# A Self-adaptive PSO-Based Dynamic Scheduling Method on Hierarchical Cloud Computing

Shunmei Meng[1](✉), Weijia Huang[1], Xiaolong Xu[2], Qianmu Li[1], Wanchun Dou[3], and Bowen Liu[3]

[1] Department of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing, China
{mengshunmei,weijia,qianmu}@njust.edu.cn

[2] Key Laboratory of Intelligent Perception and Systems for High-Dimensional Information of Ministry of Education, Nanjing University of Science and Technology, Nanjing, China
xlxu@nuist.edu.cn

[3] State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing, China
douwc@nju.edu.cn, liubw@smail.nju.edu.cn

**Abstract.** Edge computing has been envisioned as an emerging and prospective computing paradigm for its advantage of low latency, which uses local resources. However, the edge resources are usually limited and could not meet end-users' diversified requirement. Cloud computing paradigm could provide scalable and centralized resources with high computational capabilities, but it has latency issues. Thus it is suggested to combine both computing paradigms together to improve the performance of mobile applications. In this paper, we propose a self-adaptive dynamic scheduling approach based on hierarchical heterogeneous clouds. Our scheduling mechanism considers not only schedule planning but also dynamic scheduling on heterogeneous clouds. Firstly, a self-adaptive scheduling mechanism based on a meta-heuristic optimization algorithm, PSO (Particle Swarm Optimization), is presented for schedule planning. Then a dynamic scheduling mechanism on dynamic partial workflow model is proposed for dynamic optimization during the execution. Finally, external experiments compared with other methods are conducted to demonstrate the effectiveness of our proposal.

**Keywords:** Dynamic scheduling · PSO · Mobile edge computing · Cloud · Workflow

## 1 Introduction

Cloud computing has been witnessed to be one of the most promising contemporary technologies, which is a distributed computing paradigm and can provide centralized scalable resources for Internet users [1]. It has been considered to be an effective technique that powers up the implementation of service-oriented computing systems, which drives new levels of performance and productivity in multiple domains [2]. Moreover,

X. Zhang et al. (Eds.): CloudComp 2019/SmartGift 2019, LNICST 322, pp. 89–100, 2020.
https://doi.org/10.1007/978-3-030-48513-9_7

with the development of mobile devices and mobile applications, cloud computing has been proved to be a promising tool for mobile applications where computations/data can be offloaded to these centralized resources [3].

However, migrate workloads of mobile applications to centralized clouds may lead to more time consumption as the conventional centralized cloud resources are usually deployed remotely from mobile users. With the progressive development of wireless communication technology, mobile edge computing (MEC) is conceived as another prospective computing paradigm for its advantage of low latency [4]. MEC could provide local resources for mobile users, which can power up the implementation of delay-sensitive mobile applications. While compared with centralized public cloud resources, the scale and the computing capacity of local edge resources are limited.

Motivated by the different features of the low-latency edge resources and remote scalable cloud instances, heterogeneous cloud are suggested to be jointly scheduled. There have been many researches studying on task or resource provision and scheduling problem in both cloud and MEC systems [5–7]. However, these researches mainly focus the usage of single kind of cloud, i.e., only remote centralized public cloud or only edge cloud. There are few works studying the combination of the heterogeneous cloud. Authors in [8] provide a micro-services provisioning mode for IoT (Internet of Things) applications based on heterogeneous cloud-edge computing paradigm.

In scheduling problem, resource provisioning and scheduling aims to implement the reasonable deployment of cloud resources to satisfy users' requirements. However, most of existing scheduling methods are schedule planning before execution, which ignore the dynamics during the execution. Actually, especially in mobile edge cloud, due to the mobile characteristics and the complicated network environment, there are probably many dynamics and exceptions occurred while executing. Thus reactive dynamic provision is emerged for handling the dynamics and exceptions while executing of mobile applications.

To deal with the issues mentioned above, in this paper, a self-adaptive dynamic scheduling method on hierarchical clouds is proposed to provide effective scheduling mobile applications. It aims to optimize the cost while meeting the deadline. Specifically, a heterogeneous cloud model is designed to apply both centralized and low-latency cloud instances. A self-adaptive PSO-based scheduling approach on heterogeneous clouds is proposed to deploy appropriate resources for the tasks of mobile applications. To react to the dynamics of mobile environment, a dynamic scheduling mechanism is designed on the dynamic partial workflow model. Finally, experiments are conducted to demonstrate the effectiveness of the method proposed in this paper.

The reminder of this paper is organized as follows: Sect. 2 gives the system model of our proposal. In Sect. 3, a self-adaptive PSO-based dynamic scheduling method on heterogeneous clouds is presented. Section 4 presents the empirical performance and effectiveness of the method proposed in this paper. Section 5 reviews some related researches. Finally, Sect. 6 concludes our paper and presents some researches of our future work.

## 2 System Model

### 2.1 Overview of the Heterogeneous Cloud Model

In this section, a heterogeneous cloud architecture is proposed to provide both scalable and low-latency cloud resources to meet different requirements of mobile users, which is shown in Fig. 1. It consists of two layers. Layer-1 provides the remote scalable cloud resources providing by public cloud providers. It could apply centralized resources with high availability and scalability. Layer-2 presents the local edge resources provided by local mobile devices, which could provide edge resources with low latency. Mobile users schedule their excessive tasks to low-latency edge devices by the wireless access network. And the mobile edge cloud can be connected to the remote scalable cloud in layer-1 through the Internet. When mobile requests arrive, scheduling decisions are made according to scheduling strategy to determine optimal deployment of resources to each task in the mobile applications so as to meet the requirement of mobile users.

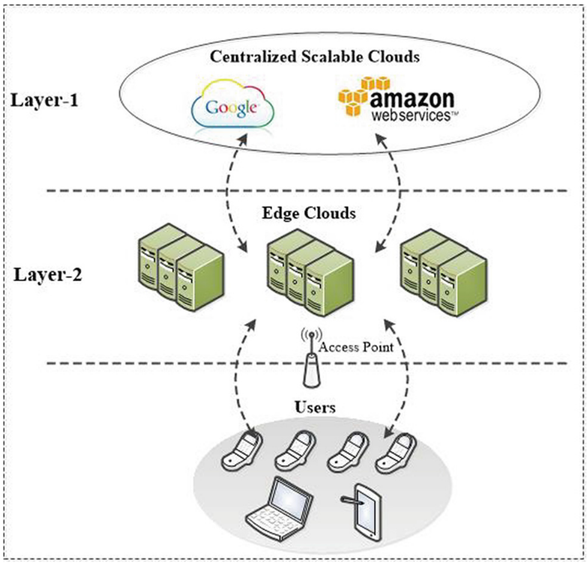


**Fig. 1.** The heterogeneous cloud architecture.

### 2.2 PSO Algorithm

The Particle Swarm Optimization (PSO) scheme is a cooperative meta heuristic method, which was proposed by Kennedy and Eberhartin in 1995 [9]. The PSO algorithm is a global search swarm intelligent method based on population mechanism. In initialization of the basic PSO, the initial particles are generated randomly. Every particle in the swarm can be represented as a two-tuple $\{x_i, v_i\}$, where $x_i = \{x_{i1}, x_{i2}, \ldots, x_{iD}\}$ is denoted as the position vector, $v_i = \{v_{i1}, v_{i2}, \ldots, v_{iD}\}$ is denoted as the velocity vector, and $D$ is the dimensional of the search space which corresponds to the number of tasks in the scheduling problem. In PSO, the moving direction of each particle can be determined by Eq. (1).

$$v_{id}^{k+1} = \omega v_{id}^{k} + c_1 r_1 \left( pbest_{id} - x_{id}^{k} \right) + c_2 r_2 \left( gbest_{id} - x_{id}^{k} \right) \tag{1}$$

where $d = 1, 2, \ldots, D(1 \leq d \leq D)$, $v_{id}^{k+1}$ and $v_{id}^{k}$ are the $d$-th velocities of particle $i$ at time $k$ and time $k+1$, respectively. $\omega$ is the inertia weight, $r_1$ and $r_2$ are random numbers ranging between the interval [0, 1], $c_1$ and $c_2$ are acceleration coefficients, $pbest_{id}$ is the personal best position of particle $i$, $gbest_{id}$ is the global best position of the entire swarm.

Then the individual particle updates its position based on its previous position $x_{id}^{k}$ and $v_{id}^{k+1}$, which is shown in Eq. (2):

$$x_{id}^{k+1} = x_{id}^{k} + v_{id}^{k+1} \tag{2}$$

Then after the iterated advances based on the updating, particles make further searches in the searching space with the new updated velocity and position. The swarm gradually approaches the optimal solution. In our work, we will apply the PSO algorithm into our scheduling problem to find optimal solutions.

## 3 Self-adaptive PSO-Based Dynamic Scheduling Method

In this work, inspired by the advantages (such as high-precision solutions and fast convergence) of PSO mechanism, a self-adaptive PSO-based dynamic scheduling method is proposed to deal with the scheduling and provisioning problem for mobile users. Specifically, a schedule planning method based on the PSO algorithm is presented firstly. Then to deal with the mobility, a dynamic scheduling method based on the dynamic partial workflow model is presented to minimize the execution cost dynamically during the execution of the application workflow.

### 3.1 Self-adaptive PSO-Based Scheduling Mechanism

Although PSO has been used in many domains. It is still not easy to get optimal solution based on standard PSO for its robustness, i.e., the ability to track changes in convergence state of the particle swarm. To address this issue, an adaptive PSO algorithm with better search capability is applied to solve the scheduling problem in our proposal. An improved initial population generation strategy and an adaptive parameter setting strategy are designed to improve the search capability and avoid local optimal solution.

(1) Improved Initialization Strategy
In our PSO-based scheduling method, the critical path (CP) strategy is applied to filter the poor solution and get a better initial population, which consists of two phases:

1) *Determine the CP of the workflow:* The critical path of the workflow must contain the exit task $t_{exit}$. The CP-finding process starts form the exit task $t_{exit}$ and stop at the entry task $t_{entry}$. The latest finish time (LFT) of $t_{exit}$ should be no more than the deadline *UD*. In the critical path, each individual task *t* has a critical predecessor task. And the critical predecessor task $t_{cp}$ is the task which has the latest arrival time to task *t*. After determining the critical parent $t_{cp}$ of task *t* in *CP*, $t_{cp}$ will be added to the critical path *CP*. Repeatedly, the overall critical path for the workflow will be found.
2) *CP-based Initialization:* To improve the quality of initial particles, only the random initial particle where the finish time of the critical path is less than the deadline *UD* will be seen as a good initial solution. Otherwise the initial particle will be regenerated until the finish time of the critical path is less than the deadline *UD*. As the number of tasks in the critical path is far less than the number of tasks of the workflow. Thus it will consume little time to filter the bad initial particles.

(2) Adaptive Parameters Settings:
For self-adaptive learning of PSO, an adaptive inertia weight is adopted for velocity and position updating. The inertia weight in the *k*-th iteration can be determined by Eq. (3).

$$\omega_k = \omega_{max} - \frac{(\omega_{max} - \omega_{\min})}{K} * k \tag{3}$$

where $\omega_{max}$ and $\omega_{\min}$ are the maximum and minimum weight factor values, *K* is the total number of iterations and *k* is the current number of the iterations.

The acceleration parameters $c_1$ and $c_2$ are also adaptively changing, which are varied with the number of iterations and changed asynchronously with the time.

$$\begin{aligned} c_1 &= \tfrac{k}{K}(c_{1f} - c_{1i}) + c_{1i} \\ c_2 &= \tfrac{k}{K}(c_{2f} - c_{2i}) + c_{2i} \end{aligned} \tag{4}$$

where $c_{1f}, c_{2f}, c_{1i}, c_{2i}$ are constants which are empirically decided, and $c_{1i}$ and $c_{2i}$ are the initial values of $c_1$ and $c_2$. Then based on the adaptive inertia weight and the adaptive acceleration parameters described above, the velocity and position of particles of the swarm can be updated based on the Eq. (1) and (2).

(3) Evaluation Function
In PSO-based scheduling problem, an evaluation function is needed to calculate the fitness value of each particle in the swarm. As our goal is to optimize the overall execution cost while meeting the deadline, and the fitness function can be defined as follows.

$$\begin{aligned} & EC = \sum\nolimits_{i=1}^{n} c_i \\ & s.t. \sum\nolimits_{i=1}^{n} ET_i \le UD \end{aligned} \tag{5}$$

where $c_i$ and $ET_i$ represent the execution cost and the execution time of task $t_i$, respectively. The execution time $ET_i$ of task $t_i$ contains both the output/input data transmission time between task to resource and the execution time on the selected resource.

Calculate the fitness values of each scheduling solution based on the evaluation function. For each particle, if the fitness value of its current position outperforms than the best position of this particle *pbest*, then update *pbest* with current position. And if the fitness value of its current position is better the best position of the swarm *gbest*, then reset *gbest* with the new position of the particle. Then update the velocity and position of all particles based on the updating equations. The search will be iterated until the stop criterion is satisfied.

## 3.2 Dynamic Scheduling Based on Dynamic Partial Workflow Model

To deal with the influence of mobility of users on scheduling during the execution, a dynamic resource scheduling method is designed. It is based on the adaptive PSO mechanism and a dynamic workflow model to provision appropriate resources for mobile applications dynamically.

(1) Mobility model of mobile users
In this section, a commonly used mobility model named random waypoint [10] is applied to formulate the movement of users in our problem. Based on random waypoint mobility scheme, the movement of users could be modeled as the moving trajectory between different waypoints where mobile users move at pre-set speeds and stop at next waypoint for some time period. More details about the random waypoint model can be referred to reference [10].

Figure 2 gives the moving track of a mobile user, which is designed according to the random waypoint mobility scheme. User starts by choosing its first waypoint $p_1$ randomly and stop at $p_1$ for some time period $t_r$ and $t_r$ is within $[PT_{min}, PT_{max}]$. Then the user selects another waypoint, i.e., the second waypoint $p_2$, and moves to it at a random speed between $[pv_{min}, pv_{max}]$. User will also pause at $p_2$ for another random time period $\mathrm{T}_2$ which is between $[PT_{min}, PT_{max}]$. This procedure will be repeated for the whole execution process of the mobile application.

(2) Generating dynamic workflows based on the mobility model
Since the local edge resources are constantly changing as the user moves, then the optimized services to each individual task may vary with the movement of mobile users while executing. Thus it is necessary to update the scheduling for the uncompleted tasks dynamically at each waypoint. To address this issue, dynamic partial workflow model is designed for rescheduling, where a dynamic partial workflow will be constructed at each waypoint.

The dynamic partial workflow at a waypoint is a sub workflow cut-off from the original application workflow, which only contains the uncompleted tasks (tasks that are being executed) and unexecuted tasks (tasks that are not yet executed). And each waypoint has a dynamic partial workflow.

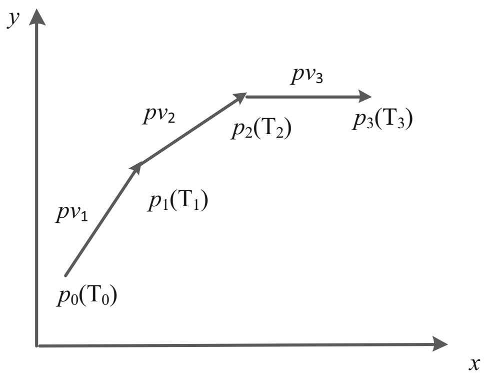


**Fig. 2.** User moving trajectory based on RWP model.

Different waypoint has different local edge cloud resources, thus re-scheduling should only be conducted for the remaining tasks. At a waypoint $p_i$, the dynamic workflow has three kinds of tasks: completed tasks, executing tasks and unexecuted tasks. The dynamic partial workflow at each waypoint only contains executing tasks and unexecuted tasks at that time.

(3) Rescheduling based on the dynamic partial workflow
Due to the local edge resources are changing with the movement of mobile users, real-time optimal scheduling strategy should be applied for the uncompleted tasks. And reactive scheduling strategy should be re-planed on the DPW workflow. In our work, the executing tasks in the DPW workflow will continue to be executed in the original assigned resources. The unexecuted tasks will be re-provisioned with real-time resources according to the above PSO-based scheduling approach at current waypoint.

## 4 Experiment

Experiments are performed to evaluate the effectiveness of the proposed dynamic scheduling model. Experimental settings are described in the following firstly.

### 4.1 Experiment Settings

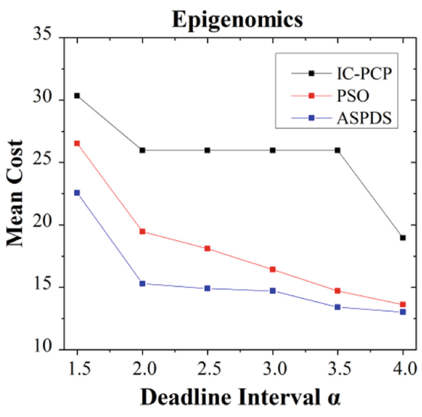
In this paper, to design the hierarchical heterogeneous cloud, Amazon T2 instances are used to simulate remote centralized cloud with scalable resources, and a set of PC nodes are used to simulated as edge resources in Layer-2. It is assumed that the prices of each edge resource and their process capacity are usually positively related.

Two scientific workflows from different areas, i.e., CyberShake and Epigenomics, are applied to model the mobile application workflow with multiple tasks. The details of the five scientific workflows can be referred to reference [11]. In our experiment, the parameters related to $c_1$ and $c_2$ (acceleration coefficients in Eq. (6)) are empirical values. Which is set as: $c_{1f} = 0.5$, $c_{2f} = 2.5$, $c_{1i} = 2.5$, $c_{2i} = 0.5$.

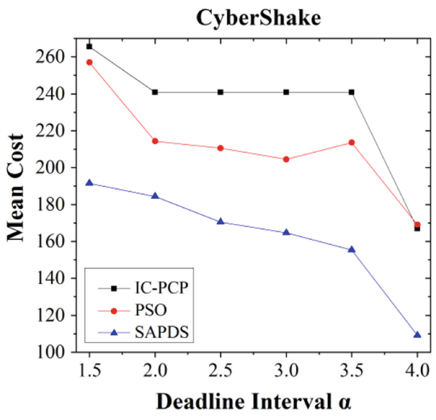
## 4.2 Experimental Result

To validate the performance of our proposal, our method is compared (denoted as SAPDS) with two other comparative methods: IC-PCP [12] and standard PSO [9]. IC-PCP introduced in [12] is designed for deadline-constrained scheduling problem, which also aims to minimize the execution cost and satisfy the deadline.

Figure 3 (a) and Fig. 3 (b) provide the performance of the three methods in the mean execution cost on six different deadlines for Epigenomics and CyberShake, respectively. In our experiment, the deadlines are set between the slowest execution time *se* (all tasks are executed on the slowest resource) and the fastest execution time *fe* (all tasks are executed on the fastest resource), which is similar to the research [13]. The difference between the slowest and fastest runtime *se* - *fe*, denoted as *D'*, is divided by 4 to obtain an interval size *D'*/4. As shown in Fig. 3, all of the mean execution costs of the three methods decrease with the increase of the deadline in both of the two workflows. And our method, i.e., SAPDS, has lower execution cost than IC-PCP and standard PSO. This is because IC-PCP only executes the scheduling only once, which didn't consider dynamic scheduling during execution. Compared with standard PSO, our method not only adopt critical path to improve the quality of initial particles but also apply adaptive parameter setting strategies.



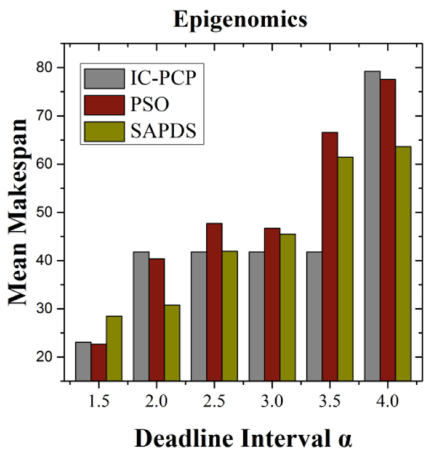
(a) Mean cost for Epigenomics



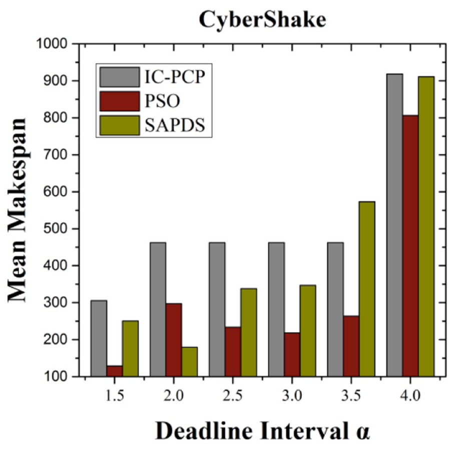
(b) Mean cost for CyberShake

**Fig. 3.** Comparison in mean cost under different deadlines for Epigenomics and CyberShake (number of initial particles = 100, number of iterations = 80).

Figure 4 presents the simulation result of the makespan (total execution time) of the three methods under different deadlines where the number of initial particles and the number of iterations are fixed. In Epigenomics, it can be found that our method mostly has less makespan in different deadlines compared with other two methods. While in CyberShake, the performance of our method in makespan under different deadlines is not stable. This is because the CyberShake workflow is more complex than Epigenomics workflow, and the tasks of CyberShake are dependent with each other more closely.



(a) Mean makespan for Epigenomics

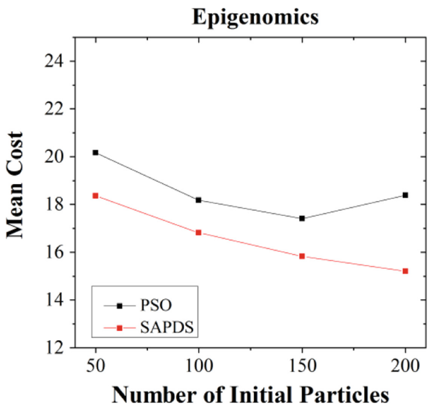


(b) Mean makespan for CyberShake

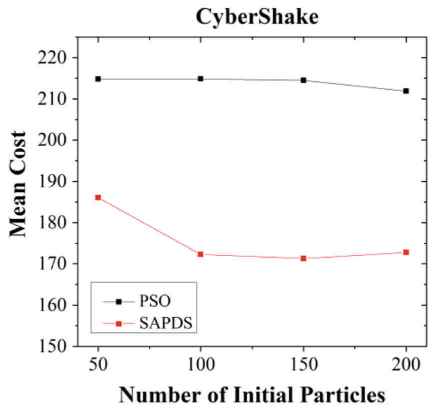
**Fig. 4.** Comparison in mean makespan under different deadlines for Epigenomics and CyberShake (number of initial particles = 100, number of iterations = 80).

From Fig. 3 and Fig. 4, according to the goal of our proposal, i.e., minimize the overall and satisfy the deadline, we can see that our method could obtain optimized execution cost with fine makespan under different deadlines.

Figure 5(a) and Fig. 5 (b) present the result of PSO and SAPDS in mean cost with the change of the initial particles, respectively. We can see that the execution costs of both of PSO and SAPDS decrease with the increase of initial particles at the beginning and then slowly get steady. Besides, with the change of the number of initial particles, our method constantly consumes less cost than PSO.



(a) Mean cost for Epigenomics



(b) Mean cost for CyberShake

**Fig. 5.** Comparison in mean cost with the increase of initial particles for Epigenomics and CyberShake (deadline interval = 2.0, number of iterations = 80).

Overall we can observe that our method can get optimized execution cost with fine makespan in most cases. That is because that our method adopts adaptive strategies and dynamic scheduling.

## 5 Related Work

In this section, some related work is discussed in the following. There have been some researches focusing on scheduling researching in edge clouds or heterogeneous clouds. Liu et al. [14] introduce a stochastic scheduling mechanism for mobile edge computing systems. Chen et al. [15] propose research multi-user multi-task scheduling problem in green mobile edge cloud system and utilize Lyaponuv optimization approach to decide the energy harvesting policy. The literature [16] considers the scheduling issue by incorporating both the edge cloud and the remote cloud. But it didn't consider the impact of dynamics (the mobility of mobile users) to scheduling.

On the other hand, there are also some researches that have considered the dynamics and uncertainties during the execution of tasks. In our previous work, we have discussed the influence of the uncertainties to scheduling in cloud environment and propose a failure-tolerance scheduling approach to handle the exceptions during the executing of application tasks [17]. The literature [18] and [19] also research uncertainty or failure-aware method for scheduling and task provisioning in both cloud and mobile cloud systems. Sahni et al. [20] provide a dynamic cost-aware heuristic-based scheduling method, which use the VM performance variability and instance acquisition delay to get a just-in-time schedule strategy. Juarez et al. [21] present an energy-aware dynamic scheduling system for parallel task-based applications and aim to minimize both energy consumption and the execution time.

## 6 Conclusion

In this paper, a self-adaptive PSO-based dynamic scheduling approach is proposed to deal with the scheduling problem during the execution of workflow. Firstly, a hierarchical heterogeneous cloud architecture is proposed. Then to deal with resource provisioning and scheduling problem in heterogeneous clouds, a self-adaptive PSO-based scheduling mechanism is presented to optimize the execution cost and satisfy the deadline, where self-adaptive strategies are adopted to accelerate the convergence speed and get optimized solutions. Moreover, a dynamic scheduling mechanism based on dynamic partial workflow model is proposed to deal with the mobility of mobile users during the execution of workflow, which aims to get an optimal solution with changes in edge cloud resources. Finally, experiments are conducted and demonstrate the efficiency of our dynamic scheduling method. In our future work, we will do further research in multi-objective dynamic problem in heterogeneous clouds and consider more dynamics and uncertainties during the execution of workflows.

**Acknowledgment.** This paper is supported by the National Science Youth Foundation of China under Grant No. 61702264, the Fundamental Research Funds for the Central Universities No. 30919011282, the Fundamental Research Funds for the Central Universities No. 30918014108.

## References

1. Rimal, B.P., Maier, M.: Workflow scheduling in multi-tenant cloud computing environments. IEEE Trans. Parallel Distrib. Syst. **28**(1), 290–304 (2016)
2. Qi, L., et al.: Finding all you need: web APIs recommendation in web of things through keywords search. IEEE Trans. Comput. Soc. Syst. (2019). https://doi.org/10.1109/tcss.2019.2906925
3. Xu, X., Liu, Q., Zhang, X., Zhang, J., Qi, L., Dou, W.: A blockchain-powered crowdsourcing method with privacy preservation in mobile environment. IEEE Trans. Comput. Soc. Syst. (2019). https://doi.org/10.1109/tcss.2019.2909137
4. Abbas, N., Zhang, Y., Taherkordi, A., Skeie, T.: Mobile edge computing: a survey. IEEE Internet Things J. **5**(1), 450–465 (2017)
5. Zhu, Z., Zhang, G., Li, M., Liu, X.: Evolutionary multi-objective workflow scheduling in cloud. IEEE Trans. Parallel Distrib. Syst. **27**(5), 1344–1357 (2015)
6. You, C., Huang, K., Chae, H., Kim, B.H.: Energy-efficient resource allocation for mobile-edge computation offloading. IEEE Trans. Wireless Commun. **16**(3), 1397–1411 (2016)
7. Qi, L., Chen, Y., Yuan, Y., Fu, S., Zhang, X., Xu, X.: A QoS-aware virtual machine scheduling method for energy conservation in cloud-based cyber-physical systems. World Wide Web **23**(2), 1275–1297 (2019). https://doi.org/10.1007/s11280-019-00684-y
8. Filip, I.D., Pop, F., Serbanescu, C., Choi, C.: Microservices scheduling model over heterogeneous cloud-edge environments as support for IoT applications. IEEE Internet Things J. **5**(4), 2672–2681 (2018)
9. Kennedy, J., Eberhart, R.: Particle swarm optimization (PSO). In: Proceedings of the IEEE International Conference on Neural Networks, Perth, Australia, pp. 1942–1948 (1995)
10. Kumar, N., Zeadally, S., Chilamkurti, N., Vinel, A.: Performance analysis of Bayesian coalition game-based energy-aware virtual machine migration in vehicular mobile cloud. IEEE Netw. **29**(2), 62–69 (2015)
11. Bharathi, S., Chervenak, A., Deelman, E., Mehta, G., Su, M.H., Vahi, K.: Characterization of scientific workflows. In: 2008 Third Workshop on Workflows in Support of Large-Scale Science, pp. 1–10 (2008)
12. Abrishami, S., Naghibzadeh, M., Epema, D.H.: Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds. Future Gener. Comput. Syst. **29**(1), 158–169 (2013)
13. Rodriguez, M.A., Buyya, R.: Deadline based resource provisioning and scheduling algorithm for scientific workflows on clouds. IEEE Trans. Cloud Comput. **2**(2), 222–235 (2014)
14. Liu, J., Mao, Y., Zhang, J., Letaief, K.B.: Delay-optimal computation task scheduling for mobile-edge computing systems. In: 2016 IEEE International Symposium on Information Theory (ISIT), pp. 1451–1455 (2016)
15. Chen, W., Wang, D., Li, K.: Multi-user multi-task computation offloading in green mobile edge cloud computing. IEEE Trans. Serv. Comput. **12**, 726–738 (2018)
16. Zhao, T., Zhou, S., Guo, X., Niu, Z.: Tasks scheduling and resource allocation in heterogeneous cloud for delay-bounded mobile edge computing. In: 2017 IEEE International Conference on Communications (ICC), pp. 1–7 (2017)
17. Meng, S., Li, Q., Wu, T., Huang, W., Zhang, J., Li, W.: A fault-tolerant dynamic scheduling method on hierarchical mobile edge cloud computing. Comput. Intell. (2019). https://doi.org/10.1111/coin.12219
18. Chen, H., Zhu, X., Liu, G., Pedrycz, W.: Uncertainty-aware online scheduling for real-time workflows in cloud service environment. IEEE Trans. Serv. Comput. (2018)
19. Deng, S., Huang, L., Taheri, J., Zomaya, A.Y.: Computation offloading for service workflow in mobile cloud computing. IEEE Trans. Parallel Distrib. Syst. **26**(12), 3317–3329 (2014)

20. Sahni, J., Vidyarthi, D.P.: A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment. IEEE Trans. Cloud Comput. **6**(1), 2–18 (2015)
21. Juarez, F., Ejarque, J., Badia, R.M.: Dynamic energy-aware scheduling for parallel task-based application in cloud computing. Future Gener. Comput. Syst. **78**, 257–271 (2018)