Wireless Personal Communications (2020) 110:1887–1913
https://doi.org/10.1007/s11277-019-06817-w

Multi-Objective Task Scheduling Using Hybrid Whale Genetic Optimization Algorithm in Heterogeneous Computing Environment

Gobalakrishnan Natesan[1] · Arun Chokkalingam[2]

Published online: 17 October 2019

Abstract

The system of cloud computing comprises of several servers that are inter-connected in a datacenter, provisioned dynamically to cater on-demand services through the front-end interface for the clients. Improvement in virtualization technology has made cloud computing a viable option for various application services development. Cloud datacenters process the tasks on the basis of pay as you use manner. Task scheduling is one of the important research challenges in cloud computing. The formulation of task scheduling probes has been depicted to be NP-hard hence identifying the solution for a bigger problem is intractable. The dissimilar feature of cloud resources makes task scheduling non-trivial. NP-hard problem arises due to the dynamic behavior of the dissimilar resources identified in the cloud computing environment. Task scheduling can be optimized using a meta-heuristic algorithm. In this paper, we have combined two meta-heuristic techniques, namely Whale Optimization Algorithm (WOA) and Genetic Algorithm (GA) to devise a new hybridized algorithm called as Whale Genetic Optimization Algorithm. Our aim is to minimize the makespan and cost while scheduling the tasks. The simulation is done by using Cloudsim toolkit. The results obtained shows significant reduction in the execution time that was measured in terms of enactment amelioration rate. These results were compared with the classical WOA and standard GA. The results of the proposed technique provide higher quality solution for task scheduling.

Keywords Cloud computing · Algorithm · NP-hard · Task scheduling · Hybrid · Virtualization and optimization

✉ Gobalakrishnan Natesan
gobalakrishnanse@gmail.com

[1] Department of Information Technology, St. Joseph's College of Engineering, Sathyabama University, Chennai, Tamil Nadu, India

[2] R.M.K College of Engineering and Technology, Chennai, Tamil Nadu, India

1 Introduction

One of the most trending and the easily recognizable technologies among the academicians, researchers and businessmen is cloud computing as it allows storage of information and its utilities and allows it to be used from anywhere [1]. It abides a set of non-similar processing nodes, virtual systems and services that are provided among the opponent clients system dynamically on basis of their processing speed, accuracy and QoS needs. The service provisions of the environment of cloud form three tiers: Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) [2]. These layers stakes administration and monitoring of distinct of the final stack.

Several industries and forums send their data and services to the cloud everyday due to its high flexibility and effective structures which enable QoS sanctioned processing surroundings added with constructive services that can quickly be allowed for use with optimal effort from management [3]. As several researches concentrate on processes to provide efficient systems that has high system performance, the active researches included several attributes like security, storage performance, etc. promoting the system efficiency and making it avoided many obstacles.

Many achievements have been made in the research of the task scheduling in recent years. In this research, cloud computing is designed for execution of several applications of different types at the same time. The virtualization technology and automation management in the cloud computing makes the nodes join freely and exit operationally gradually reducing the cost of data center management. Task scheduling problem is considered as the toughest problem according to research [4, 5]. Rather than performing the regular routine of assigning tasks to resources to perform the needed operations, task scheduling works like a traditional computer OS scheduling. The main aim of a good scheduling algorithm is to improve the resource utilization rate. In NP-hard problem, the scheduling increases the efficiency, mapping various tasks of services to a set of resources. Finding solutions to the NP-hard problem is impractical with known algorithms, which requires an optimum scheduler which works with a particular computational speed. In general, NP-hard problems can be solved by inventory methods, heuristic methods or deterministic method [6].

However, enumeration needs to be used to build an optimum schedules for which it requires the process of building all scheduler that are possible and the results are compared one by one in order to select the optimal one which is not applicable for cloud scheduling due to the large number of tasks. Therefore, the proposed methods of literature for resolving problem of task scheduling that are either heuristic or meta-heuristic based [7–9]. The optimal solution can be found by heuristic based methods basis on some predefined rules and the solution quality was obtained by these methods on dependence of underlying rules and size of the problem. The solution was hence acquired by search methods of heuristics are not feasible and are operated at a very high cost. To solve highly complex problems, the meta-heuristic techniques are applied on it extensively. These techniques use a set of solutions to tresses the space of solution unlike the mathematical and heuristic techniques that uses a single candidate solution. The meta-heuristic algorithm attribute makes them perform better than the heuristic and mathematical techniques. Some of the well-known techniques for solving the issues of task scheduling in clouds are Genetic Algorithm (GA) [10], Water drop Algorithm [11], League championship Algorithm (LCA) [12], Firefly Algorithm [13], Ant Colony Optimization (ACO) [14] and Whale Optimization algorithm (WOA) [15].

In this paper, we have proposed a cloud task scheduling algorithm on WGOA to satisfy the needs of reducing both time and cost of task on the cloud environment. Hence in need to check the proposition of the scheduler, we have implemented using cloudsim simulation environment to execute a set of independent task on data centers with various models of configuration and cost.

The manuscript is organized as follows: Sect. 2 describes the related work of author's pertaining to task scheduling. Section 3 shows the solution framework model of task scheduling in cloud computing. Problem formulation and proposed hybrid WGOA technique for multi-objective task scheduling are described in Sects. 4 and 5. Section 6 describes the simulation results. Finally summary & conclusion of the proposed technique are described in Sect. 7.

2 Related Work

Several researchers have been performed on bag of independent tasks scheduling of applications on resources that are distributed in cloud computing. In such cases, many algorithms are used for scheduling and apply a variety of methods and parameters of scheduling [16–19]. In [20] author presents a multi-objective task scheduling techniques in cloud using resource cost model. The scheme use improved Ant Colony Optimization (ACO) algorithm to minimize the budget cost and execution time. Simulation outcome shows that the improved ACO technique presented is new adept of improving the performance of cloud, decrease the user cost, increase the utilization of resource and minimize the deadline violation rate.

Thamaraiselvi et al., [21] proposed a framework for scheduling bio-informatics application in the cloud environment. The author has used Particle Swarm Optimization (PSO) algorithm to schedule the bio-informatics tasks to minimize the makespan, complete the execution time of tasks within the time limit and cost. Self-adaptive learning—PSO based resource allocation framework in hybrid IaaS cloud environment was proposed by author [22]. In the hybrid IaaS environment the Cloud Service Provider (CSP) could outsource the tasks to external CSP since the resources are not enough to accommodate the tasks.

The hybridization of SOS and simulated annealing for task scheduling in cloud environment was used by the author of [23]. The hybridization algorithm objective is to reduce the response time, execution time and imbalance degree among VM's. In [24] author suggested a task scheduling scheme based on Global League Championship optimization to minimize the makespan and response time in scientific applications.

The authors of [25] proposed Intelligent Water Drops (IWD) algorithm to optimize the scheduling of workflow on the cloud where it was used and embedded inside the workflow simulation toolkit and was checked in various environments of cloud with cost models. The authors of [26] proposed DVFS technique to optimize the energy and cost consumed in HCS framework by implementation of green service level agreement made between cloud users. All empty slacks of schedule on processing elements and extended makespan.

Jiachen yang et al. [27] proposed a task scheduling algorithm having game theory designed for management energy in cloud computing. On basis of game theory algorithm the reliability of the balanced task is considered. It is implemented using cooperative game model and game strategy in the task scheduling. The authors of [28] described the pre-power determination and post-power determination algorithms to perform energy and cost constrained scheduling on several many core processor of precedence constrained parallel tasks to be performed on continuous or discrete speed levels by embedding equal speed method onto our algorithms giving high performance.

3 Solution Framework

The cloud task scheduling system has three main sections namely the policies module, the objective function and the scheduler module. The communication and interaction among the three modules are depicted and implemented in Fig. 1. This is implemented using the task scheduling technique.

The cloud consumers decide on what task has to be done and come up with a collection of tasks forming a task pool which all has to be completed on which the scheduling process has to be applied along with the objective function and then on this data is formed, the scheduler by making use of the WOA & GA techniques, provides the needed resources like storage, CPU and memory to the respective servers from the cloud resource pool. The first module of scheduling the system is the policies module on which contains scheduling process are applied which are generally set by the cloud providers. These policies of scheduling are normally a set of constraints and rules to enable resource allocation and establish SLA's for all the applications of cloud. Resource conflicts occurs on a regular basis as the cloud computing environment provides the illusion of infinite resources but in reality there are no adequate resources available to satisfy their need/tasks. Hence, this can be settled by using scheduling policies.

The second module is the objective/fitness function. It is used for determination and evaluation of the rank and quality of a schedule. It is given for all the tasks and hence a joint objective function is available for every schedule. It's main purposes is to test and make sure 'n' tasks are scheduled to 'm' cloud resources. So that the goal of minimum makespan problem is an NP hard problem. The most important component is the scheduler which forms the third module of our system and it's responsible for allocation of tasks to resources using hybrid WGOA optimized scheduling technique. Initially it collects tasks and needed resource information from the cloud information server and then secondly it judges whether the resource R_j satisfies the requirements of the users task T_i, if it does then in the end R_j is allocated to T_i.

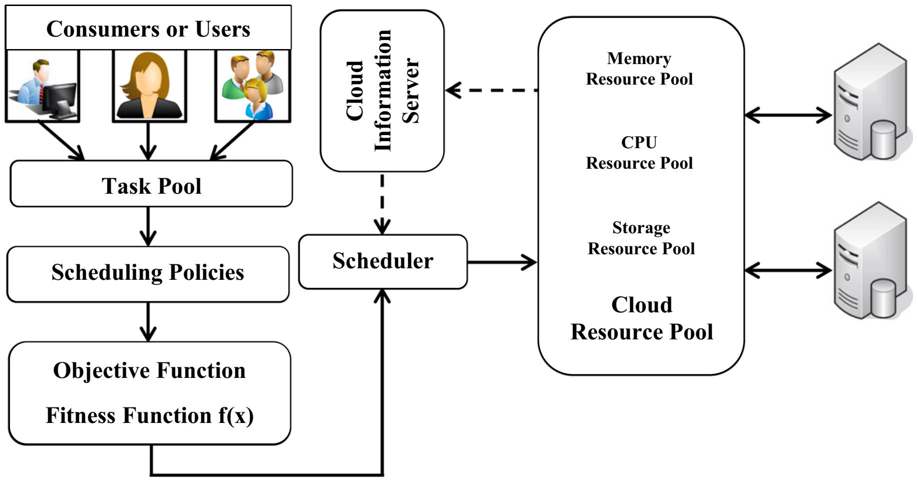


Fig. 1 Scheduling system framework model in cloud

4 Problem Formulation

The Cloud Broker (CB) receives the tasks to be scheduled and it uses the Cloud Information Server (CIS) to find the services that are needed to perform the execution of tasks that are received from the client and then scheduling of tasks is performed on the services that are discovered. For e.g., tasks $\left\{T^1, T^2, T^3, \dots T^y\right\}$ are given to CB at a particular time interval. The VMs are non-similar having different processing speeds and memory showing a task to be executed on various VMs will end in disparate costs for execution. Say VMs $\left\{M^1, M^2, M^3, \dots M^x\right\}$ are present when CB receives the tasks. FCFS is used for the execution of tasks to be scheduled present on the VMs. Our main adage is to perform task scheduling on VMs to achieve higher resource utilization, minimization of makespan and cost. Therefore expected time to complete (ETC) of the tasks on which scheduling needs to be performed on each VM is used to make decisions in scheduling by the proposed method. ETC values are depicted using MIPS ratio of the task length of VM. ETC values are generally depicted in a matrix format, where the row represents the no. of tasks to be scheduled and column represent the no. of VM's that are available of each matrix.

Every row in ETC matrix determines/depicts the given tasks execution time for each VM while execution time of each task given on VM represents each column. Because our aim is to minimize the makespan and cost by identifying the optimal group of tasks to be performed on VMs. Let C_{xy}, $y = 1, 2, 3 \dots m$, $x = 1, 2, 3 \dots n$ be consider as execution time of yth task is executing on the xth VM, where $m \rightarrow$ no. of VMs and $n \rightarrow$ no of tasks. The notations and their descriptions used in our proposed system are represented in Table 1.

4.1 Performance Metrics

The following metrics are the used to check proposed method performance.

Table 1 Notations and its description

Notation	Description
T_y	The task y, $1 \le y \le N$
R_x	The resource x, $1 \le x \le K$
K, N	The amount of resources and tasks
$CPU^{Cost}(x)$, $Storage^{Cost}(x)$, $Memory^{Cost}(x)$, $Bandwidth^{Cost}(x)$	The cost of CPU, Storage, Memory and Bandwidth of R_x
CPU^{Base}	The lowest usage of CPU base cost
$Storage^{Case}$	The base cost of storage under the lowest usage
$Memory^{Case}$	The base cost of memory under 1 GB memory
$Bandwidth^{Case}$	The base cost of bandwidth under the lowest usage
Run time cost (x)	The duration time of task T_y in resource R_x
CPU^{Tran}, $Storage^{Tran}$,	
$Memory^{Tran}$, $Bandwidth^{Tran}$	The transmission cost associated with CPU, storage, memory and bandwidth

4.1.1 Makespan

Makespan is the time when the execution of the last task is finished. It is one of the famous metrics for performance of scheduling methods. Lower makespan depicts best and optimal task scheduling of VMs.

$$makespan = minimize_{S_y=Sch}\left\{maximum_{y \in Tasks} T_y\right\} \tag{1}$$

where T_y denotes the completion time of task '*y*', the set of all possible schedules are represented as '*Sch*' and *'Tasks'* the set of whole tasks to be scheduled. To make (1) easily estimable, makespan is expressed in terms of completion time of the virtual machine. The reason for this assumption is that the time by which the final task is completed. It is equivalent to the time taken to complete its tasks by the last virtual machine. Let completion be a vector of size nb_VM completion [VM] indicates process finalizing time of VM for the previously assigned jobs as well as of those already planned for the virtual machine.

Mathematically,

$$completion\,[VM] = ready\_time[VM] + \sum_{\{y \in Tasks | schedule[y] = VM\}} ETC[y]\,[VM] \tag{2}$$

Also, makespan from 1 is as,

$$makespan = maximum\left\{completion[y]\,|y \in VM\right\} \tag{3}$$

4.1.2 Enactment Amelioration Rate (EAR) Percentage

The EAR is termed as the percentage of amelioration of performance for the technique y is proposed over the other technique z and is measured using the following equation

$$EAR(\%) = \left(Makespan\left(C_z\right) - Makespan\left(C_y\right)\right) \times \frac{100}{Makespan\left(C_y\right)} \tag{4}$$

4.1.3 Cost

Resources and services are of wide range in cloud computing. Say for e.g.: few tasks require high processing capability resources whereas few other tasks require more storage. Also, different resources have different costs, even the task costs will be varying. Hence, if we take in the task demand difference for resources, it will reflect in the relationship among the user budget and resource costs. To talk about this problem, a resource cost that divides the cost of the resource into parts of memory and CPU can be proposed.

The CPU cost can be depicted as:

$$CPU^{Cost}(x) = CPU^{Base} \times CPU\ of\ VM(x) \times Run\ time\ cost\,(x) + CPU^{Tran} cost \tag{5}$$

where CPU^{Base} is the cost base i.e. needed for a resource when it is used by the lowest utilization, run time cost is the time that the task is run on the VM and CPU^{Tran} is the cost of CPU transmission.

$$Memory^{Cost}(x) = Memory^{Base} \times Memory\ of\ VM(x) \times Run\ time\ cost\,(x) + Memory^{Tran} cost \tag{6}$$

Similarly, $Memory^{Base}$ is the base cost when 0.5 MB memory used, run time cost is duration of the task is run on the VM. $Memory^{Tran}$ is the cost of Memory transmission.

The formula of storage cost is defined as,

$$Storage^{Cost}(x) = Storage^{Base} \times Storage\ of\ VM\ (x) \times Run\ time\ cost\ (x) + Storage^{Tran} cost \tag{7}$$

$Storage^{Base}$ is the base cost of storage. The cost incurred by the storage transmission is represented by $Storage^{Tran}$. The run time cost is amount of data is stored during task execution. The bandwidth cost is defined by the formula as follows:

$$Bandwidth^{Cost}(x) = Bandwidth^{Base} \times Bandwidth\ of\ VM\ (x) \times Run\ time\ cost\ (x) + Bandwidth^{Tran} cost \tag{8}$$

The cost incurred by the bandwidth transmission is represented by $Bandwidth^{Tran}$. The runtime cost is amount of bandwidth used during task runtime.$Bandwidth^{Base}$ is base cost of bandwidth.

The depiction of cost function can be represented as follows on basis of the CPU, Bandwidth, Storage and Memory.

$$Cost(x) = CPU^{cost}(x) + Bandwidth^{cost}(x) + Storage^{cost}(x) + Memory^{cost}(x) \tag{9}$$

$$Total\ Cost(x) = \sum_{x=1}^{n} Cost(x) \tag{10}$$

4.1.4 Fitness Function Calculation

To formulate the objective function, scheduling the tasks are taken into account. The main intension of the cloud service provider is to maximize the profit, whereas the customers expect to minimize the execution time and cost cutting. Thus to solve the above problem, fitness function is calculated to evaluate the quality of feasible solutions. However, for each optimization problem the fitness function is needed to find near optimal solutions. Therefore, the fitness value is calculated for our proposed model by using Eq. (11).

$$Fitness\ Function\left(x_i^t\right) = \lambda \cdot Makespan + \delta \cdot Total\ Cost \tag{11}$$

where fitness function is denoted as $\left(x_i^t\right)$ and the values of λ and δ ranges between $1 > \lambda \geq 0$ and $1 > \delta \geq 0$ respectively.

5 Hybrid WGOA Based Scheduling Techniques

WOA is a new Meta-heuristic algorithm devised by S. Mirjalili and A. Lewis [29]. it is inspired form the natural phenomenon observed in hunting patterns of humpback whales. Humpback whales are one of the most interesting species of the whale family. The most interesting thing about them is their special hunting and foraging method called Bubble-net feeding. They prey on krill or small schools of fish by creating distinctive bubble in a circular or '9' shaped path after diving 10–15 m down. The WOA method is done in three steps.

1. Encircling the prey
2. Exploitation phase—Bubble-net attacking
3. Exploration phase—search for prey

6 Encircling the Prey

The best search agent is defined and the positions are updated as the whale gets closer to the prey. This behavior can be denoted as vector equations given below:

$$\overrightarrow{D_{dis}} = \left|\vec{C} \cdot \overrightarrow{X'_{bestpos}}(t) - \overrightarrow{X_{pos}}(t)\right| \tag{12}$$

$$\overrightarrow{X_{pos}}(t+1) = \overrightarrow{X'_{bestpos}}(t) - \vec{A} \cdot \overrightarrow{D_{dis}} \tag{13}$$

where, current iteration is denoted as ‘t’, Coefficient vectors are A and C, best solution position vector is $\overrightarrow{X'_{bestpos}}$, $\overrightarrow{X_{pos}}$ is the position vector.

To calculate Coefficient vectors $\vec{A}$ and $\vec{C}$,

$$\vec{A} = 2\vec{a} \cdot \vec{r} - \vec{a} \tag{14}$$

$$\vec{C} = 2 \cdot \vec{r} \tag{15}$$

Where $\vec{a} \rightarrow$ Linearly decreasing from [2 to 0]. $\vec{r} \rightarrow$ Random vector ranging from [0 to 1].

7 Exploitation Phase: (Bubble-Net Attack)

There are two methods to demonstrate the bubble-net attacking method:

1. Shrinking encircling mechanism
2. Spiral updating position

7.1 Shrinking Encircling Mechanism

In this method, the coefficient vector $\vec{A}$ is a arbitrary value in the interval $[-a, a]$, we decrease the value of $\vec{a}$ in subsequent iterations form [2 to 0]. The following Fig. 2 shows the possible positions of whale from (X, Y) to (X^*, Y^*).

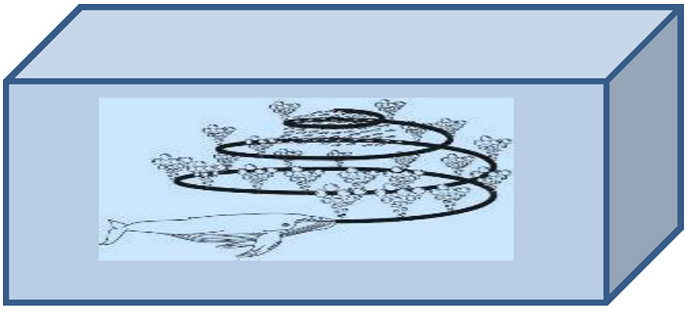
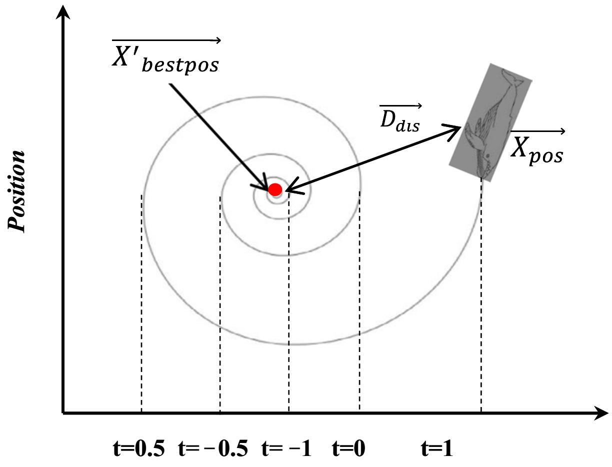


Fig. 2 Bubble net feeding methods of humpback whales



$\overrightarrow{X'_{bestpos}}$
$\overrightarrow{D_{dis}}$
$\overrightarrow{X_{pos}}$
Position
t=0.5 t=−0.5 t=−1 t=0 t=1

Fig. 3 Bubble net search spiral updating position method

7.2 Spiral Updating Position

The spiral updating position is shown in Fig. 3. In this method, the distance between the whale and the prey is calculated first and the helix shaped movement of humpback whales is traced using the given formula.

$$\overrightarrow{X_{pos}}(t+1) = \overrightarrow{D_{dis}} \cdot e^{bl} \cdot cos(2\pi l) + \overrightarrow{X'_{bestpos}}(t) \tag{16}$$

where $\overrightarrow{D_{dis}} = \left|\overrightarrow{X'_{bestpos}}(t) - \overrightarrow{X_{pos}}(t)\right|$ and designate the distance of ith whale to the prey. Here, 'b' →constant for logarithmic spiral and 'l' →random no ranging from [− 1 to1].

To elect the finest method, the following probability equation is used,

$$\overrightarrow{X_{pos}}(t+1) = \begin{cases} \overrightarrow{X'_{bestpos}}(t) - \vec{A} \cdot \overrightarrow{D_{dis}} & if \quad p < 0.5 \\ \overrightarrow{D_{dis}} \cdot e^{bl} \cdot cos(2\pi l) + \overrightarrow{X'_{bestpos}}(t) & if \quad p \geq 0.5 \end{cases} \tag{17}$$

Where *'p'* is the random number between [0,1], *'l'* is between [− 1,1] and *'b'* is a constant for the spiral shape.

8 Exploration Phase—Search for Prey

Based on variation of $\vec{A}$ vector the prey can be searched using random variables from [− 1 to 1]. This mechanism emphasizes exploration and allows the WOA algorithm to perform a global search.

Mathematically,

$$\overrightarrow{D_{dis}} = \left|\vec{C} \cdot \overrightarrow{X_{randpos}}(t) - \overrightarrow{X_{pos}}(t)\right| \tag{18}$$

$$\overrightarrow{X_{dis}}(t+1) = \overrightarrow{X_{randpos}}(t) - \vec{A} \cdot \overrightarrow{D_{dis}} \tag{19}$$

In the proposed work, genetic algorithm crossover and mutation operations are integrated with WOA. This hybridization is used to produce the near optimal solution for multi-objective optimization process.

8.1 Crossover

Crossover is one of the genetic algorithm operators. Cross-overing occurs between pairs of chromosomes. This operation is a method of replacing a few of the chromosomes in one worst chromosome by the corresponding chromosome in the other worst chromosome. In the WO algorithm half of the solution is considered as best case solution and the rest half is considered as worst case solution. For worst case solution of the WOA, the two point crossover has been employed in our work.

Consider seven whales, where the first selected whale W1 is assumed to obtain a worst case solution from the worst population and the second selected whale is W2. Each whale is coded with an integer array. This integer array is divided into segments. Some of the integers segments are replaced during the crossover operation. In the below figure seven tasks are consider and the detailed crossover operation is shown in Fig. 4.

8.2 Mutation

In addition to the two point crossover operation, a mutation process is used usually with a low probability. A higher probability results in a randomized search. This is done with the crossover solution. In particular, scramble mutation is deployed here. Scramble mutation is popular with solutions including permutation representations. Scramble mutation is to perform from the whole chromosome, a part of genes is elected and their values are jumbled or shuffled randomly as depicted in Fig. 5.

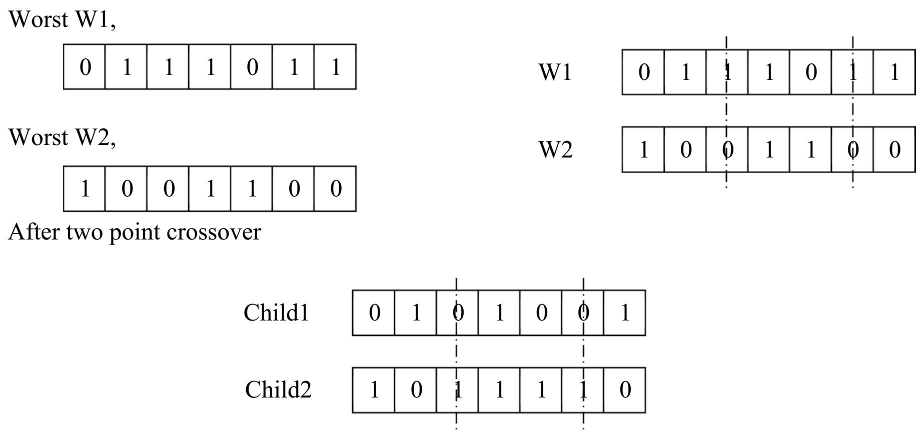
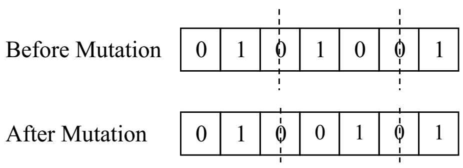


Fig. 4 GA cross over operation

Fig. 5 GA mutation operation



9 Pseudo Code of WGOA

```
Let the generated population of whales be $X_i$ (i = 1,2,…..n)
For each search agent the fitness is evaluated using Eq (11)
$\overrightarrow{X'_{bestpos}}$= the best search agent
While (t < $Max_{itr}$)
    For each search agent
        Update $a$, $l$ and $p$, coefficient vector $\vec{A}$ and $\vec{C}$
            If1 ($p$ < 0.5)
              If2 ($|\vec{A}|$ < 1)
                The current search agent position is updated by Eq (12)
                  Else if2 ($|\vec{A}|$ ≥ 1)
                      The random search agent ($\overrightarrow{X_{randpos}}$) is selected
                      The current search agent position is updated by Eq (18)
                  End if2
              Else if1 ($p$ ≥ 0.5)
                  Current search agent position is updated by Eq (16)
          End if1
        End for
Analysis if any exploration agent goes apart from the search space and amends it
For each search agent the fitness is evaluated
If4 (Better Solution)
        Update $\overrightarrow{X'_{bestpos}}$
        Use crossover and mutation to update the worst chromosome
        // Apply Two Point crossover
        Crossover
        {
            For i=1,2….N do
            Randomly select two worst chromosomes $w_1$ and $w_2$ from the initial solution or
        population
            Produce $w_{11}$ and $w_{12}$ by two point cross over
            Save $w_{11}$ and $w_{12}$ to the new solution $w_i^{New}$
         End for
        }
        //Apply scramble mutation
        Mutation
        {
            For i=1,2….N do
            Select the new solution from the cross over
            Mutate each bit of $w_i^{New}$ ;
            Save mutated solution as the new solution
            End for
        }
        t = t+1
End if4
End while
return $\overrightarrow{X'_{bestpos}}$
```

10 Experimentations and Results

The cloudsim toolkit is an event cloud simulation toolkit based on java. It is popularly named structure for constructing and processing cloud computing architectures and services and also enables the processing's of large cloud computing nodes and environment. It can supports to schedule data intensive application, computing intensive application and scientific application. Therefore, to check the proposed task scheduling algorithm, the cloud simulation tool has been used. It does not support only the checking of scheduling techniques, but also allows the analysis of several failures that can be caused. So, we used our WGOA in cloudsim. We verified our experiments by running it on a PC with Intel(R) Core(TM) i5–457 OS, 4 CPU @ 2.9 GHz, a RAM of 8 GB and a 64-bit windows OS.

10.1 Simulations and Experimentation

To demonstrate the efficiency of the algorithm on a cloud based network of virtual machines, we perform certain experiments using the CloudSim tool. We compare the proposed algorithm with the following sets:

- Simulation of proposed algorithm in comparison with the given standard algorithms:
  - First-Come-First-Serve (FCFS) algorithm
  - Min–Min algorithm
  - Max–Min algorithm
- Simulation of proposed algorithm in comparison with other meta-heuristic algorithms

10.1.1 Simulation of Proposed Algorithm in Comparison with the Standard Algorithms

The above simulation cases are considered in four sets, where, in each set, the parameters for the virtual machines are varied—with respect to the number of tasks allotted, number of virtual machines used, bandwidth values and the completion speed for tasks. The parameter setting for standard algorithm experiment is shown in Table 2.

A homogenous configuration implies a collection of systems where the parameters are fixed for all virtual machines, whereas, a heterogeneous configuration implies varied parameters for each virtual machine in the network.

10.1.2 Evaluating the Performance of the Above Simulations

SET 1

In the first set, let us assume 5 virtual machines. The number of tasks is 25, with same task lengths of 15 MI.

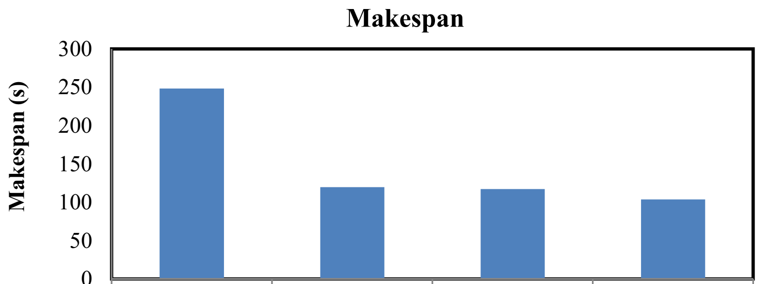
1. Homogenous network:

Here, the value of MIPS and the bandwidth is considered to be 1200.

In the above & below results shown in Figs. 6 and 7 that is obtained for both the homogenous and heterogeneous configurations respectively, it can be seen that the proposed Whale Genetic Optimization Algorithm obtains the least makespan whereas the FCFS

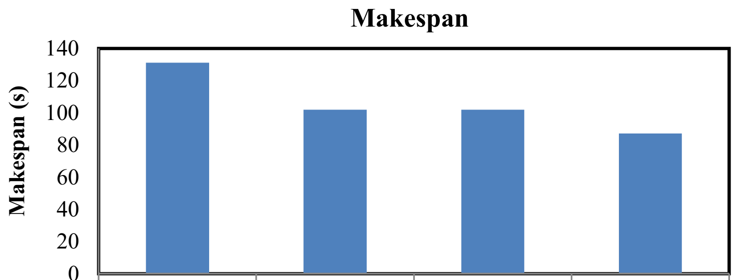
Table 2 The specific parameter settings are given for standard algorithms

Parameter	Value
No. of data centres	1
No. of hosts	2
No. of virtual machines	5,10,20,25
No. of CPUs	1
Bandwidth	Homogenous: 1200; Heterogeneous: > 500 to ≤ 1200
MIPS of CPU per virtual machine	Homogenous: 1200; Heterogeneous: > 500 to ≤ 1200
RAM	0.6 GB



	FCFS	Min-Min	Max-Min	WGOA
■ VM's=10	248.3	119.83	117.25	103.62

Fig. 6 Makespan value of homogeneous environment (MIPS & Bandwidth = 1200)



	FCFS	Min-Min	Max-Min	WGOA
■ VM's = 10	131.19	101.86	101.86	87.13

Fig. 7 Makespan value of heterogeneous environment (MIPS & Bandwidth = 700, 900 & 1100)

algorithm takes the maximum completion time. Other algorithms such as the Min–Min and Max–Min also obtain closely related results to each other and to that of the proposed WGOA. The only difference between the homogenous and heterogeneous configurations is that the ranges for the makespan are less for the heterogeneous configuration.

2. Heterogeneous network: Here, the values of MIPS and bandwidth are considered to be 700, 900 and 1100 consequently.

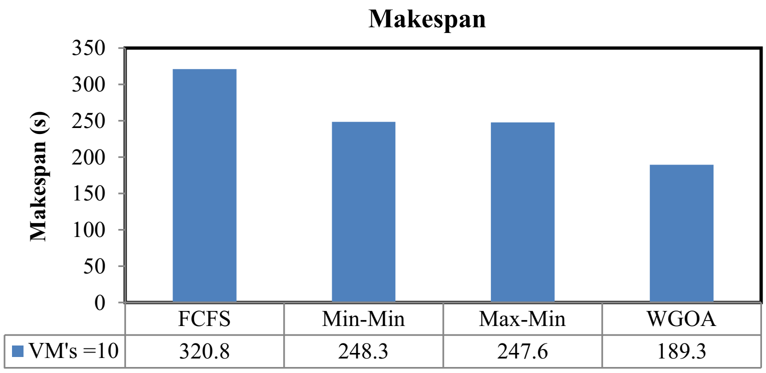
SET 2

In the second set, let us consider 10 virtual machines. The experiment has 50 tasks, with different task lengths and within the range of 10–50 MI.

1. Homogenous network:

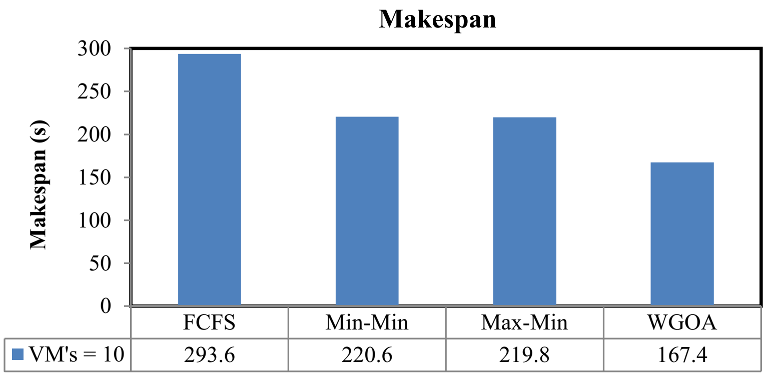
Here, the value of MIPS and the bandwidth is considered to be 1200.

Similar to set 1, the results obtained in set 2, as depicted in Figs. 8 and 9 for both the homogenous and heterogeneous configurations respectively, the proposed WGO algorithm obtains the minimum makespan whereas the FCFS algorithm takes the maximum completion time. Other algorithms such as the Min–Min and Max–Min also obtain closely related results to each other and to that of the proposed WGOA, despite the increase in the number of virtual machines and heterogeneous task lengths.



	FCFS	Min-Min	Max-Min	WGOA
■ VM's =10	320.8	248.3	247.6	189.3

Fig. 8 Makespan value of homogeneous environment (MIPS & Bandwidth = 1200)



	FCFS	Min-Min	Max-Min	WGOA
■ VM's = 10	293.6	220.6	219.8	167.4

Fig. 9 Makespan value of heterogeneous environment (MIPS & Bandwidth = 500–1200)

2. Heterogeneous network:

Here, the values of MIPS and bandwidth are considered to be between the ranges of greater than 500 and less than or equal to 1200.

SET 3

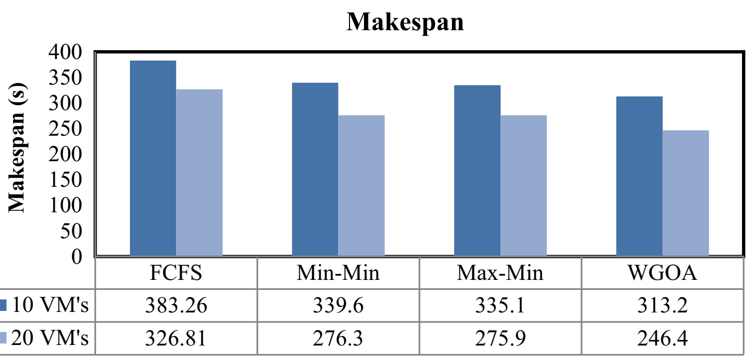
In the third scenario, 10 and 20 virtual machines are considered for this simulation. This simulation has 75 tasks, with different task lengths for heterogeneous configurations and equal tasks length for homogeneous configurations. The results obtained have been depicted in Figs. 10 and 11 for homogeneous and heterogeneous network configurations respectively. The value of MIPS and the bandwidth are equal here and the task lengths are varied instead.

1. Homogenous network:

Here, all the task lengths are same length 20 MI

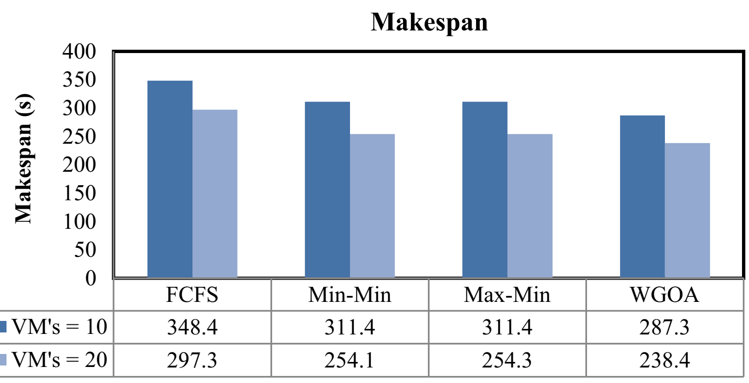
2. Heterogeneous network:

Here, all the task lengths are within the range of not more than 20 MI.



	FCFS	Min-Min	Max-Min	WGOA
■ 10 VM's	383.26	339.6	335.1	313.2
■ 20 VM's	326.81	276.3	275.9	246.4

Fig. 10 Makespan value of homogeneous environment for 75 tasks



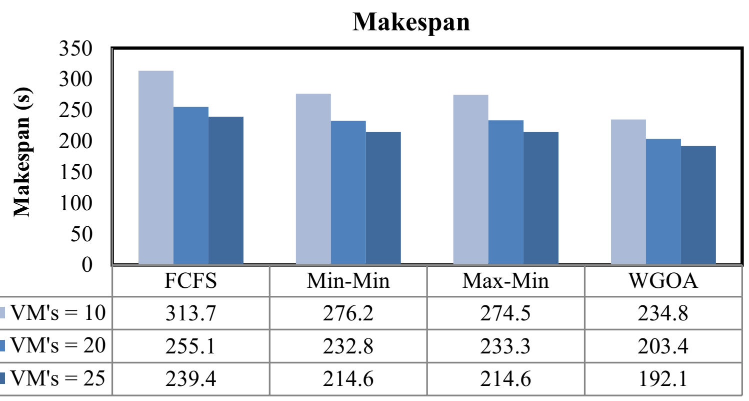
	FCFS	Min-Min	Max-Min	WGOA
■ VM's = 10	348.4	311.4	311.4	287.3
■ VM's = 20	297.3	254.1	254.3	238.4

Fig. 11 Makespan value of heterogeneous environment for 75 tasks

SET 4

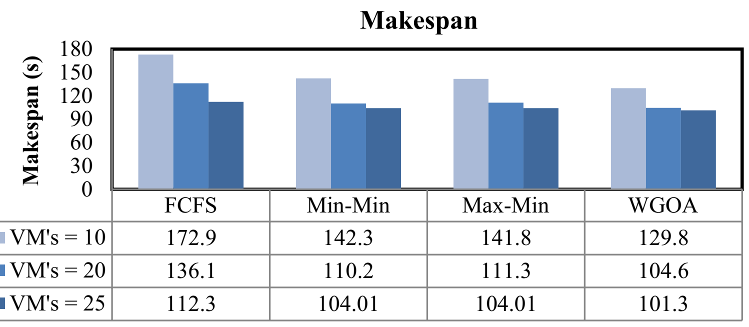
In this set of experiments, 10, 20 and 25 virtual machines are used. This simulation has 100 tasks, with the following five cases:

- All virtual machines are homogenous (MIPS = 1200 and Bandwidth = 1200) and all tasks have equal length (Task length = 25 MI). The result obtained for this case has been depicted in Fig. 12.
- All virtual machines are homogenous (MIPS = 1200 and Bandwidth = 1200) and all tasks are various length (Task length ≤ 25 MI). The result obtained for this case has been depicted in Fig. 13.
- All virtual machines are heterogeneous (MIPS & Bandwidth range is > 500 to ≤ 1200) and all tasks are same length (Task length = 25 MI). The result obtained for this case has been depicted in Fig. 14.
- All virtual machines are heterogeneous (MIPS & Bandwidth range is > 500 to ≤ 1200) and all tasks are different length (Task length ≤ 25 MI). The result obtained for this case has been depicted in Fig. 15.
- 50 tasks are of equal length (Task length = 25 MI) and 50 tasks are various length (Task length ≤ 25 MI). In this case, half of the VMs are homogenous (MIPS = 1200 and Bandwidth = 1200) and half of them are heterogeneous (MIPS & Bandwidth range is > 500



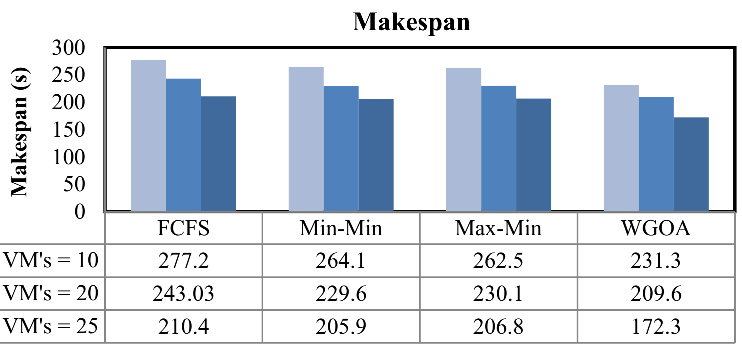
	FCFS	Min-Min	Max-Min	WGOA
▨ VM's = 10	313.7	276.2	274.5	234.8
▨ VM's = 20	255.1	232.8	233.3	203.4
▨ VM's = 25	239.4	214.6	214.6	192.1

Fig. 12 Makespan value of homogeneous environment for 100 tasks



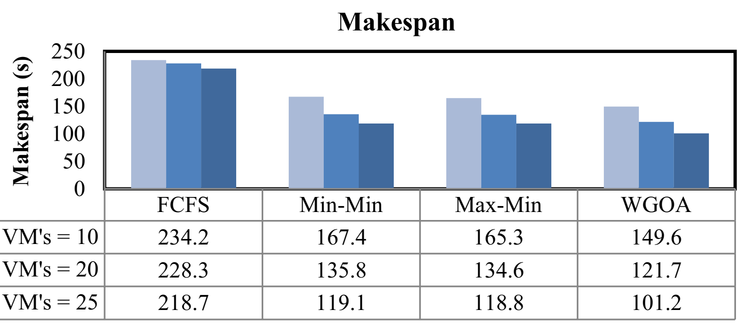
	FCFS	Min-Min	Max-Min	WGOA
▨ VM's = 10	172.9	142.3	141.8	129.8
▨ VM's = 20	136.1	110.2	111.3	104.6
▨ VM's = 25	112.3	104.01	104.01	101.3

Fig. 13 Makespan value of homogeneous environment for 100 tasks



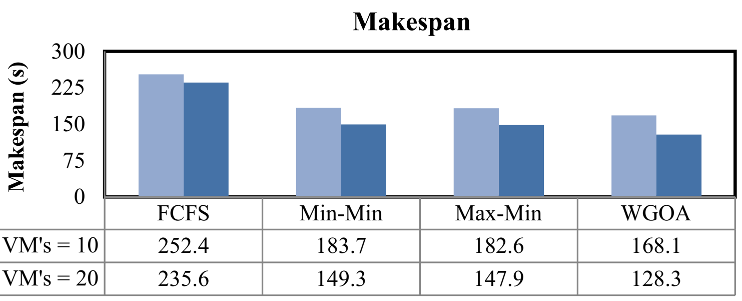
	FCFS	Min-Min	Max-Min	WGOA
■ VM's = 10	277.2	264.1	262.5	231.3
■ VM's = 20	243.03	229.6	230.1	209.6
■ VM's = 25	210.4	205.9	206.8	172.3

Fig. 14 Makespan value of heterogeneous environment for 100 tasks



	FCFS	Min-Min	Max-Min	WGOA
■ VM's = 10	234.2	167.4	165.3	149.6
■ VM's = 20	228.3	135.8	134.6	121.7
■ VM's = 25	218.7	119.1	118.8	101.2

Fig. 15 Makespan value of heterogeneous environment for 100 tasks



	FCFS	Min-Min	Max-Min	WGOA
■ VM's = 10	252.4	183.7	182.6	168.1
■ VM's = 20	235.6	149.3	147.9	128.3

Fig. 16 Makespan value of mixed environment for 100 tasks

to $\leq$ 1200). In this scenario we considered 10 and 20 VMs. The result obtained for this case has been depicted in Fig. 16.

10.2 Simulation of Proposed Algorithm in Comparison with Other Meta-Heuristic Algorithms

The proposed Whale Genetic Optimization algorithm is modeled and built to address the problems of global scheduling of tasks in the cloud computing environment. For understand and check the performance of the technique to measure the goodness and efficiency of the WGOA scheme, cost and makespan parameters are kept. Hence, the main aim of this section is to test and check the performance of proposed method with other meta-heuristic algorithms such as standard GA and classical WOA. The experiments are performed continuously up to 25 times and the average is calculated and it is used with results that are available for comparison. Table 3 represents the setting the parameter of the selected techniques to perform scheduling. The simulations are then used in four datacenters with 20 hosts. Each host consists, 5 VMs and 10 VMs. If each host has 5 VMs, totally 50 VMS are considering for our simulation. Similarly, each host has 10 VMs the total VMs are 100 for our experiment. The parameter settings of the cloud simulator are depicted in Table 4. An assumption of simulation is considered that the tasks are independent and has no constraint on priority among tasks during non-preemptive and execution. Three various data sets were considered to check the performance of the proposed method. They are normal distribution, uniform distribution and HPC2 N [23]. Equal number of tasks of various sizes namely small, medium and large can be portrayed using uniform distribution. Task of size medium, few small and large ones can be effectively portrayed using normal distribution. The performance of distributed systems can be evaluated by HPC2 N set log, which is one of the standard organized workloads.

Table 3 Algorithm parameters

S. No	Algorithm	Parameter	Value
1	WOA	*A*	[2, 0]
		L	[− 1, 1]
		B	1
		R	[0, 1]
2	GA	Crossover rate	0.4
		Mutation rate	0.1

Table 4 Simulation environment setup details

S. no	Entity type	Parameter	Value
1	Data centre	No. of datacenter	4
		No. of hosts	10
2	Host	Host memory	10 GB
		Host storage	50 GB
		Host bandwidth	2500 Mbps
3	Virtual machine (VMs)	No. of VM's	50 and 100
		Policy	Time-shared
		CPU capability	300–6000 MIPS
		VMM	Xen
		No. of CPU	1 on each
		VMs RAM	0.5 GB
		Operating system(s)	Windows and linux
4	Task	No. of tasks	100–500
		Length	100–16,000 MIPS

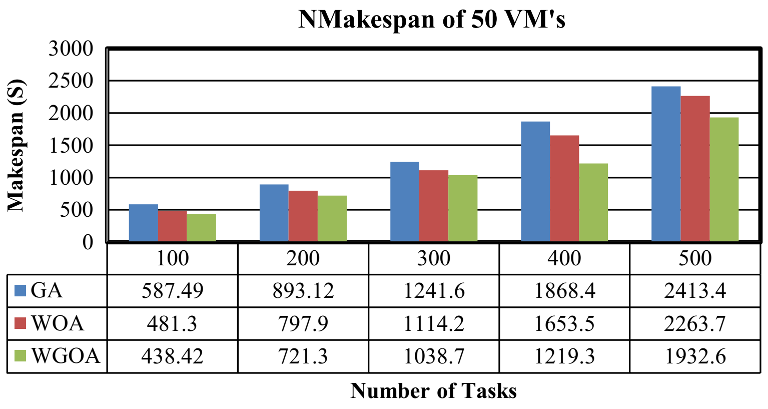
10.2.1 Makespan Performance Evaluation

In this section, we present simulation results to explain the efficiency of the proposed hybrid WGOA task scheduling in terms of makespan. Cloudlet length (task size) was generated from normal, uniform and HPC2 N distribution. Tasks of varying numbers are chosen as input to portray the scalability of the algorithm. We vary the number of tasks from 100 to 500 and present a comparison between standard GA and classical WOA. In this simulation, tasks were executed 50 times and an average was calculated. For all three distributions 50 VMs and 100 VMs were considered for task execution. Figures 17 and 18 shows the performance results of makespan for normal distribution.

Similarly Figs. 19 and 20 depicts the makespan of uniform distribution and Figs. 21 and 22 represent the HPC2 N distribution makespan. From the performance analysis, it was inferred that the proposed hybrid WGOA optimization method has better capability to attain near optimal value or gives better makespan for the tested task executions, when compared with standard GA and classical WOA techniques.

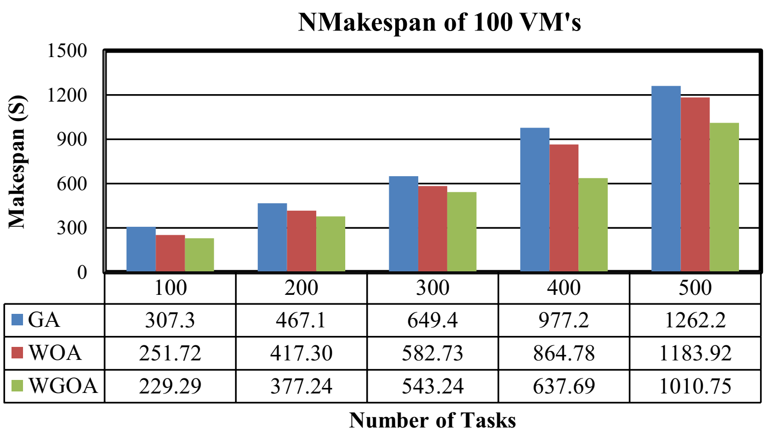
Table 5 show the normal distribution EAR (%) on total makespan of the proposed hybrid WGOA when compared with the classical WOA and standard GA schedulers. The proposed hybrid WGOA produces 17.94%, 30.90% makespan improvements when compared to classical WOA and standard GA respectively. A uniform distribution dataset is tested with our proposed hybrid method with classical WOA, and standard GA. The proposed hybrid WGOA approach makespan is fares 11.51%, 25.83% more in terms of EAR (%) when compared with standard approaches and the results are shown in the Table 6.

Similarly when HPC2 N dataset is given as test input between our proposed hybrid techniques and standard approaches, our proposed hybrid WGOA approach resulted in an improved performance of 7.44% and 22.34%. The obtained results are presented in the Table 7.



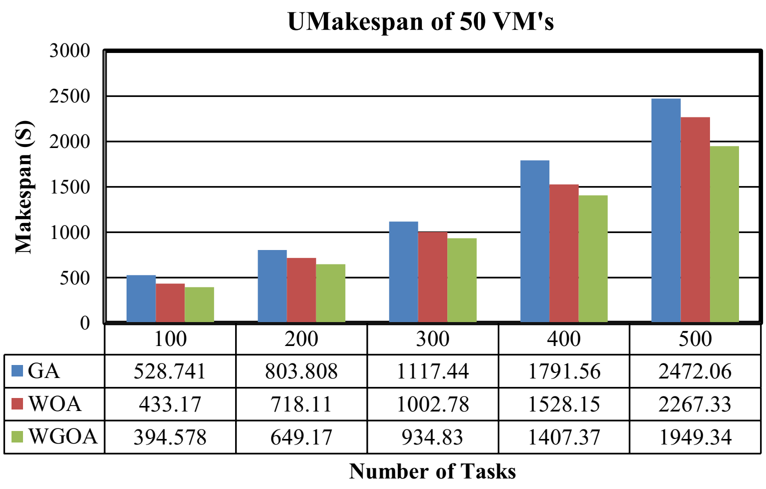
	100	200	300	400	500
🟦 GA	587.49	893.12	1241.6	1868.4	2413.4
🟥 WOA	481.3	797.9	1114.2	1653.5	2263.7
🟩 WGOA	438.42	721.3	1038.7	1219.3	1932.6

Fig. 17 Makespan of normal distribution 50 VMs



	100	200	300	400	500
🟦 GA	307.3	467.1	649.4	977.2	1262.2
🟥 WOA	251.72	417.30	582.73	864.78	1183.92
🟩 WGOA	229.29	377.24	543.24	637.69	1010.75

Fig. 18 Makespan of normal distribution 100 VMs

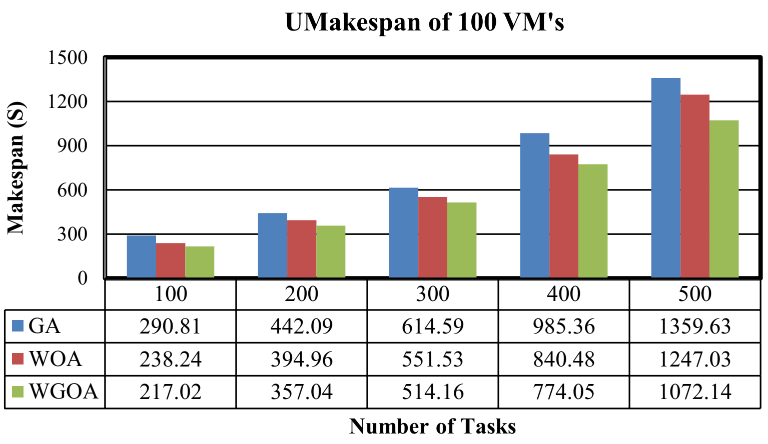


	100	200	300	400	500
🟦 GA	528.741	803.808	1117.44	1791.56	2472.06
🟥 WOA	433.17	718.11	1002.78	1528.15	2267.33
🟩 WGOA	394.578	649.17	934.83	1407.37	1949.34

Fig. 19 Makespan of uniform distribution 50 VMs

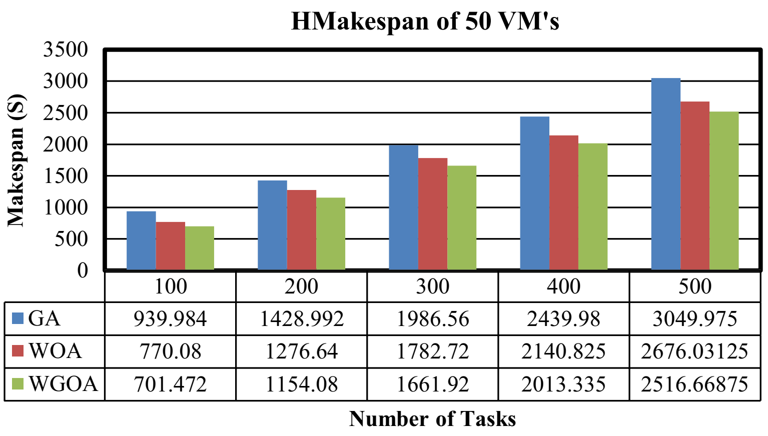
10.2.2 Cost

The second objective cost is simulated and the costs across the three methods are compared. The results of comparison are summarized in the figures provided. In Figs. 23 and 24, the costs of normal distribution for 200 and 400 tasks at varying deadlines are shown. Costs of uniform distribution for 100 and 300 tasks are depicted in Figs. 25 and 26. Similarly, the costs of HPC2 N of 200 and 500 tasks at different deadlines are shown in Figs. 27 and 28.



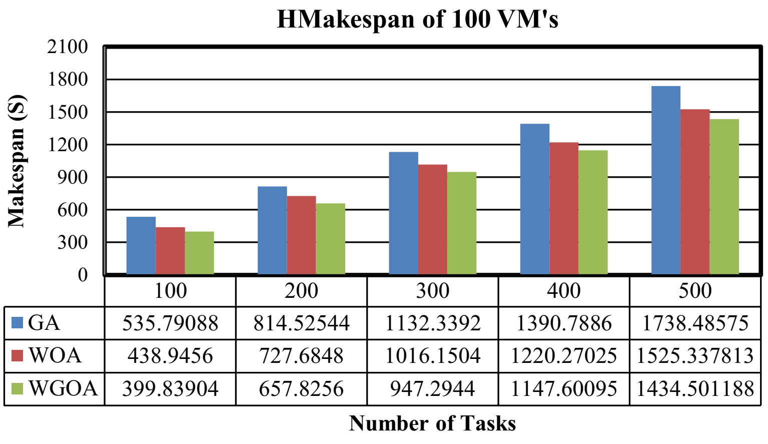
	100	200	300	400	500
🟦 GA	290.81	442.09	614.59	985.36	1359.63
🟥 WOA	238.24	394.96	551.53	840.48	1247.03
🟩 WGOA	217.02	357.04	514.16	774.05	1072.14

Fig. 20 Makespan of uniform distribution 100 VMs



	100	200	300	400	500
🟦 GA	939.984	1428.992	1986.56	2439.98	3049.975
🟥 WOA	770.08	1276.64	1782.72	2140.825	2676.03125
🟩 WGOA	701.472	1154.08	1661.92	2013.335	2516.66875

Fig. 21 Makespan of HPC2 N distribution 50 VMs



	100	200	300	400	500
🟦 GA	535.79088	814.52544	1132.3392	1390.7886	1738.48575
🟥 WOA	438.9456	727.6848	1016.1504	1220.27025	1525.337813
🟩 WGOA	399.83904	657.8256	947.2944	1147.60095	1434.501188

Fig. 22 Makespan of HPC2 N distribution 100 VMs

Table 5 EAR of normal distribution makespan for 50 VMs

	GA	WOA	WGOA
Total makespan	7004.01	6310.6	5350.32
EAR% over GA		10.98802	30.90824
EAR% over WOA			17.94809

Table 6 EAR of uniform distribution makespan for 50 VMs

	GA	WOA	WGOA
Total makespan	6713.609	5949.54	5335.288
EAR% over GA		12.84249	25.83405
EAR% over WOA			11.51301

Table 7 EAR of HPC2 N distribution makespan for 50 VMs

	GA	WOA	WGOA
Total makespan	9845.491	8646.296	8047.476
EAR% over GA		13.86946	22.3426
EAR% over WOA			7.441097

For normal distribution we considered 200 and 400 tasks with varied deadlines. From the simulation, the maximum cost acquired by WGOA varies from 115.92–50.4 and 220.5–95.76. Similarly, for WOA the maximum costs are 128.83–67.67 and 244.77–116.25 and that of GA are 1151.50–77.15 and 287.85–146.59 both of which are higher than the costs of WGOA.

In uniform distribution, for 100 and 300 tasks on a deadline of 10–100, the maximum acquired cost of WGOA is in the range of 104.33–45.36 and 187.79–81.65 respectively.

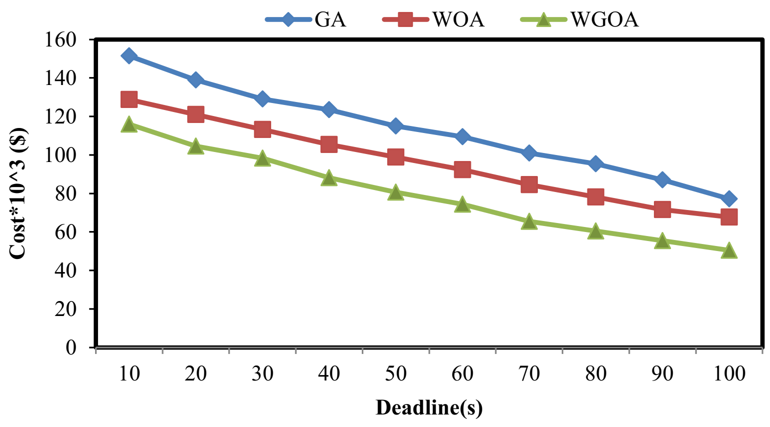


Fig. 23 Cost of normal distribution (Tasks = 200)

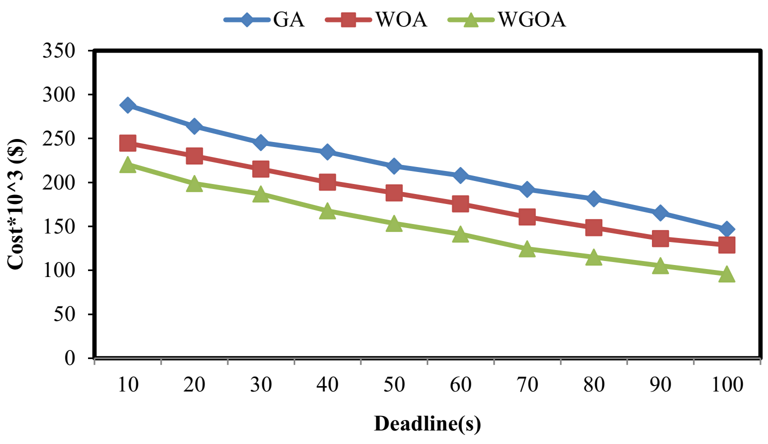


Fig. 24 Cost of normal distribution (Tasks = 400)

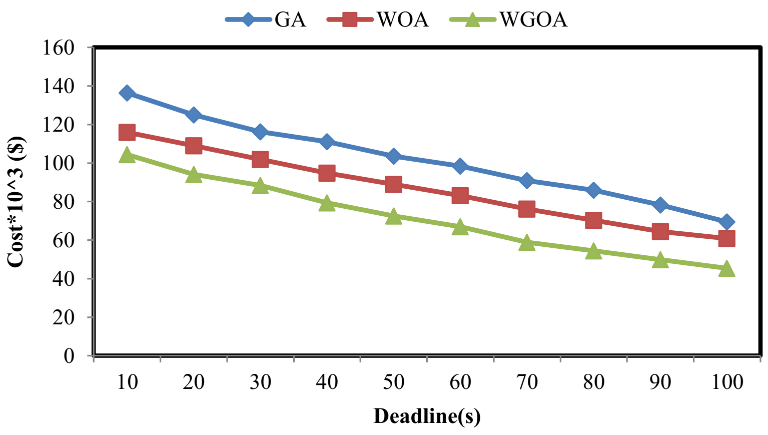


Fig. 25 Cost of uniform distribution (Tasks = 100)

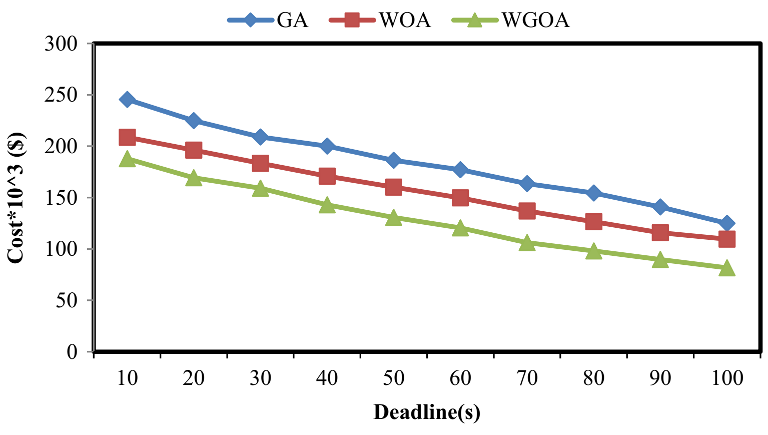


Fig. 26 Cost of normal distribution (Tasks = 300)

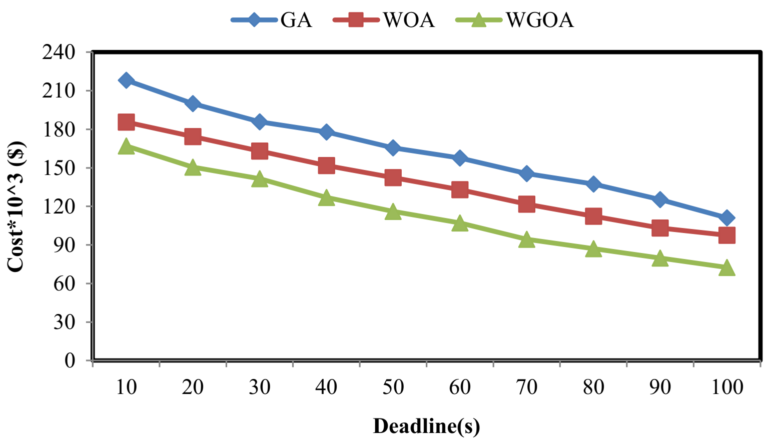


Fig. 27 Cost of HPC2N distribution (Tasks = 200)

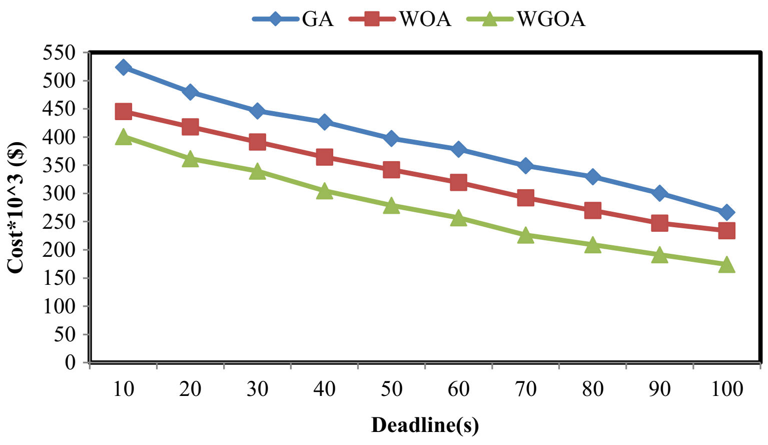


Fig. 28 Cost of HPC2N distribution (Tasks = 500)

Likewise, the maximum cost of WOA is in the range of 115.95–60.90 and 208.70–109.62 and that of GA varies from 136.35 to 69.44 and 245.43 to 124.99.

For HPC2 N distribution the cost statistics are quite similar to the above stated algorithms. For 200 and 500 tasks, the maximum acquired costs are of the scale of 166.92–72.58 and 400.62–174. 18 in the deadline of 10–100, while using WGOA. The maximum cost for WOA is from 185.51 to 97.44 and 445.23–233.86 and for GA, it lies in the scale of 218.16–110.10 and 266.64–525.34–266.64

11 Conclusion

In this paper, we propose a new task scheduling algorithm suited for cloud computing environment called hybrid WGOA. WGOA is a fusion, or in other words, a hybridization of two algorithms namely, Genetic Algorithm and Whale Optimization Algorithm. WGOA ensures the optimization in terms of three conflicting factors, makespan, cost and EAR.

The effectiveness of WGOA has been studied by comparing it with classical WOA and standard GA and then following it with a simulation of three different datasets (Normal, Uniform and HPC2 N). By far, WGOA has excelled in performance in all the three distributions. WGOA can produce much better solutions in comparison with the WOA and GA. From this, it is understood that the proposed method performs scheduling of tasks better than the other methods. From the simulation, the proposed method has been provides near optimal solution and significantly reducing the maksepan, cost and EAR. In future work, we intend to consider other objectives such as energy consumption, security and reliability in addition to the objectives in this work.

References

1. Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation computer systems, 25*(6), 599–616.
2. Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *Journal of internet services and applications, 1,* 7–18.
3. Jennings, B., & Stadler, R. (2015). Resource management in clouds: Survey and research challenges. *Journal of Network and Systems Management, 23*(3), 567–619.
4. Mustafa, S., Nazir, B., Hayat, A., & Madani, S. A. (2015). Resource management in cloud computing: Taxonomy, prospects, and challenges. *Computers & Electrical Engineering, 47,* 186–203.
5. Kalra, M., & Singh, S. (2015). A review of metaheuristic scheduling techniques in cloud computing. *Egyptian Informatics Journal, 16*(3), 275–295.
6. Tsai, J. T., Fang, J. C., & Chou, J. H. (2013). Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. *Computers & Operations Research, 40*(12), 3045–3055.
7. Ali, H. G. E. D. H., Saroit, I. A., & Kotb, A. M. (2017). Grouped tasks scheduling algorithm based on QoS in cloud computing network. *Egyptian Informatics Journal, 18*(1), 11–19.
8. Manvi, S. S., & Shyam, G. K. (2014). Resource management for infrastructure as a service (IaaS) in cloud computing: A survey. *Journal of Network and Computer Applications, 41,* 424–440.
9. Nzanywayingoma, F., & Yang, Y. (2018). Efficient resource management techniques in cloud computing environment: A review and discussion. *International Journal of Computers and Applications, 41*(3), 165–188.
10. Gutierrez-Garcia, J. O., & Sim, K. M. (2012). GA-based cloud resource estimation for agent-based execution of bag-of-tasks applications. *Information Systems Frontiers, 14*(4), 925–951.
11. Niu, S. H., Ong, S. K., & Nee, A. Y. (2013). An improved intelligent water drops algorithm for solving multi-objective job shop scheduling. *Engineering Applications of Artificial Intelligence, 26*(10), 2431–2442.
12. Abdulhamid, S. M., & Latiff, M. S. A. (2014). *League championship algorithm based job scheduling scheme for infrastructure as a service cloud.* Preprint arXiv:1410.2208.
13. Karthikeyan, S., Asokan, P., Nickolas, S., & Page, T. (2015). A hybrid discrete firefly algorithm for solving multi-objective flexible job shop scheduling problems. *International Journal of Bio-Inspired Computation, 7*(6), 386–401.
14. Dorigo, M., & Sttzl, T. (2004). *Ant colony optimization*. Brighton: Bradford Co.
15. Ebrahimi, A., & Khamehchi, E. (2016). Sperm whale algorithm: An effective meta-heuristic algorithm for production optimization problems. *Journal of Natural Gas Science and Engineering, 29,* 211–222.

16. Eswaraprasad, R., & Raja, L. (2017). A review of virtual machine (VM) resource scheduling algorithms in cloud computing environment. *Journal of Statistics and Management Systems, 20*(4), 703–711.
17. Natesan, G., & Chokkalingam, A. (2017). Opposition learning-based grey wolf optimizer algorithm for parallel machine scheduling in cloud environment. *International Journal of Intelligent Engineering and Systems, 10*(1), 186–195.
18. Pradeep, K., & Jacob, T. P. (2017). CGSA scheduler: A multi-objective-based hybrid approach for task scheduling in cloud environment. *Information Security Journal: Global Perspective*, *27*(2), 77–91.
19. Ma, T., Chu, Y., Zhao, L., & Ankhbayar, O. (2014). Resource allocation and scheduling in cloud computing: Policy and algorithm. *IETE Technical Review, 31*(1), 4–16.
20. Zuo, L., Shu, L. E. I., Dong, S., Zhu, C., & Hara, T. (2015). A multi-objective optimization scheduling method based on the ant colony algorithm in cloud computing. *IEEE Access, 3,* 2687–2699.
21. Somasundaram, T. S., & Govindarajan, K. (2014). CLOUDRB: A framework for scheduling and managing high-performance computing (HPC) applications in science cloud. *Future Generation Computer Systems, 34,* 47–65.
22. Zuo, X., Zhang, G., & Tan, W. (2014). Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud. *IEEE Transactions on Automation Science and Engineering, 11*(2), 564–573.
23. Abdullahi, M., & Ngadi, M. A. (2016). Hybrid symbiotic organisms search optimization algorithm for scheduling of tasks on cloud computing environment. *PLoS ONE, 11*(6), e0158229.
24. Latiff, M. S. A., Abdul-Salaam, G., & Madni, S. H. H. (2016). Secure scientific applications scheduling technique for cloud computing environment using global league championship algorithm. *PLoS ONE, 11*(7), e0158102.
25. Elsherbiny, S., Eldaydamony, E., Alrahmawy, M., & Reyad, A. E. (2018). An extended intelligent water drops algorithm for workflow scheduling in cloud computing environment. *Egyptian Informatics Journal*, *19*(1), 33–55.
26. Kumar, N., & Vidyarthi, D. P. (2017). An energy aware cost effective scheduling framework for heterogeneous cluster system. *Future Generation computer systems, 71,* 73–88.
27. Yang, J., Jiang, B., Lv, Z., & Choo, K. K. R. (2017). A task scheduling algorithm considering game theory designed for energy management in cloud computing. *Future Generation computer systems*.
28. Li, K. (2017). Scheduling parallel tasks with energy and time constraints on multiple manycore processors in a cloud computing environment. *Future Generation computer systems*.
29. Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95,* 51–67.

Gobalakrishnan Natesan pursued his Bachelor's degree in Information Technology at Anna University, Tamilnadu, India in 2005. He then obtained his Master's degree in Software Engineering from Bharathidhasan University, Tamilnadu, India in 2008. Currently, he is a research scholar in Sathyabama University, Chennai, India and working as Assistant Professor in the Department of Information Technology, St. Joseph's College of Engineering, Tamilnadu, India. His current research interests are Cloud computing, Virtualization and Big Data.

Dr. Arun Chokkalingam received his Ph.D. from Anna University, Chennai, India. Currently, he is working as a Professor in the Department of Electronics and Communication Engineering, R.M.K College of Engineering & Technology, Tamilnadu, India. He has more than 15 years of teaching experience in various Engineering Colleges. He has published research papers in various National, International Conferences and Journals. His current research interests are Cloud Computing, Image Processing, VLSI and Optimization Techniques. He is a life time member of ISTE.