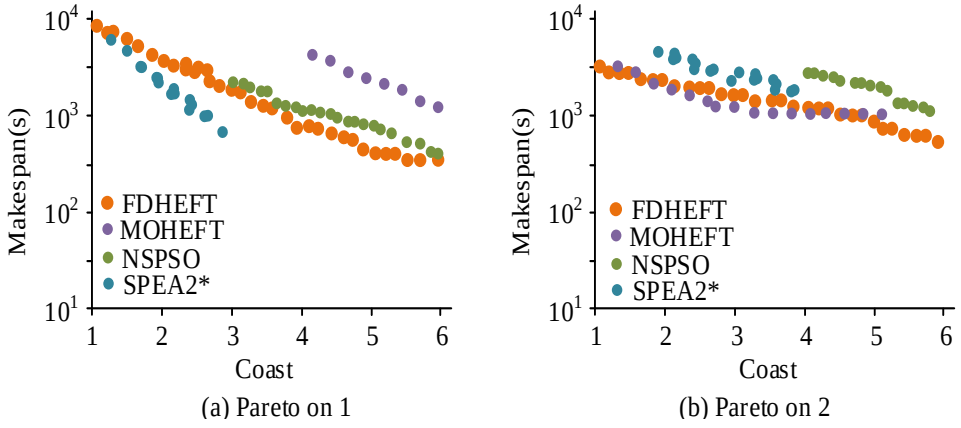


Fig. 10. Pareto frontal solution.

Fig. 9(a) shows the HV ratio diagram of the MOHEFT, NSPSO, and SPEA2* models to the FDHEFT model. This indicates that the ratios are all less than 1, so FDHEFT is slightly better than the SPEA2* model and significantly better than the MOHEFT and NSPSO models. Fig. 9(b) shows the time ratio diagram of the three models of MOHEFT, NSPSO, and SPEA2* to the FDHEFT model. This indicates that the time ratios of MOHEFT, NSPSO, and SPEA2* models are all greater than 1. Therefore, the running time of all three models is greater than that of the FDHEFT model. To test the scheduling effect of the FDHEFT model on other workflows, two types of workflow are randomly generated, numbered 1 and 2. On the randomly generated workflow, the Pareto frontier solutions of the FDHEFT model and other models are shown in Fig. 10.

Fig. 10 shows that in a randomly generated workflow, the FDHEFT model has the best Pareto frontier solution compared to the other three models, with continuous and numerous Pareto frontier solutions. Therefore, this proves the effectiveness of the FDHEFT model in dual objective scheduling.

C. MEAC Model Testing and Analysis

To verify the performance of the MEAC model, the MOHEFT model and the SPEA2* model are compared and tested with the MEAC model, with HV value and running time as detection indicators. The HV values and time results of different models are shown in Table III.

TABLE III. HV VALUE AND TIME COMPARISON

-	Index	MEAC	Ratio	SPEA2*	Ratio	MOHEFT	Ratio
Montage	Times(s)	8.98	-	12.94	-30.6%	13.43	-33.1%
	HV	9.73	-	8.49	14.6%	8.13	19.7%
Epigenomics	Times(s)	156.21	-	178.49	-12.5%	188.45	-17.1%
	HV	9.43	-	8.27	14.0%	8.16	15.6%
LIGO	Times(s)	178.49	-	188.43	-5.3%	197.23	-9.5%
	HV	8.97	-	8.19	9.5%	7.93	13.1%
CyberShake	Times(s)	23.59	-	30.47	-22.6%	32.52	-27.5%
	HV	8.81	-	8.47	4.0%	8.09	8.9%

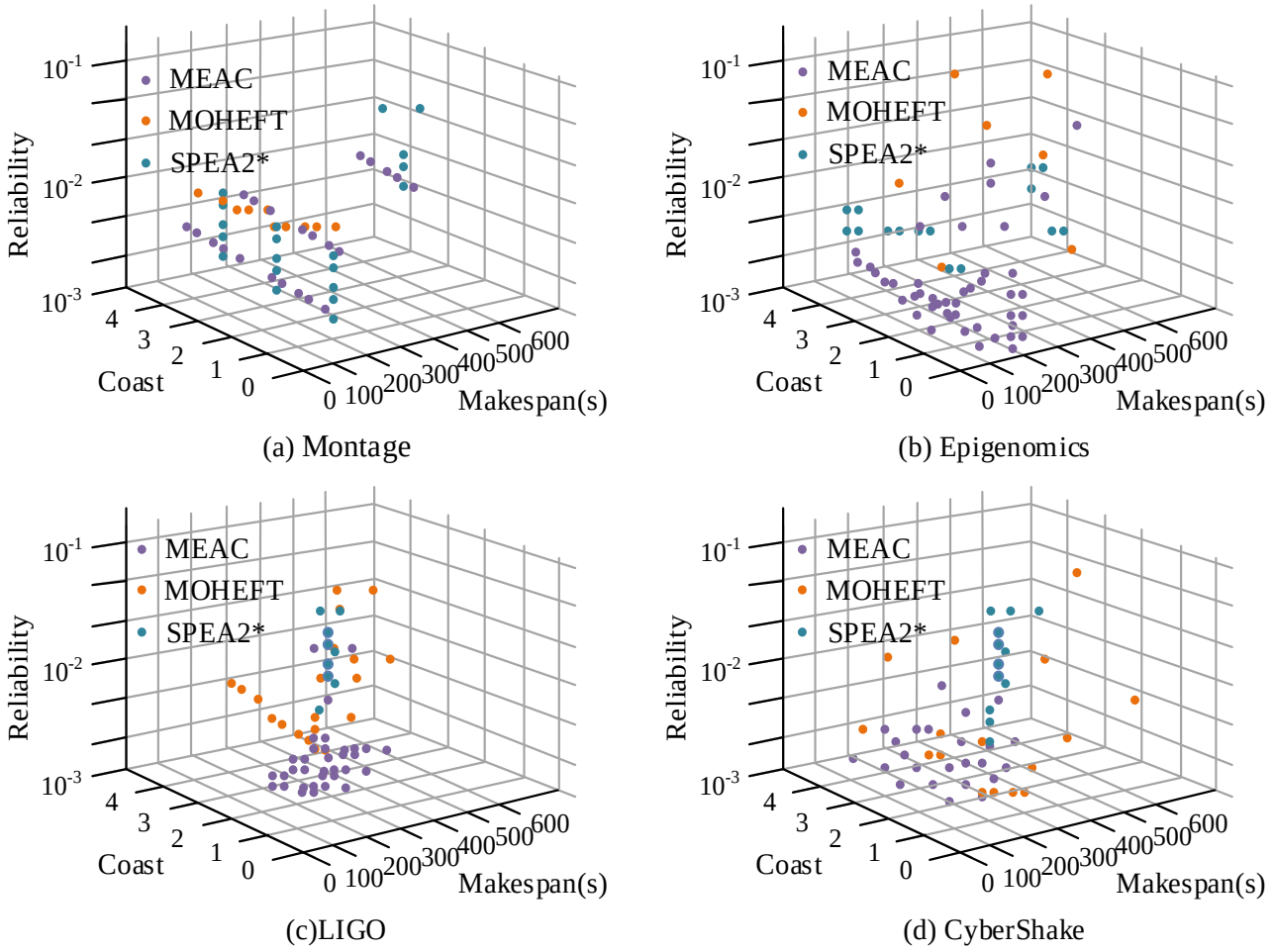


Fig. 11. Pareto frontier solutions for three objectives of each model

Table III shows that in the Montage workflow, the MEAC model reduces time by 30.6% and 33.1% compared to the MOHEFT model and the SPEA2* model, while the HV value increases by 14.6% and 19.7%. On the Epigenomics workflow, the MEAC model reduces time by 12.5% and 17.1% compared to the MOHEFT model and the SPEA2* model. The MEAC model increases HV values by 14.0% and 15.6%. In the LIGO workflow, the MEAC model reduces time by 5.3% and 9.5% compared to the MOHEFT model and the SPEA2* model, while the HV value increases by 9.5% and 13.1%. On the CyberShake workflow, the MEAC model reduces time by 22.6% and 27.5% compared to the MOHEFT model and the SPEA2* model, while the HV value increases by 4.0% and 8.9%. Three models are tested on the standard generation workflow, and the Pareto frontier solutions of each model's three objectives are shown in Fig. 11.

In Fig. 11, each point represents a feasible scheduling solution for the model. Fig. 10 shows that the experimental results of the MEAC model have more points, indicating that more scheduling schemes can be found and better solution sets can be extracted for workflow scheduling.

V. Conclusion

To solve the workflow target scheduling problem in CC IaaS environment, this study adopts a heuristic scheduling model based on deadline to construct a single objective workflow scheduling model based on deadline. Based on fuzzy dominated sorting and evolutionary algorithms, a time-cost dual objective and a time-cost reliability three objective workflow scheduling model is constructed. The experimental results showed that the total execution time of the DCWS model in four standard workflows was 92s, 106s, 113s, and 105s, respectively. In Montage, the DCWS model reduced time by 30.8%, 24.2%, and 14.8% compared to the IC-PCP, JIT, and PDC models, respectively. In Epigenomics, the DCWS model reduced time by 24.3%, 22.1%, and 13.1% compared to the IC-PCP, JIT, and PDC models, respectively. In LIGO, the DCWS model reduced time by 18.1%, 14.5%, and 12.4% compared to the IC-PCP, JIT, and PDC models, respectively. In CyberShake, the DCWS model reduced time by 23.3%, 19.2%, and 4.2% compared to the IC-PCP, JIT, and PDC models, respectively. DCWS had the highest throughput in standard workflows, with throughput rates of 144b/s, 138b/s, 140b/s, and 142b/s. Compared with other comparative models, the FDHEFT model had higher HV values and less execution time, as well as better Pareto frontier solutions. On the Montage workflow, the MEAC model reduced time by 30.6% and 33.1% compared to the MOHEFT model and the SPEA2* model, while the HV value increased by 14.6% and 19.7%. On the Epigenomics workflow, the MEAC model reduced time by 12.5% and 17.1% compared to the MOHEFT model and the SPEA2* model, which increased HV values by 14.0% and 15.6%. In the LIGO workflow, the MEAC model reduced time by 5.3% and 9.5% compared to the MOHEFT model and the SPEA2* model, while the HV value increased by 9.5% and 13.1%. On the CyberShake workflow, the MEAC model reduced time by 22.6% and 27.5% compared to the MOHEFT model and the SPEA2* model, while the HV value increased by 4.0% and 8.9%. There are shortcomings in this study, as there are less randomly generated workflow data, and more experimental data will be considered in the future.

References

[1] Martey S, Zhang G. Ensuing Security in a Proposed Tertiary Institution Cloud Computing Environment: Introducing a NoHype Framework to the Private Cloud as a Way of Securing the IaaS Model. International Journal of Computer Applications, 2020, 177(43):10-16.

[2] Dhakal A, Sharma S, Pokhrel A, Poudel A. Variability and Heritability Estimate of 30 Rice Landraces of Lamjung and Tanahun Districts, Nepal. Indonesian Journal of Agricultural Science, 2020, 21(1):1-10.

[3] Mukhopadhyay B, Bose R, Roy S. A Novel Approach to Load Balancing and Cloud Computing Security using SSL in IaaS Environment. International Journal of Advanced Trends in Computer Science and Engineering, 2020, 9(2):1-9.

[4] Suryateja P S. Analysis of Host Resources Utilization by OpenStack in Ubuntu Environment. Emerging Science Journal, 2020, 4(6):466-492.

[5] Sharma A. Data Storage in Cloud Environment: Challenges and Issues. International Journal of Scientific Research in Agricultural Sciences, 2020, 3(2):8-18.

[6] Modisane P, Jokonya O. Evaluating the benefits of Cloud Computing in Small, Medium and Micro-sized Enterprises (SMMEs). Procedia Computer Science, 2021, 181(1):784-792.

[7] Chen P, Xia Y, Yu C. A Novel Reinforcement-Learning-Based Approach to Workflow Scheduling Upon Infrastructure-as-a-Service Clouds. International Journal of Web Services Research, 2021, 18(1):21-33.

[8] Ramadhan M, Latip R, Hussin M, Hamid N A W A. Comparative Analysis of PSO-derived Workflow Scheduling Algorithms in Cloud Computing based on QoS Requirements. INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN ENGINEERING & TECHNOLOGY, 2020, 11(12):1387-1399.

[9] Li Y, Ma Y, Zeng Z. A Novel Approach to Location-Aware Scheduling of Workflows Over Edge Computing Resources. International Journal of Web Services Research, 2020, 17(3):56-68.

[10] Ghafouri R, Movaghar A. An adaptive and deadline-constrained workflow scheduling algorithm in infrastructure as a service clouds. Iran Journal of Computer Science, 2022, 1(5):17-39.

[11] Wang Y, Zuo X. An Effective Cloud Workflow Scheduling Approach Combining PSO and Idle Time Slot-Aware Rules. IEEE/CAA Journal of Automatica Sinica, 2021, 8(5):1079-1094.

[12] Kumar D S K D. Optimal workflow scheduling in cloud computing based on hybrid bacterial evolutionary and bees mating optimization algorithm. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021, 12(3):4762-4775.

[13] Aggarwal A, Dimri P, Agarwal A. IFFO: An Improved Fruit Fly Optimization Algorithm for Multiple Workflow Scheduling Minimizing Cost and Makespan in Cloud Computing Environments. Mathematical Problems in Engineering, 2021, 2021(3):1-9.

[14] Kaur M, Kadam S. Bio-Inspired Workflow Scheduling on HPC Platforms. Tehnički Glasnik, 2021, 15(1):60-68.

[15] Pratiksha. Review on Workflow Scheduling In Cloud Environment: A Comprehensive Study. International Journal of Innovative Research in Computer and Communication Engineering, 2021, 9(2):2347-5552.

[16] Paknejad P, Khorsand R, Ramezanpour M. Chaotic improved PICEA-g-based multi-objective optimization for workflow scheduling in cloud environment. Future Generation Computer Systems, 2021, 117(10):12-28.

[17] Lin T, Wu P, Gao F M. Energy-Saving Cloud Workflow Scheduling Based on Optimistic Cost Table. International Journal of Simulation Modelling, 2020, 19(3):505-516.

[18] Djigal H, Jun F, Lu J. IPPTS: An Efficient Algorithm for Scientific Workflow Scheduling in Heterogeneous Computing Systems. IEEE Transactions on Parallel and Distributed Systems, 2020, 32(5):1-1.

[19] Haidri R A, Katti C P, Saxena P C. Cost effective deadline aware scheduling strategy for workflow applications on virtual machines in cloud computing. Journal of King Saud University - Computer and Information Sciences, 2020, 32(6):666-683.

[20] Krishan R, Kumar V. Optimization of Resource Aware Task- Scheduling Approaches in Cloud Computing. Journal of Green Engineering, 2020,

10(3):1077-1096.

[21] Kapoor N, Lacson R, Khorasani R. Workflow Applications of Artificial Intelligence in Radiology and an Overview of Available Tools. Journal of the American College of Radiology, 2020, 17(11):1363-1370.

[22] Ouldkablia M E, Kechar B, Bouzefrane S. IoT-Based Smart Home Process Management Using a Workflow Approach. International Journal of Information Technology and Web Engineering, 2020, 15(2):50-76.

[23] Long T, Ma Y, Wu L. A novel fault-tolerant scheduling approach for collaborative workflows in an edge-IoT environment. Digital Communication and Networks, 2022, 8(6):911-922.

[24] Barma M, Modibbo U M. Multiobjective mathematical optimization model for municipal solid waste management with economic analysis of reuse/recycling recovered waste materials. Journal of Computational and Cognitive Engineering, 2022, 1(3): 122-137.

[25] Guo Y, Mustafaoglu Z, Koundal D. Spam Detection Using Bidirectional Transformers and Machine Learning Classifier Algorithms. Journal of Computational and Cognitive Engineering, 2023, 2(1): 5-9.