A Hybrid Bio-Inspired Algorithm for Scheduling and Resource Management in Cloud Environment

Shridhar G. Domanal, *Student Member, IEEE*,
Ram Mohana Reddy Guddeti, *Senior Member, IEEE*, and Rajkumar Buyya, *Fellow, IEEE*

Abstract—In this paper, we propose a novel HYBRID Bio-Inspired algorithm for task scheduling and resource management, since it plays an important role in the cloud computing environment. Conventional scheduling algorithms such as Round Robin, First Come First Serve, Ant Colony Optimization etc. have been widely used in many cloud computing systems. Cloud receives clients tasks in a rapid rate and allocation of resources to these tasks should be handled in an intelligent manner. In this proposed work, we allocate the tasks to the virtual machines in an efficient manner using Modified Particle Swarm Optimization algorithm and then allocation / management of resources (CPU and Memory), as demanded by the tasks, is handled by proposed HYBRID Bio-Inspired algorithm (Modified PSO + Modified CSO). Experimental results demonstrate that our proposed HYBRID algorithm outperforms peer research and benchmark algorithms (ACO, MPSO, CSO, RR and Exact algorithm based on branch-and-bound technique) in terms of efficient utilization of the cloud resources, improved reliability and reduced average response time.

Index Terms—Bio-inspired algorithms, load balancing, resource management, task scheduling, virtual machines

1 Introduction

Cloud computing is the emerging technology in distributed environment consisting of several data centers, servers, virtual machines, load balancers etc. which are connected intelligently. Further, the cloud deals with many things such as storing and retrieving of documents, sharing of multimedia, lending the related resources on pay-as-you-go model and much more [1], [2]. Even though there is much advancement in the era of computers and Internet of Things (IoT) with respect to responsiveness, reliability and flexibility, still there is a room for improvement in scheduling, optimal resource allocation and management algorithms since these algorithms come under NP-hard and NP-complete complexity classes. Hence, there is a need to address these set of challenging problems using different techniques. Efficient task scheduling and resource management is a challenging problem of distributed computing but it is still in its infant stage in spite of exhaustive research in recent years [1].

In a cloud environment, there are different service models such as SaaS, IaaS, PaaS and XaaS where X refers to anything as a service (Ex: Security as a Service). All these service models need to satisfy the incoming tasks of the clients within the service level agreements (SLA). To execute these tasks, we need to deploy Virtual Machines (VMs) in the cloud and these VMs play a key role in serving the incoming client tasks [3], [4]. Since cloud is a pay-as-you-go model hence several VMs can be created and destroyed inside the physical machines (PMs).

Recently, Amazon developed a novel Elastic cloud based solution in which all the components (PMs, VMs, load balancers etc.) shrink when there is a less load, and expand when there is an increase in the load (here load refers to the incoming tasks for the cloud) [5], [6]. Thus, this elastic cloud resolves the problem of allocating the resources depending on the load. But the efficiency of the VMs depends on the scheduling and load balancing techniques rather than allocation of resources dynamically for its execution. Even though the property of elasticity improves the performance of the cloud, but this service has its own limitations with respect to heterogeneity. For example, scaling of cloud resources with the different system configurations needs to be considered for heterogeneous environment. If the cloud has to work seamlessly, then the load balancer has to play a vital role in satisfying the above mentioned services given by any cloud [7]. In this work, we mainly focus on the Bio-Inspired algorithms for solving the challenging issues of task scheduling, resource allocation and management near optimally. Our approach follows the process of hot-

- *S.G. Domanal is with the Department of Information Technology, National Institute of Technology Karnataka, Surathkal, Mangalore, Karnataka 575025, India. E-mail: shridhar.domanal@gmail.com.*
- *R.M.R. Guddeti is with the Department of Information Technology and Central Computer Center, National Institute of Technology Karnataka, Surathkal, Mangalore, Karnataka 575 025, India. E-mail: profgrmreddy@gmail.com.*
- *R. Buyya is with the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, Department of Computing and Information Systems, University of Melbourne, Vic 3010, Australia. E-mail: rbuyya@unimelb.edu.au.*

Manuscript received 10 Dec. 2016; accepted 1 Mar. 2017. Date of publication 8 Mar. 2017; date of current version 12 Feb. 2020.
Digital Object Identifier no. 10.1109/TSC.2017.2679738

plugging in which the resource units such as Memory and CPU will be added and removed from the VMs without interrupting the current state of the VMs.

The core idea of our proposed methodology lies in achieving the following two goals:

- *Efficient Task Scheduling for Load Balancing:* By designing a novel Modified Particle Swarm Optimization algorithm, the tasks can be efficiently scheduled with balanced load on VMs.
- *Dynamic Resource Allocation and Management*: By designing a novel HYBRID Bio-Inspired algorithm using Modified Particle Swarm Optimization (MPSO) and Modified Cat Swarm Optimization (MCSO) techniques, the cloud resources can be efficiently allocated to execute the clients' tasks.

There is a significant tradeoff between the aforementioned goals in the context of efficient utilization of VMs and managing the resources allocated to each task so that the resources can be utilized efficiently. Scheduling and load balancing play an important role in allocating the incoming tasks and managing the resources dynamically as demanded by tasks respectively in the cloud environment. In the present scenario of cloud, efficient utilization of VMs and effective utilization of the resources (CPU and Memory) are very important. These resources must be used in an intelligent manner to attain high throughput and low response time for execution of incoming tasks which are converging from all over the world [8], [9].

Nowadays, many clients demand for several resources (CPU, Memory etc.) on pay-as-you-go model and satisfying these types of tasks is very tedious and needs proficiency in managing these tasks from IaaS platform. Since, the efficiency of the cloud should not be hindered, hence throughput of the cloud should be directly proportional to the efficient utilization of cloud resources. Special focus is needed when these tasks change their demand for resources which are uncertain and handling the resources demand would be a great challenge in the cloud environment.

Peer research algorithms such as Round Robin, First Come First Serve, and Throttled [10], [11] etc. have been used for task scheduling and resource management. These algorithms are rule based techniques and suffer from underutilization of VMs, resources; resulting in overall delay in executing the tasks. Further, Bio-Inspired techniques such as Particle Swarm Optimization (PSO) [12], Ant Colony Optimization (ACO) [13], Cat Swarm Optimization (CSO) [14], Honey Bee etc. [15] are much suitable for solving the Scheduling, Resource Allocation and Management problems which come under NP-hard/NP-complete complex classes. Hence, by applying Bio-Inspired techniques, we would enhance the efficiency in terms of reliability, flexibility and time (response and execution).

In this paper, we mainly focus on scheduling the tasks using MPSO and allocation & managing the cloud resources using MPSO, MCSO and HYBRID Bio-Inspired (MPSO+MCSO) algorithms. Our proposed MPSO is more efficient for task scheduling when compared to other proposed algorithms. Our proposed HYBRID Bio-Inspired algorithm (MPSO+MCSO) outperforms peer research algorithms in terms of reduced execution time and average response time with efficient utilization of cloud resources.

Hence, our main/key research contributions in this paper are as follows:

1) A novel Bio-Inspired load balancing algorithm is designed for efficient scheduling of the tasks using MPSO technique.
2) An efficient dynamic resource allocation and management approach is designed using both MPSO and MCSO techniques.
3) A combined HYBRID (MPSO+MCSO) heuristic approach is proposed for managing the cloud resources efficiently.
4) Performance evaluation of proposed HYBRID approach with benchmark exact algorithm along with statistical hypothesis analysis.

The remainder of the paper is organized as follows. Section 2 begins with related work on task scheduling and resource management, Section 3 deals with proposed methodology and HYBRID heuristic algorithms. Section 4 describes the performance evaluation and simulation scenarios followed by experimental results & analysis and finally we conclude with future directions in Section 5.

2 Related Work

In this section, we briefly summarize the state of the art scheduling and resource management algorithms available in the literature [16], [17], [18], [19], [20]. Many researchers have contributed towards efficient algorithms for load balancing, scheduling and resource management, but still there is a scope for further improvement since the aforementioned algorithms are NP-hard/NP complete complex class problems. Al Nuaimi et al. [10] presented efficient algorithms for assigning the clients' tasks to the nodes or virtual machines in the cloud environment. This approach aims to enhance the overall performance of the cloud.

Gougarzi et al. [21] proposed a resource allocation problem that aims to minimize the total energy cost of cloud computing system while meeting the specified client-level SLAs in a probabilistic sense. Here, authors applied a reverse approach to put a penalty if the client does not meet the SLA agreements. Authors implemented a heuristic algorithm to solve the aforementioned resource allocation problem.

Radojević and Žagar [11] introduced Central Load Balancing Decision Model in the cloud environment which automates the complete scheduling process and reduces the role of human administrator. It lacks in finding the node capabilities, configuration details and complete system has no backup thus resulting in single point of failure.

Nishant et al. [13] proposed Ant Colony Optimization technique for Scheduling the incoming tasks effi-ciently by utilizing the under-loaded nodes in the cloud environment. After a few iterations, this ACO algorithm slows down and fails in changing the node status for scheduling the future tasks. Shojaee et al. [14] proposed New Cat Swarm Optimization for scheduling the tasks in the distributed environment with better performance of task allocation. Here, authors focus was on both seeking and tracing modes of CSO and this leads to delay in execution of tasks. Jeyarani et al. [17] proposed Self Adaptive MPSO technnique for efficient scheduling of the VMs in the homogeneous cloud environment.

Domanal and Reddy [22], [23] proposed a modified throttled algorithm that efficiently schedules the incoming client tasks to the virtual machines. The authors focus was on the response time of the tasks. Later, the same authors proposed another VM-Assign load balancing algorithm which focuses not only on the response time, but also on the efficient utilization of VMs present in the cloud. In both algorithms, scheduling of incoming tasks to the VMs are addressed efficiently, but the authors did not consider the resources such as CPU and Memory which are inevitably demanded by clients tasks across the world.

Gonçalves et al. [24] proposed a Distributed Cloud Resource Allocation System (D-CRAS) which ensures an automatic monitoring and control of resources of the cloud to guarantee the optimal functioning while meeting the SLA requirements, but authors did not compare their work with State-of-the-art technologies. Gonçalves et al. [24] presented an Heterogeneous Earliest Finish Time (HEFT) based HYBRID scheduling algorithm that decides which resources should be taken on lease from public cloud to accomplish the execution of a task within its deadline and with minimum cost for the user. Later, the same authors modified their algorithm with sub-deadlines for resource provisioning and compared with greedy approach and min-min algorithms.

Zhou et al. [25] implemented a resource allocation policy for load balancing in virtual machine cluster. This resource allocation policy not only monitors the real-time resource utilization of CPU, Memory etc. but also uses instant resource reallocation for VMs running on the PM using VM migration.

Zuo et al. [12] proposed a resource allocation framework in which an IaaS provider can outsource its tasks to External Clouds (ECs) when its own resources are insufficient to meet the current tasks demand. There is no guarantee that ECs could be used all the time, but the key issue is how to allocate users' tasks to maximize the profit of IaaS provider while guaranteeing the Quality of Service (QoS). Authors formulated this problem as an Integer Programming (IP) model, and it is solved by a Self-Adaptive Learning Particle Swarm Optimization (SLMPSO)-based scheduling approach. Authors did not highlight the overhead caused during the communication between IaaS providers and the external cloud. These aforementioned issues motivated us to propose novel HYBRID Bio-Inspired technique in this paper.

In this proposed work, we focus on scheduling of in-coming tasks over the VMs (here the task refers to execu-tion of a task with resource demand) using Bio-Inspired MPSO technique. Thus achieving better efficiency with respect to optimal allocation tasks to the VMs with better average response time and balanced load. Further, we consider the resource demands in terms of CPU & Memory and managing of these resources are taken care by our proposed Bio-Inspired techniques such as MPSO, MCSO and HYBRID (MPSO+MCSO). For resource management, we aim to achieve better Average Response time and less communication overhead between cloud resource pool and VMs. If the VMs have sufficient resources to execute the tasks then it proceeds, otherwise it lends from the cloud resource pool. In this work, the main investigation is how Bio-Inspired techniques play a major role in assigning and managing these resources efficiently. Table 1 summarizes some important existing works.

TABLE 1
Summary of Existing Works

Authors	Methodology	Remarks
Huang et al. [26]	Job Scheduling in distributed cloud using Dominant Resource Fairness (DRF)	Efficient scheduling is done without load balancing of VMs
Zheng and Wang [27]	Task Scheduling and Resource Allocation using Pareto based fruit Fly Optimization algorithm (PFOA)	Authors did not consider other heuristic approaches for performance evaluation
Rasti-Barzoki and Hejazi [28]	Distributed Scheduling and Resource Allocation using Pseudo-Polynomial Dynamic Programming Algorithm	Authors considered supply chain management application. However, heuristics approaches can be added to suit multiple platforms
Akbari and Rashidi [29]	Task Scheduling in Heterogeneous Cloud Environment using Multi-Objective Scheduling Cuckoo Optimization Algorithm (MOSCOA)	Random assignment of tasks to processors during scheduling
Pillai and Rao [30]	Resource Allocation mechanism for machines in cloud based on the uncertainty principle of game theory	No comparison with other state-of-the-art algorithms
Xu et al. [31]	Energy-aware Resource Allocation method for workflow executions	Authors did not consider Resource-aware scheduling
Wang et al. [32]	Load Balancing Min-Min (LBMM) for load balancing in cloud environment	Execution time of the tasks is not considered and leads to bottleneck for task scheduling
Xu and Li [33]	Stable matching framework based Resource management for mapping virtual machines to physical servers	Authors did not consider dynamic resources demands and job scheduling
Tsai et al. [1]	Hyper Heuristics Scheduling Algorithm in cloud environment	Heterogeneous environment is not considered for scheduling

3 Proposed Methodology

We proposed improved MPSO for task scheduling. We also proposed MPSO, MCSO and HYBRID (MPSO+MCSO) techniques for resource allocation and management.

3.0.1 Modified Particle Swarm Optimization

Particle Swarm Optimization was developed by Eberhart and Kennedy [34] in 1995 and it has been widely used stochastic optimization technique based on the behavior of animals and birds. In MPSO, the particle is represented by its position and velocity; these particles keep track of local best (LB) and global best (GB) values; fitness function determines the LB and GB values. In the scheduling approach, particles refer to VMs; LB refers to under-loaded VM from each cluster and GB refers to the minimum value among all LB values. The algorithm iterates continuously to get the new LB and GB values. In MPSO, GB does not remain the same in each iteration when compared to PSO. The position and velocity of a particle are updated based on the following Equations (1) and (2) respectively [35]

$$x(t+1) = x(t) + v(t) \tag{1}$$

$$x(t+1) = v(t) + c_1 r_1 (LB - x(t)) + c_2 r_2 (GB - x(t)), \tag{2}$$

where,
x(t) is current position of particle / Current load of a VM.
LB is least under loaded VM from cluster.
GB is least under loaded VM from all LB values.
c_1 and c_2 acceleration coefficients, usually $c_1 = c_2 = 2$.
r_1 and r_2 are random numbers between (0, 1).

3.0.2 Modified Cat Swarm Optimization

CSO technique [36] deals with two important phases like Seeking mode and Tracing mode. In the proposed MCSO, we mainly concentrate on Seeking mode rather than on Tracing mode as it is similar to the MPSO approach. MCSO is used for resource allocation management based on Seeking mode only. However, MCSO cannot be efficiently used for scheduling if it uses both Seeking and Tracing modes. MCSO deals with four different types of memories like Seeking Memory Pool (SMP), Seeking Range of selected Dimension (SRD), Counts to Dimension Change (CDC) and Self Position Consideration (SPC). These memory pools play an important role in assigning the cloud resources to VMs in the current work. The usage details of these memory pools are given in the Section 3.1.3 along with the proposed MCSO algorithm. The main difference between MPSO and MCSO is that the seeking mode of MCSO overcomes the limitations of the MPSO by comparing the future resource demands with excess resources present in the VMs. The complete details are given in Section 4.2.2.

3.0.3 Example for Explaining Our Proposed Work

In this proposed model, tasks are scheduled on the VMs and resources such as CPU & Memory are also utilized efficiently by our proposed algorithms. Here, we used different types of VMs on a PM which can communicate with the scheduler for the efficient functioning of proposed algorithms. The tasks are coming at a batch of ten and their inter arrival time remains constant. Initially, tasks are scheduled efficiently on VMs and then allocation of required resources will take place.

Fig. 1 shows an example for explaining our proposed work. The incoming task demands *n* number of cloud resources where *n = 0, 1, 2, 3, 4, 5, 6, 7, 8, 9* and allocation of *n* resources are provided by either cloud resource pool or by the VMs. Each cluster contains different VMs with the best two *n* resource values represented as *excess_res1* and *excess_res2*, respectively and the remaining resources available with VMs are stored in *excess_res3[]*. During resource allocation, the on demand resources are compared with *excess_res1*, *excess_res2* and *excess_res3[]*, respectively by our proposed MPSO, MCSO and HYBRID (MPSO+MCSO) algorithms. The *excess_res1* and *excess_res2* mappings are used by the MPSO algorithm where as *excess_res3[]* mapping is used by the MCSO algorithm. On the other hand, HYBRID (MPSO+MCSO) algorithm uses all the three aforementioned resource mappings.

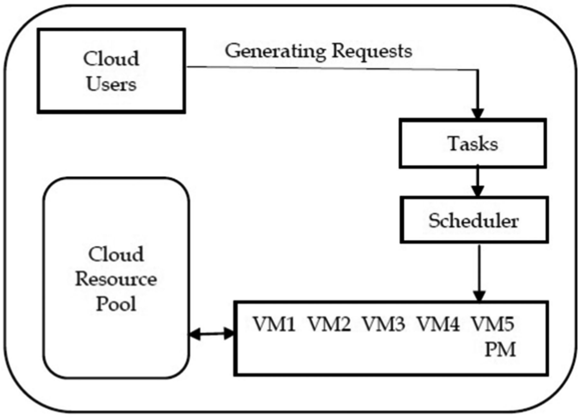


Fig. 1. Example for explaining our proposed work.

3.0.4 Branch-and-Bound Based Exact Algorithm

To check the performance, proposed algorithms are compared with bench mark solution based on Branch-and-Bound based Exact algorithm [37]. The exact algorithm gives the global optimum solution for efficient utilization of cloud resources. Hence, we compared our proposed algorithms with Exact algorithm, and details of the Exact algorithm are as follows. Let *'N'* be the node at level *'l'* of the search tree along with *'lb'* as lower bound and *'ub'* as upper bound. The root node N0 corresponds to an empty solution and each node at level $l \geq 1$ corresponds to the partial solution. Initially at root level, all VMs are not allocated and child node is created by comparing the MIPS of each VMs along with resource demands. Further, the child node N^+ is created only if the following conditions are met.

- MIPS(VMs) $\leq$ MIPS(task)
- Resources(VMs) $\leq$ Resource Demand(task)
- Resources(VMs) $\neq$ Resource Demand(task)

Hence, if the above conditions are met, then *'lb'* is computed. Later, if it reaches *'ub'*, then child node is pruned and the best fit VMs are chosen for the scheduling, resource allocation and thus execution of tasks is carried out.

3.1 Proposed Model

In the proposed work, incoming tasks, i.e., $\{x_1, x_2, x_3, \ldots, x_n\}$ have to be scheduled on the virtual machines such as $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$. Further, cloud resources demanded by these requests are intelligently handled by proposed algorithms. Hence our objectives are as follows.

Objective 1. Efficient utilization of VMs.

$$\begin{aligned} &VM\ \ Type(i) = Number\ of\ tasks \\ &i.e.\ x_1, x_2, x_3, \ldots, x_n\ handled\ by\ VMs. \end{aligned} \tag{3}$$

VM Type(i) refers to type of VM used in the experiment. We have used five different types of VMs (small, Medium, Large, X-Large, Extra Large) and more details are given in Table 4 of Section 4.

TABLE 2
Notations and Definitions

Notations	Definitions
$VMType_{(i)}$	Type of VM on which the task is being executed
RD	Number of resources demanded by tasks
R	Resources (CPU and Memory)
t_1	Time at which task enters the VM
t_2	Time at which task completes the execution
excess_res1	First highest remaining resource from the cluster
excess_res2	Second highest resource from the cluster
Needed_res[]	Number of resource demands from the request
excess_res3[]	Remaining resources other than best excess_res1 & excess_res2

Objective 2. Reducing the Average Response.

Average Response Time (*AR_Time*) is the total amount of time taken for responding to a task and *AR_Time* is calculated by the following

$$AR\_Time = t_2\left[\sum_{x=1}^{n} VM\ \ Type(i)\right] - t_1\left[\sum_{x=1}^{n} VM\ \ Type(i)\right] \quad (4)$$

$$FV = \frac{(R)\sum_{vm=0}^{k} VM\ \ Type(i)}{\sum_{x=1}^{n}(RD)}. \quad (5)$$

For resource allocation and management, the fitness value (FV) is given by the Equation (5). Here, at each iteration, we find the FV and then choose the suitable VMs for assigning the task for the execution. If FV value is feasible then it takes unused resources from the VMs otherwise it assigns the resources from the cloud resource pool as explained in the example. Table 2 gives the notations and definitions used in the proposed methodology.

3.1.1 Scheduling of VMs Using MPSO Algorithm

Here, we use a MPSO technique for scheduling the VMs against incoming tasks. Cloud receives tasks in a rapid rate from the outside world, assigning and executing these tasks is a challenging issue.

Number of incoming requests/tasks, i.e., $\{x_1, x_2, x_3, \ldots, x_n\}$ have to be scheduled on VMs, i.e., $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$. Here, our proposed MPSO algorithm is deployed for scheduling the tasks in a balanced way. The MPSO algorithm plays an important role in assigning the incoming tasks to the VMs as efficiently as possible. We experimented this work for a private cloud which receives the tasks in a batch of ten (can be extended) and these tasks are assigned to the VMs. Clustering depends on the number of VMs taken for experimentation. Algorithm 1 gives the complete details of task scheduling by MPSO.

In every iteration, each cluster will identify the least loaded VM referred to as local best (LB_z) and the smallest among these VMs referred to as global best. The next task is allocated to the VM that is associated with GB. If the GB remains the same in the subsequent iteration, then GB is updated with second least LB_z from the cluster list C_z. The same process is continued until all the tasks are executed. The time complexity of this algorithm is *0(n.z)*. Since z is a constant hence the time complexity of MPSO algorithm will be *0(n)* in polynomial time. The flow diagram of the proposed MPSO algorithm is shown in Fig. 2.

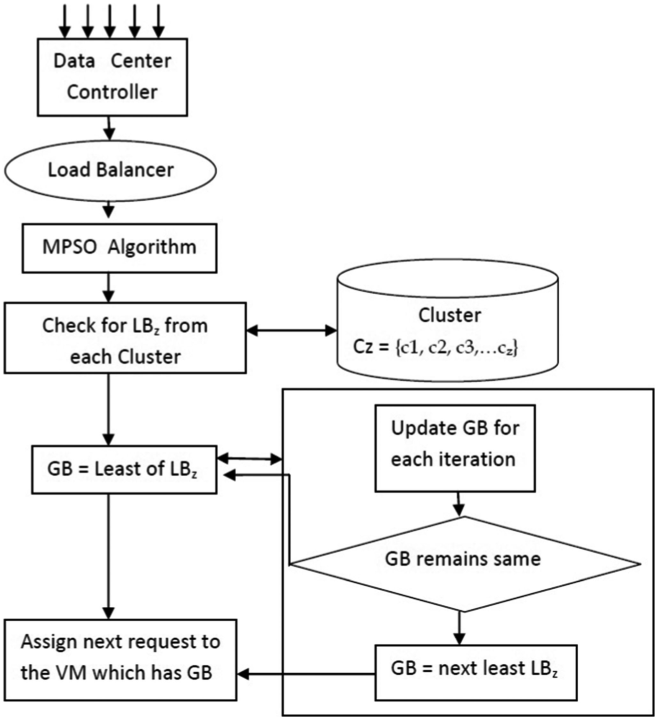


Fig. 2. Flow diagram of MPSO based scheduler.

Algorithm 1. Task Scheduling Using MPSO

```
0:  Initialisation: $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$ count = 0
                    Local Best ($LB_z$) = 0
                    Global Best (GB) = 0
                    say VMs = $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$
                    say Clusters, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
                    Cluster size = k/$C_z$
1:  for all incoming requests $\{x_1, x_2, x_3, \ldots, x_n\}$
2:      each cluster $C_z$ = least loaded VM
3:      Assign each one of them as $LB_z$ from $C_z$
4:  end for
5:  Assign GB = least $LB_z$
6:  Next task allocated to VM which contains GB
7:  if (Next allocation == last used GB) then skip
8:      goto step 2 for next least $LB_z$
9:  else
10:     goto step 6
```

3.1.2 Resource Allocation and Management Using Modified Particle Swarm Optimization Algorithm

Tasks demand for dynamic resources at a rapid rate in the cloud environment and satisfying these demands is a challenging task. VMs must have enough resources to execute clients' tasks and providing these resources to VMs is managed by the proposed MPSO. Let us assume that there is a cloud resource pool *(Res_pool)* which provides the resources demanded by the tasks and it acts as a resource repository. Each of the VMs has at least two minimum resources

(CPU and Memory) for executing a task. In the first iteration, for all VMs, resources are given by cloud resource pool and from the second iteration onwards it depends on the resources as demanded by the tasks. Each of the VMs will not use all the resources for execution of tasks and these unused excess resources which remain with the VMs can be utilized for the future tasks demands. Our proposed MPSO algorithm plays an important role in assigning these unused resources either from the VMs or from the cloud resource pool.

Algorithm 2. Resource Allocation and Management Using MPSO Algorithm

```
Initialisation: Res_pool, Needed_res, min_res = 2
        sum_res = 0
        say VMs = $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$
        say Clusters, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
        Cluster size = k/$C_z$
for all incoming requests $\{x_1, x_2, x_3, \ldots, x_n\}$
    Res_demand ← sum of resources from requests
    Needed_res ← resource demand for each request
    for all available VMs
        sum_res ← sum_res + Res_demand
    end for
    if (iteration 1)
        Res_pool ← Res_pool - sum_res
    end if
    else
        for all Cluster size, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
            excess_res1 ← first best of $c_z$
            excess_res2 ← second best of $c_z$
        end for
        for all available VMs $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$
            for all Cluster size, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
                if (excess_res1 == Needed_res[])
                    VM executes using excess_res1
                else
                    Res_pool ← Res_pool − Needed_res[]
                if (excess_res2 == Needed_res[])
                    VM executes using excess_res2
                else
                    Res_pool ← Res_pool − Needed_res[]
            end for
        end for
        Res_pool ← Res_pool + remaining Needed_res[]
end for
```

For better efficiency, let us form a clusters $\{c_1, c_2, c_3, \ldots, c_z\}$ from the VMs $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$ which operate in both sequential and parallel modes. For parallel mode execution, we need at least two clusters. For subsequent assignment of resources to the VMs, let us take the best two excess resources, i.e., *excess_res1* and *excess_res2* (VMs which have unused extra resources) from each cluster and check for match with the next resource demands. If the next resource demands match with the best values (unused resources), then VMs start executing the task immediately, or else the resources are taken from the resource pool for the execution. After each iteration, if there are any unmatched/unused resources then these resources can be released to the resource pool. Once VMs complete the execution of a task, then the minimum resources available with VMs can be released to the cloud resource pool. The main focus is on utilizing the resources from the VMs rather than lending the resources from the resource pool. In doing so, we dynamically exploit the resources and thus communication overhead between the cloud resource pool and VM is reduced significantly. Algorithm 2 describes the resource allocation and management using the proposed MPSO technique.

Usually in MPSO algorithm, the local and global best may result in minimum or maximum values and further it is application specific. In the Algorithm 1, we took only one best (minimum: load on VM) value from each cluster; but in the Algorithm 2, our main objective is to choose the best (maximum) two values and thus resulting in more optimization if it matches the subsequent resource demands from the tasks.

Even though Algorithm 2 improves the allocation of resources intelligently but it has the following limitations

- *excess_res1 and excess_res2 may not match exactly with subsequent future resource demands.*
- *Increase in the communications overhead between VMs and cloud resource pool.*

3.1.3 Resource Allocation and Management Using Modified Cat Swarm Optimization Algorithm

The aforementioned limitations of Algorithm 2 can be addressed by using our proposed MCSO algorithm (Algorithm 3). Cats always remain calm and move slowly and this behavior of cats is referred to as seeking mode. When the presence of prey (resource match happens) is sensed, then cats chase it with high speed, and this behavior is represented by the tracing mode. The tracing mode acts similar to that of MPSO algorithm, but seeking mode awaits the opportunity to capture a prey. The details of the Algorithm 3 are as follows.

In the proposed MCSO algorithm, we mainly concentrate on seeking mode rather than tracing mode. In Algorithm 2, it matches only with the *excess_res1* and *excess_res2* values from each cluster, but the remaining excessive resources from each cluster are not considered for the assignment and this issue is addressed in the proposed MCSO algorithm as shown in Algorithm 3.

The seeking mode of MCSO passes through four different types of memories. The excess resources available with VMs other than *excess_res1* and *excess_res2* are stored in the seeking memory pool. Using SMP, the cats await the opportunity in order to find the exact match with the future resource demands from the tasks. If there is a match, then status is stored in the seeking range of selected dimension. If SRD has a new update, then VMs start executing the task and this status is referred to as counts to dimension change. Cats position is changing in every update of seeking mode of MCSO and it is stored in self position consideration. The tracing mode operation is applied to the outcome of seeking mode and the procedure is same as that of Algorithm 2. The limitations of Algorithm 2 can be overcome by seeking mode of MCSO. There might be a situation in which all remaining excessive resources (other than *excess_res1* and *excess_res2*) may match with future resource demands; hence, time to lend the resources from the resource pool may be reduced considerably.

Algorithm 3. Resource Allocation and Management Using MCSO Algorithm (Seeking Mode only)

```
Initialisation: Res_pool, Needed_res, min_res = 2
                sum_res = 0
                say VMs = $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$
                say Clusters, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
                Cluster size = k/$C_z$
for all incoming requests $\{x_1, x_2, x_3, \ldots, x_n\}$
  Res_demand ← sum of resources from requests
  Needed_res[] ← resource demand for each request
  for all available VMs
    sum_res ← sum_res + Res_demand
  end for
  if (iteration 1)
    Res_pool ← Res_pool − sum_res
  end if
  else
    for all Cluster size, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
      excess_res3[] ←  rest of first and second best from
                         each cluster
    end for
    for all available VMs $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$
      for all Cluster size, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
      while (size of(excess_res3[])
        if (excess_res3[] == Needed_res[])
          VM executes using excess_res3[]
        else
          Res_pool ← Res_pool − Needed_res[]
      end while
      end for
    end for
  Res_pool ← Res_pool + remaining Needed_res[]
end for
```

Thus, the MCSO algorithm overcomes the aforementioned limitations of the MPSO algorithm. The seeking mode matching ratio of MCSO is greater than the best two, i.e., excess_res1 and excess_res2 matching policies of the MPSO and thus MCSO will improve the dynamic allocation and management of cloud resources. However, the MCSO algorithm has the following limitations.

- *The values of excess_res3[] may not (rare case) match with the subsequent resource demands.*
- *If the above condition is true, then the algorithm will be slower and communication overhead between VM and cloud resource pool will be increased.*

3.1.4 Resource Allocation and Management Using HYBRID (MPSO+MCSO) Algorithm

The limitations of the MPSO and MCSO algorithms can be overcome by our proposed HYBRID (MPSO+MCSO) Bio-Inspired algorithm which combines the merits of both MPSO and MCSO techniques. Thus HYBRID approach provides a better efficiency in terms of allocation of resources with reduced total execution time. Further, communication overhead between VMs and resource pool is marginally decreased. In MPSO and MCSO, we compare only exact matches of *excess_res1*, *excess_res2* with the *Needed_res[]* from future resource demands. In the worst-case, HYBRID approach may not work efficiently and degrades the performance of the resource allocation and thus the system will respond with high delay. To overcome this situation, we consider the *excess_res3[]* which contains the remaining resources other than *excess_res1* and *excess_res2*. The complete flow of HYBRID (MPSO+MCSO) technique is given in Algorithm 4.

Algorithm 4. Resource Allocation and Management Using HYBRID (MPSO+MCSO) Bio-Inspired Algorithm

```
 0: Initialisation: Res_pool, Needed_res, min_res = 2
                    sum_res = 0
                    say VMs = $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$
                    say Clusters, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
                    Cluster size = k/$C_z$
 1: for all incoming requests $\{x_1, x_2, x_3, \ldots, x_n\}$
 2:   Res_demand ← sum of resources from requests
 3:   Needed_res ← resource demand for each request
 4:   for all available VMs
 5:     sum_res ← sum_res + Res_demand
 6:   end for
 7:   if (iteration 1)
 8:     Res_pool ← Res_pool − sum_res
 9:   end if
10:   else
11:     for all Cluster size, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
12:       excess_res1 ← first best of $c_z$
13:       excess_res2 ← second best of $c_z$
14:       excess_res3[] ← rest of first and second best of $c_z$
15:     end for
16:     for all available VMs $\{vm_0, vm_1, vm_2, \ldots, vm_k\}$
17:       for all Cluster size, $C_z = \{c_1, c_2, c_3, \ldots, c_z\}$
18:       if (excess_res1 ≥ Needed_res[])
19:         VM executes using excess_res1
20:       else
21:         Res_pool ← Res_pool − Needed_res[]
22:       if (excess_res2 ≥ Needed_res[])
23:         VM executes using excess_res2
24:       else
25:         Res_pool ← Res_pool − Needed_res[]
26:       while (size 0f(excess_res3[]))
27:       if (excess_res3[] ≥ Needed_res[])
28:         VM executes using excess_res3[]
29:       else
30:         Res_pool ← Res_pool − Needed_res[]
31:       end while
32:     end for
33:   end for
34:   Res_pool ← Res_pool + remaining Needed_res[]
35: end for
```

In the proposed HYBRID approach, we modify the condition by checking the upper bounds of *excess_res1*, *excess_res2* and *excess_res3[]* of Algorithms 2 and 3. In doing so, we can decide how many resources from *excess_res1*, *excess_res2* and *excess_res3[]* should be given to the forthcoming task and the remaining (unmatched) extra resources can be returned to cloud *Res_pool*. Thus, our proposed HYBRID algorithm outperforms MPSO and MCSO algorithms when considered individually. Algorithm 4 behaves intelligently from line numbers 12 to 30 as it combines the features of both Algorithms 2 and 3 with optimum modifications. We need to apply both approaches simultaneously, so that the line numbers 18-25 and 26-29 execute in parallel using Python threads during the execution (applying both

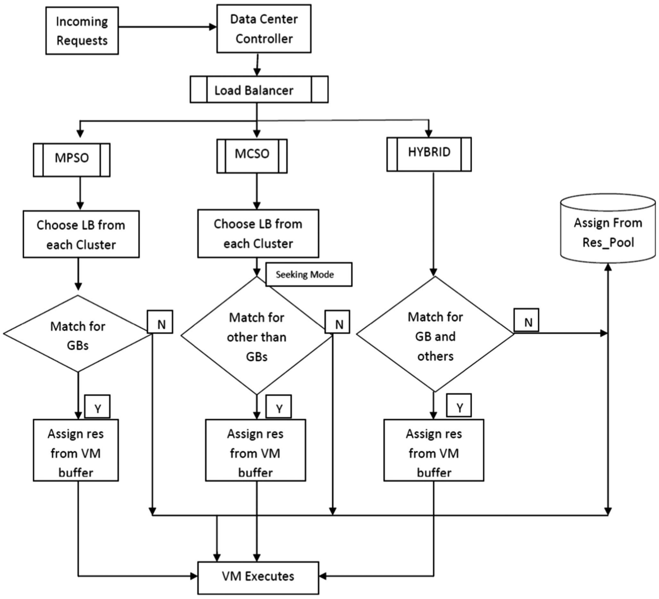


Fig. 3. Overall flow diagram of proposed algorithms.

MPSO and MCSO). Hence, the matching possibility of *excess_res3[]* with future resource demands is more than that of MPSO and MCSO algorithms when taken separately. The results and analysis of all proposed algorithms are discussed in Section 4. Fig. 3 gives the overall flow diagram of MPSO, MCSO and HYBRID (MPSO+MCSO) approaches.

4 Performance Evaluation

4.1 Experimental Setup

The proposed work is experimented on a simulator which gives the real-time scenario of cloud environment. The complete simulation scenario is written in Python language and here after it is named as Python Simulator, in short, PySim. Fig. 4 shows the block diagram of PySim, which is used for experimenting the proposed Bio-Inspired algorithms. The experimental setup consists of four physical machines which are interconnected with different configurations among them.

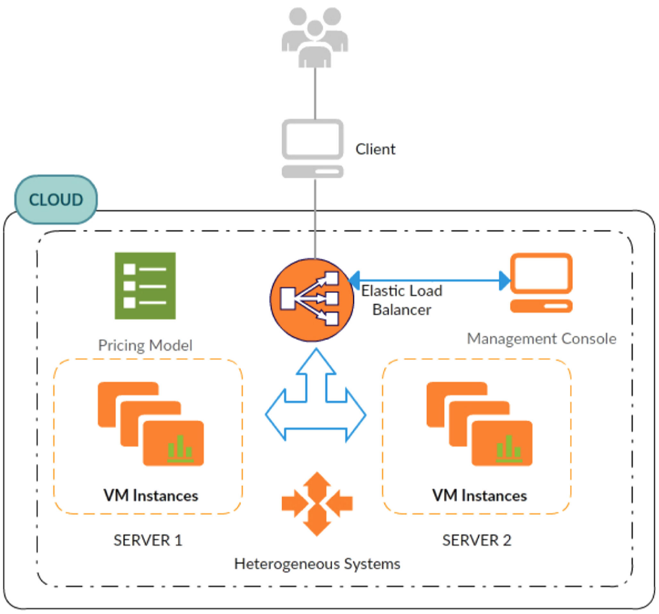


Fig. 4. Block diagram of PySim.

TABLE 3
Configuration Details of PySim

Machine	GHz	RAM (GB)	Storage (GB)
Client Machine	2.80	4	100
Load Balancer	3.40	8	100
Server 1	3.40	16	200
Server 2	3.40	16	200

The configuration details of customized simulation setup are given in Table 3 and it consists of four different machines in which one machine act as a task sender (Client Machine), one machine acts as a load balancer and other two machines act as servers with several VMs which execute the incoming tasks. Here, the task refers to executing a job which demands several resources. We considered CPU and Memory as two types of resources which are needed for executing a task. Resources are given by the cloud resource pool which has many resources. The systems used in the customized setup are with different system configurations such as the load balancer with i7 processor of 3.40 GHz clock speed and 8 GB RAM; the server machines with i7 processor of 3.40 GHz clock speed and 16 GB RAM and other client machine which generates tasks with a dual core processor of 2.80 GHz clock speed and 4 GB of RAM.

For resource allocation and management strategy, the tasks are coming from the clients' side with the same inter arrival time. We considered a queue size of ten tasks and each task demands cloud resources in random fashion. Allocation and management of these resources are managed by a load balancer based on our proposed algorithms. For experimentation purpose, we assumed CPU and Memory as mandatory resources for execution and each of the VMs needs at least two resources (i.e., CPU and Memory) for execution of a task. In the first iteration, resources are taken from the cloud resource pool which acts as global hub of resources.

Experimentation is carried out with different sets of tasks and VMs. The performance evaluation is carried out in terms of efficient utilization of available virtual machines, average response time and optimum usage of cloud resources. If the resources are not in use, then these unused resources are given back to the cloud resource pool. We consider Round Trip Time (RTT) of 1 second when the VMs take the resources from cloud resource pool. Clusters are made depending on the number of VMs and each of the VMs has its own buffer to store the excess resources.

In this work, the parameters such as execution time, response time and reliability are considered for analysis. Here, the execution time refers to the time spent by the task using the cloud resources; response time refers to the time when the task becomes active till it completes the execution. Reliability refers to the consistency of the proposed algorithms with different scenarios during the experimentation.

For scheduling of VMs, we considered the applications like Facebook users, Twitter users, Internet users etc. For experimentation purpose, we used the internet users from six different continents of the world (six different geographic locations) during peak and non-peak hours [38], [39]. Physical machine has a capacity to host several virtual machines

TABLE 4
Configurations of VMs

VM Type	MIPS	RAM (GB)	Storage (GB)
Small	500	0.5	20
Medium	1,000	1	30
Large 1	1,500	2	40
X. Large	2,000	3	50
Extra Large	2,500	4	50

TABLE 5
Workload Setup for Scheduling

User Base	Region	Online Users during peak hours	Online Users during non-peak hours
North America	0	135,000	13,500
South America	1	125,000	12,500
Europe	2	255,000	25,500
Asia	3	535,000	53,500
Africa	4	30,000	3,000
Oceania	5	10,000	1,000

TABLE 6
Utilization of VMs

Sl. No.	Throttled	Round Robin	ACO	Exact Algorithm	Proposed MPSO
VM1	1,182	254	356	253	254
VM2	76	254	289	254	254
VM3	8	253	389	254	254
VM4	2	253	180	254	253
VM5	0	254	54	253	253

TABLE 7
Average Response Time of Algorithms

Algorithms	Average Response Time
Throttled	365.52 ms
Round Robin	364.85 ms
ACO	362.67 ms
Exact Algorithm	365.87 ms
Proposed MPSO	360.11 ms

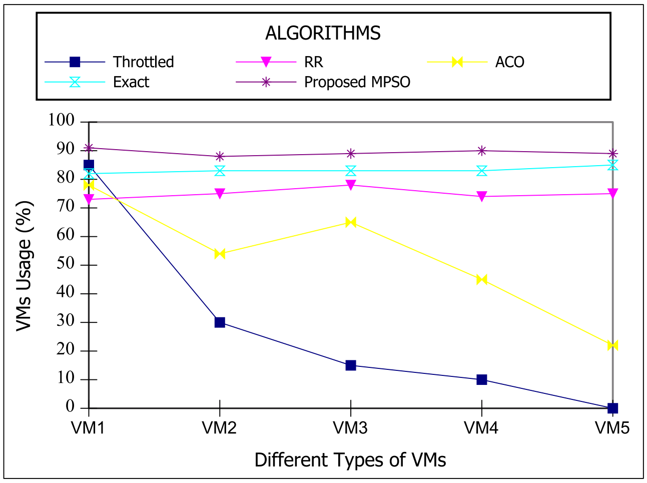


Fig. 5. Average utilization of VMs.

which are needed for an application. In this regard, we used five different types of VMs and Table 4 shows the configuration details of VMs in terms of Million Instructions Per Second (MIPS), Random Access Memory (RAM) and Storage in Gigabyte (GB). Table 5 shows the workload setup used for conducting the experiment for scheduling.

4.2 Experimental Results and Analysis

Results of our proposed Bio-Inspired algorithms are analyzed with respect to efficient utilization of VMs, Average Response Time during Scheduling, Resource Allocation and optimum usage of cloud resources. Further, the proposed algorithms are compared with Branch-and-Bound based Exact Algorithm. We also carried out statistical analysis for null hypothesis using T-Test. Further, the time complexity of the proposed HYBRID algorithm is also analysed.

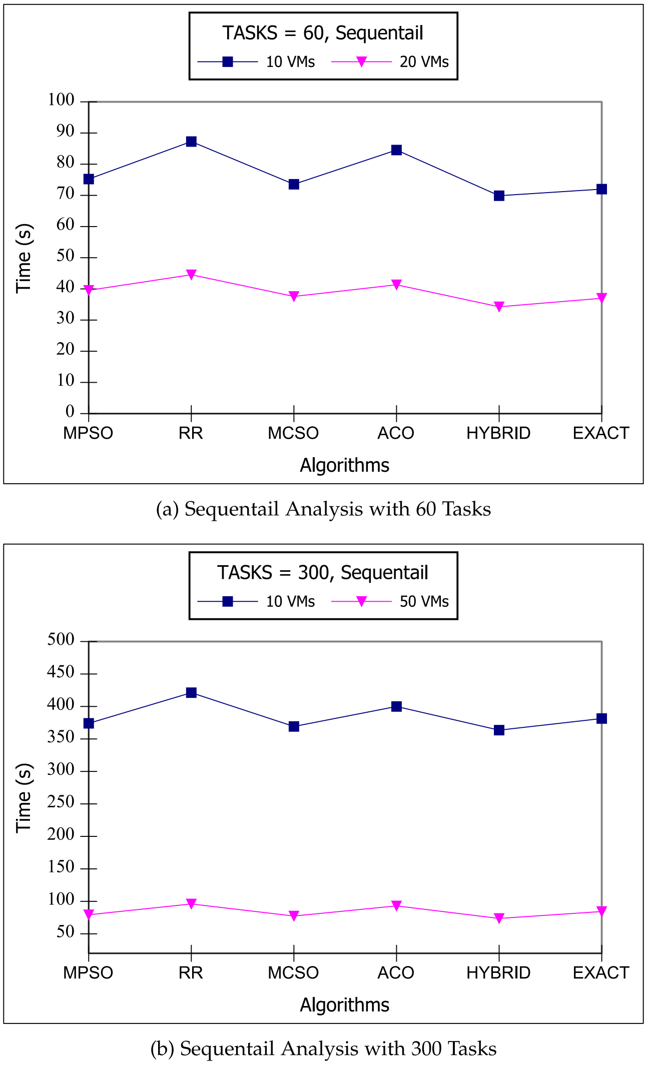
4.2.1 Efficient Load Balancing of VMs by Scheduling

As explained in Section 3, we use the proposed MPSO algorithm for scheduling the tasks to the VMs. Hypothetical examples like web application tasks from different parts of the continents are considered as input and details are given in Table 5. Table 6 shows the number of tasks handled by the VMs for different algorithms.

From Table 6, we can observe that both Round Robin and proposed MPSO algorithms efficiently schedule the tasks as compared to Throttled algorithm. But, if we compare Round Robin and proposed MPSO algorithms, the results are same, but in Round Robin, we do not check the state of the VM (BUSY/AVAILABLE) which leads to the queuing of the incoming task on the server. The proposed MPSO algorithm gives better results by checking the state of the VM and it has cluster based comparison of allocating tasks to a VM and thus avoids the server queuing. In Ant Colony Optimization, the tasks can be assigned to VMs which have less pheromone content. In Exact algorithm, the scheduling of the tasks is similar to that of proposed MPSO but it needs more Average Response time. The same steps are repeated with different number of clusters when there are more VMs. For the aforementioned experiment, we used two clusters with equal number of VMs on each cluster and the Average Response time in milli seconds (ms) for the same setup is given in Table 7.

It is clearly observed from the Tables 6 and 7 that the proposed MPSO algorithm is not only efficient in balancing the load on the virtual machines but also achieves better Average Response time. The experiment is repeated for different sets of virtual machines, tasks and the obtained results are consistent.

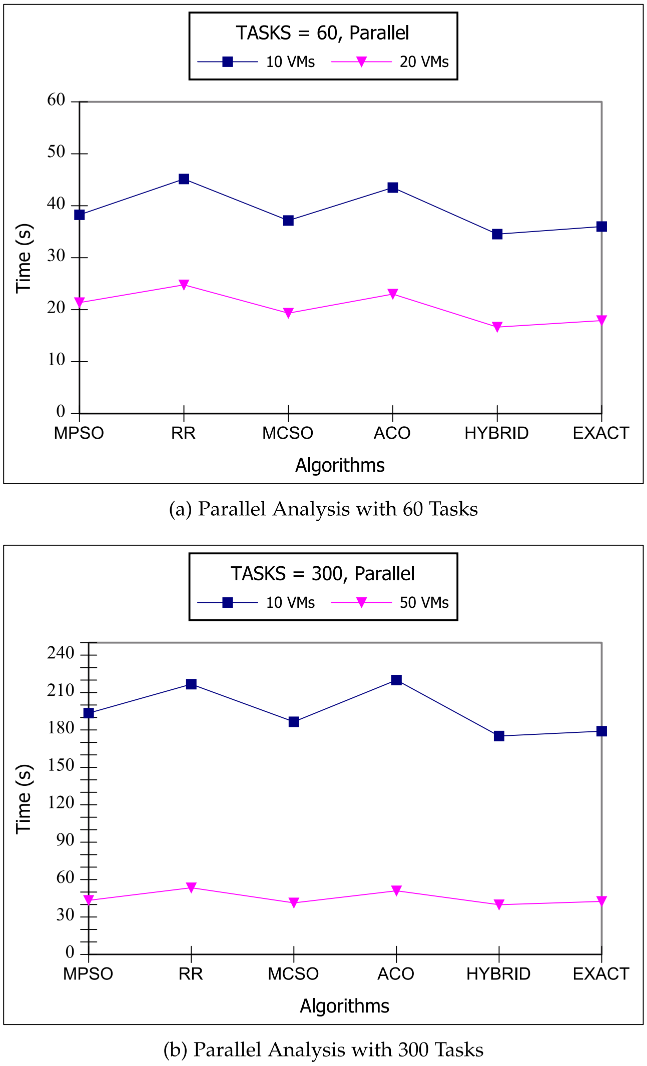
Next, we analysed the utilization of VMs with different combinations 1,000, 2,000 and 3,000 tasks. Fig. 5 shows the utilization of VMs using different algorithms used in the experiment. It is observed from Fig. 5 that Throttled and ACO algorithms are not consistent in utilizing the VMs when compared to other algorithms. Even though RR and Exact Algorithms utilize the VMs efficiently but our proposed MPSO algorithm is more efficient in utilizing the



(a) Sequentail Analysis with 60 Tasks

(b) Sequentail Analysis with 300 Tasks

Fig. 6. Sequential analysis.



(a) Parallel Analysis with 60 Tasks

(b) Parallel Analysis with 300 Tasks

Fig. 7. Parallel analysis.

VMs and Average Response time. If MCSO algorithm uses two modes (Seeking and Tracing mode) for scheduling then Seeking mode takes more time when compared to that of the tracing mode. Hence, we have not considered proposed MCSO for scheduling. Similarly our proposed HYBRID (MPSO+MCSO) is also not considered for scheduling since it combines both MPSO and MCSO. Next, we will analyze the resource allocation and management based on our proposed algorithms (MPSO, MCSO and HYBRID) and the details are given in Section 4.2.2.

4.2.2 Efficient Resource Allocation and Management

VMs facilitate the resources demanded by the tasks which are managed by the proposed techniques. As explained in Section 4, Incoming tasks have the same inter-arrival time and arrive in a batch of ten periodically and demand for different cloud resources. Proposed algorithms are experimented with different number of VMs and tasks. In the first iteration, VMs take resources from cloud resource pool and from the second iteration onwards the proposed algorithms such as MPSO, MCSO and HYBRID (MPSO+MCSO) allocate the resources either from cloud resource pool or from the unused resources of VMs. We experimented the proposed algorithms with two clusters having equal number of VMs. Once the tasks are served by the VMs, then the excess resources are stored in the individual buffers of VMs. Further, these resources are used for serving the upcoming task demands. As discussed earlier, initially VMs start executing the tasks by lending the resources from the cloud resource pool. Experiments are conducted for evaluating the Average Response Time in both sequential and parallel modes; and the corresponding results are shown in Figs. 6 and 7, respectively.

In Fig. 6a, we used 10 and 20 VMs in two clusters with 60 incoming tasks. In Fig. 6b we used 10 and 50 VMs in two clusters with 300 incoming tasks. In MPSO, for the best match (GB) case approach, the two best VMs from each cluster are taken for resources matching with the upcoming demands. For example, *excess_res1* and *excess_res2* from each cluster are compared with resource demands. Further, the reamining VMs will follow the MCSO approach. For example, *excess_res3[]* from each cluster are compared with the upcoming demands.

Further, the experiment is carried out in parallel approach by using threads (depend on the number of clusters) which are executed asymmetrically. Here, threads play a vital role in handling the resources effectively. Each cluster has several threads with which the execution speed can be improved.

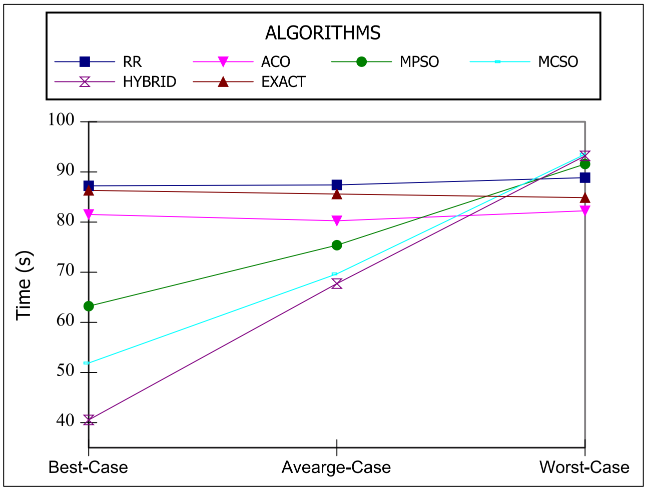


Fig. 8. Execution time analysis of proposed algorithms.

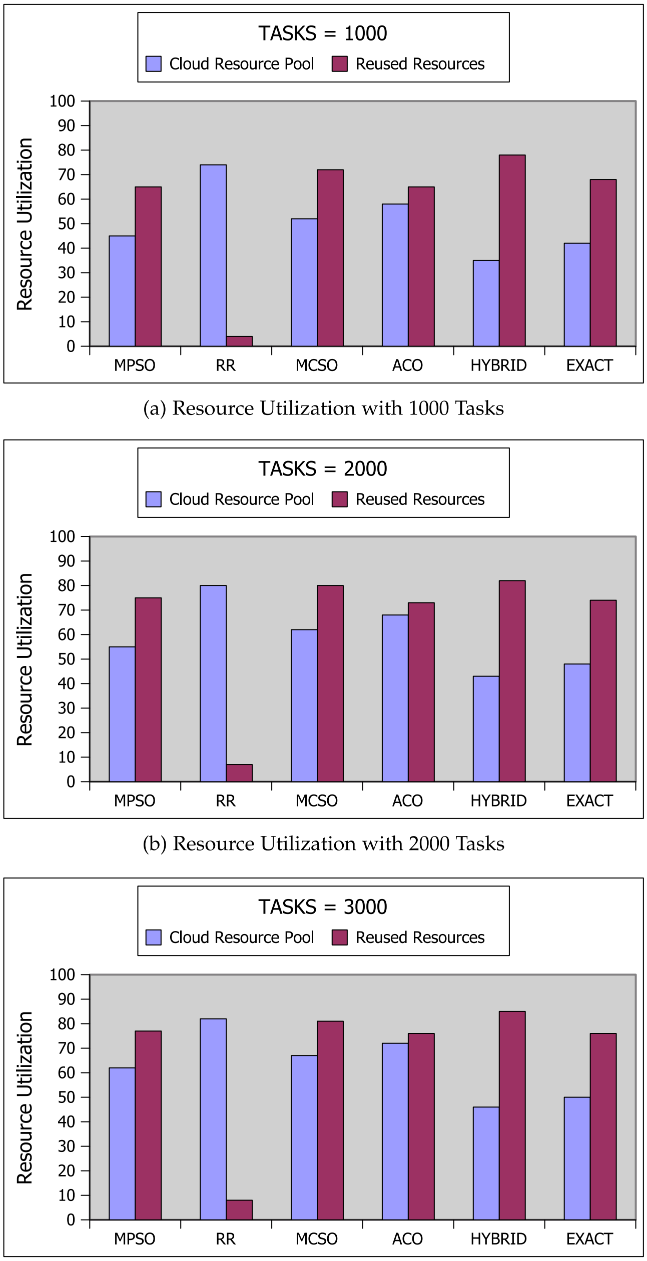
In Fig. 7a, we used 10 and 20 VMs in two clusters with 60 incoming tasks and in Fig. 7b, we used 10 and 50 VMs in two clusters with 300 incoming tasks. It is clearly observerd from Fig. 7 that the parallel mode execution will reduce the average response time as compared with the sequential mode.

Figs. 6 and 7 illustrate that MPSO, MCSO and HYBRID (MPSO+MCSO) algorithms give feasible solutions with respect to average response time when compared with branch-and-bound based Exact Algorithm and further our HYBRID algorithm takes less time when compared with all other algorithms.

Fig. 8 shows the execution time analysis for different cases (Best, Average and Worst) of all the proposed algorithms. We experimented with different combinations of tasks and varied numbers of VMs using the proposed and state-of-the-art benchmark algorithms. It is observed from Fig. 8 that, the worst-case execution time for all the proposed algorithms is a rare case in which the matching never happens with excess resources present in the buffer with the future resource demands. Since RR takes the resources from the cloud resource pool for most of the time, hence the execution time of RR will remain the same for all the cases.

In ACO, the resources match with the VMs which have highest pheromone content. Since there will not be any perfect resource match for most of the time due to high pheromone evaporation rate and hence ACO takes same execution time for all cases. The Exact algorithm tries to match with all possible combinations and then allocates resources with possible solution. Hence the Exact algorithm also takes same execution time for all cases.

The worst-case execution time of MPSO, MCSO and HYBRID (MPSO+MCSO) approaches is continuously increasing because of the delay in the mismatch comparison of resources from the VMs buffer to the future resources demand. In the MPSO, the best-case is considered when the best two values from each cluster match with the upcoming resource demands. In average-case, resources match is varying so that MPSO can work efficiently when compared to RR and ACO. In MCSO, the best-case is considered when rest of the values (other than *excess_res1* and *excess_res2*) are matched with excess resources from VMs buffer to the future resource demands. Hence, more resources got matched when compared to MPSO and thus MCSO outperforms both MPSO, ACO and RR.



(a) Resource Utilization with 1000 Tasks

(b) Resource Utilization with 2000 Tasks

(c) Resource Utilization with 3000 Tasks

Fig. 9. Average resource utilization.

Further, HYBRID approach takes less execution time in both best and average-cases. And if all the resource mappings are true with HYBRID approach then it outperforms both MPSO and MCSO when considered separately. Hence, HYBRID (MPSO+MCSO) approach is more efficient in terms of resource allocation and management in the cloud environment.

As mentioned earlier, the resources are taken either from the cloud resource pool or from the unused resources of the respective VMs from the clusters. Accordingly, proposed algorithms are analyzed with respect to resource utilization factor. Figs. 9a, 9b, and 9c show the resource utilization with 1,000, 2,000 and 3,000 tasks respectively. In all three cases, RR takes maximum resources from the cloud resource

TABLE 8
Results of T-TEST Analysis

Algorithms	P-values
RR-HYBRID	0.005
ACO-HYBRID	0.0035
MPSO-HYBRID	0.0015
MCSO-HYBRID	0.001
EXACT-HYBRID	0.0045

pool. ACO takes almost equal amount of resources from both cloud resource pool and unused resources from VMs. Our proposed MCSO provides better resource utilization as compared to proposed MPSO. On the other hand, the proposed HYBRID (MPSO+MCSO) algorithm is more efficient in utilising the unused resources from clusters rather than taking it from the cloud resource pool. All the proposed algorithms are compared with bench mark Exact algorithm. It is clearly observed from Fig. 9 that our proposed HYBRID (MPSO+MCSO) algorithm achieves high resource utilization efficiency when compared with all other state-of-the-art methods considered for performance evaluation.

4.3 Statistical Hypothesis Analysis

In the statistical hypothesis, two entities are evaluated about a given workload or population so that we can determine the best supported entity for different cases. Hence, we made statistical analysis by evaluating the proposed HYBRID (MPSO+MCSO) algorithm with state-of-the-art benchmark algorithms. For statistical hypothesis, we used T-Test analysis for resource allocation and management. The results of T-Test analysis are shown in Table 8. For the null hypothesis analysis, Threshold value of P or Significance Level of α are considered. In this experiment, we considered α = 0.05 which is the standard cutoff for null hypothesis. The P-value of HYBRID (MPSO+MCSO) algorithm in comparison with other algorithms is less than α = 0.05 and thus it rejects the null-hypothesis. Hence, our proposed HYBRID (MPSO+MCSO) algorithm is more efficient when compared to all other algorithms considered for Resource Allocation and Management.

4.4 Time Complexity Analysis

Our proposed HYBRID algorithm is based on MPSO and MCSO techniques. In this experiment, n number of tasks which are scheduled on heterogeneous VMs hosted on C_z clusters. Therefore, the time complexity of our proposed HYBRID (MPSO+MCSO) algorithm is [O(number of tasks) $*$ O(number of clusters) $*$ O(number of m iterations)] $*$ [O(MPSO) + O(MCSO)], i.e., [$O(n)$ $*$ $O(C_z)$ $*$ $O(m)$] $*$ [$O(vm_k,C_z)$ + $O(vm_k,C_z)$]. It is reduced to $O(n * C_z * m * vm_k)$. Finally, the time complexity is $O(n * vm_k)$, which is $O(n) < O(n * vm_k) < O(n^2)$ since C_z (clusters are fixed) and m are constants.

5 Conclusions and Future Work

In this work, we proposed three Bio-Inspired (MPSO, MCSO and HYBRID) algorithms for efficient scheduling and resource management in a cloud environment. The MPSO algorithm is more efficient in scheduling the tasks when compared to other algorithms. On the other hand, our proposed HYBRID (MPSO+MCSO) approach is more efficient in allocating the resources to the VMs when compared to other algorithms. Our proposed HYBRID algorithm not only reduces the average response time but also increases the resource utilization by approximately 12 percent when compared to other state-of-the-art benchmark algorithms. In future, our focus will be on more effective dynamic scheduling in which tasks will be entering the cloud with different inter arrival times.

References

[1] C.-W. Tsai, W.-C. Huang, M.-H. Chiang, M.-C. Chiang, and C.-S. Yang, "A hyper-heuristic scheduling algorithm for cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 2, pp. 236–250, Apr.–Jun. 2014.

[2] Y. Wang and W. Shi, "Budget-driven scheduling algorithms for batches of MapReduce jobs in heterogeneous clouds," *IEEE Trans. Cloud Comput.*, vol. 2, no. 3, pp. 306–319, Jul.–Sep. 2014.

[3] N. G. Duffield, P. Goyal, A. Greenberg, P. Mishra, K. Ramakrishnan, and J. E. Van der Merwe, "Resource management with hoses: Point-to-cloud services for virtual private networks," *IEEE/ACM Trans. Netw.*, vol. 10, no. 5, pp. 679–692, Oct. 2002.

[4] S. Yu, C. Wang, K. Ren, and W. Lou, "Achieving secure, scalable, and fine-grained data access control in cloud computing," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.

[5] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1107–1117, Jun. 2013.

[6] B. Guan, J. Wu, Y. Wang, and S. U. Khan, "CIVSched: A communication-aware inter-VM scheduling technique for decreased network latency between co-located VMs," *IEEE Trans. Cloud Comput.*, vol. 2, no. 3, pp. 320–332, Jul.–Sep. 2014.

[7] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "Dynamic heterogeneity-aware resource provisioning in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 14–28, Jan.–Mar. 2014.

[8] A. J. Younge, G. Von Laszewski, L. Wang, S. Lopez-Alarcon, and W. Carithers, "Efficient resource management for cloud computing environments," in *Proc. Int. Green Comput. Conf.*, 2010, pp. 357–364.

[9] M. Polverini, A. Cianfrani, S. Ren, and A. V. Vasilakos, "Thermal-aware scheduling of batch jobs in geographically distributed data centers," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 71–84, Jan.–Mar. 2014.

[10] K. Al Nuaimi, N. Mohamed, M. Al Nuaimi, and J. Al-Jaroodi, "A survey of load balancing in cloud computing: Challenges and algorithms," in *Proc. 2nd Symp. Netw. Cloud Comput. Appl.*, 2012, pp. 137–142.

[11] B. Radojević and M. Žagar, "Analysis of issues with load balancing algorithms in hosted (cloud) environments," in *Proc. 34th Int. Convention MIPRO*, 2011, pp. 416–420.

[12] X. Zuo, G. Zhang, and W. Tan, "Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud," *IEEE Trans. Autom. Sci. Eng.*, vol. 11, no. 2, pp. 564–573, Apr. 2014.

[13] K. Nishant, et al., "Load balancing of nodes in cloud using ant colony optimization," in *Proc. UKSim 14th Int. Conf. Comput. Model. Simul.*, 2012, pp. 3–8.

[14] R. Shojaee, H. R. Faragardi, S. Alaee, and N. Yazdani, "A new cat swarm optimization based algorithm for reliability-oriented task allocation in distributed systems," in *Proc. 6th Int. Symp. Telecommun.*, 2012, pp. 861–866.

[15] P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2292–2303, 2013.

[16] T.-Y. Wu, W.-T. Lee, Y.-S. Lin, Y.-S. Lin, H.-L. Chan, and J.-S. Huang, "Dynamic load balancing mechanism based on cloud storage," in *Proc. Comput. Commun. Appl. Conf.*, 2012, pp. 102–106.

[17] R. Jeyarani, N. Nagaveni, and R. V. Ram, "Design and implementation of adaptive power-aware virtual machine provisioner (APA-VMP) using swarm intelligence," *Future Generation Comput. Syst.*, vol. 28, no. 5, pp. 811–821, 2012.

[18] A. Kundu and C. Ji, "Swarm intelligence in cloud environment," in *Proc. 3rd Int. Conf. Swarm Intell.*, 2012, pp. 37–44.

[19] Z. Liu and X. Wang, "A PSO-based algorithm for load balancing in virtual machines of cloud computing environment," in *Proc. 3rd Int. Conf. Swarm Intell.*, 2012, pp. 142–147.

[20] S. Bilgaiyan, S. Sagnika, and M. Das, "Workflow scheduling in cloud computing environment using cat swarm optimization," in *Proc. IEEE Int. Advance Comput. Conf.*, 2014, pp. 680–685.

[21] H. Goudarzi, M. Ghasemazar, and M. Pedram, "SLA-based optimization of power and migration cost in cloud computing," in *Proc. 12th IEEE/ACM Int. Symp. Cluster Cloud Grid Comput.*, 2012, pp. 172–179.

[22] S. G. Domanal and G. R. M. Reddy, "Load balancing in cloud computingusing modified throttled algorithm," in *Proc. IEEE Int. Conf. Cloud Comput. Emerging Markets*, 2013, pp. 1–5.

[23] S. Domanal and G. Reddy, "Optimal load balancing in cloud computing by efficient utilization of virtual machines," in *Proc. 6th Int. Conf. Commun. Syst. Netw.*, 2014, pp. 1–4.

[24] G. Gonçalves, et al., "CloudML: An integrated language for resource, service and request description for D-clouds," in *Proc. IEEE 3rd Int. Conf. Cloud Comput. Technol. Sci.*, 2011, pp. 399–406.

[25] W. Zhou, S. Yang, J. Fang, X. Niu, and H. Song, "VMCTune: A load balancing scheme for virtual machine cluster using dynamic resource allocation," in *Proc. 9th Int. Conf. Grid Cloud Comput.*, 2010, pp. 81–86.

[26] D. Huang, C. Zhu, H. Zhang, and X. Liu, "Resource intensity aware job scheduling in a distributed cloud," *China Commun.*, vol. 11, no. 14, pp. 175–184.

[27] X.-l. Zheng and L. Wang, "A Pareto based fruit fly optimization algorithm for task scheduling and resource allocation in cloud computing environment," in *Proc. IEEE Congr. Evol. Comput.*, 2016, pp. 3393–3400.

[28] M. Rasti-Barzoki and S. R. Hejazi, "Pseudo-polynomial dynamic programming for an integrated due date assignment, resource allocation, production, and distribution scheduling model in supply chain scheduling," *Appl. Math. Model.*, vol. 39, no. 12, pp. 3280–3289, 2015.

[29] M. Akbari and H. Rashidi, "A multi-objectives scheduling algorithm based on cuckoo optimization for task allocation problem at compile time in heterogeneous systems," *Expert Syst. Appl.*, vol. 60, pp. 234–248, 2016.

[30] P. S. Pillai and S. Rao, "Resource allocation in cloud computing using the uncertainty principle of game theory," *IEEE Syst. J.*, vol. 10, no. 2, pp. 637–648, Jun. 2016.

[31] X. Xu, W. Dou, X. Zhang, and J. Chen, "EnReal: An energy-aware resource allocation method for scientific workflow executions in cloud environment," *IEEE Trans. Cloud Comput.*, vol. 4, no. 2, pp. 166–179, Apr.–Jun. 2016.

[32] S.-C. Wang, K.-Q. Yan, W.-P. Liao, and S.-S. Wang, "Towards a load balancing in a three-level cloud computing network," in *Proc. 3rd IEEE Int. Conf. Comput. Sci. Inf. Technol.*, 2010, vol. 1, pp. 108–113.

[33] H. Xu and B. Li, "Anchor: A versatile and efficient framework for resource management in the cloud," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 6, pp. 1066–1076, Jun. 2013.

[34] R. C. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in *Proc. 6th Int. Symp. Micro Mach. Human Sci.*, 1995, vol. 1, pp. 39–43.

[35] J. Kennedy and R. C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *Proc. IEEE Int. Conf. Syst. Man Cybern. Comput. Cybern. Simul.*, 1997, vol. 5, pp. 4104–4108.

[36] A. Bouzidi and M. E. Riffi, "Cat swarm optimization to solve job shop scheduling problem," in *Proc. 3rd IEEE Int. Colloq. Inf. Sci. Technol.*, 2014, pp. 202–205.

[37] S. Martello and P. Toth, *Knapsack Problems: Algorithms and Computer Implementations*. Hoboken, NJ, USA: Wiley, 1990.

[38] I. W. Stat, "Internet users," [Online]. Available: http://www.internetworldstats.com/stats.htm

[39] S. Bakers, "Blog," [Online]. Available: http://www.socialbakers.com/blog/878-new-number-1-asia-became-the-largest-continent-on-facebook

Shridhar G. Domanal received the BE and MTech degrees from the Visveswaraya Technological University, Belgaum, Karnataka, India, in 2009 and 2011, respectively. Currently, he is working toward the full-time PhD in the National Institute of Technology Karnataka, Surathkal, Mangalore, India. He focuses on scheduling and resource management in cloud computing. He is a student member of the IEEE and has nine International Conference publications so far.

Ram Mohana Reddy Guddeti received the BTech degree from S.V. University, Tirupati, Andhra Pradesh, India, in 1987, the MTech degree from the Indian Institute of Technology, Khargpur, India, in 1993, and the PhD degree from the University of Edinburgh, United Kingdom, in 2005. Currently, he is the professor and head in the Department of Information Technology, National Institute of Technology Karnataka Surathkal, Mangalore, India. His research interests include affective computing, big data and cognitive analytics, bio-inspired cloud and green computing, Internet of Things and smart sensor networks, social multimedia, and social network analysis. He has more than 160 research publications in reputed and peer reviewed International Journals, Conference Proceedings, and Book Chapters. He is a senior member of the IEEE and the ACM, a lLife fellow of the IETE (India), and a life Member of ISTE (India)

Rajkumar Buyya a professor of computer science and software engineering, future fellow of the Australian Research Council, and director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He has authored more than 500 publications and four text books including "*Mastering Cloud Computing*" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. He also edited several books including "*Cloud Computing: Principles and Paradigms*" (Wiley Press, USA, Feb 2011). He is one of the highly cited authors in computer science and software engineering worldwide (h-index=97, g-index=202, 44,100+ citations). Microsoft Academic Search Index ranked, he as the world's top author in distributed and parallel computing between 2007 and 2015. "*A Scientometric Analysis of Cloud Computing Literature*" by German scientists ranked, he as the World's Top-Cited (#1) Author and the World's Most-Productive (#1) Author in Cloud Computing Software technologies for Grid and Cloud computing developed under His leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world. He has led the establishment and development of key community activities, including serving as foundation chair of the IEEE Technical Committee on Scalable Computing and five IEEE/ACM conferences. These contributions and international research leadership of, he are recognized through the award of "2009 IEEE TCSC Medal for Excellence in Scalable Computing" from the IEEE Computer Society TCSC. Manjrasoft's Aneka Cloud technology developed under his leadership has received "2010 Frost & Sullivan New Product Innovation Award" and recently Manjrasoft has been recognised as one of the Top 20 Cloud Computing companies by the Silicon Review Magazine. He served as the foundation editor-in-chief of the *IEEE Transactions on Cloud Computing*. He is currently serving as co-editor-in-chief of the *Journal of Software: Practice and Experience*, which was established more than 40 years ago. He is a fellow of the IEEE. For further information on him, please visit his cyberhome: www.buyya.com.