

W-Scheduler: whale optimization for task scheduling in cloud computing

Karnam Sreenu¹ · M. Sreelatha²

Received: 12 April 2017 / Revised: 20 June 2017 / Accepted: 14 July 2017 / Published online: 28 July 2017
© Springer Science+Business Media, LLC 2017

Abstract One of the important steps in cloud computing is the task scheduling. The task scheduling process needs to schedule the tasks to the virtual machines while reducing the makespan and the cost. Number of scheduling algorithms are proposed by various researchers for scheduling the tasks in cloud computing environments. This paper proposes the task scheduling algorithm called W-Scheduler based on the multi-objective model and the whale optimization algorithm (WOA). Initially, the multi-objective model calculates the fitness value by calculating the cost function of the central processing unit (CPU) and the memory. The fitness value is calculated by adding the makespan and the budget cost function. The proposed task scheduling algorithm with the whale optimization algorithm can optimally schedule the tasks to the virtual machines while maintaining the minimum makespan and cost. Finally, we analyze the performance of the proposed W-Scheduler with the existing methods, such as PBACO, SLPSO-SA, and SPSO-SA for the evaluation metrics makespan and cost. From the experimental results, we conclude that the proposed W-Scheduler can optimally schedule the tasks to the virtual machines while having the minimum makespan of 7 and minimum average cost of 5.8.

Keywords Cloud computing · Task scheduling · Multi-objective model · Whale optimization algorithm · Makespan

1 Introduction

Due to the availability of big data, the requirement of cloud computing has increased in several areas, such as business and the cloud computing is the on-demand process in the recent years. Cloud computing [1,2] allows the users to access the resources, such as storage, servers, and applications from the internet [3]. The service provider of the cloud is used to manage the services, and these services are accessed by the users over the internet [4]. The cloud offers several services to the users. The most significant services are Platform as a Service (PaaS) [5], Infrastructure as a Service (IaaS) [6], Expert as a Service (EaaS) [7], and Software as a Service (SaaS) [8,9]. The users of the cloud have different jobs, and these jobs are performed simultaneously by the resources available in the cloud. The performance of the cloud computing can be improved by allocating resources to the jobs in an optimized manner. One of the critical processes of the cloud computing [10,11] is how to schedule the tasks, and the task scheduling produces great impacts on the entire cloud by affecting the Quality of Service (QoS). The task scheduling process maintains the balance among the requirements of the users and the utilization of resources. Each task requires memory, computing time and response time in different scales and the cloud computing environment has the heterogeneous resources which are distributed in the cloud geographically. The task scheduling process is affected by the above features of the cloud environment [3].

The effective task scheduling process must reduce the makespan [12] of the application. Thus, there is a need for the algorithms for scheduling tasks in the cloud which optimally allocate the tasks to the resources at the same time minimize the makespan. In cloud computing, the task scheduling process is considered as the NP-complete problem [13] where, the time required for finding the solution

✉ Karnam Sreenu
karnamsreenu.27@gmail.com

¹ Department of Computer Science and Engineering, ANU College of Engineering, Acharya Nagarjuna University, Guntur, Andhra Pradesh, India

² Department of Computer Science and Engineering, RVR & JC College of Engineering, Guntur, Andhra Pradesh, India

changes by the size of the problem [14]. Task scheduling algorithms are classified into two groups namely heuristic and meta-heuristic algorithms. Now a day, meta-heuristic algorithms are popular for task scheduling, and they find the solutions which are near to the optimal solution. Heuristic algorithms find the solution by excluding some paths in the solution space [15]. There are three types of heuristic algorithm. They are list scheduling, duplication based scheduling, and clustering based scheduling [16]. The list scheduling is performed in two steps. In the first step, it assigns the priority values to the tasks and in the second step, the tasks are allocated to the processors depends on the priority values of the tasks.

In the duplication based algorithms, the identical copies of the tasks are generated which are used to decrease the application's makespan, and the duplicated copies of the tasks are allocated to the same processor [16] to reduce the cost needed for performing the computation. The clustering based algorithms groups the tasks into the cluster and the cluster of tasks are allocated to the processors. These algorithms had infinite processors and designed to work in a homogeneous system. In contrast, the meta-heuristic algorithms find the solution by utilizing the random choices [17]. The best example for the meta-heuristic algorithm is the genetic algorithm [8]. In the literature works [3,4,8,18,19], the time required for allocating the resources is increased when the number of task increases. Hence, the existing algorithms were not fitting for cloud centers with a large amount of data. The WOA [20] allows the task scheduling of the big data since it provides better performance in the unknown search space.

This paper proposes the W-Scheduler for scheduling tasks to the virtual machines in the cloud computing environment. The proposed task scheduling method is based on the multi-objective model and the whale optimization algorithm. The multi-objective model calculates the cost function of CPU and memory of all the virtual machines and then, calculates the budget cost function by adding cost function of both CPU and memory. Then, it calculates the fitness by adding the makespan and the budget cost function. Based on the fitness value, it allocates the tasks to the virtual machines. The whale optimization algorithm begins with the group of random solutions. Initially, it assumes that the current solution is the best solution and performs the searching process based on the current solution. This process is repeated until the best solution reaches.

The major contributions of this paper are:

- The primary contribution of this work is the design of the multi-objective model for calculating the fitness value.
- The secondary contribution of this work is the design of the proposed W-Scheduler based on the WOA algorithm. The proposed W-Scheduler optimally allocates the tasks

to the virtual machines while maintaining the minimum makespan and minimum cost.

The organization of the paper is described as follows: Sect. 2 presents the motivation of the proposed W-Scheduler for scheduling tasks to the virtual machines in the cloud environments. Section 3 describes the system model of the cloud environment. Section 4 presents our proposed task scheduling mechanism. The results and discussion are presented in Sects. 5, and 6 concludes the paper.

2 Motivation

In this section, the various works related to the task scheduling problem in cloud computing and the challenges associated with the task scheduling are discussed.

2.1 Review of related works

The review of the research papers in the field of task scheduling in cloud environments is discussed here. HE Hua et al. [3] have suggested an adaptive multi-objective task scheduling (AMTS) approach based on particle swarm optimization (PSO) algorithm. This method optimally allocates the resources to the tasks with less time and requires average energy. The disadvantage of this method is that it did not schedule the task dynamically. Bahman Keshanchi et al. have proposed a task scheduling approach based on the genetic algorithm in [8]. Along with the genetic algorithm, this method makes use of the heterogeneous earliest finish time (HEFT) searching. The disadvantage of this method is that it takes more time to detect solutions. Xue Lin et al. [4] have proposed the task scheduling algorithm based on the dynamic voltage and frequency scaling (DVFS) technique. This method performs task scheduling with less delay and energy, and the linear-time rescheduling algorithm does the movement of the task. Xiaolong Xu et al. [18] have proposed a task scheduling method in which the resources are allocated to the tasks based on probabilistic matching (PM) and Improved Simulated Annealing (ISA). This method provides optimal allocation of resources to the tasks.

Haitao Yuan et al. [19] have suggested a task scheduling method based on profit maximization algorithm (PMA). In this method, all the arrived tasks are processed in the public cloud and also in the private cloud. The difficulty in the profit maximization algorithm was removed by the simulated annealing particle swarm optimization algorithm (SAPSO). This method did not work on the real cloud environments. Yibin Li et al. [21] have suggested an Energy-aware Dynamic Task Scheduling (EDTS) algorithm based on the Dynamic Voltage Scaling (DVS) technique, and also they have proposed a Critical Path Assignment (CPA) algorithm for finding

the critical paths. The advantage of this method is that it had greater efficiency. The drawback of this method is that it was designed for only Android devices and needs improvements. Zhifeng Zhong et al. [22] have presented task scheduling algorithm based on Greedy Particle Swarm Optimization (GPSO). It had some advantages, such as better convergence rate, balanced workload. This method considered only the task size and the virtual machines' ability but did not consider the other factors, such as bandwidth. Chunling Cheng et al. [23] have suggested a task scheduling method based on the Vacation Queuing Theory for minimizing the need for energy. The disadvantage of this method is that the performance of the task scheduling was low.

Hongyan Cui et al. [24] presented the task scheduling in the cloud environment based on the Markov model. Various objects such as reliability, makespan, and flow time found for the scheduling is found as the optimization problem. They have proposed the Genetic Algorithm-based Chaotic Ant Swarm (GA-CAS) algorithm for scheduling the tasks. This model has an improved rate of convergence. Sanjaya K. Panda et al. [25] proposed the task scheduling for the cloud platforms by considering the factor task allocation. The proposed algorithm depends on the Min–Min and Max–Min algorithm to make it suitable for the multi-cloud environment. Factors such as transfer cost and time stamp in the task scheduling were not discussed in this work.

2.2 Challenges

One of the critical problems in cloud computing is the scheduling of tasks to the resources. The existing task scheduling algorithms did not solve the various crucial factors associated with the task scheduling problem. The time required for allocating the resources was increased when the number of task increases. Hence, the existing algorithms were not fitting for cloud centers with a large amount of data. Variation in tasks and time adjustment are the major challenges during the process of task scheduling. Most of the existing systems perform task scheduling by adjusting the tasks. They did not evaluate the time overhead. In some situations, the estimated workload of the system was smaller than the actual workloads of the system which leads to performance degradation of the task scheduling algorithms. The sudden oscillation in the estimation of workload also influences the stability of the system [18].

3 System model

This section describes the system model of the proposed task scheduling mechanism in the cloud environment. Figure 1 represents the system model of task scheduling mechanism in the cloud computing. The task manager collects the tasks

from the various users. The users submit the task requests to the task manager. The task manager manages the database to store every user request. The task manager organizes the user tasks and provides the status of the task to the user. The task manager contains the information about the status of the virtual machine. The task manager provides these task requests to the task scheduler. Task scheduler is a device which provides the priority to the incoming tasks. The task scheduler analyzes the memory requirement, cost, deadline, and the required budget of the tasks. The cloud environment contains many physical machines. The virtual machine present in the physical machine can process many tasks. The task scheduler allocates tasks to the virtual machines presented in the cloud environment.

Assume that, the cloud which consists of 100 physical machines and each physical machine consists of 10 virtual machines. This can be represented as,

$$\text{Cloud, } C = \{P_1, P_2 \dots, P_{100}\} \quad (1)$$

where, C represents the cloud and $\{P_1, P_2 \dots, P_{100}\}$ represents the physical machines presented in the cloud. The following equation can represent the physical machine P_1 .

$$P_1 = \{V_1, V_2 \dots, V_j \dots, V_{10}\} \quad (2)$$

where, $\{V_1, V_2 \dots, V_j \dots, V_{10}\}$ represents the virtual machines presented in the physical machine P_1 . Each virtual machine has the central processing unit (CPU) and the memory. Here, the capacity of the CPU is 1860 MIPS (Millions Instructions per second) or 2660 MIPS. That is, the CPU can be able to perform 1860 or 2660 millions of instructions in one second. The size of the memory is 4 GB. The total number of tasks is 100, and each task has the different user cost of CPU, the user cost of memory, deadline, and budget cost.

$$\text{Task} = \{T_1, T_2 \dots, T_i \dots, T_{100}\} \quad (3)$$

where, T_1, T_2 are the first and second tasks respectively. T_i represents the i th task and T_{100} represents the 100 th task. Figure 2 explains the operation of the task scheduler. The task manager provides the tasks to the scheduler. Each task provided by the user contains the following parameters, memory requirement, cost, deadline, and the required budget. Based on this parameter the task scheduler prioritizes the task and schedules it accordingly. In Fig. 2, the memory requirement, cost, deadline, and the required budget of the task T_i are analyzed by the scheduler and provided accordingly to the virtual machine for processing.

In Fig. 2, the term C_i^U represents the cost of CPU of the task T_i which was defined by the users, M_i^U is the cost of the memory of task T_i defined by the users, D_i^U represents the deadline for the task T_i , and B_i^U is the budget cost of T_i .

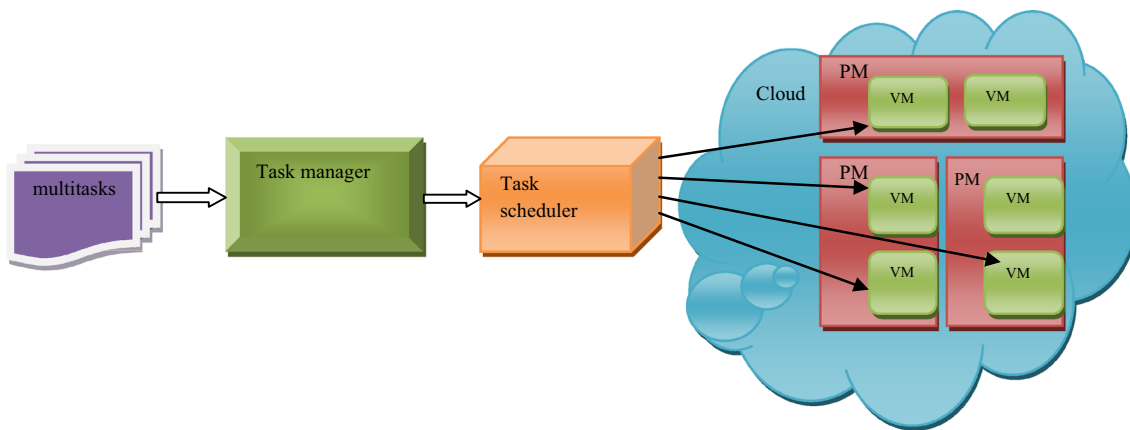
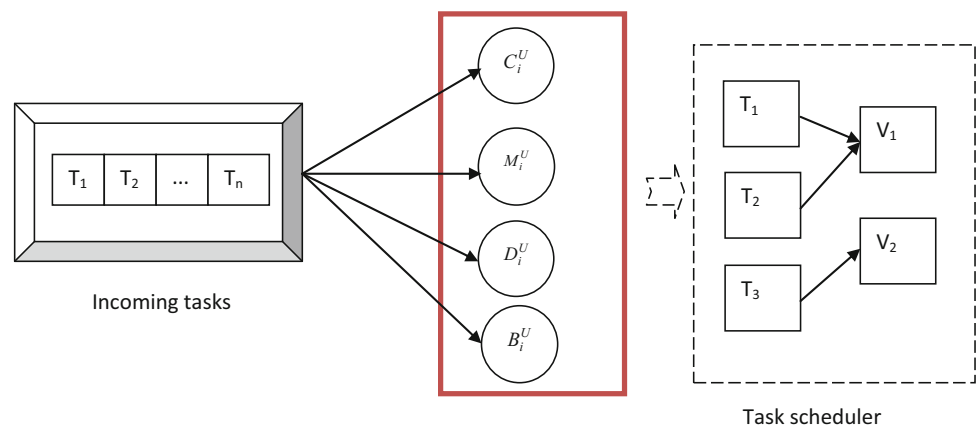


Fig. 1 System model of task scheduling mechanism in the cloud computing

Fig. 2 Scheduling of the task T_i



4 Proposed task scheduling method based on whale optimization algorithm

This section describes the proposed task scheduling method for scheduling tasks to the virtual machines in the cloud computing environments. The proposed task scheduling method is based on the multi-objective model [26] and the whale optimization algorithm [20]. Figure 3 shows the block diagram of the proposed W-Scheduler for scheduling tasks to the virtual machines in the cloud environments. The multi-objective model calculates the cost function of CPU and memory of all the virtual machines and then, calculates the budget cost function by adding cost function of both CPU and memory. Then, it calculates the fitness by adding the makespan and the budget cost function. Based on the fitness value, it allocates the tasks to the virtual machines. The whale optimization algorithm begins with the group of random solutions. Initially, it assumes that the current solution is the best solution and performs the searching process based on the current solution. This process is repeated until the best solution reaches. The objective of the task scheduling is to optimally sched-

ule the tasks to the resources while obtaining the minimum makespan and minimum budget cost. Makespan represents the total time needed for executing all the tasks.

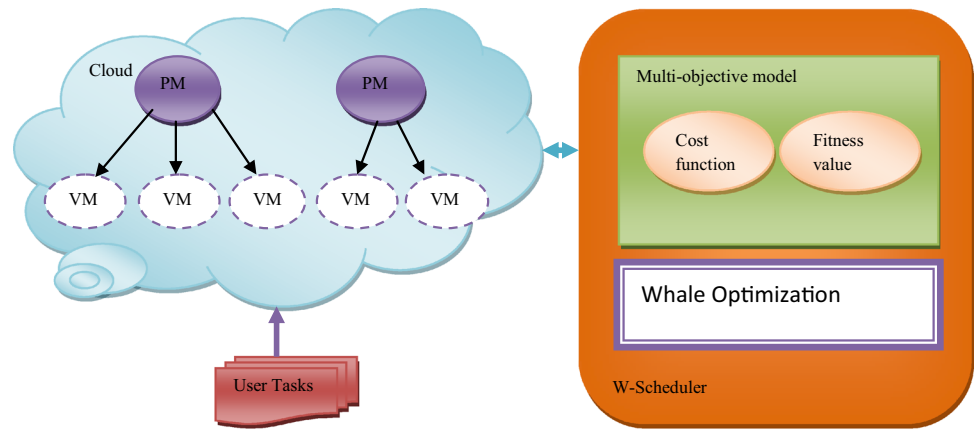
4.1 Multi-objective model for task scheduling

The multi-objective model [26] for scheduling the tasks to the virtual machines is described here. At first, the multi-objective model calculates the cost function of CPU and memory of all the virtual machines presented in the cloud environment and calculates the budget cost. The budget cost function is calculated by adding the cost function of CPU and memory. Then, the fitness is calculated by combining the budget cost function and the makespan of the scheduling process.

4.1.1 Fitness calculation

The fitness is calculated to find the quality of the optimal solutions, and the solutions must have minimum makespan and minimum cost function. At first, the cost function of

Fig. 3 Block diagram of the proposed W-Scheduler for task scheduling



CPU and memory is calculated. The following equations can calculate the cost functions of CPU and memory of the virtual machine V_j .

$$C(x) = \sum_{j=1}^{|VM|} C^{\text{cost}}(j) \tag{4}$$

where, $C^{\text{cost}}(j)$ is the cost of the CPU of the virtual machine V_j , $|VM|$ represents the total number of virtual machines. Then, the $C^{\text{cost}}(j)$ is calculated as follows,

$$C^{\text{cost}}(j) = C_{\text{base}} \times C_j \times t_{ij} + C_{\text{Trans}} \tag{5}$$

where, C_{base} is the base cost, C_j represents the CPU of the virtual machine V_j , and t_{ij} represents the time in which the task T_i is processed in the resource R_j . C_{Trans} is the transmission cost of the CPU. Here, C_{base} and C_{Trans} are constant.

$$C_{\text{base}} = 0.17/\text{hr} \tag{6}$$

$$C_{\text{Trans}} = 0.005 \tag{7}$$

The cost function of the memory is calculated by,

$$M(x) = \sum_{j=1}^{|VM|} M^{\text{cost}}(j) \tag{8}$$

where, $M^{\text{cost}}(j)$ represents the cost of the memory of the virtual machine V_j , $|VM|$ represents the total number of virtual machines. Then, $M^{\text{cost}}(j)$ is calculated as follows,

$$M^{\text{cost}}(j) = M_{\text{base}} \times M_j \times t_{ij} + M_{\text{Trans}} \tag{9}$$

where, M_{base} represents the base cost of the memory, M_j represents the memory of the virtual machine V_j , and t_{ij} represents the time in which the task T_i is processed in the

resource R_j . M_{Trans} is the transmission cost of the memory. The value of M_{base} and M_{Trans} are constant.

$$M_{\text{base}} = 0.05\text{GB}/\text{hr} \tag{10}$$

$$M_{\text{Trans}} = 0.5 \tag{11}$$

Then, the budget cost function of the user can be calculated by adding the cost function of CPU and memory of the virtual machine.

$$B(x) = C(x) + M(x) \tag{12}$$

where, $B(x)$ is the budget cost function of the user, $C(x)$ is the cost function of the CPU, and $M(x)$ is the cost function of the memory. Then, the fitness is calculated by the following equation,

$$H(x) = F(x) + B(x) \tag{13}$$

where, $F(x)$ represents the makespan that is the performance function and it should be less than or equal to the deadline of the task. Makespan can be represented by the equation (14).

$$F(x) \leq \sum_{i=1}^{|T|} D_i \tag{14}$$

where, D_i represents the deadline for the task T_i . In equation (13), $B(x)$ represents the budget cost function of the tasks that includes both the CPU cost and memory cost and it should be less than or equal to the user's budget cost. The budget cost function can be represented as follows,

$$B(x) \leq \sum_{i=1}^{|T|} B_i \tag{15}$$

where, B_i represents the user's budget cost of the task T_i .

4.2 Solution encoding

The aim of the task scheduling is to allocate all the 100 tasks to the ten virtual machines while considering the deadline of the task and the budget cost of the tasks. Assume that, an array of 100 tasks and each task of an array is initialized to the values in between one to ten. If the first element of an array is set to one, then the task one is allocated to the virtual machine V_1 . If the first element of the array is set to eight, then the task one is allocated to the virtual machine V_8 . Similarly, all the tasks are allocated to the virtual machines $V_1 - V_{10}$.

T_1	T_2	...	T_i	...	T_{100}
V_1	V_2		V_8		V_2

4.3 Whale optimization algorithm

The whale optimization algorithm [20] is described for optimally allocating the tasks to the virtual machines. The whale optimization algorithm begins with the group of random solutions. Initially, it assumes that the current solution is the best solution and continues the process based on the current solution. This process is repeated until the best solution reaches.

Step 1: Initialization In this step, the population of the search agent is initialized, and the best search agent is selected randomly. The initial population is defined as, R_j ($j = 1, 2, \dots, k$) and the best search agent is represented as R^* .

Step 2: Fitness calculation The fitness is calculated by the Eq. (13).

Step 3: Encircling prey In this step, the humpback whales realize the position of the prey and surrounding them. Then, it assumes that current solution is the best prey and the search agents update their position according to the position of the current best agent. This can be represented as follows,

$$\vec{P} = \left| \vec{Q} \bullet \vec{R}^*(x) - \vec{R}(x) \right| \tag{16}$$

where, x represents the current iteration, \vec{R} represents the position vector and \vec{R}^* represents the position vector of the best solution. \vec{Q} represents the coefficient vector. The Eq. (16) represents the position of the current best search agent. Then, the new position of the search is calculated by the Eq. (17)

$$\vec{R}(x + 1) = \vec{R}^*(x) - \vec{M} \bullet \vec{P} \tag{17}$$

where, \vec{M} represents the coefficient vector. $||$ indicates the absolute value and \bullet indicates the element by element multiplication. Then, \vec{M} and \vec{Q} are calculated by the Eqs. (18) and (19).

$$\vec{M} = 2 \vec{m} \bullet \vec{n} - \vec{m} \tag{18}$$

$$\vec{Q} = 2 \bullet \vec{n} \tag{19}$$

where, the value of \vec{m} is decreased from 2 to 0 and \vec{n} represents the random vector in $[0, 1]$. The value of \vec{M} and \vec{Q} are modified to visit the surrounding places of the best search agent.

Step 4: Exploitation phase This phase consists of two steps namely, (1) shrinking encircling mechanism, (2) spiral updating position. In shrinking encircling mechanism, the value of \vec{M} is set to $[-1, 1]$ and the new position of the agent is represented by the agent's initial position and the current best agent's position. In spiral updating position, the spiral can be updated by the following equation,

$$\vec{R}(x + 1) = \vec{P}' \bullet h^{st} \bullet \cos(2\pi t) + \vec{R}^*(x) \tag{20}$$

where, s represents the constant and t is the value in $[-1, 1]$. \bullet represents the element by element multiplication. \vec{P}' is calculated as follows,

$$\vec{P}' = \left| \vec{R}^*(x) - \vec{R}(x) \right| \tag{21}$$

where, \vec{R} represents the position vector and \vec{R}^* represents the position vector of the best solution. Then, the position of the search agent can be updated between either the encircling mechanism or the spiral position.

$$\vec{R}(x + 1) = \begin{cases} \vec{R}^*(x) - \vec{M} \bullet \vec{P} & ; \text{if } p < 0.5 \\ \vec{P}' h^{st} \bullet \cos(2\pi t) + \vec{R}^*(x) & ; \text{if } p \geq 0.5 \end{cases} \tag{22}$$

where, p is the random number in between $[-1, 1]$.

Step 5: Exploration phase In this step, the position of the search agent is updated by the randomly chose agent. This can be represented as,

$$\vec{P} = \left| \vec{Q} \bullet \vec{R}_{rand} - \vec{R} \right| \tag{23}$$

$$\vec{R}(x + 1) = \vec{R}_{rand} - \vec{M} \bullet \vec{P} \tag{24}$$

where, \vec{R}_{rand} represents the random position vector.

Fig. 4 Algorithm of proposed W-Scheduler

W-Scheduler Algorithm	
1	Input: Tasks T , Virtual machines V
2	Output: The tasks are allocated to the virtual machines
3	Parameters: R^* , m, M, Q, t , and p
4	Begin
5	Initializes the population
6	Fitness value obtained from the sub method
7	Initialize the current best agent R^*
8	Update m, M, Q, t , and p
9	If ($p < 0.5$)
10	If ($ M < 1$)
11	Update the position of the search agent using equation 17
12	Else if ($ M \geq 1$)
13	Update the position of the search agent by the randomly chosen agent using equation 24
14	End if
15	End if
16	If ($p \geq 0.5$)
17	Update the position of the search agent using equation 20
18	End if
19	If (any search agent go outside of the search region)
20	Update R^*
21	Set $x = x + 1$
22	End if
23	End
24	Sub method:
25	Begin
26	Input: Tasks T , Virtual machines V
27	Output: Fitness value
28	For(all the virtual machines)
29	Find cost function of CPU by equations 4 to 7
30	Find cost function of Memory by equations 8 to 11
31	Find the budget cost function of the user by equation 12
32	Find the fitness value by equation 13
33	End for
34	End

Step 6: *Termination* If the search agents go outside of the search region, then R^* is updated and set $x = x + 1$. This process is repeated until the best solution reaches.

Figure 4 shows the algorithm for the proposed W-Scheduler for task scheduling in cloud computing. The inputs are the tasks and the virtual machines. The aim of the proposed task scheduler is to allocate the tasks to the virtual machines optimally. This scheduling is based on the Whale Optimization Algorithm. At first, the population of the search

agents is initialized. Then, the fitness value is calculated by the multi-objective model. Then, the current best agent is initialized, and the other search agents update their position towards the position of the current best search agent. If the probability value is greater than or equal to 0.5, then the position of the search agent is updated by the equation number 20. If the probability value is less than 0.5, then the position of the search agents are updated by the Eq. (17). This process is continued until the optimal solution reaches.

5 Results and discussion

This section presents the experimental results of the proposed W-Scheduler for scheduling tasks to the virtual machines in the cloud environments and the comparative discussion of the proposed method with the existing methods, such as PBACO [26], SLPSO-SA [27], and SPSO-SA [27].

5.1 Experimental setup

The experimentation of the proposed W-Scheduler is performed in a personal computer with Intel Core i3 processor and 2GB memory using Windows 8 operating system. The proposed method is implemented using Java with cloudsim and the performance is evaluated using the makespan and cost.

5.2 Evaluation metrics

The evaluation metrics considered for analyzing the performance of the proposed W-Scheduler algorithm are makespan and cost.

Makespan Makespan represents the total time needed for executing all the tasks. The makespan of the scheduler must be minimum.

Cost The cost represents the total cost needed for scheduling the tasks to the virtual machines.

5.3 Experimental results

The experimental results of the proposed W-Scheduler are discussed here. The proposed method is evaluated based on the evaluation metrics makespan and average cost.

5.3.1 Makespan

Figure 5 represents the makespan of the proposed method on iteration 25 and 50. Figure 5a represents the makespan of the proposed W-Scheduler on 25 th iteration for the population sizes of 10, 15, and 20. When the numbers of tasks are 100, 200, and 400, the makespan of the proposed W-Scheduler is nine for the population sizes of 10, 15, and 20. When the number of tasks is 300, the makespan of the proposed method is nine for the population sizes of 10 and 20, and the makespan of the proposed method is eight if the population size is 15. Figure 5b represents the makespan of the proposed W-Scheduler on 50 th iteration with a number of physical machines 20 for the population sizes of 10, 15, and 20. When the number of jobs is 100, the makespan of the proposed W-Scheduler is eight for the population sizes of $R=10$ and $R=20$, and the makespan is nine if the population size is $R=15$. When the number of jobs is 200, the makespan of the proposed W-Scheduler is eight for the population sizes

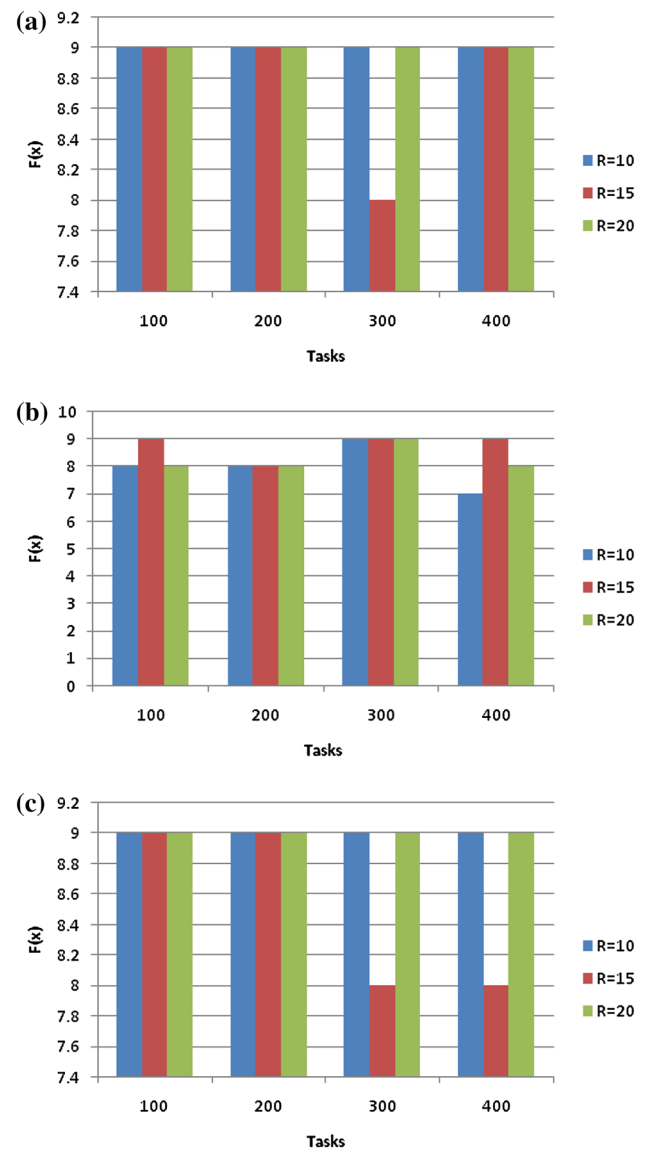


Fig. 5 Illustration of the makespan of the proposed W-Scheduler on iteration 25 and 50. **a** Makespan of the proposed W-Scheduler for $R=10$, $R=15$, and $R=20$ on iteration 25 ($PM=40$), **b** Makespan of the proposed W-Scheduler for $R=10$, $R=15$, and $R=20$ on iteration 50 ($PM=20$) and **c** Makespan of the proposed W-Scheduler for $R=10$, $R=15$, and $R=20$ on iteration 50 ($PM=30$)

of $R=10$, $R=15$, and $R=20$. When the number of jobs is 300, the makespan of the proposed W-Scheduler is nine for the population sizes of $R=10$, $R=15$, and $R=20$. When the number of jobs is 400, the makespan of the proposed W-Scheduler is seven, nine, and eight if the population size is $R=10$, $R=15$, and $R=20$ respectively. Figure 5c represents the makespan of the proposed W-Scheduler on 50 th iteration with a number of physical machines 30 for the population sizes of 10, 15, and 20. When the numbers of jobs are 100 and 200, the makespan of the proposed W-Scheduler is nine for all the population sizes $R=10$, $R=15$, and $R=20$.

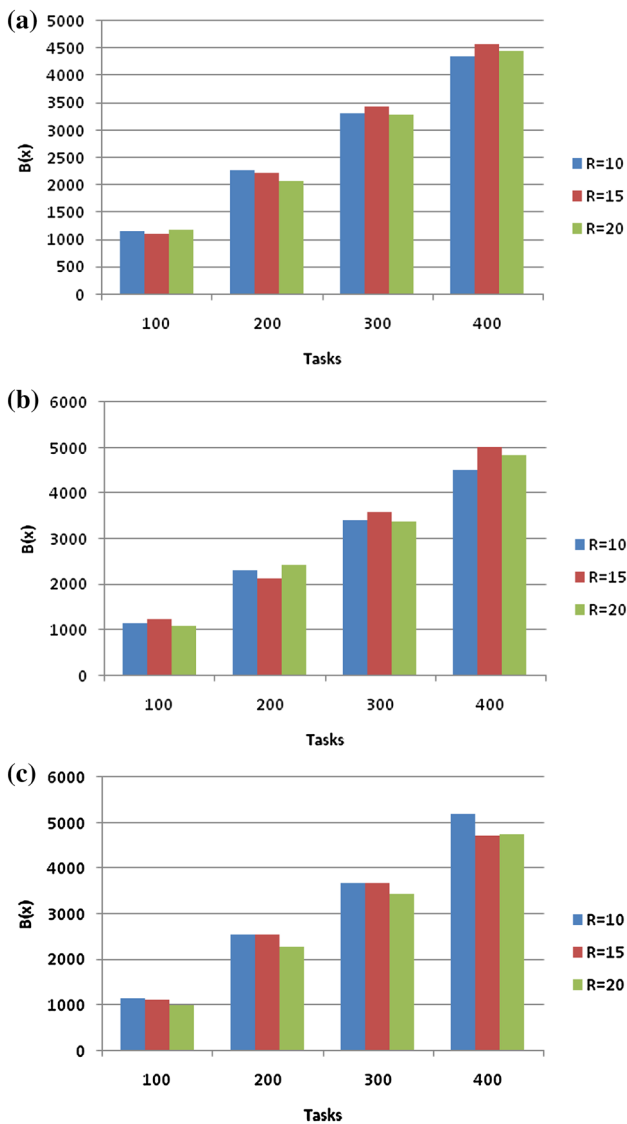


Fig. 6 Illustration of the budget cost of the proposed W-Scheduler on iteration 25 and 50. **a** Budget cost of the proposed W-Scheduler for R = 10, R = 15, and R = 20 on iteration 25 (PM = 40), **b** Budget cost of the proposed W-Scheduler for R = 10, R = 15, and R = 20 on iteration 25 (PM = 20), **c** Budget cost of the proposed W-Scheduler for R = 10, R = 15, and R = 20 on iteration 25 (PM = 30)

When the numbers of jobs are 300 and 400, the makespan of the proposed Scheduler is nine for the population sizes of R = 10 and R = 20 and eight for the population size of R = 15. From the Fig. 5a, b, c, the proposed method has a minimum makespan.

5.3.2 Budget cost

Figure 6 shows the illustration of the budget cost of the proposed W-Scheduler on iterations 25 and 50 for the population sizes of R = 10, R = 15, and R = 20. Figure 6a shows the budget cost of the proposed W-Scheduler on iteration 25. When

the number of jobs is 100, the budget cost of the proposed W-Scheduler is 1143.776, 1101.894, and 1160.496 for the population sizes of R = 10, R = 15, and R = 20 respectively. If the population size is R = 10, the budget cost of the proposed W-Scheduler is 2255.188, 3315.382, and 4360.176 when the numbers of jobs are 200, 300, and 400 respectively. If the population size is R = 15, the budget cost of the proposed W-Scheduler is 2217.96, 3428.542, and 4578.822 when the numbers of jobs are 200, 300, and 400 respectively. When the numbers of jobs are 200, 300, and 400, the budget cost of the proposed W-Scheduler is 2059.852, 3284.734, and 4449.666 for the population size of R = 20. Figure 6b shows the budget cost of the proposed W-Scheduler on iteration 50 and number of physical machines is 20. For the population size of R = 10, the budget cost of the proposed W-Scheduler is 1148.05, 2309.378, 3411.798, and 4504.28 when the numbers of jobs are 100, 200, 300, and 400 respectively. If the population size is R = 15, the budget cost of the proposed W-Scheduler is 1224.028, 2130.96, 3582.504, and 5013.136 when the numbers of jobs are 100, 200, 300, and 400 respectively. When the numbers of jobs are 100, 200, 300, and 400, the budget cost of the proposed W-Scheduler is 1082.382, 2411.464, 3363.66, and 4832.49 for the population size of R = 20. Figure 6c shows the budget cost of the proposed W-Scheduler on iteration 50 and number of physical machines is 30. When the number of jobs is 100, the budget cost of the proposed W-Scheduler is 1135.686, 1093.692, and 992.054 for the population sizes of R = 10, R = 15, and R = 20. For the population sizes of R = 10, R = 15, and R = 20, the budget cost of the proposed W-Scheduler is 2541.132, 2543.642, and 2267.974 when the number of jobs is 200. When the number of jobs is 300, the budget cost of the proposed W-Scheduler is 3658.264, 3665.288, and 3429.632 for the population sizes of R = 10, R = 15, and R = 20. When the number of jobs is 400, the budget cost of the proposed W-Scheduler is 5203.61, 4702.22, and 4739.06 for the population sizes of R = 10, R = 15, and R = 20.

5.3.3 Average cost

Figure 7 shows the illustration of the average cost of the proposed W-Scheduler on iterations 25 and 50 for the population sizes of R = 10, R = 15, and R = 20. Figure 7a shows the average cost of the proposed W-Scheduler on iteration 25. When the number of jobs is 100, the average cost of the proposed W-Scheduler is 6.28, 6.17, and 6.88 for the population sizes of R = 10, R = 15, and R = 20 respectively. If the population size is R = 10, the average cost of the proposed W-Scheduler is 6.47, 6.203333333, and 6.67 when the numbers of jobs are 200, 300, and 400 respectively. If the population size is R = 15, the average cost of the proposed W-Scheduler is 5.8, 6.003333333, and 6.3525 when the numbers of jobs are 200, 300, and 400 respectively. When the numbers of jobs are 200,

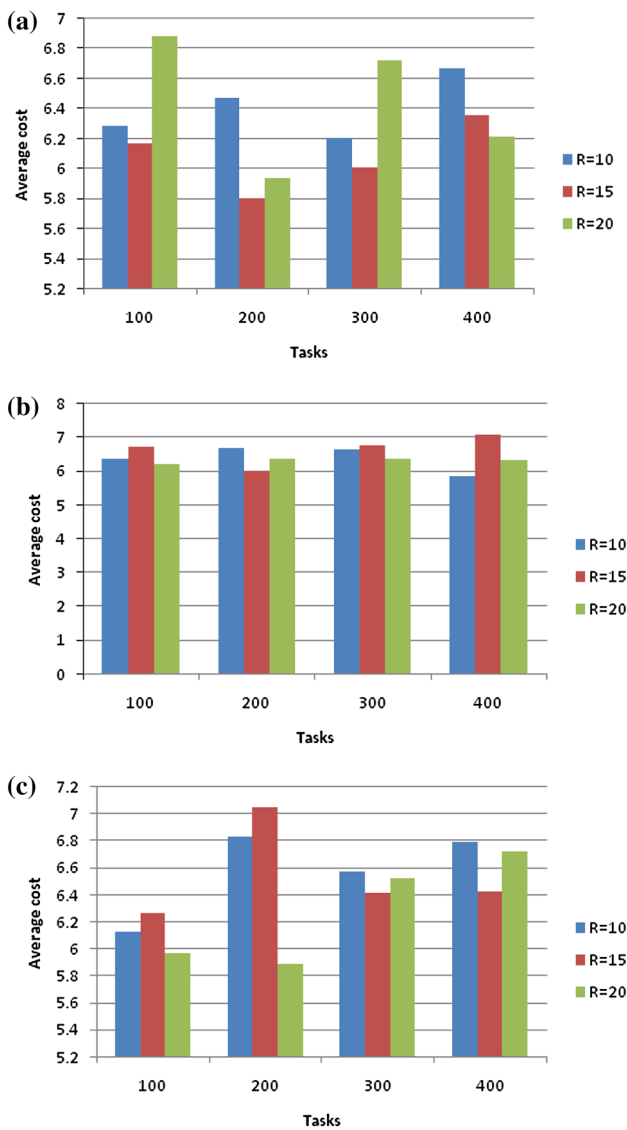


Fig. 7 Illustration of the average cost of the proposed W-Scheduler on iteration 25 and 50. **a** Average cost of the proposed W-Scheduler for R = 10, R = 15, and R = 20 on Iteration 25 (PM = 40), **b** Average cost of the proposed W-Scheduler for R = 10, R = 15, and R = 20 on Iteration 25 (PM = 20) and **c** Average cost of the proposed W-Scheduler for R = 10, R = 15, and R = 20 on Iteration 25 (PM = 30)

300, and 400, the average cost of the proposed W-Scheduler is 5.93, 6.723333333, and 6.2075 for the population size of R = 20. Figure 7b shows the average cost of the proposed W-Scheduler on iteration 50 and number of physical machines is 20. When the number of jobs is 100, the average cost of the proposed W-Scheduler is 6.35, 6.74, and 6.21 for the popula-

tion sizes of R = 10, R = 15, and R = 20. For the population sizes of R = 10, R = 15, and R = 20, the average cost of the proposed W-Scheduler is 6.695, 6, and 6.36 when the number of jobs is 200. When the number of jobs is 300, the average cost of the proposed W-Scheduler is 6.63, 6.773333333, and 6.366666667 for the population sizes of R = 10, R = 15, and R = 20. When the number of jobs is 400, the average cost of the proposed W-Scheduler is 5.85, 7.07, and 6.3375 for the population sizes of R = 10, R = 15, and R = 20. Figure 7c shows the average cost of the proposed W-Scheduler on iteration 50 and number of physical machines is 30. For the population size of R = 10, the average cost of the proposed W-Scheduler is 6.13, 6.83, 6.573333333, and 6.7875 when the numbers of jobs are 100, 200, 300, and 400 respectively. If the population size is R = 15, the average cost of the proposed W-Scheduler is 6.26, 7.055, 6.413333333, and 6.425 when the numbers of jobs are 100, 200, 300, and 400 respectively. When the numbers of jobs are 100, 200, 300, and 400, the average cost of the proposed W-Scheduler is 5.97, 5.885, 6.52, and 6.725 for the population size of R = 20.

5.4 Comparative discussion

Table 1 shows the comparative discussion of the proposed W-Scheduler with the existing methods, such as PBACO [26], (SLPSO-based scheduling approach) SLPSO-SA [27], and (Standard PSO-based scheduling approach) SPSO-SA [27]. From the table, the proposed W-Scheduler has the minimum makespan of 7. The makespan of the PBACO is 15, and the makespan of the SLPSO-SA and SPSO-SA are 30, 15 respectively. From the table, the proposed W-Scheduler has the minimum makespan. The average cost of the PBACO, SLPSO-SA, and SPSO-SA are 16, 14, and 13 respectively while the average cost needed for our proposed W-Scheduler is 5.8 which is smaller than the other existing methods. From the table, we can conclude that the proposed W-Scheduler can optimally schedule the tasks to the virtual machines while requiring the minimum makespan and minimum average cost.

6 Conclusion

This paper presents the task scheduling algorithm for scheduling the tasks to the virtual machines in the cloud computing environments based on the multi-objective model

Table 1 Comparative discussion of the proposed W-Scheduler with the existing methods, such as PBACO, SLPSO-SA, and SPSO-SA

	W-Scheduler	PBACO [26]	SLPSO-SA [27]	SPSO-SA [27]
Makespan	7	15	30	15
Average cost	5.8	16	14	13

Bold values indicate the better performance

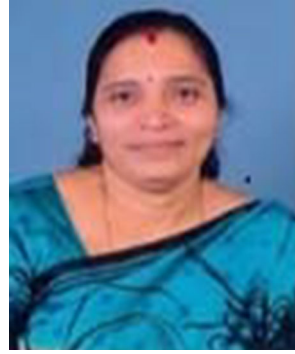
and the whale optimization algorithm. Initially, the multi-objective model calculates the fitness value by calculating the cost function of the CPU and the memory. Then, the budget cost function is calculated by adding the cost function of both the CPU and the memory. Finally, the fitness value is calculated by adding the makespan and the budget cost function. Then, the whale optimization algorithm is presented for optimally scheduling the tasks to the virtual machines. The whale optimization algorithm assumes that the current solution is the best solution and finds the optimal solution based on the best search agent. The performance analysis of the proposed W-Scheduler is performed with the existing methods, such as PBACO, SLPSO-SA, and SPSO-SA for the evaluation metrics makespan and cost. From the experimental results, we conclude that the proposed W-Scheduler can optimally schedule the tasks to the virtual machines while consuming minimum makespan of 7 and minimum average cost of 5.8.

References

- Mell, P., Grace, T.: The NIST definition of cloud computing. *Natl. Inst. Stand. Technol.* **53**(6), 50 (2009)
- Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, Randy, Konwinski, Andy, Lee, Gunho, Patterson, David, Rabkin, Ariel, Stoica, Ion, Zaharia, Matei: A view of cloud computing. *Commun. ACM* **53**(4), 50–58 (2010)
- Hua, H.E., Guangquan, X.U., Shanchen, P.A.N.G., Zenghua, Z.H.A.O.: AMTS: adaptive multi-objective task scheduling strategy in cloud computing. *China Commun.* **13**(4), 162–171 (2016)
- Lin, X., Wang, Y., Xie, Q., Pedram, M.: Task scheduling with dynamic voltage and frequency scaling for energy minimization in the mobile cloud computing environment. *IEEE Trans. Serv. Comput.* **8**(2), 175–186 (2015)
- Navimipour, N.J., Rahmani, A.M., Navin, A.H., Hosseinzadeh, M.: Expert cloud: a cloud-based framework to share the knowledge and skills of human resources. *Comput. Hum. Behav.* **46**, 57–74 (2015)
- Malawski, M., Juve, G., Deelman, E., Nabrzyski, J.: Algorithms for cost- and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. *Future Gener. Comput. Syst.* **48**, 1–18 (2015)
- Navimipour, N.J.: A formal approach for the specification and verification of a trustworthy human resource discovery mechanism in the expert cloud. *Expert Syst. Appl.* **42**(15–16), 6112–6131 (2015)
- Keshanchi, B., Souri, A., Navimipour, N.J.: An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. *J. Syst. Softw.* **124**, 1–21 (2017)
- Alkhanak, E.N., Lee, S.P., Khan, S.U.R.: Cost-aware challenges for workflow scheduling approaches in cloud computing environments: taxonomy and opportunities. *Future Gener. Comput. Syst.* **50**, 3–21 (2015)
- Rimal, B.P., Jukan, A., Katsaros, D., Goeleven, Y.: Architectural requirements for cloud computing systems: an enterprise cloud approach. *J. Grid Comput.* **9**(1), 3–26 (2011)
- Rimal, B.P., Choi, E. and Lumb, I.: A taxonomy and survey of cloud computing systems. In: *Proceedings of the Fifth International Joint Conference on IEEE*, pp. 44–51 (2009)
- Navimipour, N.J., Rahmani, A.M., Hosseinzadehet, M.: Expert grid: new type of grid to manage the human resources and study the effectiveness of its task scheduler. *Arab. J. Sci. Eng.* **39**(8), 6175–6188 (2014)
- Ullman, J.D.: NP-complete scheduling problems. *J. Comput. Syst. Sci.* **10**(3), 384–393 (1975)
- Xua, Y., Li, K., He, L., Truong, T.K.: A DAG scheduling scheme on heterogeneous computing systems using double molecular structure-based chemical reaction optimization. *J. Parallel Distrib. Comput.* **73**(9), 1306–1322 (2013)
- Yuming, X., Li, K., Jingtong, H., Li, K.: A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. *Inf. Sci.* **270**, 255–287 (2014)
- Khan, M.A.: Scheduling for heterogeneous systems using constrained critical paths. *J. Parallel Comput.* **38**(4–5), 175–193 (2012)
- Gupta, S., Agarwal, G., and Kumar, V.: Task scheduling in multiprocessor system using genetic algorithm. In: *Proceedings of Second International Conference on Machine Learning and Computing (ICMLC)* (2010)
- Xiaolong, X., Cao, L., Wang, X.: Resource pre-allocation algorithms for low-energy task scheduling of cloud computing. *J. Syst. Eng. Electron.* **27**(2), 457–469 (2016)
- Yuan, H., Bi, J., Tan, W., Li, B.H.: Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds. *IEEE Trans. Autom. Sci. Eng.* **14**(1), 337–348 (2017)
- Mirjalili, S., Lewis, A.: The whale optimization algorithm. *Adv. Eng. Softw.* **95**, 51–67 (2016)
- Li, Y., Chen, M., Dai, W., Qiu, M.: Energy optimization with dynamic task scheduling mobile cloud computing. *IEEE Syst. J.* **11**(1), 96–105 (2017)
- Zhong, Z., Chen, K., Zhai, X., Zhou, S.: Virtual machine-based task scheduling algorithm in a cloud computing environment. *Tsinghua Sci. Technol.* **21**(6), 660–667 (2016)
- Cheng, C., Li, J., Wang, Y.: An energy-saving task scheduling strategy based on vacation queuing theory in cloud computing. *Tsinghua Sci. Technol.* **20**(1), 28–39 (2015)
- Cui, Y.L., Liu, X., Ansari, N., Liu, Y.: Cloud service reliability modeling and optimal task scheduling Hongyan. *IET Commun.* **11**(2), 161–167 (2017)
- Panda, S.K., Gupta, I., and Jana, P.K.: Task scheduling algorithms for multi-cloud systems: allocation-aware approach. *Inf. Syst. Front.* pp. 1–19 (2017)
- Zuo, L., Shu, L., Dong, S., Zhu, C., Hara, Takahiro: A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *Big Data Serv. Comput. Intell. Ind. Syst.* **3**, 2687–2699 (2015)
- Zuo, X., Zhang, G., Tan, W.: Self-adaptive learning PSO-based dead-line constrained task scheduling for hybrid IaaS cloud. *IEEE Trans. Autom. Sci. Eng.* **11**(2), 564–573 (2014)



Karnam Sreenu is a Research Scholar in Computer Science and Engineering at ANU College of Engineering, Acharya Nagarjuna University, Guntur, India. He is currently working as Assistant Professor in Department of Information Technology, Sreenidhi Institute of Science and Technology, Hyderabad, India. He received B.Tech. and M.Tech. from Jawaharlal Nehru Technological University, Hyderabad. His research interest is Cloud Computing.



M. Sreelatha is a Professor in CSE at RVR & JC College of Engineering, India. She received M.Tech. from NIT Warangal and Ph.D. from Andhra University in Computer Science & Engineering. Her research interests include Data Mining, Cloud Computing and Information Security.