Hindawi
Security and Communication Networks
Volume 2020, Article ID 6697640, 11 pages
https://doi.org/10.1155/2020/6697640

Research Article

A Novel Coevolutionary Approach to Reliability Guaranteed Multi-Workflow Scheduling upon Edge Computing Infrastructures

Zhenxing Wang,[1] **Wanbo Zheng**,[2] **Peng Chen**,[3] **Yong Ma**,[4] **Yunni Xia**,[1] **Wei Liu**,[5] **Xiaobo Li**,[6] **and Kunyin Guo**[1]

[1]*Software Theory and Technology Chongqing Key Lab, Chongqing University, Chongqing, China*
[2]*School of Mathematics, Kunming University of Science and Technology, Kunming, Yunnan 650500, China*
[3]*School of Computer and Software Engineering, Xihua University, Chengdu, Sichuan 610065, China*
[4]*School of Computer and Information Engineering, Jiangxi Normal University, Nanchang 330022, China*
[5]*Shanghai Jiaotong University Chongqing Research Institute, Chongqing 401121, China*
[6]*Chongqing Animal Husbandry Techniques Extension Center, Chongqing 401121, China*

Correspondence should be addressed to Peng Chen; chenpeng@mail.xhu.edu.cn, Yong Ma; may@jxnu.edu.cn, and Yunni Xia; xiayunni@hotmail.com

Received 14 October 2020; Revised 4 November 2020; Accepted 23 November 2020; Published 8 December 2020

Academic Editor: Honghao Gao

Recently, mobile edge computing (MEC) is widely believed to be a promising and powerful paradigm for bringing enterprise applications closer to data sources such as IoT devices or local edge servers. It is capable of energizing novel mobile applications, especially the ultra-latency-sensitive ones, by providing powerful local computing capabilities and lower end-to-end delays. Nevertheless, various challenges, especially the reliability-guaranteed scheduling of multitask business processes in terms of, e.g., workflows, upon distributed edge resources and servers, are yet to be carefully addressed. In this paper, we propose a novel edge-environment-based multi-workflow scheduling method, which incorporates a reliability estimation model for edge-workflows and a coevolutionary algorithm for yielding scheduling decisions. The proposed approach aims at maximizing the reliability, in terms of success rates, of services deployed upon edge infrastructures while minimizing service invocation cost for users. We conduct simulative experimental case studies based on multiple well-known scientific workflow templates and a well-known dataset of edge resource locations as well. Simulative results clearly suggest that our proposed approach outperforms traditional ones in terms of workflow success rate and monetary cost.

1. Introduction

Edge computing is an evolving computing paradigm offering a more efficient alternative: data is processed and analyzed closer to the point where it is created. It enables computation as a service model and prepares a proximity-based and mobility-aware resource provisioning model of virtualized resources applicable on demand [1, 2] The edge service providers are equipped with computational facilities, which allow them to provide necessary spaces required by commercial and noncommercial users. Recently, the edge computing paradigm has evolved as an increasingly popular force for supporting and enabling business process and scientific workflow execution [3–5]. A workflow is a set of dependent or independent tasks illustrated as a directed acyclic graph (DAG) [6–8], in which the nodes indicate the tasks and a directed arch represents the interdependency among the corresponding tasks. Workflow scheduling involves mapping workflow tasks to computational resources for execution, and the resulting optimization problem is well acknowledged to be NP-hard.

Recently, as novel bioinspired and genetic algorithms are becoming increasingly versatile and powerful, a great deal of research efforts are paid to applying them in dealing with edge-environment-oriented workflow scheduling problem [9–11]. However, it remains a great challenge to develop efficient scheduling algorithm with good scheduling performance, low service-level-agreement (SLA) violation rate, and high user-perceived quality of service.

In this paper, we propose a novel edge-environment-based multi-workflow scheduling approach by leveraging a multi-workflow-reliability estimation model and preference-inspired coevolutionary algorithms, i.e., PICEA-g, for yielding scheduling decisions. We show through simulative studies as well that our proposed method clearly outperforms traditional ones in terms of multiple metrics.

2. Literature Review

2.1. Related Work. It is widely believed that to arrange multitask business processes or workflows upon distributed nodes or computing resources with Quality of Service (QoS) constraints, e.g., reliability, is an NP-hard problem [12, 13]. It is therefore extremely time-consuming to yield optimal schedules through traversal-based algorithms. Fortunately, heuristic and metaheuristic strategies with polynomial complexity are capable of producing approximate or near-optimal solutions at the cost of acceptable optimality loss.

For example, Wang et al. [14] proposed a look-ahead genetic algorithm (LAGA), which utilized reliability-based reputation scores for optimizing the makespan and the reliability of a workflow application. Wen et al. [15] aimed at solving the problem of deploying workflow applications over federated clouds while meeting the reliability, security, and cost requirements. Wu et al. [16] proposed a soft error-aware and energy-efficient task scheduling method for workflow applications in DVFS-enabled cloud infrastructures under reliability and completion time constraints. Cao et al. [17] proposed a soft error-aware VM selection and the task scheduling approach to minimize the execution cost of cloud workflows under makespan, reliability, and memory constraints while considering soft errors in cloud data centers. Garg et al. [18] proposed a new scheduling algorithm called the reliability and energy-efficient workflow scheduling algorithm, which jointly optimized lifetime reliability of application and energy consumption and guaranteed the user-specified QoS constraint. Nik et al. [19] proposed a scheduling approach, which included four algorithms for minimizing the workflow execution cost while also meeting the user-specified deadline and reliability.

To minimize the overall error probability in a multi-server mobile edge computing (MEC) network, where the wireless data transmission/offloading was carried by finite blocklength (FBL) codes, Zhu et al. [20] characterized the FBL reliability of the transmission phase and investigated the extreme event of queue length violation in the computation phase by applying extreme value theory and provided an optimal framework for deciding time allocation and server selection. Peng et al. [8] proposed a novel method to evaluate the resource reliability in mobile edge computing environment and addressed the workflow scheduling problem by using a Krill-based algorithm. Kouloumpris et al. [21] considered an architecture consisting of an edge node, an intermediate node (hub), and the cloud infrastructure and then used a mathematical programming-based framework to derive an application-reliability-optimal task allocation based on multiple operational constraints. Wang et al. [22] developed a reinforcement-learning-based approach to the multi-workflow scheduling method. However, they considered the centralized cloud environment as the underlying infrastructure and thus ignored the overhead for inter-edge-node data transmission. For a similar optimization objective, Wang et al. [23] and Saeedi et al. [24] employed an immune-based PSO algorithm for scheduling workflows over centralized clouds.

3. Models and Systems

3.1. System Architecture. An edge computing system usually consists of an edge computing agent (ECA) and multiple edge servers. The edge computing agent manages all resources and each edge server owns several virtual machines (VMs), each of which can usually handle a workflow task that a user offloads at a time. An edge server usually has limited capacity for storage and computation. Due to the requirement of signal strength and channel stability, as illustrated in Figure 1, it is usually believed that an edge server can cover a limited circular range and thus users can only offload their tasks to the reachable edge servers in terms of such coverage ranges.

As can be seen in Figure 2, instead of considering the monolithic task configurations, we consider that user requests can be structured and process-like requests can be expressed as workflows with different constructs. A workflow refers to a directed acyclic graph (DAG), $G = (T, E)$. T denotes the task set $T = \{t_1, t_2, \ldots t_n\}$, E is the set of edges between tasks, and $e_{ij} = (t_i, t_j)$ is a priority constraint, indicating that t_i is the precedent task of t_j.

The notations used in this paper are shown in Table 1.

3.2. Problem Formulation. In engineering, reliability is the probability of a system or component to perform its required functions under the stated conditions and with dependable outcomes. Guaranteeing reliability of computing systems and applications is a challenging problem due to the fact that faults are hard to avoid due to hardware failure, software bugs, transient faults, devices that work in high temperature, and so on. The reliability issue of edge-environment-based multi-workflow can be further complicated due to the fact that structured and process-based task flows are more susceptible to varying types of faults, especially transmission errors and faults occurring when wireless communications between edge nodes and users are required.

As shown in Figure 3, the reliability of a workflow is usually structure dependent as follows:

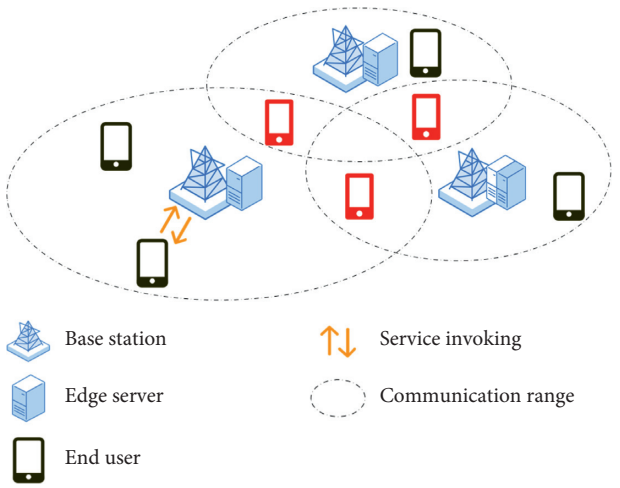


Figure 1: Proximity constraint example.

$$R_\alpha(r_j, n_\alpha) = \prod_{j=1}^{n_\alpha} r_j,$$
$$R_\beta(r_j, n_\beta) = 1 - \prod_{j=1}^{n_\beta}(1 - r_j), \quad (1)$$
$$R_\gamma(r_j, n_\gamma) = \sum_{j=1}^{n_\gamma} \lambda_j \cdot r_j,$$

where n_α denotes the number of tasks in a sequential routing, n_β is the number of tasks succeeded by a split point in a parallel routing, and n_γ is that of a selective routing, respectively. For a task executed on the edge server p, its reliability can be estimated as its success rate of execution, i.e., the probability that its time-to-failure (TTF_p) exceeds its completion time:

$$r_j = \sum_{p=1}^{m} \sum_{k=1}^{m_p} x_{pk} \cdot \mathrm{Prob}(\mathrm{TTF}_p > \mathrm{FT}(T_{pk})), \quad (2)$$

where

$$x_{pk} = \begin{cases} 1, & \text{if } \mathrm{VM}_{pk} \text{ is selected for the task,} \\ 0, & \text{otherwise,} \end{cases}$$
$$\mathrm{FT}(T_{ijpk}) = \mathrm{FT}(\mathrm{prior}(T_{ijpk}), \mathrm{FT}(\mathrm{pred}(T_{ijpk}))) + t_{ijpk}. \quad (3)$$

To estimate the monetary cost of workflows, we first have to estimate the cost for renting server p:

$$V_p^{\mathrm{rent}} = \sum_{k=1}^{m_p} \max[\mathrm{FT}(T_{pk})] \cdot C_p^r, \quad (4)$$

where $\max[\mathrm{FT}(T_{pk})]$ is the completion time of the task executing queue on VM_{pk} and C_p^r is the charge per unit time for renting server p.

The transmission time for the task i can be estimated as $\Delta_i = \Delta_i^{\mathrm{ul}} + \Delta_i^{\mathrm{dl}} + \Delta_i^{\mathrm{bh}}$, which is composed of three parts [25], where Δ_i^{ul} indicates the uplink communication time, Δ_i^{dl} is the the downlink time, and Δ_i^{bh} is the the backhaul link time. According to [26, 27], Δ_i^{bh} can be infinitesimal, and the downlink time Δ_i^{dl} can usually be a constant ξ. Therefore, Δ_i can be expressed as

$$\Delta_i = \frac{d_i}{\omega_p \cdot \eta_p^i} + \xi, \quad (5)$$

where $\eta_p^i = \lambda/\mathrm{disk}(i, p)$ is decided by the distance between the task (user) and the server; as the distance increases, the bit error rate increases and the average transmission speed decreases [27]. And, ω_p indicates the averaged bandwidth of the server p and d_i is the the data size of task i. If the transmission price per unit time of the server p as C_p^t, then the transmission fees can be estimated as

$$V_p^{\mathrm{commu}} = \frac{d_i}{\omega_p \cdot \eta_p^i} \cdot C_p^t. \quad (6)$$

Based on the described system configuration, the problem that we are interested in is thus, for given proximity constraints of server-user communications and deadline, how to schedule workflows with higher reliability and lower cost. The resulting formulation is thus

$$\begin{aligned} &\mathrm{Min}\ f_1 = \mathrm{reliability} = \prod_{i=1}^{n} R_i, \\ &\mathrm{Min}\ f_2 = \mathrm{cost} = V_p^{\mathrm{rent}} + V_p^{\mathrm{commu}}, \end{aligned} \quad (7)$$

subject to

$$\begin{aligned} &\mathrm{ST}(T_{ij}) \geq \max[\mathrm{FT}(T_{il})], \quad T_{il} \in \mathrm{pred}(T_{ij}) \text{ and } l \in \{1, \ldots, n_i\}, \\ &T(W_i) \leq D(W_i), \\ &\mathrm{dist}_{ip} \leq \mathrm{cov}_p, \\ &x_{ijpk} \leq 1. \end{aligned} \quad (8)$$

4. PICEA-g for Multi-Workflow Scheduling

4.1. Preference-Inspired Coevolutionary Algorithms Using Goal Vectors. It has long been known that preference-based approaches are useful for the generation of trade-off surfaces in objective subspaces of interest to the decision maker. Wang et al. [28] offered one realization of such approach named preference-inspired coevolutionary algorithm using goal vectors (PICEA-g), which had been testified to outperform four other best-in-class multiobjective evolutionary algorithms, e.g., NSGA-II, MOEA, HypE, and MOEA/D.

PICEA-g is a coevolutionary approach in which the usual population of candidate solutions is considered evolvable with a set of goal vectors during the search. In this algorithm, optimality of candidate solutions is decided by a Pareto-dominance model. To be specific, a family of goal vectors and a population of candidate solutions coevolved during the search process. A candidate solution gains fitness by meeting a set of goal vectors in the objective space, but the fitness contribution must be shared with other solutions satisfying those goal vectors. Goal vectors only gain fitness

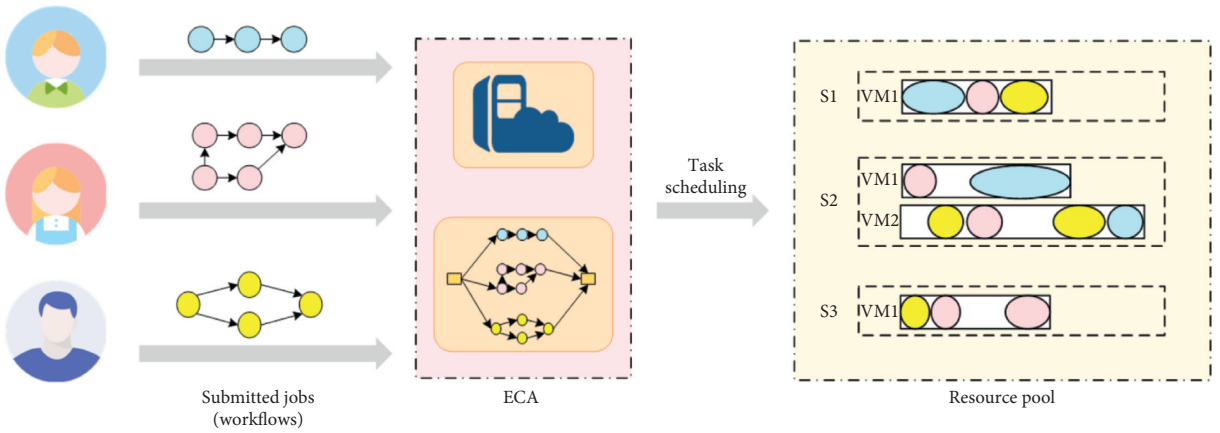


Figure 2: Edge computing deployment example.

Table 1: Notations and description.

Notation	Description
N	The total number of workflows
M	The total number of edge servers
R_i	The reliability of the workflow
λ_j	The selection probability for a task
x_{pk}	A boolean variable indicating whether VM_{pk} is selected for a task
ST (T_{pk})	The start time of the task on VM_{pk}
$MTTF_p$	The mean time-to-failure of the server p
D (W_i)	User-defined deadline of the workflow i
pred (T_{ij})	All predecessor node tasks of T_{ij} in the workflow i
cov_p	The coverage area of the server p
n_i	The total number of tasks in the workflow
m_p	The total number of virtual machines in the server p
r_j	The success rate of the task
T_{pk}	The task executed on VM_{pk}
t_{pk}	The execution time of the task on VM_{pk}
FT (T_{pk})	The completion time of the task on VM_{pk}
prior (T_{pk})	The prior task of T_{pk} in the execution queue of VM_{pk}
T (W_i)	The finish time of the workflow i
$dist_{ip}$	The distance between the server p and device i
–	–

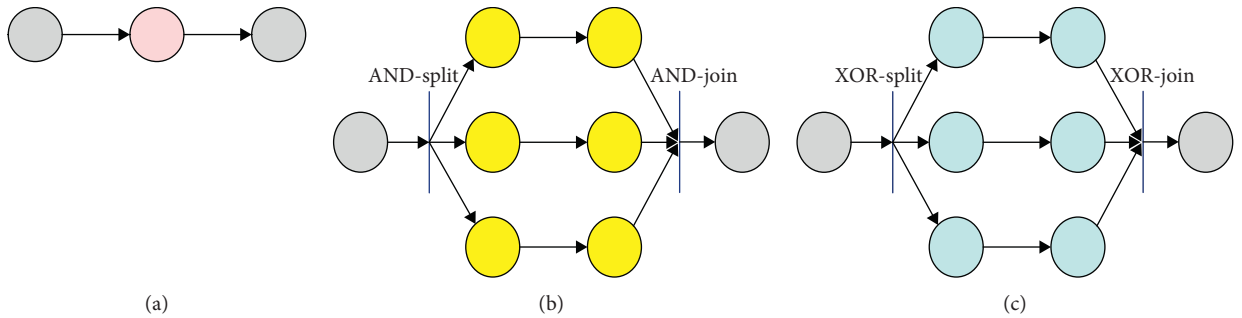


Figure 3: Routing patterns: (a) sequential routing; (b) parallel routing; (c) selective routing.

by being satisfied by a candidate solution, but the fitness is reduced the more the time a goal being satisfied by other solutions in the population. Ultimately, the population of candidate solutions and the goal vectors coevolve toward the Pareto optimal front. The fitness F_s of a candidate solution s and the fitness F_g of a preference g can be calculated by (9)–(11) as follows:

$$F_s = 0 + \sum_{g \in G \cup G_C | s \leq g} \frac{1}{n_g}, \tag{9}$$

where n_g denotes the number of solutions that satisfy preference g. In this formulation, when s fails to satisfy any g, the fitness F_s is defined as 0. And,

$$F_g = \frac{1}{1+\alpha}, \tag{10}$$

where

$$\alpha = \begin{cases} 1, & n_g = 0, \\ \dfrac{n_g - 1}{2N_S - 1}, & \text{otherwise}, \end{cases} \tag{11}$$

where N_S is the population size of candidate solutions.

A $(\mu + \lambda)$ elitist framework is usually used for implementing the above model as shown in Figure 4. As can be seen, a population of N_S candidate solutions and a set of N_G preferences, denoted by S and G, respectively, are evolved for a fixed number of generations, maxGen. In each generation t, genetic variation operators are implemented on parents $S(t)$ to produce N_S offspring, $Sc(t)$. Meanwhile, N_G new goal vectors, $Gc(t)$, are randomly regenerated based on the predefined bounds. Then, $S(t)$ and $Sc(t)$ and $G(t)$ and $Gc(t)$ are pooled, respectively, whereafter the combined population is sorted according to the fitness. Finally, a truncation-selection is applied to select the best N_S candidate solutions and N_G vectors as the new population, $S(t+1)$ and $G(t+1)$.

4.2. Encoding. For a workflow application, a chromosome is a data structure in which a scheduling solution is encoded. We use a two-dimensional string to represent a scheduling solution. One dimension of the string represents the index of resources, which depicts the task-resource mapping, while the other dimension denotes the order between tasks.As illustrated in Figure 5, in this solution, there are tasks from three workflows, namely, w_1, w_2, and w_3, which are assigned to virtual machines on two edge servers. For instance, VM_{21} is executing four tasks with the processing sequence of $t_{12} \longrightarrow t_{13} \longrightarrow t_{21} \longrightarrow t_{24}$. The decoding scheme can be described as the reverse of encoding.

4.3. Initialization. Two constraints are applied here to generate uniformly feasible chromosomes to improve the quality of the initial population, meanwhile, accelerating the convergence rate, i.e., the topological constraint and the proximity constraint. Based on the constraints, the initial population is generated as follows:

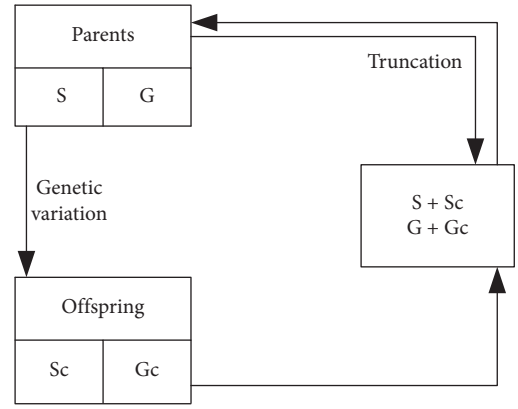


Figure 4: $(\mu + \lambda)$ elitist framework.

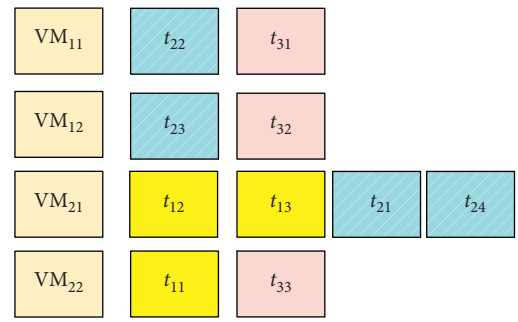


Figure 5: Solution coding.

(1) Firstly, each workflow is converted into a task list $T = \{t_0, \ldots, t_{j-1}, t_j\}$ after topological sort.

(2) Secondly, a resource r_i from $R = \{r_0, \ldots, r_{i-1}, r_i\}$ is selected as the computing resource VM, only if r_i is available for t_j. Then, t_j is assigned to a VM.

(3) Repeat the above steps until all workflow tasks are assigned. Then, a chromosome is generated.

When the population size reaches the defined value, the initialization process stops.

The initial goal vectors are randomly generated as objective vectors in the objective space within predefined bounds. In practice, the bounds are estimated via preliminary single-objective optimizations.

4.4. Population Update. The iterative update of population consists of discrete steps described below, until the termination condition is satisfied.

4.4.1. Genetic Variation. The genetic variation changes the workflow task allocation information to maintain diversity in the population. In our proposed genetic variation operation, a solution is mutated intelligently based on a resource priority heuristic. To generate a promising offspring solution, Dongarra et al. [29] have proven that the resource, which has the minimal multiplication value of some key

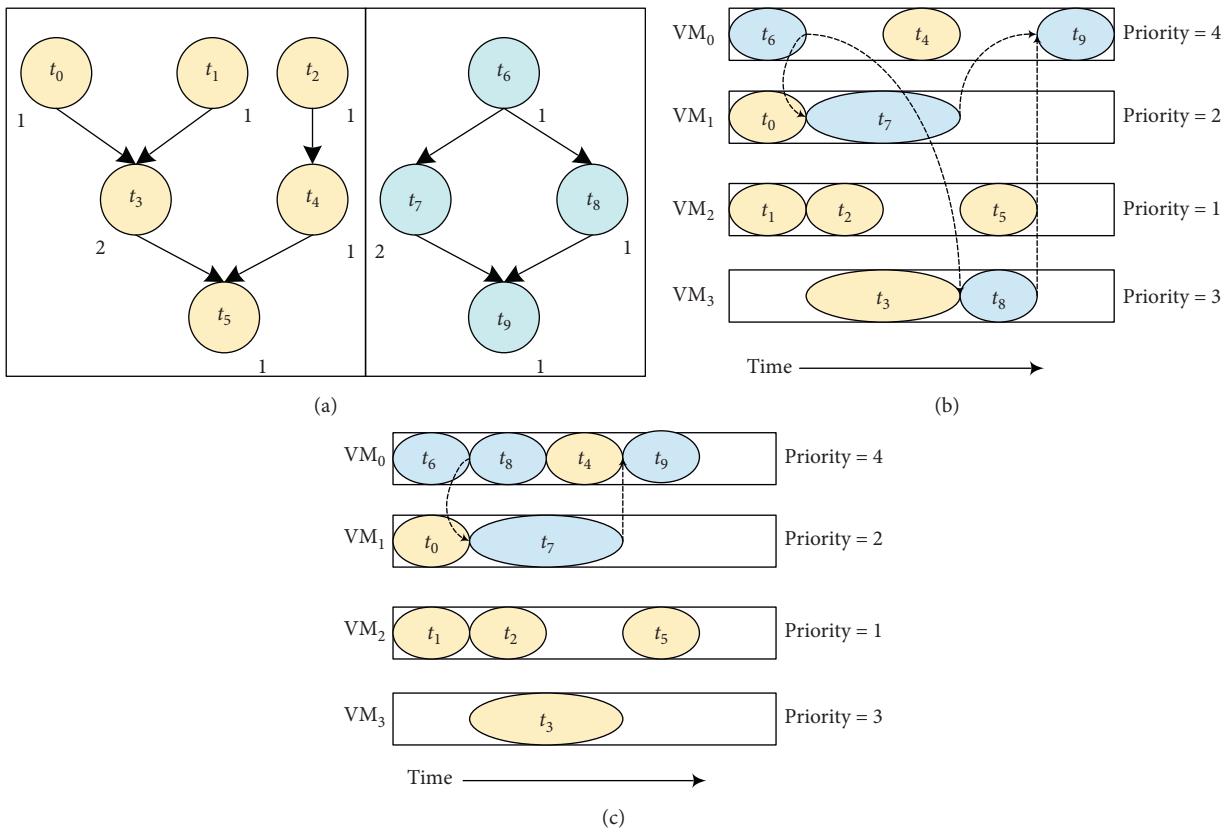


Figure 6: Genetic variation: (a) workflow example; (b) original scheduling; (c) scheduling after mutation.

performance indicators, should have a higher priority to be selected in the scheduling. Hence, we have

$$\Gamma_p = \frac{C_p^r \cdot C_p^t \cdot \mathrm{dist}(i, p)}{\mathrm{MTTF}_p \cdot \omega_p}. \tag{12}$$

Then, we let $1/\Gamma$ indicate the priority of the server p. The genetic variation operation randomly selects one task in the solution and reassigns it to any available server with a higher priority. As an example shown in Figure 6(b), task t_8 is originally scheduled to VM_4, whose priority is 3. Thus, the genetic variation reassigns it to VM_1 with a higher priority of 4.

According to the precedence constraint, we insert t_8 into the position behind t_6, as shown in Figure 6(c).

Simultaneously, N_G are new preference sets and $Gc(t)$ are randomly regenerated based on the initial bounds.

4.4.2. Fitness Calculation. Fitness calculation is based on the distribution of function value vectors and goal vectors in the target space. Assume that there are two candidate solutions s_1 and s_2, their offspring s_3 and s_4, two existing preferences g_1 and g_2, and two new preferences g_3 and g_4 (i.e., $N_S = N_G = 2$) as shown in Figure 7.

The process to calculate the fitness F_s of a candidate solution s and fitness F_g of a preference g is shown in Table 2.

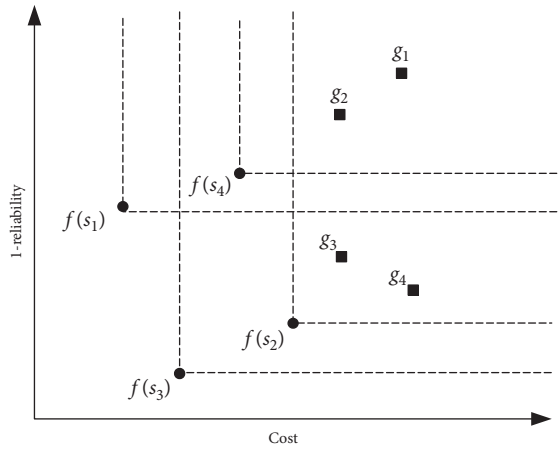
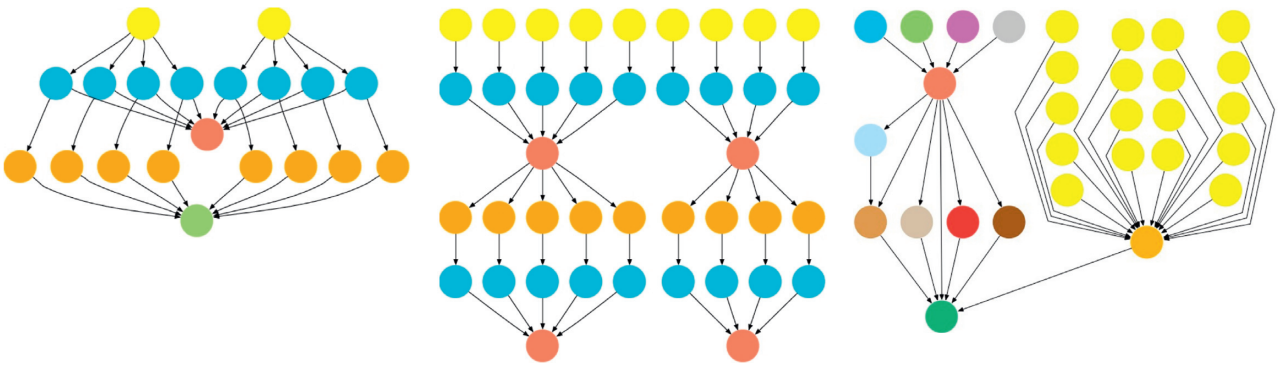


Figure 7: Example for Pareto-dominance relation.

Table 2: Fitness calculation in Figure 7.

Notation	$s1$	$s2$	$s3$	$s4$
$G\{g \mid s \le g\}$	$g1$, $g2$	$g1$, $g2$, $g3$, $g4$	$g1$, $g2$, $g3$, $g4$	$g1$, $g2$
F_s	1/2	3/2	3/2	1/2
	$g1$	$g2$	$g3$	$g4$
n_g	4	4	2	2
α	1	1	1/3	1/3
F_g	1/2	1/2	3/4	3/4



(a) (b) (c)

Figure 8: The case templates of workflows: (a) CyberShake; (b) LIGO; (c) SIPHT.

Objective function: $F(x)$;
Algorithm-related parameters: N_S, N_G, itermax;
Generate initial population S and initial goal vectors G;
$F_S = F(S)$;
while $t <$ itermax **do**
  Generate new population Sc from S by genetic variation
  $F_{Sc} = F(Sc)$
  Merge S and Sc into Joint S
  Merge F_S and F_{Sc} into Joint F
  Find Pareto Nondominated ndJoint S from Joint S
  Generate a new goal vector Gc
  Merge G and Gc into Joint G
  Evaluate the fitness of Joint S and Joint G
  if size(ndJoint S) $< N_S$ **then**
    Set the fitness of ndJoint S as max value
    Updtae S by truncation selection from Joint S
  else
    Update S by truncation selection from ndJoint S
  end
  Update G by truncation selection from Joint G
  if terminate condition satisfied, **then**
    Break
  end
end

Algorithm 1: PICEA using goals (PICEA-g).

Table 3: Resource configurations and the price-per-minute of edge servers.

Edge server types	Vcpu	Memory (g)	Unit-price/minute
tp1	1 core	1	0.0558 cents
tp2	1 core	2	0.1262 cents
tp3	2 core	4	0.1675 cents

4.4.3. Truncation Selection. Truncation selection aims to select the best N_S candidate solutions from the union population according to their fitness. However, some solutions with higher fitness may be Pareto-dominated. Therefore, we identify all nondominated solutions before the selection. If the number of nondominated solutions does not exceed the population size, then we assign the maximum fitness to all the nondominated solutions. However, if more than N_S nondominated solutions are found, we then disregard the dominated solutions prior to applying truncation selection (implicitly, their fitness is set to zero).

4.4.4. Termination Conditions. This phase is a major part of the proposed algorithm, which can specify the final solutions. In this article, the termination condition is examined in two

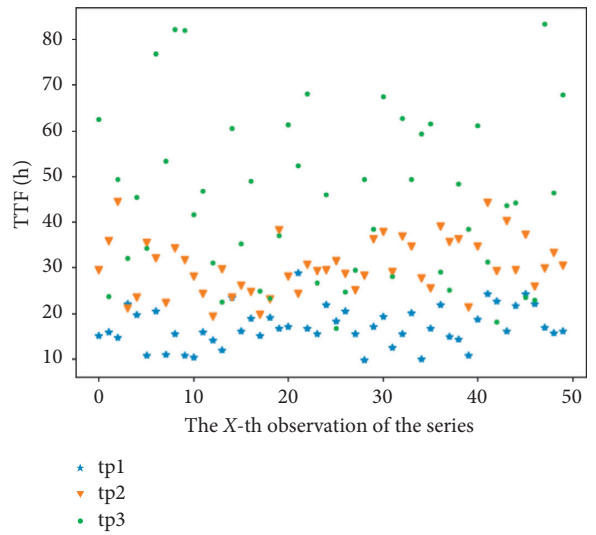


Figure 9: The TTF data collected at different types of edge servers.

Figure 10: Edge servers and users deployment.

stages: (1) as soon as a maximum iteration criterion is met, the proposed algorithm terminates and (2) T is a threshold value for terminating algorithm, set to 0.9 in our study. In every generation, after calculating the fitness of the populations, if the fitness function value is less than T, the algorithm continues; otherwise, it terminates. Whenever the algorithm ends, a set of optimal solutions is presented to the user. According to all levels presented in this article, the final solution is the best solution for all objectives including reliability and cost.

Algorithm 1 presents all the operations of the PICEA-g algorithm.

5. Performance Evaluation

To evaluate the effectiveness and correctness of our proposed method, we conduct extensive simulative experiments and show through simulative results that our proposed method outperforms traditional ones. We actually intended in the beginning to employ a real-world edge-workflow-scheduling environment to test our developed algorithms. However, we found out that such an edge environment for executing real-world scientific workflow is yet to come. Consequently, we have to rely on simulations and simulative datasets in for the model validation and comparison purpose.

These simulative experiments are based on three well-known workflow templates [30], namely, CyberShake, LIGO, and SIPHT, as shown in Figure 8.

We consider that all edge servers are with 3 different types of resource configurations and charging plans, i.e., tp1, tp2, and tp3, as shown in Table 3. We collected historical time-to-failure (TTF) records of three types as illustrated in Figure 9 as the input reliability data for edge servers. Then, the MTTF of each type of edge servers can be estimated by a Monte Carlo method [31].

We assume as well that edge servers and users are located according to the EUA dataset [2] as shown in Figure 10.

We compare our proposed method with three existing approaches, namely, NSGA-II [32], MOEA/D [33], and SPEA-II [34]. Figure 11 shows the solutions obtained by abovementioned approaches for different workflow cases where the x and the y axes represent the resulting success rate and cost. Figure 12 shows the comparison of Pareto optimal solutions of different methods with varying numbers of edge servers.

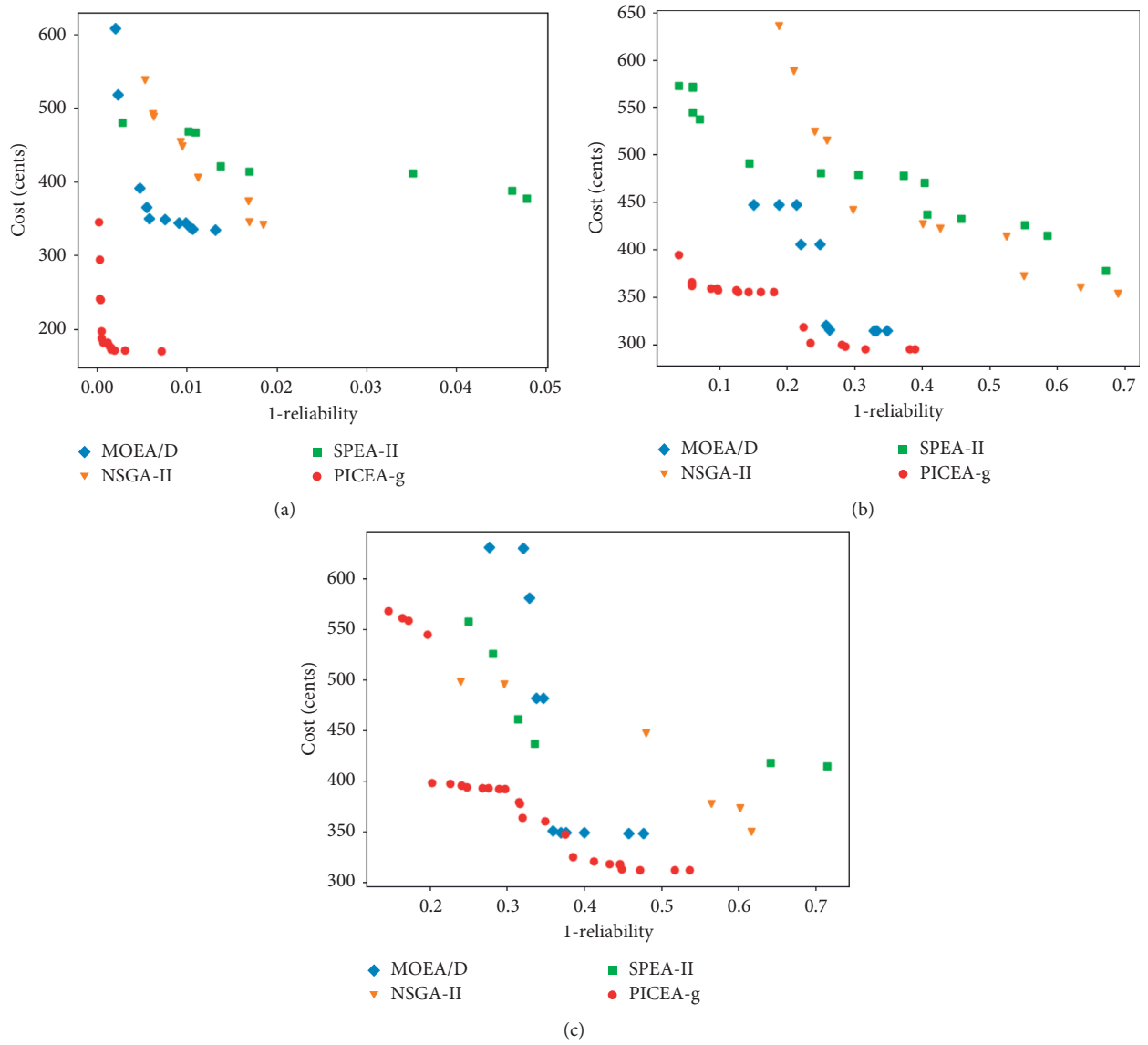


Figure 11: Experiment results with different workflow structures: (a) CyberShake; (b) LIGO; (c) IPHT.

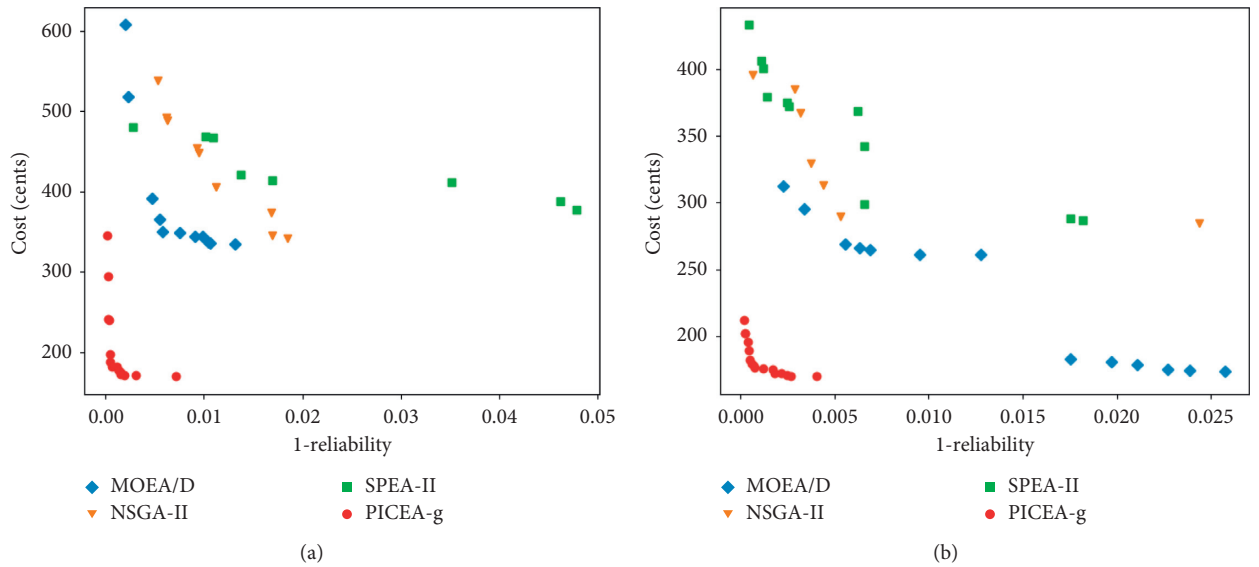
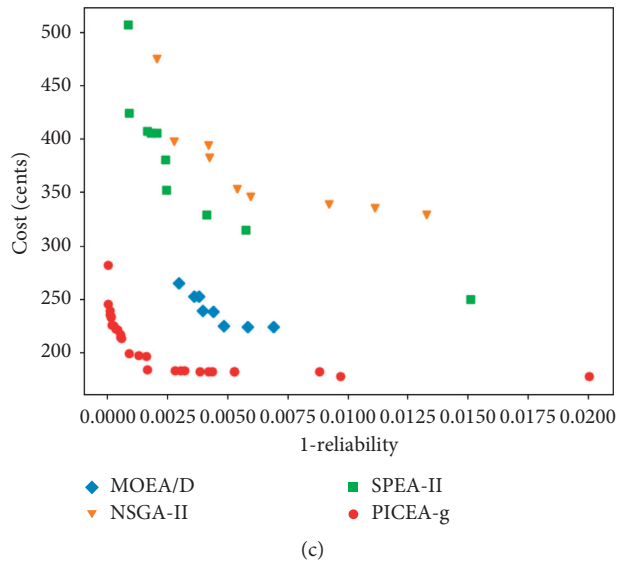


Figure 12: Continued.



(c)

Figure 12: Experiment results with the number of edge servers (a) 8 servers; (b) 10 servers; (c) 12 servers.

As can be seen from Figures 11 and 12, (1) it is evident that our method achieves better Pareto optimal fronts than its peers, regardless of workflow cases or the number of edge servers and (2) our method acquire more feasible solutions than its peers due to the fact that multiple goal vectors help to identify the solution population toward the Pareto front.

6. Conclusion

In this paper, we address the problem of reliability-guaranteed multi-workflow scheduling in the edge computing environment. We develop a reliability-driven scheduling strategy based on the PICEA-g algorithm. Extensive simulations based on several well-known workflow templates and a real-world edge-server-location dataset clearly indicate that our proposed method outperforms its counterparts in terms of different performance metrics.

Data Availability

The EUA dataset used to support the findings of this study is available at https://github.com/swinedge/eua-dataset.

Disclosure

Zhenxing Wang and Wanbo Zheng are co-first authors.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Authors' Contributions

Zhenxing Wang and Wanbo Zheng contributed equally to this work.

Acknowledgments

This work was in part supported by the Chongqing Research Program of Technology Innovation and Application under Grants cstc2019jscx-msxm0652 and cstc2019jscx-fxyd0385; Key Research and Development Plan of Jiangxi Province (No. 20181ACE50029); Science and Technology Program of Sichuan Province under Grant 2020JDRC0067/2020YFG0326; and the Talent Program of Xihua University under Grant Z202047.

References

[1] H. Gao, L. Kuang, Y. Yin, B. Guo, and K. Dou, "Mining consuming behaviors with temporal evolution for personalized recommendation in mobile marketing apps," *Mobile Networks and Applications*, vol. 25, no. 4, pp. 1233–1248, 2020.

[2] Q. He, G. Cui, X. Zhang et al., "A game-theoretical approach for user allocation in edge computing environment," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 3, pp. 515–529, 2020.

[3] H. Liu, Y. Ma, P. Chen et al., "Scheduling multi-workflows over edge computing resources with time-varying performance, A novel probability-mass function and DQN-based approach," Edited by W. Ku, Y. Kanemasa, M. A. Serhani, and L. Zhang, Eds., in *Proceedings of the Web Services–ICWS 2020-27th International Conference, Held as Part of the Services Conference Federation, SCF 2020*, vol. 12406, pp. 197–209, pp. 197–209, Honolulu, HI, USA, September, 2020.

[4] Y. Ma, J. Zhang, S. Wang et al., "A novel approach to cost-efficient scheduling of multi-workflows in the edge computing environment with the proximity constraint," Edited by S. Wen, A. Y. Zomaya, and L. T. Yang, Eds., in *Proceedings of the Algorithms and Architectures for Parallel Processing-19th International Conference, ICA3PP 2019*, vol. 11944, pp. 655–668pp. 655–, Melbourne, Australia, December, 2019.

[5] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: vision and challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

[6] C. Hoffa, G. Mehta, T. Freeman et al., "On the use of cloud computing for scientific workflows," in *Proceedings of the 2008 IEEE Fourth International Conference on EScience*, pp. 640–645, IEEE, Indianapolis, IN, USA, December, 2008.

[7] G. Juve, E. Deelman, G. B. Berriman, B. P. Berman, and P. Maechling, "An evaluation of the cost and performance of scientific workflows on amazon EC2," *Journal of Grid Computing*, vol. 10, no. 1, pp. 5–21, 2012.

[8] Q. Peng, H. Jiang, M. Chen, J. Liang, and Y. Xia, "Reliability-aware and deadline-constrained workflow scheduling in mobile edge computing," in *Proceedings of the 2019 IEEE 16th International Conference on Networking, Sensing and Control (ICNSC)*, pp. 236–241, IEEE, Banff, Canada, May, 2019.

[9] X. Lyu, W. Ni, H. Tian et al., "Optimal schedule of mobile edge computing for internet of things using partial information," *IEEE Journal on Selected Areas in Communications*, vol. 35, no. 11, pp. 2606–2615, 2017.

[10] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *IEEE Transactions on Wireless Communications*, vol. 16, no. 9, pp. 5994–6009, 2017.

[11] L. Tong, Y. Li, and W. Gao, "A hierarchical edge cloud architecture for mobile computing," in *Proceedings of the 35th Annual IEEE International Conference on Computer Communications, INFOCOM 2016*, pp. 1–9, IEEE, San Francisco, CA, USA, April, 2016.

[12] X. Sun, S. Wang, Y. Xia, and W. Zheng, "Predictive-trend-aware composition of web services with time-varying quality-of-service," *IEEE Access*, vol. 8, pp. 1910–1921, 2020.

[13] X. Yang, S. Zhou, and M. Cao, "An approach to alleviate the sparsity problem of hybrid collaborative filtering based recommendations: the product-attribute perspective from user reviews," *Mobile Networks and Applications*, vol. 25, pp. 376–390, 2019.

[14] X. Wang, C. S. Yeo, R. Buyya, and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1124–1134, 2011.

[15] Z. Wen, J. Cala, P. Watson, and A. Romanovsky, "Cost effective, reliable and secure workflow deployment over federated clouds," *IEEE Transactions on Services Computing*, vol. 10, no. 6, pp. 929–941, 2017.

[16] T. Wu, H. Gu, J. Zhou, T. Wei, X. Liu, and M. Chen, "Soft error-aware energy-efficient task scheduling for workflow applications in DVFS-enabled cloud," *Journal of Systems Architecture*, vol. 84, pp. 12–27, 2018.

[17] E. Cao, S. Musa, J. Zhang et al., "Reliability aware cost optimization for memory constrained cloud workflows," in *Proceedings of the 19th International Conference, ICA3PP 2019*, S. Wen, A. Y. Zomaya, and L. T. Yang, Eds., vol. 11945, pp. 135–150, Springer, Melbourne, Australia, December, 2019.

[18] R. Garg, M. Mittal, and L. H. Son, "Reliability and energy efficient workflow scheduling in cloud environment," *Cluster Computing*, vol. 22, no. 4, pp. 1283–1297, 2019.

[19] S. S. M. Nik, M. Naghibzadeh, and Y. Sedaghat, "Cost-driven workflow scheduling on the cloud with deadline and reliability constraints," *Computing*, vol. 102, no. 2, pp. 477–500, 2020.

[20] Y. Zhu, Y. Hu, T. Yang, and A. Schmeink, "Reliability-optimal offloading in multi-server edge computing networks with transmissions carried by finite blocklength codes," in *Proceedings of the 17th IEEE International Conference on Communications Workshops, ICC Workshops 2019*, pp. 1–6, IEEE, Shanghai, China, May, 2019.

[21] A. Kouloumpris, M. K. Michael, and T. Theocharides, "Reliability-aware task allocation latency optimization in edge computing," in *Proceedings of the 25th IEEE International Symposium on On-Line Testing and Robust System Design, IOLTS 2019*, D. Gizopoulos, D. Alexandrescu, P. Papavramidou, and M. Maniatakos, Eds., pp. 200–203, IEEE, Rhodes, Greece, July, 2019.

[22] Y. Wang, H. Liu, W. Zheng et al., "Multi-objective workflow scheduling with deep-q-network-based multi-agent reinforcement learning," *IEEE Access*, vol. 7, pp. 39974–39982, 2019.

[23] P. Wang, Y. Lei, P. R. Agbedanu, and Z. Zhang, "Makespan-driven workflow scheduling in clouds using immune-based PSO algorithm," *IEEE Access*, vol. 8, pp. 29281–29290, 2020.

[24] S. Saeedi, R. Khorsand, S. Ghandi Bidgoli, and M. Ramezanpour, "Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing," *Computers & Industrial Engineering*, vol. 147, Article ID 106649, 2020.

[25] A. Al-Shuwaili, O. Simeone, A. Bagheri, and G. Scutari, "Joint uplink/downlink optimization for backhaul-limited mobile cloud computing with user scheduling," *IEEE Transactions on Signal and Information Processing Over Networks*, vol. 3, no. 4, pp. 787–802, 2017.

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

[27] Z. Lan, W. Xia, W. Cui et al., "A hierarchical game for joint wireless and cloud resource allocation in mobile edge computing system," in *Proceedings of the 10th International Conference on Wireless Communications and Signal Processing, WCSP 2018*, pp. 1–7, IEEE, Hangzhou, China, October, 2018.

[28] R. Wang, R. C. Purshouse, and P. J. Fleming, "Preference-inspired coevolutionary algorithms for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 4, pp. 474–494, 2013.

[29] J. J. Dongarra, E. Jeannot, E. Saule, and Z. Shi, "Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems," in *Proceedings of the SPAA 2007: 19th Annual ACM Symposium on Parallelism in Algorithms and Architectures*, P. B. Gibbons and C. Scheideler, Eds., pp. 280–288, ACM, San Diego, CA, USA, June, 2007.

[30] G. Juve, A. Chervenak, E. Deelman, S. Bharathi, G. Mehta, and K. Vahi, "Characterizing and profiling scientific workflows," *Future Generation Computer Systems*, vol. 29, no. 3, pp. 682–692, 2013.

[31] https://wiki.mbalib.com/wiki/Monte_Carlo_method.

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

[33] Q. Zhang and H. Li, "MOEA/D: a multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[34] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: improving the strength pareto evolutionary algorithm multiobjective optimization," in *Proceedings of the Evolutionary Methods for Design, Optimization and Control with Applications to Industrial Problems EUROGEN'2001*, Athens, Greece, September, 2001.