A Novel Probabilistic-Performance-Aware Approach to Multi-workflow Scheduling in the Edge Computing Environment

Yuyin Ma[1], Ruilong Yang[1(✉)], Yiqiao Peng[2], Mei Long[4], Xiaoning Sun[1], Wanbo Zheng[3], Xiaobo Li[5], and Yong Ma[6]

[1] School of Computers, Chongqing University, Chongqing, China
yangrl@cqu.edu.cn
[2] Bashu Secondary School, Chongqing, China
[3] Kunming University of Science and Technology, Kunming, China
[4] ZBJ NETWORK Co. Ltd., Chongqing, China
[5] Chongqing Animal Husbandry Techniques Extension Center, Chongqing, China
[6] School of Computer and Information Engineering, Jiangxi Normal University Nanchang, Nanchang, China

Abstract. Edge computing is a decentralized computing infrastructure in which data, calculation, storage and applications are located somewhere between the data source and the computing facilities. While the edge servers enjoy the close proximity to the end-users to provide services at reduced latency and lower energy costs, we use from limitations in computational and radio resources, which calls for smart, quality-of-service (QoS) guaranteed and efficient task scheduling methods and strategies. For addressing the edge-environment-oriented multi-workflow scheduling problem, in this paper, we propose a probabilistic-QoS-aware approach to multi-workflow scheduling over edge servers with time-varying QoS. Our proposed method leveraged a probability-mass function-based QoS aggregation model and a discrete firefly algorithm for generating the multi-workflow scheduling plans. In order to prove the effectiveness of our proposed method, we conducted an experimental case study based on varying types of workflows and a real-world dataset for edge server positions. It can be seen that our method clearly outperforms its competitors in terms of completion time, cost, and deadline validation rate.

Keywords: Edge computing · Workflow scheduling · Probabilistic model · Quality-of-service (QoS)

This work is supported in part by the Graduate Scientific Research and Innovation Foundation of Chongqing, China (Grant No. CYS20066 and CYB20062), and the Fundamental Research Funds for the Central Universities (China) under Project 2019CDXYJSJ0022.

H. Gao et al. (Eds.): CollaborateCom 2020, LNICST 349, pp. 640–655, 2021.
https://doi.org/10.1007/978-3-030-67537-0_38

1 Introduction

The edge computing paradigm is evolving towards a highly efficient computing infrastructure. Different from traditional solutions, it is featured by a ubiquitous, heterogeneous collection of elastic computational entities in terms of, e.g., edge servers. It provisions the platforms for building complex and on-demand applications, in terms of, e.g., workflows with reduced prices compared to traditional parallel computing techniques, e.g., grids. Consequently, great growth in in the number of active research work regarding performance-aware scheduling methods for edge-environment-based workflows can be seen recently scheduling multi-task-based processes [7,11,18] over the edge platform refers to assigning workflow tasks into appropriate edge nodes or servers for execution. A workflow can usually be described as a Directed-Acyclic-Graph (DAG) with multiple tasks that meets the constraint of execution orders.

It is widely believed that to assign tasks within multi-workflows to distributed platforms is an NP-hard problem. It is thus impractical to yield optimal scheduling solutions by using traversal-based strategies. Recently, as novel bio-inspired 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. However, for simplicity, most existing contributions in this direction consider that edge servers are with static and invariable performance. However, edge and cloud servers in real-world can show unstable and time-varying performance. For example, Schad *et al.* [12] obversed that Amazon EC2 cloud services are subject to performance variations of 24%, 20% and 19% for CPU performance, I/O performance and network performance, respectively. Jakson *et al.* [4] showed that the difference between the maximum and minimum runtime of servers is 7,900 s, or approximately 42% of the mean runtime within EC2.

As can be seen from the above analysis, existing heuristic and bio-inspired algorithms with static and time-invariant performance models can be ineffective in dealing with real-world edge-environment-oriented workflow scheduling requirements, where performance of edge servers and platform-level infrastructures themselves are with highly unstable and time-varying performance. To overcome this limitation, in this work, we propose a probabilistic-performance-aware approach to edge-environment-oriented multi-workflow scheduling. Instead of considering single-point and static performance, our proposed method captures the dynamics of performance of edge servers by leveraging the probability mass functions (PMF) of historical performance data and utilizes a firefly algorithm for optimizing the workflow scheduling plans via maximizing the probability that the process completion duration and cost meets the deadline constraint.

To validate our proposed method, we perform extensive simulative studies based on various widely-used scientific workflow templates and a position dataset for urban edge servers. Simulative results show that our proposed method beats its peers in terms of multiple performance metrics.

2 Related Work

The main objective of workflow scheduling algorithms is to identify the best resources in the edge environment for the applications (tasks) of workflows. To achieve this, the pertinent objectives for satisfying users performance constraints include reducing execution time and total execution cost. Recently, considerable research works were carried out in this direction.

For instance, Zhang *et al.* [18] developed a Two-stage Cost Optimization algorithm to schedule workflows on edge clouds. The algorithm first leveraged a BF algorithm for obtaining the initial scheduling strategy and then further optimized the scheduling plans by the first stage. Their algorithm aims to minimize the system cost while meeting the delay requirements of workflows. Kim *et al.* [7] studied the trade-off between execution cost and workflow delays in the mobile computing system and proposed a intelligent-control-based algorithm for achieving near-optimal trade-offs. The trace-driven simulation showed that the algorithm can achieve 71% saving of execution cost and 82% gain of as opposed to its peers. Pandey *et al.* [11] proposed a particle swarm optimization (PSO) algorithm for load-balancing of cloud servers, while minimizing the execution cost, i.e., communication cost plus cloud resource cost of workflows.

kaur *et al.* [6] leveraged a multi-objective bio-inspired procedure (MOBFOA) by augmenting the traditional BFOA with Pareto-optimal fronts. Their method deals with the reduction of ow-time, completion duration, and operational expenditure. Zhang *et al.* [17] considered a multi-objective genetic optimization (BOGA) and optimized both electricity consumption and DAG reliability. Casas *et al.* [1] considered an augmented GA with the Efficient Tune-In (GA-ETI) mechanism for the optimization of turnaround time. Verma *et al.* [13] employed a non-dominated-sorting-based Hybrid PSO approach and aimed at minimizing both turnaround time and expenditure. Zhou *et al.* [19] introduced a fuzzy dominance sort based heterogeneous completion time minimization approach for the optimization of both cost and turnaround time of DAG executed on IaaS clouds.

3 System Model and Problem Formulation

3.1 The System Model

As shown in Fig. 1, an edge computing environment can be seen as a collection of multiple edge servers usually deployed near base stations. By this way, users are allowed to offload compute-intensive and latency-sensitive applications, e.g., Augmented Reality (AR), Virtual Reality (VR), Artificial Intelligence (AI), to edge servers. With in an edge computing environment, there are m users, denoted by $U = \{u_1, u_2, \ldots, u_m\}$, and n edge servers stations, denoted by $ES = \{e_1, e_2, \ldots, e_n\}$. Each user has an application, in terms of a batch of tasks organized by a workflow, to be executed, and users mobile device is allowed to offload tasks to nearby edge servers.

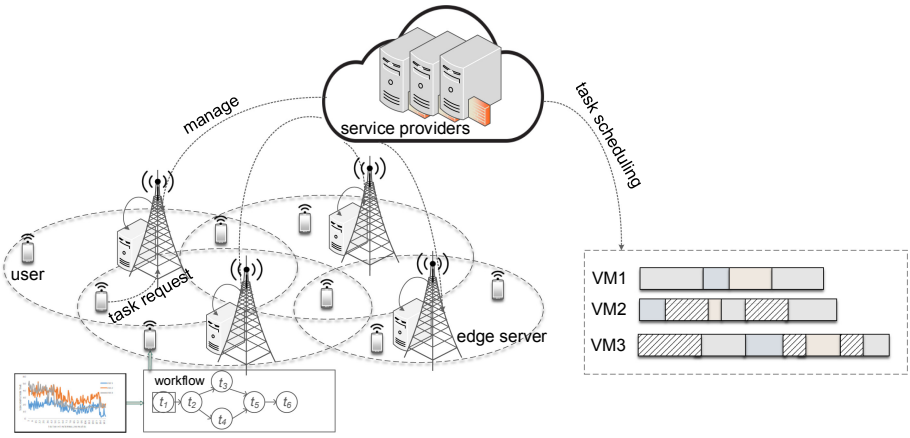


Fig. 1. The system architecture

3.2 The Probabilistic Performance Model

In this work, instead of considering static and time-invariant performance of edge servers, we consider time-varying performance of them. To be specific, we consider that the historical execution time of a certain workflow task upon an edge server, i.e., a discrete random variable X with *Dom(X)*, can be described by an empirical probabilistic distribution.

Consequently, the cumulative distribution function (CDF) of execution time and cost can be calculated as follows:

$$\begin{aligned} P(X \leq c) =& P(X \leq floor(c)) + f_X(ceil(c)) \\ & \times \frac{c - floor(c)}{ceil(c) - floor(c)} \end{aligned} \tag{1}$$

where c is deadline value, $Min(Dom(X)) < c \leq Max(Dom(X))$, $ceil(c) = Min\{c \mid c \in Dom(X)\ and\ c \geq x\}$ and $floor(c) = Max\{c \mid c \in Dom(X)\ and\ c < x\}$.

Table 1. The performance aggregation function of different workflow structural patterns

Workflow Patterns	Response Time	Cost
Sequence	$\sum_{i=1}^{n} RT(t_i)$	$\sum_{i=1}^{n} C(t_i)$
Parallel	$max_{1 \leq i \leq n} RT(t_i)$	$\sum_{i=1}^{n} C(t_i)$
Loop	$k \times RT(t_i)$	$k \times C(t_i)$

Let $w_i(t_{1j_1}, t_{2j_2}, \ldots, t_{nj_n})^{et}$ and $w_i(t_{1j_1}, t_{2j_2}, \ldots, t_{nj_n})^{cost}$ are the execution time and cost of a workflow w_i when it select edge servers of $(j_1, j_2, \ldots, j_n)$. The probability that the resulting workflow completion time and cost meet the deadline constraint, can be estimated according to the following performance aggregation functions and probabilistic performance aggregation rules in Tables 1 and 2 [3].

Table 2. The probabilistic performance aggregation rules

QoS aggregation	Probability
$Z = X + Y$	$Dom(Z) = \{z_1, z_2, \ldots, z_k\}, Max(m,n) \leq k \leq mn$[1], Each $z_i, 1 \leq i \leq k$, is the sum of some $x \in Dom(X)$ *and* $y \in Dom(Y), f_z(z_i) = \sum_{x+y=z_i} f_X(x) f_Y(y)$
$Z = X \cdot Y$	$Dom(Z) = \{z_1, z_2, \ldots, z_k\}, Max(m,n) \leq k \leq mn$[1], Each $z_i, 1 \leq i \leq k$, is the product of some $x \in Dom(X)$ *and* $y \in Dom(Y), f_z(z_i) = \sum_{x \cdot y=z_i} f_X(x) f_Y(y)$
$Z = MAX(X,Y)$	$Dom(Z) = Dom(X) \cup Dom(Y)$. $f_Z(z) = f_X(z) \cdot \sum_{y<z, y \in Dom(Y)} f_Y(y)$ if $z \in Dom(X)$ *and* $z \notin Dom(Y)$; $f_Z(z) = f_Y(z) \cdot \sum_{x<z, x \in Dom(X)} f_X(x)$ if $z \in Dom(Y)$ *and* $z \notin Dom(X)$; $f_Z(z) = f_X(z) \cdot \sum_{y \leq z, y \in Dom(Y)} f_Y(y) + f_Y(z) \cdot \sum_{x<z, x \in Dom(X)} f_X(x)$ if $z \in Dom(X)$ *and* $z \in Dom(Y)$

[1] $m = |Dom(X)|, n = |Dom(Y)|$.

3.3 Problem Description

Based on the above analysis, the problem of probabilistic-performance-aware multi-workflow scheduling can be described as follows: given multiple workflows $w_{i_{1 \leq i \leq m}}$, we are interested to identify an edge server assignment plan $(t_{1j_1}, t_{2j_2}, \ldots, t_{nj_n})$ of w_i, with the highest probability that the workflow completion time and cost meet the deadline constraint.

$$\begin{aligned} max \quad f = & \prod P(w_i(t_{1j_1}, \ldots, t_{nj_n}))^{et_i} \leq C_i^{et_i} \\ & \times \prod P(w_i(t_{1j_1}, \ldots, t_{nj_n}))^{cost_i} \leq C_i^{cost_i} \end{aligned} \tag{2}$$

s.t.

$$d_{ij} \leq cov_j, i \in \{1, \ldots, m\} \; and \; j \in \{1, \ldots, n\} \tag{3}$$

$$x_{ij} \leq 1, \; x_{ij} = \begin{cases} 1, & if \; e_j \; is \; selected \; for \; task \; t_i \\ 0, & otherwise \end{cases} \tag{4}$$

where $C_i^{et_i}(C_i^{cost_i})$ is the deadline constraint for completion time and cost of each workflow, d_{ij} is the distance between e_j and w_i, and cov_j is the coverage area of the j_{th} the server.

4 Firefly Algorithm

The firefly algorithm (FA) [15,16] is a meta-heuristic searching technology proposed, which simulates the luminous characteristics and attraction behavior of the fireflies. In this algorithm, fireflies are considered the sample points in the problem domain, and each firefly moves towards a brighter firefly that ultimately finds the optimal location. The individual renewal equation in FA consists of two parts: (1) the full attraction model; and (2) a randomly disturbed searching step size. The full attraction model requires every firefly to learn from all superior individuals.

In this work, we leverage the discrete derivative of FA, i.e., DFA, for solving the probabilistic-performance-aware workflow scheduling problem. The details are described as follows.

4.1 Encoding

In DFA, a schedule is an individual, described as a vector of integer values. The length of a vector is the same as the number of tasks in a workflow. The i_{th} element of the individual indicates to which server the i_{th} task of the workflow is scheduled to execute. Figure 2 gives a sample of an individual coding and a given workflow deployment, assuming that the workflow consists of eight tasks and is within the coverage range of by e_2, e_3, and e_4. In this schedule, t_1, t_4, t_6 are scheduled to be executed on e_3, t_2, t_3, t_7 are scheduled on e_2, and t_5, t_8 are scheduled on e_4, respectively.

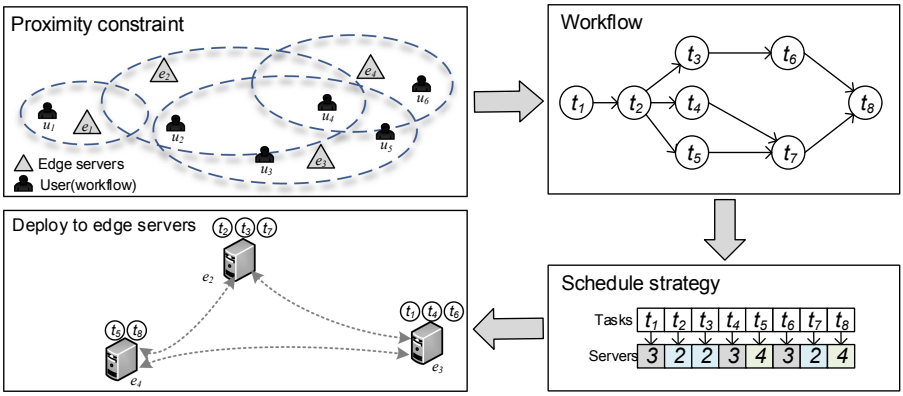


Fig. 2. An example of encoding

4.2 Population Initialization and Firefly Evaluation

The initialization of population is affected by the coverage range constraints due to the fact that, users can only offload tasks to reachable, in terms of the coverage range edge server.

As described in Sect. 3, our goal is to find the highest probability that the deadline constraint is met. The constraint is interpreted as a penalty function, where the highest probability is integrated into (5) as an evaluation of the fitness value of an individual. In case that the user is not reachable to servers, the fitness value is decided by the number of tasks that violate the deadline constraint.

$$F = \frac{1}{m} \times \begin{cases} \sum_{i=1}^{m} f_i, & if\ d_{ij} \leq cov_j \\ \sum_{i=1}^{m} f_i \times (\sqrt[n]{f_i})^{num}, & otherwise \end{cases} \tag{5}$$

where n denotes the total number of tasks in w_i, and $num \in \{1, 2, \ldots, n\}$, is the number of tasks that are unreachable by edge servers.

4.3 Individuals Update

The stipulation of update is that the darker firefly moves towards the brighter one. The population is updated iteratively and the scheduling strategy keeps being optimized. An example showing the update process is given in Table 3. The movement process is as follows.

Distance Calculation. The distance between any two fireflies p and p_{best} is measured by its corresponding hamming distance.

$$dis(x_i, x_j) = x_{id} \oplus x_{jd},\ d \in \{1, 2, \ldots, n\} \tag{6}$$

where OR operator for exclusion is applied to calculate the hamming distance [9].

β-step Update. β-step indicates the operation of firefly movement towards brighter ones with the steps given below.

Step 1: Decide the hamming distance of individuals as dis_1;
Step 2: Decide the attractiveness, $\beta(r)$, according to (7);
Step 3: Yield $|dis_1|$ random numbers between 0 and 1. If the random number is smaller than $\beta(r)$, the element of the p is substituted by the corresponding element of the brightest firefly;
Step 4: move towards the brightest one.

$$\beta(r) = \frac{\beta_0}{(1 + \gamma \times dis^2)} \tag{7}$$

α-step Update. The α-step update must be after the β-step update, according to (8), which is a process of random disturbance to avoid the solution space falling into the local optimization. Algorithm 1 presents all the operations of the discrete firefly algorithm.

$$x_i = x_i + \alpha(rand_{int}) \tag{8}$$

Table 3. Solution updation

Updation	Edge server assignment
Current firefly p	$\{2, 3, 2, 2, 3, 3, 4, 2\}$
Best firefly p_{best}	$\{3, 2, 2, 3, 4, 3, 2, 2\}$
Distance dis_1	5
Attractiveness $\beta(r)$	0.24
$rand(\ )between(0, 1)$	$\{\boxed{0.13}, 0.29, \boxed{0.03}, \boxed{0.11}, 0.67\}$
firefly p after β-step	$\{3, 3, 2, 3, 4, 3, 4, 2\}$
firefly p after α-step	$\{3, 3, 2, \boxed{4}, \boxed{3}, 3, 4, 2\}$

Table 4. Resource configurations and the price-per-minute of edge servers

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

5 Performance Evaluation

To evaluate the effectiveness of our proposed method, we conduct simulative experiments based on multiple workflow templates [5], namely, ***CyberShake***, ***Inspiral***, and ***Sipht***, as shown in Fig. 3.

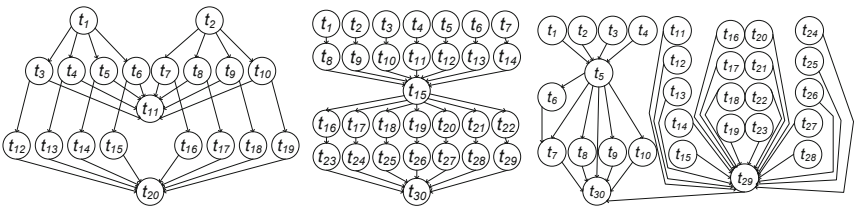
We consider that all edge servers are with 3 different types of resource configurations and charging plans, i.e., ***tp1***, ***tp2***, and ***tp3***, as shown in Table 4. We tested the completion time of tasks on three types of edge servers at different periods, i.e., (a), (b) and (c), as shown in Fig. 4. As can be seen, the period of 4(a) shows the weakest performance fluctuations while 4(c) shows the greatest. The positions for edge servers and users are based on the dataset given in [2,8,14] and illustrated in Fig. 5.

Algorithm 1: Firefly Algorithm

Input: Algorithm related parameters: α, β_0, γ, $iter_{max}$
Output: Global QoS probability F_{best}, and schedule strategy S
Initial population of firelies: x_i, $i \in \{1,2,..,n\}$;
while t < $iter_{max}$ **do**
  for i=1:n **do**
    for j=1:i **do**
      if $F(x_i) > F(x_j)$ **then**
        move firefly j towards firefly i;
      else
        move firefly i towards firefly j;
      end
      Update the attractiveness of all fireflies;
      Evaluate new solution and update $F(x_i)$;
    end
    Rank the fireflies and find the current Global QoS probability F_{best}, and schedule strategy S;
  end
end
return F_{best}, S;

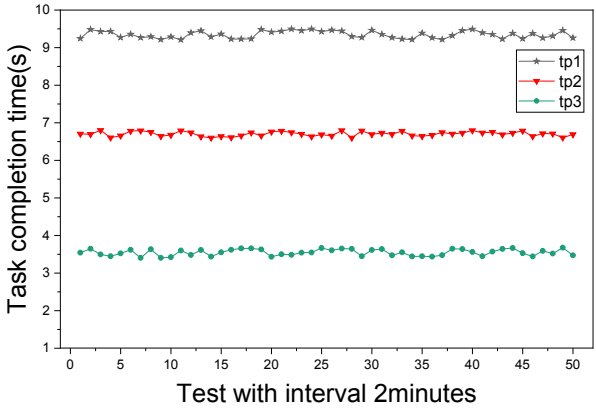
For comparison, we consider *pure* FA, GA, Greedy, and Random as baseline algorithms:

- ***pure* FA** [10]: the method is a heuristic algorithm, it is used to solve the multi-workflow scheduling problem with proximity constraint. It is noted that the QoS value in the method is constant. We use the mean QoS value as the static value of the firefly algorithm.
- **GA** [20]: the method is a heuristic search algorithm for workflow scheduling in cloud.
- **Greedy:** the method schedules unassigned tasks to the available lowest cost edge servers.
- **Random:** the method randomly selects a edge server for unassigned task.

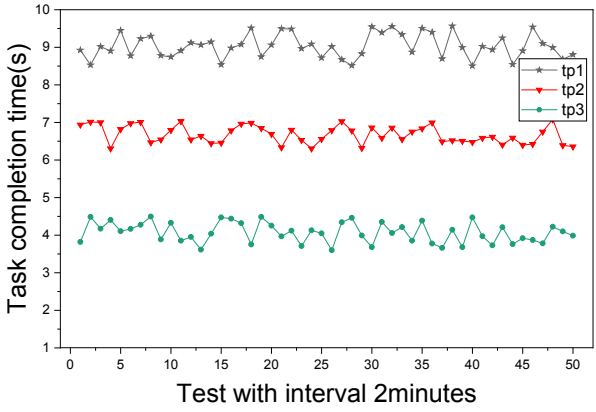


(a) CyberShake (b) Inspiral (c) Sipht

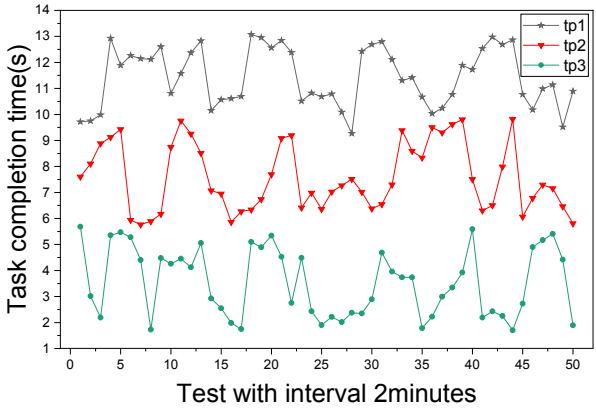
Fig. 3. Workflow process models



(a) Stable



(b) Moderate



(c) Fluctuant

Fig. 4. The completion time of workflow tasks at different types of edge servers in different periods

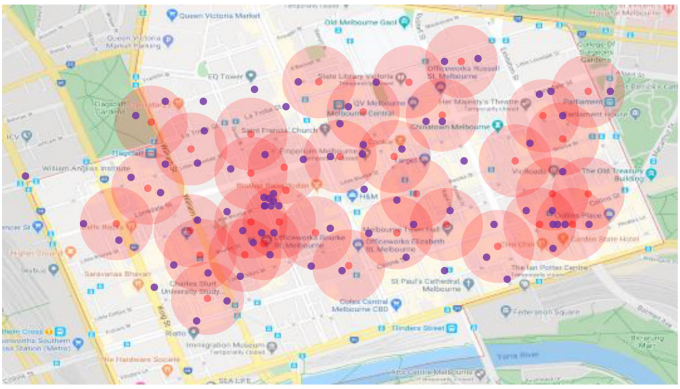
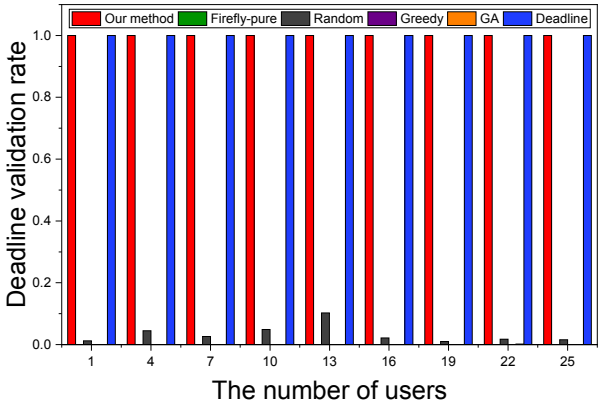


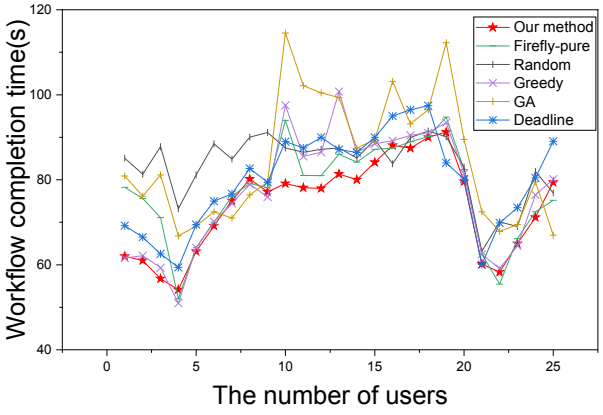
Fig. 5. Edge servers and users deployment

At the periods of 4(a), 4(b) and 4(c), we show in Figs. 6, 7 and 8 the comparison of scheduling performance of different methods in terms of deadline constraint satisfaction rate, workflow completion time, and cost. As can be observed, our method beats Random, *pure* FA, Greedy and GA in terms of average deadline validation rate. Moreover, our method clearly achieves lower workflow completion time and cost. To be specific, the cost of our method is 1.7%, 1.4%, and 3.4% lower than *pure* FA on average at three periods, respectively; 5.7%, 5.4%, and 8.4% lower than Random; 5.3%, 6.1%, and 5.6% lower than Greedy; and 2.2%, 1.4%, and 3.5% lower than GA. The workflow completion time of our method is 4.5%, 45.8%, and 49.7% lower than that of *pure* FA on average; 12.8%, 26.4%, and 62.3% lower than Random; 4.4%, 23%, and 34.7% lower than Greedy; and 14.5%, 51.4%, and 37.1% lower than GA, respectively.

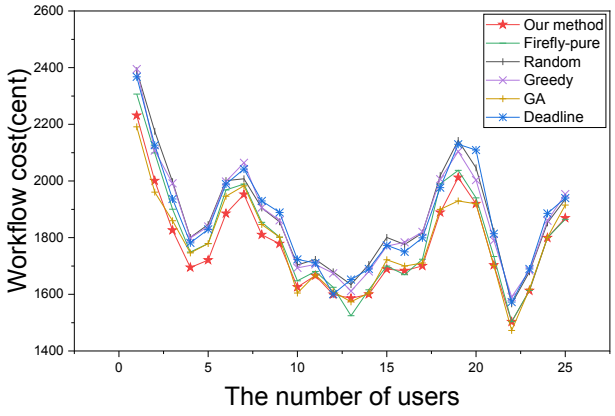
The advantage of our proposed method is achieved because of the fact that traditional workflow scheduling algorithms are designed on the cloud without taking into account the real-time performance fluctuations of edge servers. However, this is not real in the edge computing environment where each server fluctuates in performance. From the above data, it can be seen that our method has a greater advantage when server performance fluctuates sharply. This is because our method takes performance into consideration to schedule workflow makes it superior to its peers in terms of completion time and cost.



(a) deadline validation rate

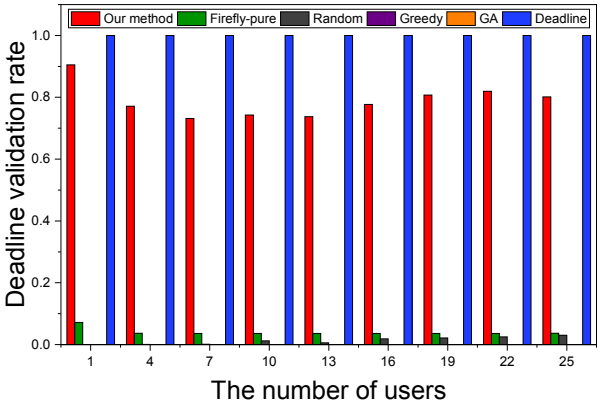


(b) completion time(s)

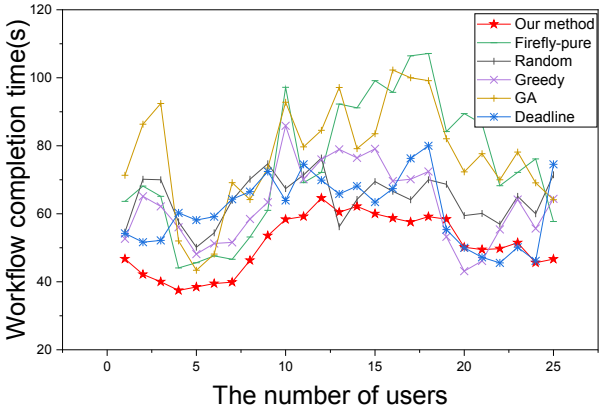


(c) cost(cent)

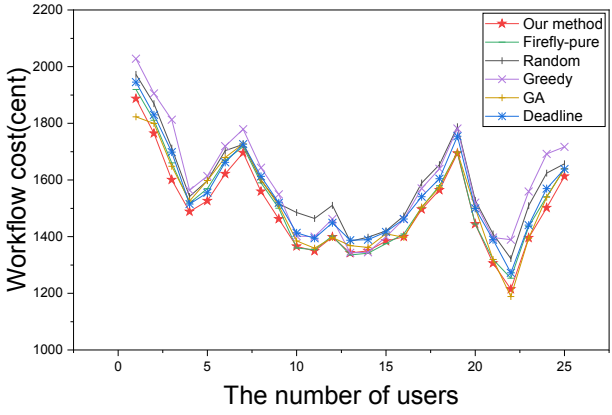
Fig. 6. The comparison of different methods at the period of 4(a) as the input performance data.



(a) deadline validation rate

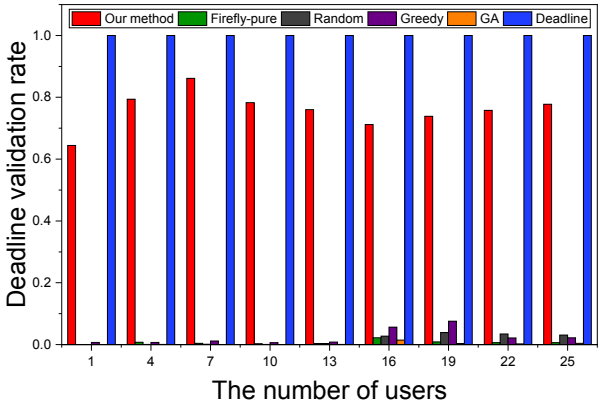


(b) completion time(s)

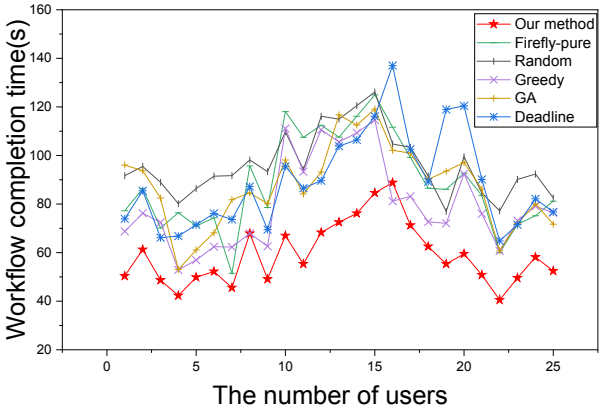


(c) cost(cent)

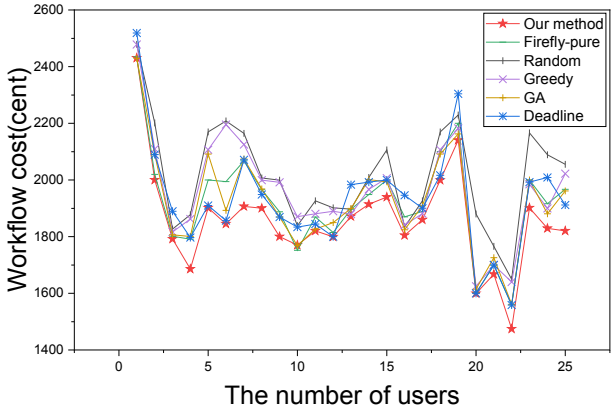
Fig. 7. The comparison of different methods at the period of 4(b) as the input performance data.



(a) deadline validation rate



(b) completion time(s)



(c) cost(cent)

Fig. 8. The comparison of different methods at the period of 4(c) as the input performance data.

6 Conclusion

In this manuscript, we introduce a novel probabilistic-performance-aware method to multi-workflow scheduling in the edge computing environment. Instead of considering static and time-invariant performance of edge servers, our approach fully exploits the real-time performance fluctuations of them and leverages a probabilistic-performance-distribution-based mechanism in feeding a discrete-FA-based optimization method for generating scheduling plans. Experimental results based on several test cases, and a real-world edge-server-location dataset show that our proposed method clearly outperforms traditional approaches in terms of multiple performance metrics.

References

1. Casas, I., Taheri, J., Ranjan, R., Wang, L., Zomaya, A.Y.: GA-ETI: an enhanced genetic algorithm for the scheduling of scientific workflows in cloud environments. J. Comput. Sci. **26**, 318–331 (2018)
2. He, Q., et al.: A game-theoretical approach for user allocation in edge computing environment. IEEE Trans. Parallel Distrib. Syst. **31**(3), 515–529 (2020)
3. Hwang, S.Y., Wang, H., Tang, J., Srivastava, J.: A probabilistic approach to modeling and estimating the QoS of web-services-based workflows. Inf. Sci. **177**(23), 5484–5503 (2007)
4. Jackson, K.R., et al.: Performance analysis of high performance computing applications on the amazon web services cloud. In: 2010 IEEE Second International Conference on Cloud Computing Technology and Science, pp. 159–168. IEEE (2010)
5. Juve, G., Chervenak, A., Deelman, E., Bharathi, S., Mehta, G., Vahi, K.: Characterizing and profiling scientific workflows. Future Gener. Comput. Syst. **29**(3), 682–692 (2013)
6. Kaur, M., Kadam, S.: A novel multi-objective bacteria foraging optimization algorithm (MOBFOA) for multi-objective scheduling. Appl. Soft Comput. **66**, 183–195 (2018)
7. Kim, Y., Kwak, J., Chong, S.: Dual-side optimization for cost-delay tradeoff in mobile edge computing. IEEE Trans. Veh. Technol. **67**(2), 1765–1781 (2017)
8. Lai, P., et al.: Optimal edge user allocation in edge computing with variable sized vector bin packing. In: Pahl, C., Vukovic, M., Yin, J., Yu, Q. (eds.) ICSOC 2018. LNCS, vol. 11236, pp. 230–245. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-03596-9_15
9. Lunardi, W.T., Voos, H.: An extended flexible job shop scheduling problem with parallel operations. ACM SIGAPP Appl. Comput. Rev. **18**(2), 46–56 (2018)
10. Ma, Y., et al.: A Novel approach to cost-efficient scheduling of multi-workflows in the edge computing environment with the proximity constraint. In: Wen, S., Zomaya, A., Yang, L.T. (eds.) ICA3PP 2019. LNCS, vol. 11944, pp. 655–668. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38991-8_43
11. Pandey, S., Wu, L., Guru, S.M., Buyya, R.: A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: 2010 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 400–407. IEEE (2010)
12. Schad, J., Dittrich, J., Quianeruiz, J.: Runtime measurements in the cloud: observing, analyzing, and reducing variance. Proc. VLDB Endow. **3**(1), 460–471 (2010)

13. Verma, A., Kaushal, S.: A hybrid multi-objective particle swarm optimization for scientific workflow scheduling. Parallel Comput. **62**, 1–19 (2017)
14. Xia, X., Chen, F., He, Q., Grundy, J.C., Abdelrazek, M., Jin, H.: Cost-effective app data distribution in edge computing. IEEE Trans. Parallel Distrib. Syst. **32**(1), 31–44 (2021)
15. Yang, X.-S.: Firefly algorithms for multimodal optimization. In: Watanabe, O., Zeugmann, T. (eds.) SAGA 2009. LNCS, vol. 5792, pp. 169–178. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04944-6_14
16. Yang, X.S.: Firefly algorithm, stochastic test functions and design optimisation. arXiv preprint arXiv:1003.1409 (2010)
17. Zhang, L., Li, K., Li, C., Li, K.: Bi-objective workflow scheduling of the energy consumption and reliability in heterogeneous computing systems. Inf. Sci. **379**, 241–256 (2017)
18. Zhang, Y., Chen, X., Chen, Y., Li, Z., Huang, J.: Cost efficient scheduling for delay-sensitive tasks in edge computing system. In: 2018 IEEE International Conference on Services Computing (SCC), pp. 73–80. IEEE (2018)
19. Zhou, X., Zhang, G., Sun, J., Zhou, J., Wei, T., Hu, S.: Minimizing cost and makespan for workflow scheduling in cloud using fuzzy dominance sort based HEFT. Future Gener. Comput. Syst. **93**, 278–289 (2019)
20. Zhu, Z., Zhang, G., Li, M., Liu, X.: Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans. Parallel Distrib. Syst. **27**(5), 1344–1357 (2016)