Experimental Performance Analysis of Cloud Resource Allocation Framework using Spider Monkey Optimization Algorithm

Mohit kumar[1], Kalka Dubey[2], Samayveer Singh[3], Jitendra Kumar Samriya[1], Sukhpal Singh Gill[4]
[1]Department of Information technology, NIT Jalandhar, India
[2]Department of Computer Science and Engineering, RGIPT Jais, Amethi, India
[3]Department of Computer Science and Engineering, NIT Jalandhar, India
[4]School of Electronic Engineering and Computer Science, Queen Mary, University of London
*Mohit Kumar, Corresponding author, Department of IT, NIT, Jalandhar, kumarmohit@nitj.ac.in

Abstract: The cloud services demand has increased exponentially in the last decade due to its plethora of services. It becomes a significant platform to compute large and diverse applications over the internet. On the contrary, on-demand resource allocation to a variety of applications becomes a serious issue due to dynamic workload conditions and uncertainty in the cloud environment. Several existing state of art techniques often fails to allocate the optimal resources to forthcoming demands, leading to an imbalance workload over cloud platform, degrading the performance. This article introduces a secure and self-adaptive resource allocation framework that addressed the mentioned issues and allocates the most suitable resources to users' applications while ensuring the deadline constraints. Further, the proposed framework is integrated with a metaheuristic algorithm named enhanced spider monkey optimization (SMO) algorithm that is based on the intelligent foraging behavior of spider monkeys. The proposed algorithm finds an optimal resource for the user's application using the fission-fusion approach and improves multiple influential parameters like time, cost, degree of load balancing, energy consumption, task rejection ratio, etc . The experimental CloudSim based results verified that the proposed framework performs superior to state of art approaches like PSO, GSA, ABC and IMMLB.

Keywords: *Virtual machines, Resource allocation, Makespan, Service Cost, Spider monkey optimization*

1. Introduction

The cloud paradigm is offering a plethora of services in the form of virtualized resources to end-users pay per used basis. It becomes a primary solution to process the complex and big data applications cost-effectively due to which end-users started to compute their applications from an on-premise platform to a cloud platform. There are four cloud deployment models (public, private, hybrid, and community) that depict the sharing of computational resources inside the cloud. The cloud models have been adopted widely for business applications, academic, data-intensive, and scientific applications due to their versatile characteristics like on-demand resource provisioning, multi-tenant environment, economical, scalable, and elastic, etc. [1]. Virtualization is the core technology in the cloud paradigm that offered the sharing of resources and applications in a multi-tenant environment by deploying several heterogeneous virtual machines (VMs) over a single physical host. It offers significant benefits to industries as well as IT companies to get the required services from the cloud rather than buying the whole infrastructure. The cloud paradigm consists of various types of features such as dynamic infrastructure, resource provisioning and de-provisioning, easy to use, on-demand availability of computational resources, automatic software updates, etc. that accomplish the demands of customers and service providers but still, it is in infancy due to inefficient scheduling and allocation strategy, security and privacy threats and imbalance workload among the cloud resources, etc. These issues do not only affect the quality of services (QoS) parameters [2], but also degrade the service performance as well as increase the service level agreement (SLA) violations.

Despite offering several predominant solutions by cloud paradigm, it is facing security and privacy issues especially data breaches, account hijacking, sniffer attacks, DoS attack, exploited system vulnerabilities, and permanent data loss etc. at network, applications, and virtualization levels [29]. Hence, it is the responsibility of the cloud service provider (CSP) to address the basic issues like confidentiality, authentication, integrity, and availability as shown in figure 1. The virtual machine manager (hypervisor) allows the virtualization concept in the cloud paradigm through multi-tenancy, but it also brings security risks like VM-to-VM attack, data visibility, hazard in virtual networking, single point of failure, etc. Therefore, it becomes necessary to secure the cloud infrastructure and preserve the basic security requirement to protect the user's data. Several cloud resource allocation frameworks have been proposed and implemented in the cloud environment, but all of them focused only on the allocation of resources and none of them

considered security issues. Hence, we have proposed a framework that allows only legitimate users to access the cloud resources and protect the user's data over the networks before the allocation of resources.

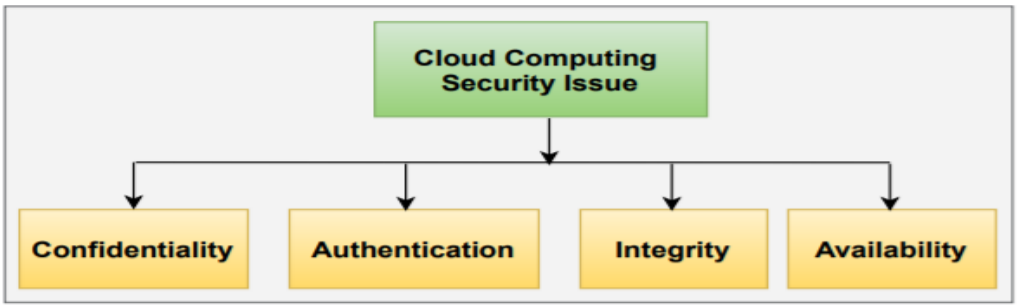

Figure 1 security issues in the cloud paradigm

Resource allocation becomes a prominent problem in cloud paradigm due to uncertainty, heterogeneous nature of computational resources, dynamism, high demand, and volume of requests [3]. As the service demand of end-users is increasing at cloud platforms that leads to an increase in the workload and required an efficient resource allocation technique to handle this workload. The existing state of art approaches are facing several challenges like over and underutilization of resources, lack of decision-making capability, inflexible nature, SLA violation, lack of pricing transparency, low throughput, high power consumption, and service cost during the allocations of diverse and complex applications over the VMs. Hence, we required a resource allocation framework that maps the virtual machines with physical resources and allocates the best virtual machine to upcoming demand for accomplishing the objectives of the service provider as well as end-users [4]. The adaptive resource allocation approach plays a significant part in the cloud paradigm whose goal is to alloacate the miscellaneous workload over the resources based on their capacity that directly influence valuable QoS parameters. In addition, resource allocation techniques should have decision-making capability at run time to avoid the probability of under or overutilization of cloud datacenter resources and maintain the SLA conditions without any violations. If utilization of resources is not up to marks as per the SLA, then it is not beneficial for cutomers as well as service provider in terms of cost and profit of resources.

On-demand resource allocation is a complex problem and come under the class of NP-Complete in a cloud environment [5]. Since increases in the size of the problem linearly, will lead to increases in the complexity of the algorithm exponentially. Resource allocation problem in the context of Cloud computing has been investigated widely, its performance is affected by a large number of factors like a high volume of requests and diverse applications, heterogeneous workload, dynamism and network conditions, elastic resource provisioning and de-provisioning, unpredictable demand, and lack of transparency in the pricing plan. Hence, an efficient resource allocation strategy is required to fulfill the objectives of the service provider and end-users. Despite such importance, there is very little work in Cloud resource allocation that satisfies users demand while ensuring the maximum profit for CSP. Although a wide variety of heuristics algorithms are proposed by the researchers to address the Cloud resource allocation problem but did not provide an acceptable solution to the problem within a finite time interval. Hence nature-inspired techniques are needed to search for the best possible solution to the problem and optimize the QoS metric. In this paper, we have proposed an Adaptive **C**loud Resour**c**e Allocation Fram**e**work u**s**ing an Enhanced **S**MO algorithm that provision the resources based upon demand as well as the budget of users, schedules the workload at resources in a well-organized way using decision making capability to optimize various QoS parameters using enhanced SMO algorithm. The key contribution of this research work is given as follows:

- The enhanced SMO-based framework is proposed to search the best solution of scientific applications and fulfill the demand of end-users.
- A trusted cloud resource allocation framework is proposed that validates the authenticity of end-users using
- an enhanced security mechanism.

- We examine and develop the power consumption and cost optimization model for computational resources that can effectively cut down energy consumption and total deployment cost.
- Performance and effectiveness of developed framework are validated at miscellaneous workload by comparing nature-inspired algorithms namely PSO, ABC, GSA, and heuristic IMMLB algorithm.
- The obtained experimental results reveal that the proposed secure and adaptive framework surpass the rival algorithms in terms of QoS metrics used.

The remaining portion of the article is arranged as follows: Section 2 gives the methodical literature review of heuristic and metaheuristic approaches. System and workload model, problem statement with objective function is described in section 3. The proposed an adaptive framework for the allocation of optimal resources is narrated in section 4. SMO and enhanced SMO algorithm along with its working principle in the field of the cloud are depicted in section 5. Discuss the simulation set up, assess & analyze the computational results of the proposed algorithm with baseline algorithm is discussed in section 6, in the end, conclude the article along with future work in section 7.

2. Related work

Allocation of optimal resources to end-users demands is a significant research topic in the area of cloud paradigm whose main aim is to search the best possible virtual machines to an application (tasks) so that algorithms can optimize required QoS parameters. There are several techniques (heuristic and metaheuristic) that have been proposed to fulfill the research gap in the field of cloud [6-21]. Heuristic approaches provide the exact solution of the deterministic problem, but cannot solve a hard optimization problem with acceptable performance. On the contrary, some of the meta-heuristic techniques efficiently explore the whole search area and provide the ideal solution to a hard optimization problem.

2.1 Heuristic Algorithms: Authors have classified the heuristic approach broadly into two ways named static and dynamic, while meta-heuristic algorithms are randomization based and always dynamic. An improved min-min algorithm has been reported by Chen et al. [6] to reduce the makespan time and upsurge the cloud datacenter resources utilization. The performance of the algorithm is tested Matlab simulation environment and computational results proved that it performs superior to the min-min algorithm. Rescheduling of tasks was the main issue with this algorithm that sometimes increase the makespan time of applications (tasks). Priority-based and dynamic load balancing algorithms have been introduced by M. Kumar and S.C. Sharma [7-8] to share the workload as per the resources capacity and monitor each resource continuously to improve the QoS metric while considering deadline and priority as constraints. But both algorithms face the problem of over and underutilization at the peak time (huge requests) and non-peak time. An effective scheduling strategy is proposed by Y. Xin et al.,[9] that attempts to allocate the tasks over virtual machines based upon random schedule technique and improve influential parameters based on the budget of customers, but the proposed approach failed to improve the processing time of applications due to overhead. S. Azizi et al., [10] introduced a Greedy Randomized VM Placement strategy for a cloud environment that finds the best physical machine for VMs to cut down the power and improve other QoS parameters also. Most of the proposed dynamic heuristic techniques are complex and unable to find a suitable solution of the optimization problem in finite time.

2.2 Motivation to choose Meta-heuristic Techniques

Several Meta-heuristic techniques [33-36] have been applied in diverse domains such as industry, academic, research, and business to solve the NP-complete problem [38] [39], combinatorial optimization [40], pattern recognition [41], anti-collision problems [42], and robot motion [43]. These algorithms provide the approximate solution of the problem and have the capability to provide the compromise solution of the conflicting parameter within the limited period in the cloud environment. It is understandable that meta-heuristic algorithms have the capability to explore and exploit the search space in an efficient manner and enhance the required key performance indicator parameters in a short time. There are several soft computing approaches have been introduced to find the optimum solution of scheduling and load balancing problems in the cloud environment. Honey bee behavior inspired load balancing (HBB-LB) approach is developed by D.Babu et al. [11] with the objective to reduce the average waiting time of tasks

queue, and upsurge the throughput within defined constraint, but the proposed technique can trap in starvation condition due to waiting for a long period of time by lower priority tasks. Particle swarm optimization (PSO) based scheduling technique is proposed by R.Fahimeh et al. [12] whose aim was to optimize execution, makespan, and task transfer time by monitoring the virtual machines continuously, if any datacenter resource (virtual machine) is in the overloaded state then transfer the task from overloaded to the underloaded virtual machine rather than transferring the physical virtual machine, but algorithm trap in local optima. Some of its variants have been proposed to overcome the mentioned issue [13-14], but provide only a limited solution, not a completely satisfactory solution. Ant colony optimization (ACO) based scheduling approach has been proposed by E.Pacini et. al. [15] to process the scientific applications over the cloud resources. The proposed algorithm tried to optimize the performance metrics based upon users' requests and create virtual machines. The CloudSim based experiments demonstrated that the proposed ACO-based scheduler achieves significantly better metrics compared to baseline approaches, but this technique is costly and time taking.

Load balancing resource clustering-based approach has been introduced by M. Adhikari et al., to improve the makespan and service cost using BAT algorithm [16]. The proposed approach chooses the optimal service for the task execution and overcomes the issue of imbalanced workload, but the authors did not the trade-off solution between time and cost (multi-objectives) parameters. A new technique named Grey wolf optimization has been proposed by authors to cut down the power consumption and improve the makespan time with system performance [17], but the algorithm neither maintain the exploration not exploitation in proper way and balancing factor between these two was not appropriate to solve the problem. D. Chaudhary and B. Kumar proposed cloudy-GSA algorithm to optimize the QoS parameters like time, cost, etc [18]. The proposed algorithm is based upon Newton's gravitational law and plays an important role in scheduling the applications over the cloud resources, but the convergence rate of the algorithm is still an issue. There are many other metaheuristic algorithms [19-21] that have been introduced in a cloud environment that try to search for the best possible resource for the user demand and optimize QoS parameters. A robust canonical PSO algorithm has been proposed for the scheduling of IoT applications over the cloud resources to optimize the QoS metrices [31]. The algorithm performance is measured in terms of delay and throughput using miscellaneous workload. Authors have proposed modified PSO approach for makespan time improvement of IoT applications [32]. The proposed approach finds the optimal cloud resource for homogeneous and hetrogegnous workload. The experimental results reveal that proposed algorithm is superior over baseline approaches. The proposed cloud based resource allocation approach does not work for latency sensitive applications and need the middleware computing paradigm like fog and edge computing for services. Meta-heuristic techniques have been applied in manufacturing industy for a cutting stock problem to find the optimal solution [33]. The above mentioned real world problem is formulated in the form of objective function or fitness function by mathematical model and try to find the ideal solution using metaheuristic based optimization algorithms [34-36]. Although, entire mentioned optimization algorithms have some limitations such as premature convergence, stagnation, unbalanced workload, excessive VMs migration, and no-compromise solution, etc. yet there is a need for improvements. After the rigorous review of mentioned optimization algorithms, it has been cleared that PSO, ACO, GSA, ABC, GWO, BAT, etc are not efficient techniques to resolve the problem of resource allocation in the cloud without compromising SLA and deadline. Hence, we have proposed an enhanced SMO-based nature-inspired technique that overcomes the existing challenges and allocates the upcoming requests at resources in an efficient way based upon fitness function. Further, the Proposed technique optimizes the key performance indicator parameters within user deadline and budget.

3. System Models and Problem Statement

We will cover the system & workload model in sections 3.1 & 3.2. Further, we will formulate the problem and define the fitness as well as objective function based upon the demands of the user using a mathematical model in section 3.3.

3.1 Model of Tasks and VMs Requests

In the cloud environment, the user submits the n applications/tasks ($£_1, £_2, £_3 \ldots £_n$) to execute at a datacenter server (DC) that is composed with M workstation or physical host denoted as PH={ $PH_1, PH_2, PH_3 \ldots\ldots\ldots PH_M$}. A PH consists of m multiple heterogeneous VMs ($\rho_1, \rho_2, \rho_3 \ldots \rho_m$) to accomplish the demand of end-users. The number of VMs is deployed at a workstation depends upon the remaining resource capacity of the individual workstation as well as resources required by applications. Each task $£_i$ have the length $\ell(£_i)$ and associated with deadline $\partial(£_i)$. If the requested resource is close to the available resource and allocated frequently as per demands then there is the possibility of load imbalance at a workstation as well as server and leading decline the service performance. All the upcoming applications are scheduled over the heterogeneous resources like memory MM_{mem}, processing speed p_s, storage s_c, no. of cpu η, etc based upon the demands. End-user expects to process all tasks with maximum throughput, minimum time as well as financial cost while the service provider try to get the profit maximum from datacenters resources. The entire upcoming applications are analyzed (authentication, feasibility based upon constraints like deadline, budget, etc), and then allocate to the best resource for the processing, otherwise the request is rejected.

3.2 Workload Model: Users submit diverse applications such as computation-intensive, memory-intensive, synthetic workload, common applications request in a cloud paradigm for the services that need VM instances for the completion. Each application is submitted with different characteristics (number of VMs required, type of VMs, the time duration of VMs, deadline) and will require a different number of resources for processing in minimum time based upon their nature and requirement (CPU intensive, memory-intensive). A user can request for different types of VM instances or multiple users can request for the same VM instances for the services. Keeping in mind, synthetic workload (complex and diverse tasks) has been generated and allocated to the heterogeneous VM instances to test the performance of the proposed framework.

3.3 Problem Statement: The goal of the proposed framework is to search and allocate the optimal resources for the upcoming user workload in this article. Further, allocate the datacenter resources in an efficient way to improve the performance indicator parameters including energy efficiency, degree of imbalance, and cost, etc using an enhanced SMO-based approach within a user-defined deadline. We formulate the scheduling and allocation of applications as a computational optimization problem that tries to reduce the time, cost, and energy of datacenter resources for offering the services.

Fitness function& objective function:

In this sub-section, we will derive the mathematical model based upon the problem and define the fitness function.

3.3.1 First objective: optimize the time of tasks

Processing time of n tasks ($TPT_{£_i}^{\rho_j}$) = min $EET_{£_i\rho_j \in \theta_{£iSp}}$ (1)

$EET_{£_1\rho_j \in \theta_{£iSp}}$ is the sum of execution time ($ET_{£_1\rho_j \in \theta_{£iSp}}$), the task transfer time ($TTT_{£_1\rho_j \in \theta_{£iSp}}$) and waiting time ($WT_{£_1\rho_j \in \theta_{£iSp}}$) of tasks. S_p indicates the p[th] schedule of user applications.

Execution time $ET_{£_i\rho_j \in \theta_{£iSp}} = \sum_{i=1}^{n} \lambda_{£_i\rho_j} * \frac{\ell(£_i)}{p_s * \eta}$ (2)

$$\lambda_{£_i\rho_j} = \begin{cases} 1 & \text{if } £_i\rho_j \in \theta_{£iSp} \\ 0 & \text{otherwise} \end{cases} \qquad \forall i \in £, \ \forall j \in \rho \tag{3}$$

$$TTT_{£_i\rho_j \in \theta_{£iSp}} = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} X_{ijk} * \frac{\mathrm{VDE}_{jk}}{B_{jk}} \tag{4}$$

Value of X_{ijk} is 1 when task $£_i$ is provision from ρ_j to ρ_k otherwise 0, and VDE_{jk} represents the exchange of data between resources $\rho_j \& \rho_k$

$$\mathrm{VDE}_{jk} = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{m} X_{ijk} * \ell(£_i) \tag{5}$$

$$\text{Where} \left|\theta_{£iSp}\right| \leq m \quad (6)$$

3.3.2 Second objective function minimizes the total execution cost (Resource Pricing) ($TEC_{£_i}^{\rho_j}$) of tasks $£_i$ at resources ρ_j in scheduleS_p.

$$\text{Min } TEC_{£_i}^{\rho_j} = EEC_{£_i\rho_j \in \theta_{£iSp}} \quad (7)$$

$$EC_{£_1\rho_j \in \theta_{£iSp}} = ET_{£_1\rho_j \in \theta_{£iSp}} * C_{\rho_j} \quad (8)$$

$$TTC_{£_i\rho_j \in \theta_{£iSp}} = TTT_{£_1\rho_j \in \theta_{£_iSp}} * C_{\rho_j} \quad (9)$$

$$EEC_{£_i\rho_j \in \theta_{£iSp}} = EC_{£_i\rho_j \in \theta_{£iSp}} + TTC_{£_i\rho_j \in \theta_{£iSp}} \quad (10)$$

Where $\theta_{£iSp}$ is the number of matched resources or tasks $£_i$ in a schedule S_p.$EEC_{£_i\rho_j \in \theta_{£iSp}}$ represents the excepted execution cost, $EC_{£_i\rho_j \in \theta_{£iSp}}$ is execution cost, $TTC_{£_i\rho_j \in \theta_{£iSp}}$ is task transfer cost and C_{ρ_j} is the cost of resources per hour basis.

3.3.3 System utilization-based power consumption model:

The goal of the third objective is to cut down the power consumption of the datacenter by utilizing computing resources in an effective way. The power consumption model is defined to reduce energy consumption. Power is consumed by various devices (resources) in the datacenter such as CPU, storage, memory, transceiver, and fan, etc, but most of the power is consumed by the CPU (nearby 90%) and it is linearly correlated with utilization of CPU [22]. Hence, we need the information of processing time and CPU utilization to compute the power consumption of applications (workload). Resource utilization (U_t) is defined in terms of time as

$$U_t = \sum_{a=1}^{b} U_{t,a} \quad (11)$$

Where b is end-user applications (workload) that are required $U_{t,a}$ resources for the execution in time t[23]. Actual energy consumption (E_{actual}) of datacenter resources ρ_j is defined as

$$E_{\text{actual}} = (E_{\max} - E_{\min}) * U_t + E_{\min} \quad (12)$$

$E_{\max}$ & $E_{\min}$ represents the maximum and minimum energy consumption to execute the workload by resources ρ_j.

Time and cost are conflicting objectives in cloud computing, we cannot improve both the parameters simultaneously. Example: if try to reduce the makespan time/execution time of workload by increasing the number of virtual machine instances, then execution cost of resource is increased i.e., cheaper resources have less computing capability as compared with high end resources. If both objectives are conflicting with each other, then we must find set of compromise solution for the problem with equal priority which cannot be compared with each other [37]. Hence, this problem becomes multi-objective and we have to find a compromise solution for it. Fitness function for the entire objective is defined as

$$f(\rho_j) = \alpha_1 * EET_{£_i\rho_j \in \theta_{£iSp}} + \alpha_2 * EEC_{£_i\rho_j \in \theta_{£iSp}} + \alpha_3 * E_{\text{actual}} \quad (13)$$

Subject to

$$\alpha_1 + \alpha_2 + \alpha_3 = 1 \quad (14)$$

$$\forall i \in \{1, 2, 3 \ldots\ldots n\}, \quad \sum_{j=1}^{m} \lambda_{ij} = 1, \quad (15)$$

$$\forall i \in \{1, 2, 3\ldots\ldots\ldots n\},\ TPT_{£_i}^{\rho_j} \leq \partial(£_i) \quad (16)$$

$$\sum_{j=1}^{m} PH_{mem_j} \geq MM_{mem} \quad (17)$$

$$\sum_{j=1}^{m} PH_{cpu_j} \geq p_s * \eta_{cpu} \quad (18)$$

Value of α_1, α_2 & α_3 is depends upon user choice & preference. Equation 15-16 ensured that one task is not allocated to more than one resource and it should be executed before within deadline. Equation 17 indicates that the total memory of physical hosts in a datacenter should always be greater than the required by applications or tasks, equation 18 depicts that number of CPUs in a host to execute the upcoming tasks is always greater than the requirement of CPU by end-users tasks. Upcoming tasks are CPU intensive and independent of each other. All the tasks are non-preemptable and their processing time is VMs dependent. The deadline of user tasks (workload) is compared with the total available workload at VM (submitted workload and previously available workload) as shown in equations 19 & 20. The resource is assigned to tasks if it satisfied the condition shown in equation 20. Initial workload W_1, W_2, W_3… W_m at VM is 0 i.e. W_{ρ_j} is 0 at the starting.

$$RT_{£_1\rho_j} = \partial(£_i) - W_{\rho_j} \quad (19)$$

$$ET_{£_1\rho_j} \leq RT_{£_1\rho_j} \quad (20)$$

The workload at VMs is increased as the new application is allocated for the services

$$W_{\rho_j} = W_{\rho_j} + ET_{£_i\rho_j} \quad (21)$$

We defined the makespan time that is an important parameter to assess the proposed algorithm performance.

$$\text{Makespan time of applications} = \max\{mst_{\rho_j}\} \quad (22)$$

$$mst_{\rho_j} = \sum_{£_1 \in \theta_{\rho_j}} EET_{£_1\rho_j} \quad (23)$$

4. Design and Development of Resource Framework

The goal of the proposed framework is to allocate the applications of legitimate users only, and continuously monitor the status of entire VM instances to avoid the possibility of over or under-provisioning the resources. In addition, the proposed approach has the ability to take the decision at run time and choose the best resources for the execution of end-user applications, and improve the performance indicator parameters. It consists of various components such as Cloud security layer, resource manager, resource provisioning & de-provisioning, adaptive task scheduling and allocation strategy, and mapping the physical resource with virtual machine instances as shown in figure 2. Workings of each component are narrated below:

4.1: Cloud security layer: A trusted cloud resource allocation framework is proposed that validates the authenticity of end-users using an enhanced security mechanism (multifactor authentication). The cloud users submit the requests through the web or command-line interface and the service provider confirms that only legitimate users are allowed to access the cloud resources. Malicious or unauthenticated user requests are blocked i.e., service providers do not grant permission to access the cloud resources if the credential is not valid. The aim of the access control mechanism is to prevent the illegal end-users to access the resources for the services and it should be flexible as well as network independent. After the verification of requests, legitimate users' requests are encrypted using the advanced encryption algorithm to secure the data over the network and avoid the possibility of modern attacks. The Advanced Encryption Standard (AES) and RSA algorithm are used for secret and public key encryption during the authentication and services. After that, legitimate users can send the request for the services to the resource manager (resource broker) in

a more secure way as compared with traditional security mechanisms. Finally, the proposed Enhanced SMO adaptive task scheduling and allocation strategy apply for the optimal allocation of requested tasks to the available number of virtual machines while considering various Quality-of-Service (QoS) parameter. Acknowledgment is sent once has work has been completed. Figure 3 represents flow of information for the proposed resource allocation framework.

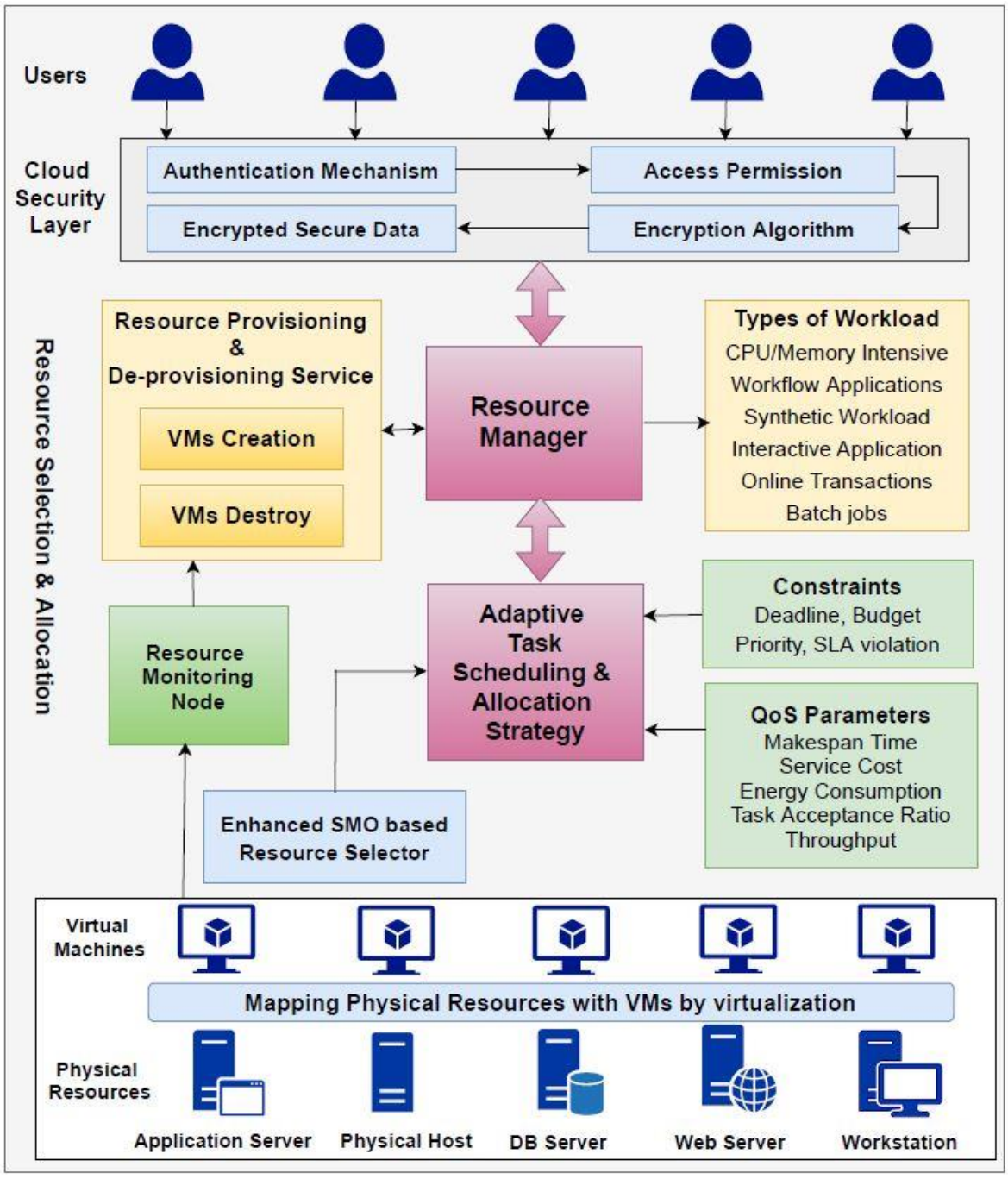

Figure 2 Secure and adaptive resource allocation framework

4.2 Resource Manager: This is the key component of the proposed cloud resource allocation framework whose objective is to interact with other components of the framework for processing the end user's tasks in an effective way without violating the constraints like deadline and budget of users. The resource manager collects the diverse

requests of legitimate users and stores them in a workload queue for processing. Initially, it analyzes every request based upon their demand and constraints, then send it to provisioning and de-provisioning components to create or deploy the VM instances so that it can select the optimum resources to fulfill the demand of requests. Further, a list of resources is forwarded to the adaptive task scheduling and allocation strategy component of the framework that will schedule and allocate the resources to diverse workloads using the proposed algorithm and return the result of QoS metrics to the resource manager.

4.3 Resource provisioning & de-provisioning: Virtual machine instances are created and destroyed by this component based upon the demand (expand or shrink) of users and the status of cloud resources. It gets the status of cloud resources from the resource monitoring node and the demand of end-users from the resource manager. The monitoring node plays a vital role in the proposed framework, it continuously monitors the cloud resources and sends the information to resource provisioning & de-provisioning component for further essential action. If the demand of users is expanded at run time, then more VM instances are deployed over the physical host to avoid the condition of under-provisioning. If the demand of end-user is shrunk at run time, then the number of VM instances is destroyed to avoid the condition of over-provisioning in a cloud datacenter.

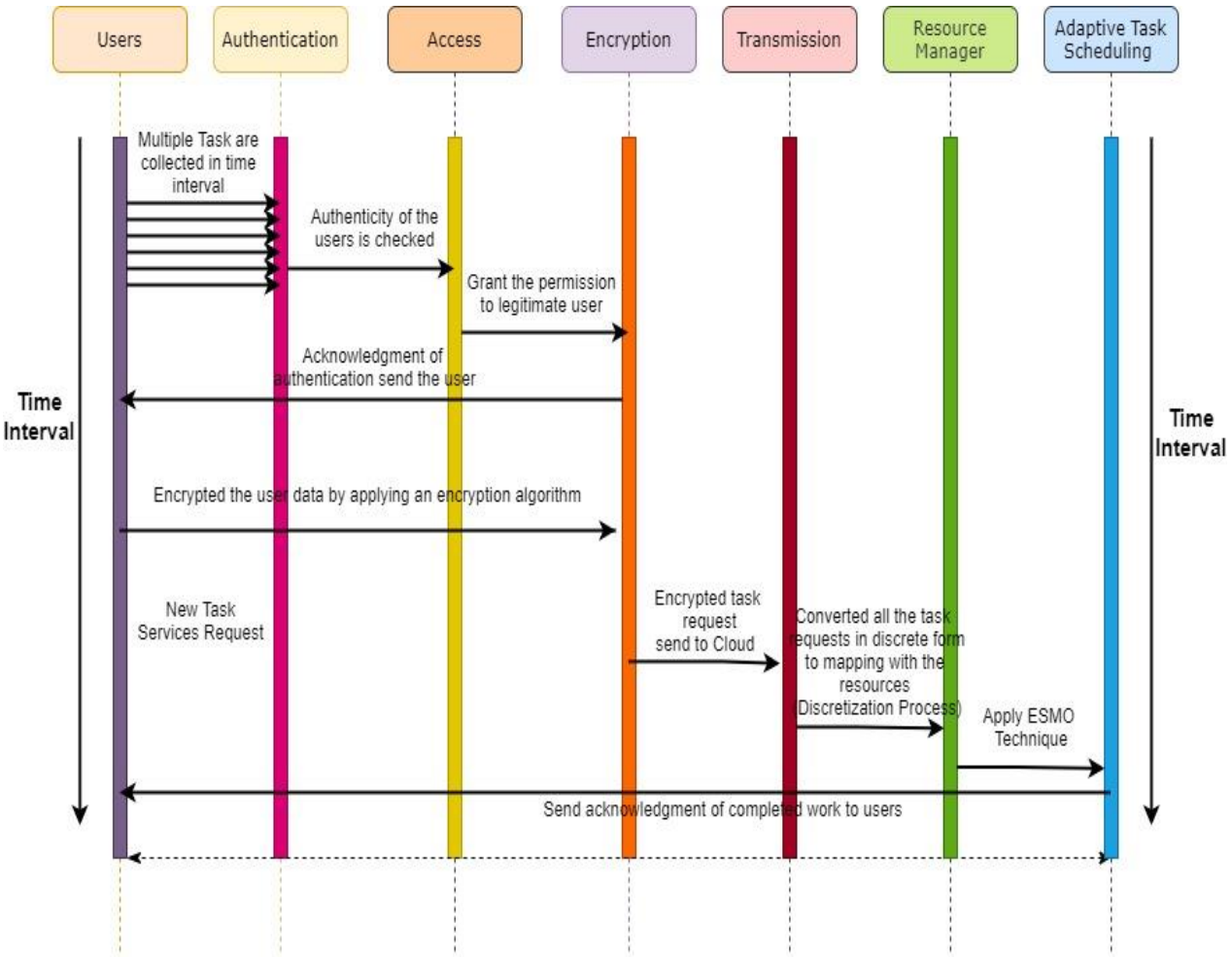

Figure 3 Information and credential flow of adaptive resource allocation framework

4.4 Adaptive task scheduling and allocation strategy: This component is responsible for scheduling and allocating of best possible resources to upcoming users' requests (workload or demand) using an enhanced SMO algorithm. It gets the list of resources from the resource manager and maps them with the diverse workload without compromising SLA and deadline. The proposed algorithm selects only optimal resources for processing the tasks and schedules

them over the VM instances to enhance the influential parameters. Further, it also takes some intelligent decisions based upon defined threshold values during the allocation of resources and avoids the possibility of an imbalanced workload. If any resource is found in an imbalanced condition, then transfer the workload to another VM instance. Some VM instances may be in slothful mode during the execution of tasks, then switch them to other modes (sleep or hibernation) to cut down the energy efficiency of the cloud datacenter. The proposed framework selects and allocates the optimum resources for the service keeping in mind, the service cost should not be surpassed to the users busget.

4.5 Mapping PMs with VMs: The efficiency of the proposed framework is also dependent upon the mapping between physical host and VM instances in the cloud datacenter. This is one of the critical issues due to heterogeneity in both virtual and Physical Machines (PMs), and the unpredictable demand of users. Virtual machine monitor (VMM) plays a significant role to deploy the multiple heterogeneous VM instances over the physical host through virtualization. The aim of VMM is to place the VM instances over the most suitable physical host and improve the utilization of resources that directly influence QoS metrics.

5. Proposed Enhanced SMO algorithm for Resource allocation

In this section, we depict the working mechanism of the SMO and enhanced SMO algorithm along with its mapping in the cloud environment (applications and resources). Further, the working mechanism of the enhanced SMO algorithm for the problem of resource allocation in the cloud paradigm is discussed.

5.1 Spider Monkey Optimization (SMO): Metaheuristic algorithms are used to serach approximate solution of the hard optimization problem in reasonable computation time. Existing meta-heuristic techniques presents better results in terms of quality and time in comparison with deterministic algorithms as discussed in related work. A new nature-inspired algorithm has been developed recently to search the real-world problem solution named spider monkey optimization (SMO). The algorithm follows the property of swarm intelligence and is inspired by intelligent foraging behavior. Usually, these monkeys tries to explore and search the best food source from several locations by exchaning the information [24-27]. We aim to search for the optimal solution of fitness that is represented by the distance of spider monkey from a food source. If they are near to the food source means the optimal solution is achieved otherwise, improvement is needed. The availability of the food is searched by the female member of the group, if there is not enough food then there is a possibility of food scarcity, which split the subgroups. Each group has a local leader and there is a global leader of the entire group. The process of fission-fusion is to divide the group into subgroups and fuse again into one group depending upon the quality of the solution. SMO algorithm consists of six iterative phases and four control parameters as depicted below:

Iterative steps of SMO algorithm are:

- **Initialization:** SMO generates a group of N spider monkeys ($SM_1, SM_2 \ldots\ldots SM_n$) with D dimensions in the search space, $R_U \sim$U(0,1)is uniformly distributed random numbers between 0 and 1 at the starting phase. Each monkey is initialized as

$$SM_{ij} = SM_{minj} + R_U(0,1)*(SM_{maxj} - SM_{minj}) \quad (24)$$

  Where SM_{ij} indicates the ith monkey of the population with jth dimension. SM_{maxj} and SM_{minj} are upper and lower bounds of search space.

- **Local Leader Phase (LLP):** This is an important phase in the SMO algorithms. Each monkey tries to update their position as per the local group members experience and local leaders. Further, the fitness value of new position is calculated, and upated if is better than the previous one as given in equation 25.

$$SM_{New,ij} = SM_{ij}+R_U(0,1)*(LL_{kj}-SM_{ij})+ R_U(-1,1)*(SM_{rj}-SM_{ij}) \quad (25)$$

  Where $LL_{kj} \in$kth local group leader and SM_{rj} is rth monkey which is arbitrarily chosen from kth group (r$\neq i$).

- **Global Leader Phase (GLP):**The entire monkey's update their position based upon the experience of global leader as well as local group member

$$SM_{New,ij} = SM_{ij} + R_U(0,1)*(GL_j - SM_{ij}) + R_U(-1,1)*(SM_{rj} - SM_{ij}) \quad (26)$$

On the basis of global and local leader member's experience, monkeys update their position, where GL_j point to position of global leader with j[th] dimension.

$$P_i = 0.9*\frac{f_i}{f_{max}} + 0.1 \quad (27)$$

Probability P_i plays an important role to update the location of monkeys and it is calculated using the value of fitness as mentioned in equation 26. Given equation state that a monkey has better chances to improve own position, if its fitness value is higher than other in the group.

- **Global Leader Learning Phase (GLL):** The greedy selection approach is applied to update the position of global leader. If global leader is not updated i.e., scarcity of food, then start the counter GLL and it is increased by 1 after every iteration.
- **Local Leader Learning phase (LLL):** In this phase, the algorithm uses greedy selection to find the best solution in the local group. If the solution is not improved then LLL count corresponding to the group is increased by 1 else it is set to 0.
- **Local leader decision phase:** If the position of LL is not improved after predefined instructions, then it can be improved either by random initialization or with the help of LL, GL and P_i as follows in equation 28.

$$SM_{New,ij} = SM_{ij} + R_U(0,1)*(GL_j - SM_{ij}) + R_U(0,1)*(SM_{ij} - LL_{kj}) \quad (28)$$

Equation 28 updates the position if $R_U(0,1) >= P_i$ otherwise equation 29 update the position

$$SM_{New,ij} = SM_{ij} + R_U(0,1)*(GL_j - SM_{ij}) + R_U(0,1)*(SM_{rj} - LL_{kj}) \quad (29)$$

- **Global leader decision phase:** The position of GL is monitored continuously, if it has not been updated in a predefined iteration (GLL), then GL reduced the size of the population in each group and start to explore the solution in more refined way.

5.2 Enhanced SMO algorithm: The conventional SMO algorithm try to make the balance between exploration and exploitation most of the times, but rate of convergence is an issue with SMO algorithm. Sometimes it traps in local minima and unable to update the solution in local leader phase. Hence, we have made some changes in local leader phase to enhance the performance of the algorithm and improve the convergence rate [28]. The updated algorithm performs better as compared with conventional SMO algorithm in the cloud environment and it is called enhance SMO algorithm. We modernize the equation of local leader phase by taking the average of both positions current as well as randomly generated that allocates a random position within a given range according to the problem. The convergence rate of the SMO algorithm is improved after performing these changes as given in equation no 30.

$$\gamma_{ij} = X_{ij} + R_U(0,1)\,(LL_{kj} - ESM_{ij}) + R_U(0,1)*\frac{SUM}{SN} \quad (30)$$

Where ESM_{ij} indicates the i[th] monkey of the population with j[th] dimension, SN is food source position, and SUM represents the difference between current and randomly generated position as shown in equation 31.

$$SUM = SUM + X_{ij} - X_{kj} \quad (31)$$

The enhanced SMO increases the possibility of a better balance between exploration and exploitation as compared with traditional SMO algorithm and deliver more reliable solution for the cloud environment. The proposed algorithm has been applied for resource allocation problems to search and allocate the best resource for services. Further, we

have observed from the computational results that the proposed framework provides a promising solution to end-user requests within deadline and outperforms state-of-the-art techniques.

5.3 Encoding technique of implemented SMO in cloud computing:

We cannot apply directly SMO approach in the cloud paradigm i.e., the algorithm cannot optimize required influential parameters without appropriate mapping. SMO technique follw the swarm intelligence approach and it is used to search for the approximate solution of NP-Complete problems. Our objective is to find the way that can encode (relate) the problem of resource allocation with SMO i.e., swarm contains the group of monkeys that search the optimal solution of problems in defined search space while resource allocation aim in a cloud environment is to deploy the diversity of applications (tasks) at heterogeneous virtual machines. The search space of resource allocation problem is considered as food searching area of SM's that consists of heterogeneous VMs, data centers (DC), physical host) and each solution (optimal VMs for the diverse tasks) of the problem is considered as the position of the monkey in the food searching area. How the "goodness" of a monkey is the measure, what a monkey can represent tasks, applications or resources or combination of task and resources (matrix form). It is essential to find the meaning and dimension of monkeys in SMO for developing the encoding scheme in a resource allocation problem. For this reason, we encode our problem in such a manner that a monkey represents the total number of tasks in a schedule that is equal to the dimension of the monkey. A coordinate system is used to determine the dimension of the monkey. For example, Table I represents 10 co-ordinates system i.e., the dimension of the monkey is 10 that is equal to the total number of tasks submitted by end-users in a schedule. Diverse tasks are deployed at heterogeneous resources (VMs) and the number of VMs in the datacenter indicates the range of monkey movement in the co-ordinate system. The range of co-ordinate is from 0 to available virtual machines that are ready to execute the tasks in a cloud environment for a particular schedule. The position of the monkey encodes a mapping of tasks with running virtual machines. We have considered 5 virtual machines (range of co-ordinate system is 0 to 4) to execute the 10 tasks as shown in table I. Mapping of upcoming tasks with running virtual machines is shown in Table II. Task $£_1$is deployed at virtual machine ρ_2 in a host; task $£_2$ is deployed at the virtual machine ρ_3 and so on shown in table II.

Table I Position of the monkeys

C_1	C_2	C_3	C_4	C_5	C_6	C_7	C_8	C_9	C_{10}
2	3	4	2	2	1	3	0	4	0

Table II Mapping task with cloud resources

$£_1 \rightarrow \rho_2$	$£_2 \rightarrow \rho_3$	$£_3 \rightarrow \rho_4$	$£_4 \rightarrow \rho_2$	$£_5 \rightarrow \rho_2$	$£_6 \rightarrow \rho_1$	$£_7 \rightarrow \rho_3$	$£_8 \rightarrow \rho_0$	$£_9 \rightarrow \rho_4$	$£_{10} \rightarrow \rho_0$

We have considered n number of tasks $T_N = £_1, £_2, £_3 \ldots £_n$ in a schedule S_p that request for m number of heterogeneous VMs (cloud resources) to execute the requested tasks as discussed in section 3. All the requested VMs can be created at a single PH or multiple PH in a datacenter, it is depending upon the configuration (computational power) of particular PH. In the cloud environment, the workload is submitted with a deadline $\partial(£_i)$ for the execution, where each tasks $£_1 \in ST_{£_i}$ has start time as well as $£_1 \in PT_{£_i}$ has end time (processing time).

5.4 Working principal of proposed framework for scheduling and allocation using enhanced SMO

The first step is to search for the best mapping between users' demand and available resources at the cloud datacenter. This mapping is done based upon several parameters such as the budget of users, deadline constraint, availability of resources, provisioning plan chosen by users, priority of applications, SLA violation, mapping PH with VMs, and other QoS parameters.

Definition 1: Specification of users and their demands

Suppose there are N users or customer C = $\{c_1, c_2, c_3 \ldots\ldots c_N\}$ and their finite demand are D = $\{d_1, d_2, d_3 \ldots\ldots d_x\}$. The demand of user is a function of time t and is represented by 5 significant attributes $\ell(£_i)$, a($£_i$), $EET_{£_1}$ $\partial(£_i)$, $B_{£_i}$.

These attributes indicate the length of i[th] task, arrival time as well as expected time of task execution, user defines the deadline and budget of the end-user for the services. Each user can submit their multi-dimensional demand anytime for the services. Customer $c_1$can request for multiple VMs with specification 4GB memory, 2 core and 10 GB storage, other users c_2 may demand for10 processing core, 16 GB memory, and 200 GB storage. Hence the demand of each user is different in the form of required resources and may be changed (expand or shrink) at the run time also. The total demand of customers D_c^+ can be represented in the form of equation 32

$$D_c^+ = \sum_{i=1}^{x} d_i \tag{32}$$

Hence, it is one of the challenging issues to provision the best resources to end-user because of the multidimensionality and heterogeneity of resources.

Definition 2: Processing Capability of Datacenter and Resources

There are k datacenters are available to process the request of end-user in the availability zone. Each datacenter (DC) consists of M physical host PH= $\{ PH_1, PH_2, PH_3 \ldots\ldots\ldots PH_M \}$. We are considering only 3 important resources R = $\{r_1, r_2, r_3\}$ of a host that satisfies the demand of the user, where r_1, r_2, r_3represents the processor with core, primary memory, and storage. The total computation capability of a DC is calculated by equation 33

$$DC_j = \sum_{j=1}^{M} R_j \tag{33}$$

Total computation capability of an availability zone C_{az}that consists of k server

$$C_{az} = \sum_{j=1}^{k} DC_j \tag{34}$$

We would like to find that C_{az} have enough resources to satisfy the demand of customers i.e.

$$\sum_{i=1}^{x} d_i \leq \sum_{j=1}^{k} DC_j \tag{35}$$

We assumed that the location of servers and users are scattered in a large geographical area. Additional constraints like bandwidth and latency may influence the performance and services of the cloud, but we are not considering these parameters as performance measurement parameters.

Definition 3: Resource mapping and provisioning

Demands of each user d_i are analyzed at the time of sla and service providers try to provision the best resources for the computation or storage as per their budget. Moreover, the service provider should select the resources in such as way that the total cost of resources should be less than the budget of the customer

$$B_{£_i} >= TEC_{£_i}^{\rho_j} \tag{36}$$

Algorithm 1: Resource Provisioning
Input: Set of task requests
Output: Selection of resources R_j based upon the budget and deadline of users

1. Start
2. Available host in datacenter $PH_2, PH_3 \ldots\ldots\ldots PH_M$
3. Total VMs at datacenters $\rho_1, \rho_2, \rho_3 \ldots \rho_m$
4. Calculate the demands of user applications ($£_1, £_2, £_3 \ldots £_n$) for the resources D_c^+ (CPU, memory, storage).
5. Demand should be less than available resources $\sum_{i=1}^{x} d_i \leq \sum_{j=1}^{k} DC_j$
6. Price of resources should not be greater than the budget of customer $B_{£_i} >= TEC_{£_i}^{\rho_j}$
7. If condition is satisfied
8. Then Start the mapping of resources with workload

   $PH_{mem_j} \geq MM_{mem}$

   $PH_{cpu_j} \geq p_s * \eta_{cpu}$

   $PH_{s_c} \geq R_{s_c}$

9. Resources is provision, only if $ET_{£_1\rho_j} \leq RT_{£_1\rho_j}$
10. If the condition is satisfied then $W_{\rho_j} = W_{\rho_j} + ET_{£_i\rho_j}$
11. Else generate the alert, resources cannot provision
12. Users will wait until the best possible resource is not found at other datacenters
13. If all the condition is satisfied successfully then resources are provided to end-user within their budget.
14. End

After the confirmation of the customer budget, the service provider searches for the optimal host or VMs to execute the service. It is time-consuming as well as very complicated to map and verify every request with existing m resources for the execution. If we try to map and execute n request over m resources within deadline then there will be m^n combination i.e., it becomes an NP-hard problem. In addition, it is challenging to solve this problem in a cloud environment where customers submit a large number of requests over the internet for the services anytime. Meta-heuristic-based scheduling algorithm is proposed for a cloud environment that assigned the tasks to VMs in a competent manner and optimizes the QoS parameters without violating the constraints like deadline and budget. Additionally, the proposed algorithm searches the best possible resource (under-loaded or idle VMs) with the help of fission and fusion mechanism that can execute the tasks in a predefined end-user budget.

Definition 4: Scheduling of resources belongs to the class of NP-hard problems

Let's consider a simple example of a scheduling problem where each host consists of R_j resources and computation capability of DC_j is defined in equation 33. Our goal is to minimize the $TPT_{£_i}^{\rho_j}$ that depends upon binary variable $\lambda_{£_i\rho_j}$ and equal to $mst_{\rho_j} = \sum_{i=1}^{n} \lambda_{£_i\rho_j} * \frac{\ell(£_i)}{p_s * \eta} + \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{m} X_{ijk} * \frac{\mathrm{VDE}_{jk}}{B_{jk}} + \sum_{i=1}^{n} WT_{£_1\rho_j \in \theta_{£\mathrm{isp}}}$. Further, we want to optimize cost and energy simultaneously with time. Hence, we have defined the problem in the form of the equation

$$\min(mst_{\rho_j} + TEC_{£_i}^{\rho_j} + E_{\text{actual}}) \quad (37)$$

Subject to: $\sum_{j=1}^{m} \lambda_{£_i\rho_j} = 1$ (38)

$$\sum_{i=1}^{x} \lambda_{£_i\rho_j} * RRV_{\rho_j} \leq Cap_{\rho_j} \quad (39)$$

$$B_{£_i\rho_j} >= TEC_{£_i}^{\rho_j} \quad (40)$$

Where Cap_{ρ_j} and RRV_{ρ_j} represent the capacity of resources and Normalized Remaining Resource Vector [10]. This can be calculated using equation 41

$$RRV_{\rho_j} = 1 - RU_{\rho_j} \quad (41)$$

Where RU_{ρ_j} measure the is cpu utilization of resources.

$$RU_{\rho_j} = \sum_{j=1}^{m} \lambda_{£_i\rho_j} * \frac{R_j}{Cap_{\rho_j}} \quad (42)$$

VMs utilization constraints:

$$\forall(\rho_i, \mathrm{a}(£_i), ET_{£_i}) \in s_p, \text{ if } \exists(\rho_j, \mathrm{a}(£_j), ET_{£_j}) \in s_p \text{ and } \rho_i = \rho_j \quad (43)$$

then $[a(£_i), a(£_i)+ET_{£_i}] \cap [a(£_j), a(£_j)+ET_{£_j}]=\varphi$ (44)

Our objective is to search the optimum combination between upcoming end-users requests and existing cloud resources i.e. $\lambda_{£_i\rho_j}$. User's demand can be mapped with more than one resource, but an enhanced SMO-based approach searches the best one based upon the defined constraint like deadline and budget. Most of the existing resource allocation approaches focused on single-objective optimization problems and do not discuss the solution of the multi-objective optimization problem (MOP). In this research work, we have considered several QoS parameters, some of which are conflicting parameters like time, service cost, and energy consumption. If the value of any one mentioned objective increases, it often leads to the expense of other conflict objectives. It is more complicated to search the solution of MOP as compared to a single objective, and problems come under the class of NP-Hard. Hence, the proposed adaptive framework has decision-making capability at the time of resource allocation and finds the acceptable (compromise) solution for MOP using the enhanced SMO technique.

Definition 5: A resource allocation algorithm is feasible if it satisfied the given conditions:

a) If $\forall £_i \in \{£_1, £_2, £_3 \ldots £_n\}$ complete their processing before the deadline of tasks $\partial(£_i)$.
b) Tasks are allocated to resources based upon their computational capacity and required resources by tasks to avoid the possibility of over or under utilization.
c) Charges for cloud services should not exceed the limit of the end-user budget.

Definition 6: Monitoring the resources of datacenter

Suppose a datacenter consists of m virtual machines to execute the n applications, where n>m and resource allocation table RAT_m is used to represent the status of resources (busy, idle). RAT_m consists of the record of every resource with multiple attributes like start time ($ST_{£_i}$), required cpu ($\eta_{£_i}$), memory (MM_{mem}), storage (s_c) and deadline $\partial(£_i)$. RAT_m is a data structure and it is updated dynamically based on some events like allocation of the new task at VMs or departure the task after the execution or task is transferred to another VM. When a task request reaches to datacenter for the services, it needs multiple resources to complete the execution, and the upcoming task (application) request is mapped with cloud resources i.e., which resources can accomplish the demand of end-user most efficiently. After the selection of the best resource, it is allocated to the user's demand, and the status of the resources is changed after the allocation and record of resources is also updated in RAT_m.

Algorithm 2: Adaptive Monitoring approach for cloud datacenter

1. Monitor the status of all the resources
2. Calculate the RAT_m
3. $\forall \rho_j \in \{\rho_1, \rho_2, \rho_3 \ldots \rho_m\}$ of a datacenter (DC)
4. A task $£_i$ is submitted with multiple attributes
5. Demand of $£_i$ is the map with the $\forall \rho_j$
6. Select the optimal resource for the service
7. Generate the record for each task request $£_i$
8. Avoid the possibility of over or under utilization
9. If any VMs consist of a higher workload than their threshold limit then it is transferred to underloaded VMs.
10. Availability of services or execution of application should be continued after the failure of a datacenter
11. If any virtual machine is in an idle state, then it is switched to a sleep state to cut down the power consumption.

The number of idle VMs can be calculated by RAT_m in a finite time interval. Further, idle resources are switched to sleep mode to cut down the energy effificncy, actually idle resources consumed around 50-60% energy of their total energy consumption in a peak state. The key feature of the monitoring node is described in algorithm 2.

Several heuristic and traditional scheduling techniques (such as sorting the task or resources in ascending or descending order and starting to map with optimal resources for the tasks) fail to provide the optimal solution to a hard computational problem. Hence, the author's move to a randomization-based meta-heuristic algorithm that searches the best possible resources for the tasks within the deadline and budget of the end-user. Hence, the key difference between our approach and the existing baseline approach is to improve the significant QoS parameters by assigning the optimal resources for the services and update the status of resources (idle, over or under load) continuously.

- **Step by step description of enhanced SMO algorithm**

The proposed algorithm is informally divided into five parts: the first part is dicates about the mathematical model for the defind problem, where we have defined the objective functions for the influtial parameters with constraint and evaluate the fitness function evaluate to measure the performance of algorithm based on influential parameters.

Algorithm 3: Enhanced SMO based scheduling algorithm

Input: Complex and diverse applications (tasks),
Output: optimize the objective and fitness function to improve the QoS parameters

Part I: Generate the tasks and VMs

1. **Start**
2. Formulate the objective and fitness function to improve the QoS parameters
3. Generate the workload $£_i, £_2, £_3 \ldots £_n$
4. Create m resources $\rho_1, \rho_2, \rho_3 \ldots \rho_m$ in datacenter

Part II: Declaration of variable

5. k represents the number of iterations, $\partial(£_i)$ represent the deadline of task
6. N = Number of Spider Monkeys
7. SM_i=ith monkey
8. SM_{new}=used to update a spider monkey
9. D= Dimension of optimization problem, MG=Max Number of Groups
10. Global Leader Limit (GLL)=2*N, Local Leader Limit (LLL)=D*N;
11. Min_monkey=10 //no of monkey in a group
12. pr =Perturbation rate

Part III: Initialization of monkeys

13. **For** i=1 to N //initialize the spider monkeys
14. **For** j =1 to D
15. Each $SM_i$with D dimension (randomly allocate the tasks to VMs)
16. $SM_{ij} = SM_{minj} + R_U(0,1)*(SM_{maxj} - SM_{minj})$
17. $SM_{New,ij} = SM_{ij} + R_U(0,1)*(LL_{kj} - SM_{ij}) + R_U(-1,1)*(SM_{rj} - SM_{ij})$ //Local leader phase
18. $SM_{New,ij} = SM_{ij} + R_U(0,1)*(GL_j - SM_{ij}) + R_U(-1,1)*(SM_{rj} - SM_{ij})$ //Global leader phase
19. Calculate the value of objective and fitness function using the equations
20. $f(\rho_j \in \lambda_{£_i\rho_j}) = (EET_{£_1\rho_j \in \theta_{£iSp}}, EEC_{£_1\rho_j \in \theta_{£iSp}}, E_{\text{actual}})$
21. End for loop
22. End of loop

Part IV: Improve the fitness function with the help of LLL, GLL and pr

23. **while**($K_{max} \neq \emptyset$ **)**
24. **Do**
25. **For** i=1 to N **(** $\forall$N in $\theta_{£iSp}$ **)** // for all the monkeys
26. **For** j =1 to D
27. $SM_{New,ij} = SM_{ij} + R_U(0,1)*(LL_{kj} - SM_{ij}) + R_U(-1,1)*(SM_{rj} - SM_{ij})$ //Local leader phase
28. **If** $f(\rho_j)_{new} < f(\rho_j)_{old}$
29. $f(\rho_j) = f(\rho_j)_{new}$

30. **If** solution is not updating in predefined iterations
31. **then** calculate γ_{ij} to improve the convergence rate
32. **Else** repeat the same
33. $SM_{New,ij} = SM_{ij} + R_U(0,1)*(GL_j - SM_{ij}) + R_U(-1,1)*(SM_{rj} - SM_{ij})$ //Global leader phase
34. $P_i = 0.9*\frac{f(\rho_j)_i}{f(\rho_j)_{max}} + 0.1$
35. **If** $f(\rho_j)_{new} < f(\rho_j)_{old}$
36. $f(\rho_j) = f(\rho_j)_{new}$
37. Apply the greedy selection process to find the better value // global & Local Leader Learning
38. If value of $f(\rho_j)$ is not improve
39. Then GLL++ and LLL++;
40. If $R_U(0,1) >= P_i$ // local Leader Decision Phase
41. Then $SM_{New,ij} = SM_{ij} + R_U(0,1)*(GL_j - SM_{ij}) + R_U(0,1)*(SM_{ij} - LL_{kj})$
42. Else $SM_{New,ij} = SM_{ij} + R_U(0,1)*(GL_j - SM_{ij}) + R_U(0,1)*(SM_{rj} - LL_{kj})$
43. If $f(\rho_j)$ is not improved in 2*N iteration *//Global leader decision phase*
44. If Number of groups < MG
45. Then Divide the population into groups //fission
46. Else Combine all the groups to make a single group //fusion
47. End if Update Local Leaders position

Part V: Termination condition

48. Repeat until all the tasks of a schedule are not executed
49. K_{max}--;
50. End of for loop
51. while loop
52. Return optimal value of $f(\rho_j)$

In addition, cpu intensive workload and heterogeneous resources are generated in line 3-4. Control parameters of SMO is declared and initialized in the second part from line 5 to 12 in algorithm 3.

- N Monkeys are initialized with their dimension D in line 13-14.
- Diverse tasks are allocated to VMs randomly at the starting in line 15.
- The number of diverse tasks is equal to the dimension of the monkey.
- Monkeys started to explore the search space to find the solution of the problem after the initialization.
- All the monkeys get their position and calculate the global as well as local leader in line 17-18.
- The value of the multi-objective fitness function is calculated in line 20 and the aim of the algorithm is to improve the QoS parameters that are defined in the fitness function.
- Part IV of the algorithm optimizes the fitness function based upon some control parameters.
- While loop is started from line 23 and continuously repeats the procedure up to line 48 to find the best possible VMs for the tasks.
- For loop is for monkeys and their dimension in line 25-26
- Line 27 computes the new position of monkeys on the basis of the local leader and local group member experience.
- Line 28-32, if fitness value $f(\rho_j)_{new}$ is better than $f(\rho_j)_{old}$ then monkey updates the position with a new one.
- Line 33-34, update the position of monkey in search space based upon the probabilistic value P_i and global as well as local members.
- Line 35-36, if new position of monkey (VMs) is better than the old one then replaces it with the new value.
- Line 37-38, apply the greedy selection procedure to improve the fitness value, increase the value of GLL and LLL, if the position (VMs) is not updated.

- The possibility of premature convergence or stagnation is avoided by re-initialization the local leader in line 40-42.
- Global leader decision phase performs the fission and fusion operation based upon the condition in line 43-47.
- This procedure is repeated until all the tasks are allocated to the best possible VMs for the execution as illustrated in part 5.
- Line 50-51, End of for & while loop
- At the last, we achieved the optimal value of the fitness function.

5. Analysis and comparison of experimental results

5. 1 Simulation setup

The simulation toolkit CloudSim 3.03 along with eclipse IDE is used to assess the performance of the proposed framework in a cloud environment [30]. Simulation is the best way to assure and validate the computational results of the developed technique and baseline approaches. The CloudSim offered all the functionality (VM placement, power model, datacenter broker, etc.) and environment (datacenter, physical host, VMs) to implement the scheduling and allocation strategy. Hence, CPU-intensive synthetic workload is generated and heterogeneous virtual machines are deployed over the physical resources (after mapping PM to VMs) based upon the user demand as well as computational capability of the datacenter. In addition, Amazon EC2 instances are also considered to evaluate the proposed framework performance in a real cloud environment. There are lots of QoS parameters that exist in the cloud paradigm for performance evaluation, but we have chosen some significant parameters based upon the service provider and end-user perspective. To validate the results, the performance of the proposed and baseline algorithm is tested at each significant parameter. Simulation setup for the synthetic workload is shown in Table III and configuration of on-demand AWS EC2 instances along with service cost is shown in Table IV. Table V represents the range of parameters of the proposed algorithm, i.e., enhanced SMO in the simulation environment to compute the QoS metrics.

Table III Simulation setup for synthetic workload

Parameter	**Setting**
DC	1
Number of hosts	[1, 40]
VMs	[1, 2000]
Virtual machine processing rate	[100, 1000] in MIPS
Synthetic workload (Cloudlets)	[1, 6000]
Cloudlets lengths	[200000 to 500000] in MI

Table IV Configuration of AWS EC2 VMs

VM	**Platform**	**vCPU**	**Memory**	**Storage (GiB)**	**Instance Storage**	**Price**
t4g.micro	Linux/UNIX	1	1	8	EBS Only	$0.0084 per Hour
a1.medium	Linux/UNIX	1	2	16	EBS Only	$0.0255 per Hour
a1.large	Linux/UNIX	2	4	32	EBS Only	$0.051 per Hour
a1.xlarge	Linux/UNIX	4	8	64	EBS Only	$0.102 per Hour

5.2 Simulation Results and Discussion

In the cloud environment, resource requirements (CPU, memory, storage) of every cloudlet (task) are different which leads to a heterogeneous workload model. To handle such situations, diverse tasks, and heterogeneous VM instances, as well as Amazon EC2 instances, have been considered. The critical parameters of VM instances vary depending upon the type, for eg. processing speed varies from 100 MIPS to 1000 MIPS, and range of memory is from 1-8 GB,

variation in cost is the function of time (per hour basis). These VMs are capable of processing diverse tasks as well as complex applications without violating the deadline and SLA using enhanced SMO-based scheduling techniques. : End user request can come at any time (continuous form) for the services at cloud platform, but it is converted to discrete form before the mapping i.e., number of requests can be store in a task queue for a timespan i.e., number of upcoming tasks are collected for a time interval and schedule the tasks over the VMs using the proposed algorithm [44]. The proposed algorithm helps service providers as well as end-user to meet their objectives by searching for the most suitable resource and assigning them to the upcoming task.

Table V SMO parameters for setting the simulation environment

Parameter	**Setting**
Population size (Number of Monkeys)	[10, 100]
Monkeys in a group	[10, 12]
Dimensions D of each Monkey	Number of Tasks
Number of iterations	[200]
Perturbation rate (pr)	[0.1, 0.9]
Global Leader Limit	[N/2, 2*N]
Local Leader Limit	[D*N]
Maximum group	[10]

The proposed enhanced SMO-based algorithm has the advantage over other population-based algorithms such as PSO, ABC, GSA, and GA on the QoS parameters like premature convergence, stagnation, trapping in local minima, etc [27]. ABC failed to explore the full search domain, while PSO optimizes the objective function at a faster rate but it sometimes traps in local minima and GSA is unable to get global optima due to less diversity. Furthermore, the baseline approaches hese algorithms are incapable to balance between exploration and exploitation. Hence, an enhanced SMO algorithm has been deployed to get the faster solution of fitness function without compromising the accuracy and efficiency. A comparative analysis of the performance of the proposed algorithm with the other meta-heuristics algorithms has been done in a simulation environment at some significant influential parameters such as makespan time, service cost, degree of load balancing etc. The outcome of simulation-based results endorses that the developed algorithm surpass to its rivals algorithm in terms of mentioned paramteers.

5.2.1 Makespan Time

Initially, the proposed framework performance is assessed at makespan time influential parameters using diverse applications/tasks and heterogeneous virtual machines. Makespan time is one of the significant parameters in the cloud paradigm that represent the running time of cloud resources for the processing of upcoming demands. The proposed enhanced SMO algorithm has been used to schedule and allocate the tasks to virtual machines using synthetic data, without compromising the budget and deadline constraints.

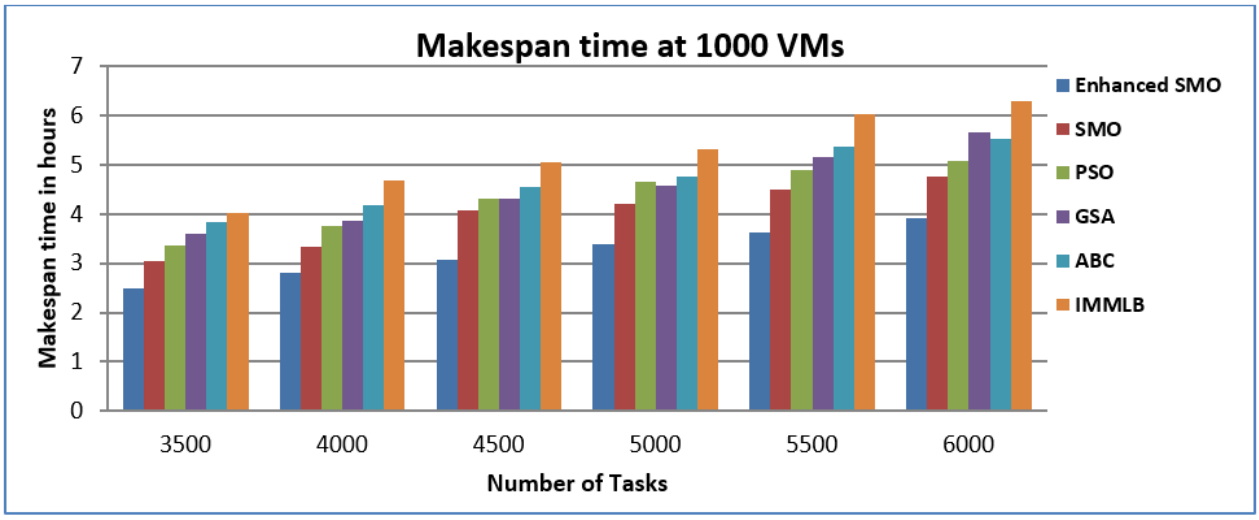

Figure 4 makespan time comparisons at variable tasks

Test case 1: we have considered 1000 heterogeneous VM instances to execute the elastic dynamic requests (3500 to 6000) in S1 and computational results are calculated using simulation toolkit. The same dataset and parameter is considered for comparative analysis of proposed enhanced SMO and its other competitors like standard SMO [24], PSO [14], GSA [18], ABC [11], and IMMLB [6]. The output of the computational experiment demonstrated that the proposed enhanced SMO-based framework significantly reduced the average makespan time up to 18.44%, 26.29%, 31.07%, 35.10%, and 38.12% as compared with PSO, GSA, ABC, and IMMLB algorithms as shown in figure 4.

To validate the performance of the developed algorithm, five more different schedules (S2 to S6) have been generated that consist of diverse workloads. Tasks of each schedule have been mapped to 1000VMs with different processing speeds to compute the average makespan time. The experimental results of almost every schedule (shown in figure 4) proved that the proposed enhanced SMO algorithm significantly outperforms the state-of-art approaches.

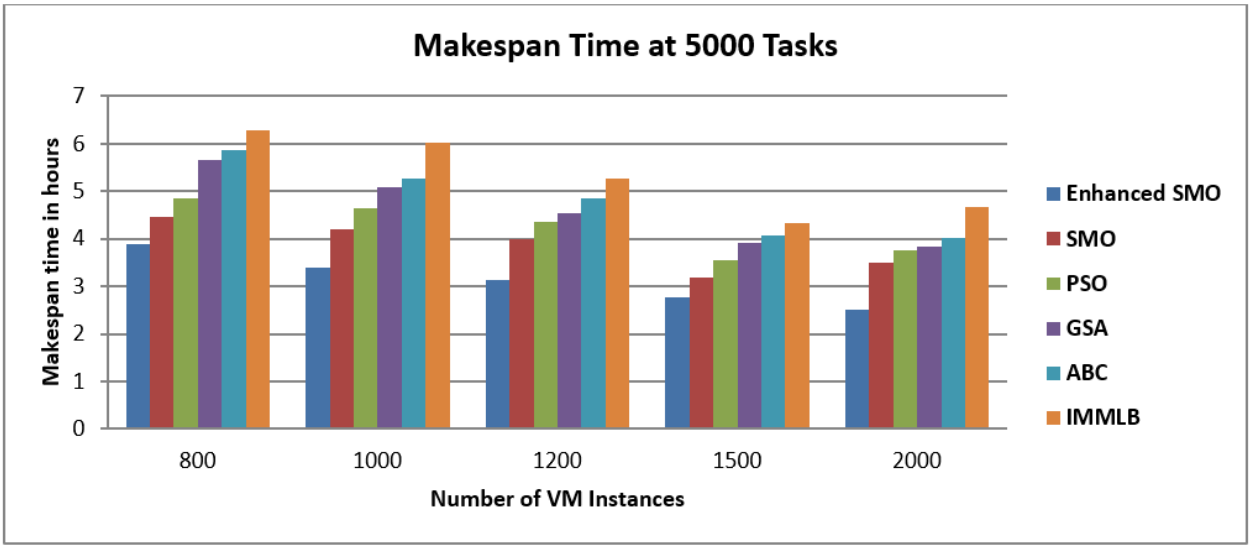

Figure 5 makespan time comparisons between algorithms at variable VMs

Test case 2: Another synthetic dataset with diverse tasks and heterogeneous cloud resources is generated, where the number of tasks is fixed (5000) but VM instanced are varying from 800 to 2000 in this computational experiment. Further, the proposed enhanced SMO-based algorithm has been applied to the mentioned synthetic dataset to assess

its efficacy. The experiment is conducted in which makespan time of tasks is decreased as the number of VMs instances is increased i.e., a smaller number of tasks is allocated to each VM by scheduling algorithms. To validate the superiority of enhanced SMO-based proposed framework is compared with baseline approaches. The experimental results shown in figure 5 proved that the proposed framework schedule and allocate the tasks in a more significant way as compared to baseline approaches (SMO, PSO, GSA, ABC, and IMMLB) existing in the cloud environment.

5.2.2 Services Cost

Figure 6-8, represents the analysis based on service cost w.r.t. different schedules in varying load conditions executing over the heterogonous VM instances. Figure 6, represents the simulation results of the proposed model on different schedules with a dynamic scenario where the number of tasks as well as VM instances are increasing. We have considered AWS Elastic compute cloud (EC2) instances (as described in Table II) to measure the performance of the proposed adaptive framework.

Test case 1: Initially, a synthetic dataset of 1000 tasks are generated and allocated to 300 heterogeneous VM instance using enhanced SMO and others baseline technique to assess the performance. The results of computational experiments verify that enhanced SMO-based algorithm execute the end-user tasks in less cost as compared to other standard algorithms like PSO, GSA, ABC, and IMMLB. Further, the end-user workload increased gradually from 1000 to 35000 and scheduled over the VMs from 300 to 800 using the mentioned scheduling algorithms. The outcome of the entire schedule proved that the proposed enhanced SMO algorithm significantly outperforms over state of art techniques for service cost metrics.

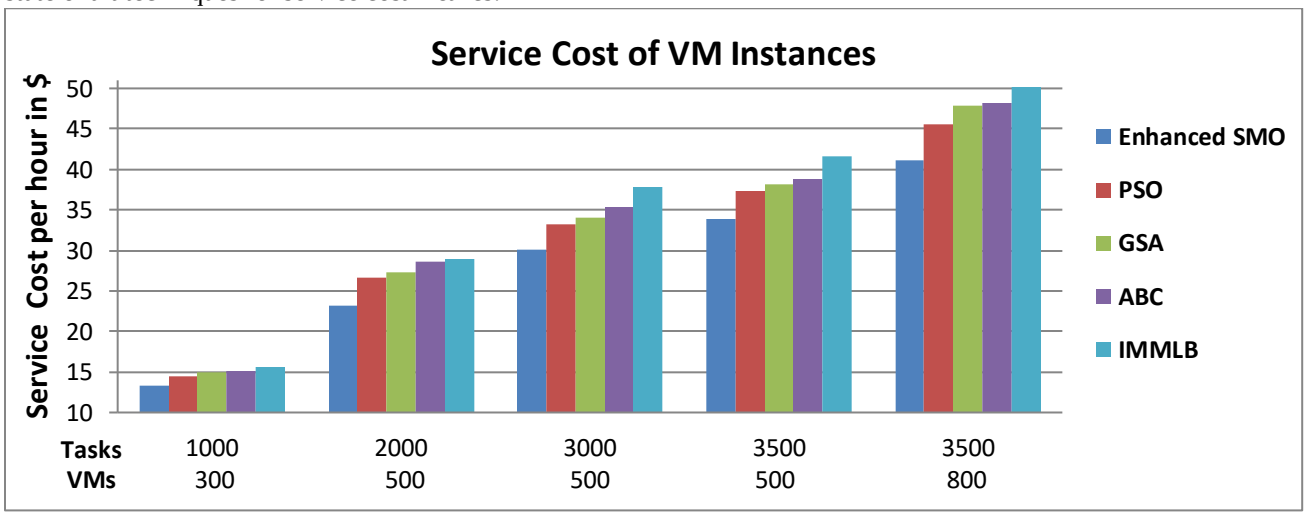

Figure 6 Service cost of VM instances

Test case 2: Next experimental scenario has been conducted to evaluate the consistency of performance in which the fixed number of VM instances is added in the datacenter while the workload is increased gradually from 3500 to 6000. The simulation has been run more than 10 times to calculate the average service cost of user requests using developed and others baseline algorithms. The computational results shown in figure 7 demonstrate that enhanced SMO-based algorithm cut down the budget of user requests up to 12.41%, 17.46%, 18.82%, and 20.91% in contrast with state of art PSO, GSA, ABC, and IMMLB algorithms.

Test case 3: The developed framework performance is computed at the fixed workload that is deployed over varying VM instances from 800 to 2000 to execute the demand of end-users. Simulation-based results shown in figure 8 confirm that the service cost of end-user requests is reduced by the developed algorithm in a more efficient way than other baseline algorithms.

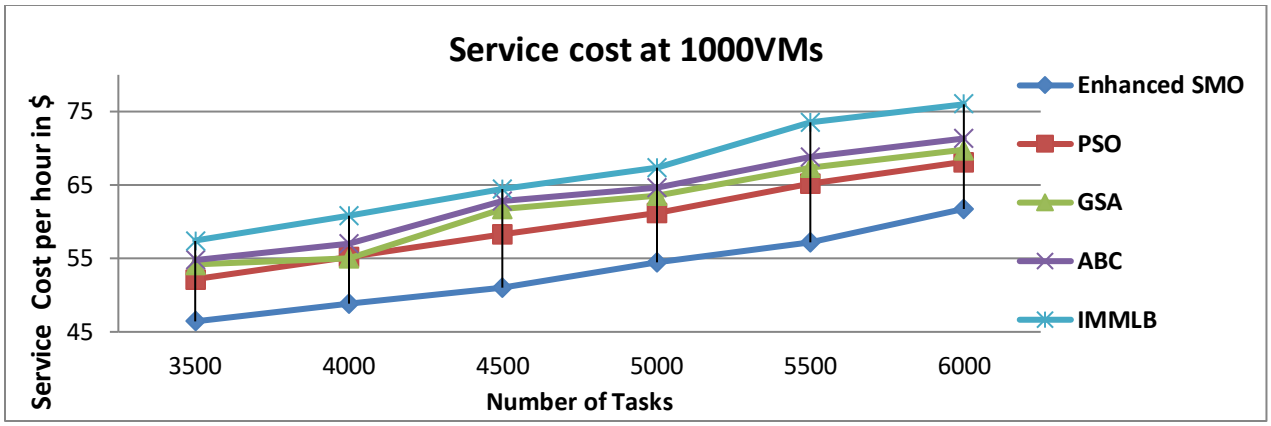

Figure 7 Service cost at variable workload

5.2.3 Task Rejection Ratio ($£_{RR}$): This is based upon the deadline of end-users and computed using equation 45. We have conducted some test case to assess the success rate of developed SMO based approach and compared with rival approaches to validate the results. Initially, 5000 tasks are scheduled over 800 VMs using the proposed algorithm to calculate the task rejection ratio where the range of deadline is 500 to 3000 seconds. The simulation result of the first schedule indicates the high task rejection rate of the enhanced SMO algorithm because they are running at a smaller number of VM instances, but it is better than the other state of art approaches. Hence, the proposed framework searches the optimal virtual machine for the diverse user requests allocation that reduces the percentage of task rejection.

$$£_{RR} = \frac{\text{No.of rejected applications}}{\text{Total number of applications}} * 100 \quad (45)$$

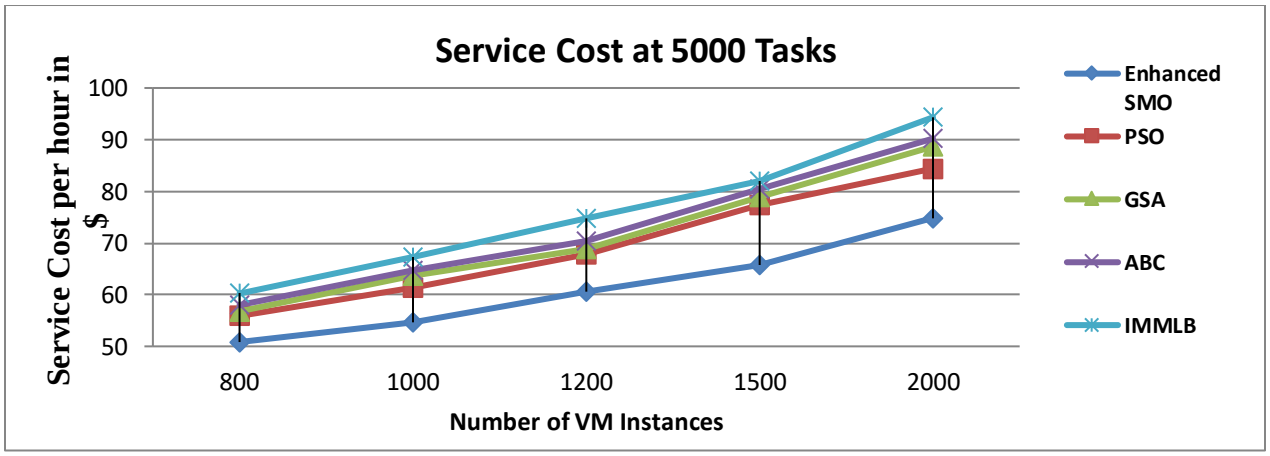

Figure 8 Service cost at variable VM instances

Furthermore, the performance of enhanced SMO-based algorithm is evaluated by varying the number of VMs from 1000 to 2000 and compared with state of art methods. The tasks rejection ratio of the proposed algorithm is decreased as the number of VMs is increased as shown in figure 9. Computational results of the entire schedule proved that the rate of task rejection in enhanced SMO-based algorithm is less than compared to others baseline approaches shown in figure 9.

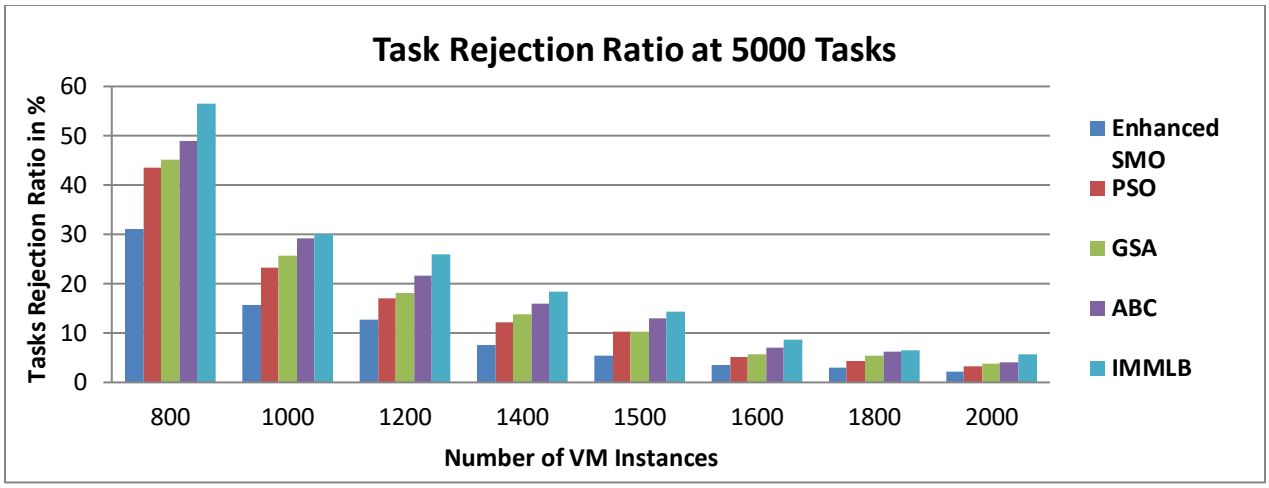

Figure 9 Task rejection ratio comparisons of proposed SMO and other algorithms

5.2.4 Degree of load balancing (DLB): This QoS metric is used to check and confirm the load distribution over the cloud resources assigned by the proposed and existing state of art approaches based upon the capacity. Equation 46 represents the way to calculate the DLB.

$$DLB=\frac{ET_{\rho_j}\max - ET_{\rho_j}\min}{ET_{\rho_j}\text{avg}} \tag{46}$$

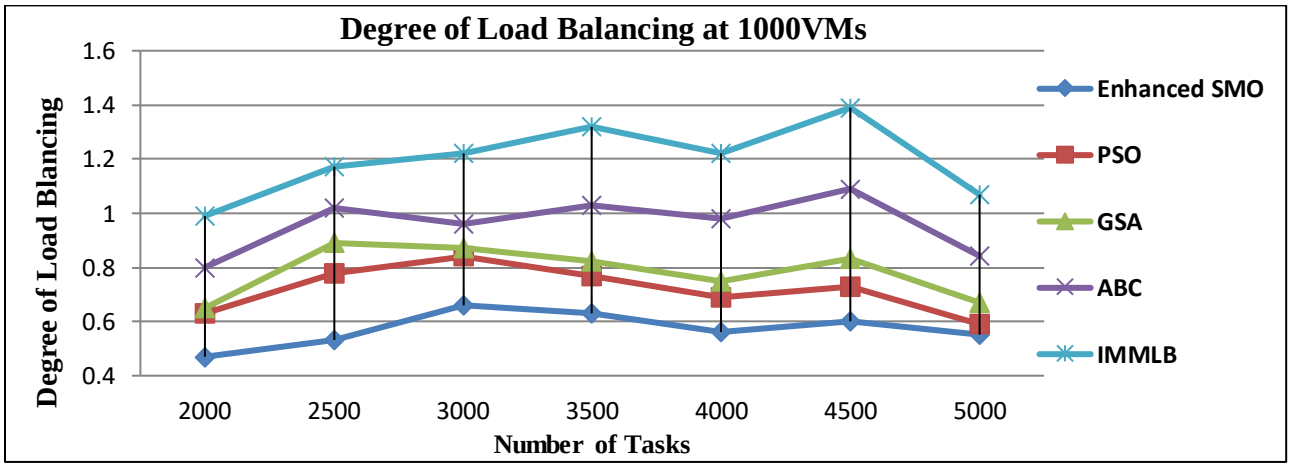

Figure 10 DLB at fixed VM Instances

$ET_{\rho_j}\max$ and $ET_{\rho_j}\min$ represents the maximum and minimum execution time of resource ρ_j in a schedule. $ET_{\rho_j}\text{avg}$ is the average execution time of all the resources. The objective of the proposed algorithm is to reduce the degree of load balancing by balancing the workload among the running instances. Initially, the degree of load balancing is calculated at 2000 tasks that are deployed over 1000VMs using an enhanced SMO algorithm for the execution. The results of enhanced SMO are compared with the baseline algorithm to verify the performance of the algorithm. Additionally, the consistency as well as efficiency of enhanced SMO based algorithm is tested by increasing the tasks from 2000 to 5000 without changing the VMs.

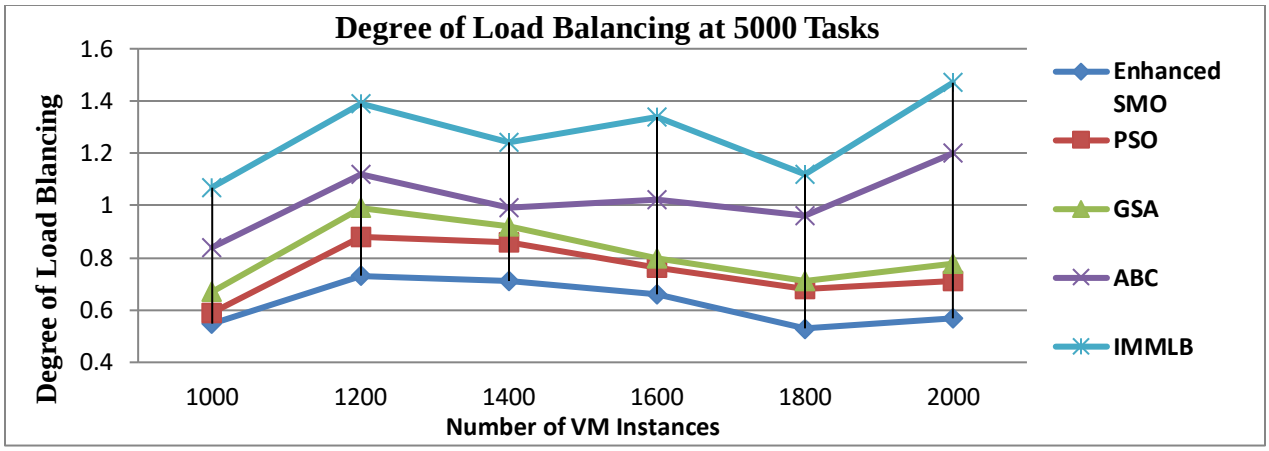

Figure 11 DLB at fixed tasks

The computational results (shown in figure 10) of the entire schedules proved that DLB of the proposed algorithm significantly outperforms over baseline approaches. In addition, one more synthetic dataset is generated that consists of fixed 5000 tasks and varies the virtual machine instances from 1000 to 2000. The effectiveness of the enhanced SMO-based algorithm is calculated at 1000 VMs at the starting and the same procedure is repeated up to 2000 VMs. We run the simulation for six schedules and get the results (shown in figure 11) that proved the proposed enhanced SMO-based algorithm achieved the better degree of load balancing in all the schedules as compared with PSO, GSA, ABC and IMMLB algorithms.

5.2.5 Throughput (Γ): It is calculated by the number of applications executed in unit time and computed by equation 47.

$$\text{Throughput } (\Gamma) = \frac{\text{successfully execution of Tasks}}{Total\ processing\ time} \tag{47}$$

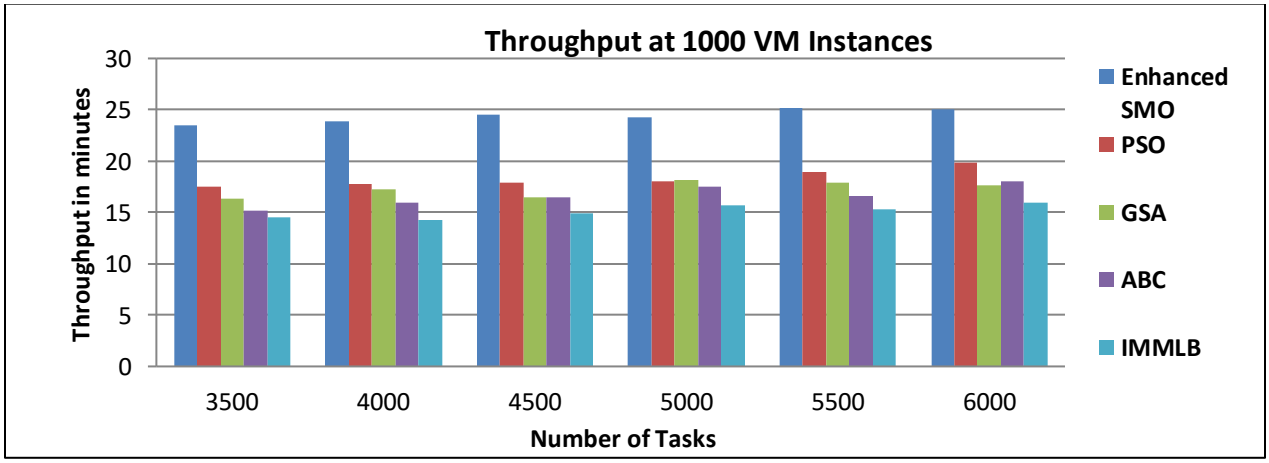

Figure 12 Throughput of proposed and other algorithms

Throughput is used to measure the performance of the proposed framework, if it is high then it can process a large number of applications in a short period of time and improve user satisfaction as well as the quality of cloud services. Hence, we have conducted six experiments to assess the performance of the proposed framework. Initially, 3500 tasks is scheduled over 1000VMs and run the simulation at least 10 times for every schedule and obtained the average throughput of the system using the enhanced SMO algorithm as well as other state-of arts. Further, the number of tasks is increasing 500 in every schedule and allocated to 1000 fixed numbers of heterogeneous VM instances using presented algorithm and baseline algorithms. The simulation results reported in figure 12 illustrate that the throughput

of the proposed enhanced SMO-based algorithm is healthier than PSO, GSA, ABC, and IMMLB algorithms. Hence, the enhanced SMO-based algorithm achieved more throughput by allocating the most suitable resource to user requests using an adaptive approach at runtime as compared with mentioned baseline approaches.

5.2.6 Energy Consumption: This is one of the significant influential parameters and used to assess the performance of datacenters in cloud paradigm. If the power consumption of a datacenter is high, it leads to increases the cost of services and decreases the margin of profit from cloud infrastructure. To cut down the energy efficiency by utilizing the resources of cloud datacenters properly becomes a serious issue. Therefore, we have proposed an enhanced SMO-based energy-efficient framework that minimizes the possibilities of idle VM instances as well as overloaded VM instances in a datacenter. The simulation experiments have been conducted to evaluate the efficacy of algorithms using diverse tasks and heterogeneous VMs.

Test case 1: Initially, 2500 diverse tasks are deployed at 1000 VMs and calculate the execution time and makespan time of VMs using the presented algorithm and other baseline algorithms. Additionally, the idle time of each VMs is calculated and defined the energy consumption by idle VMs as well as busy VMs. The simulation results of the first schedule show that the presented enhanced SMO-based algorithm utilized the datacenter resources in a better way which leads to reduce more energy consumption as compared to PSO, GSA, ABC, and IMMLB approaches as shown in figure 13.

Test case 2: Furthermore, 500 tasks are increased in each schedule from S1 to S7 and allocated to 1000 VM instances using the enhanced algorithm as well as baseline algorithms to measure the algorithms performance. The proposed enhanced SMO-based framework take the decision about the switching of VM instances to other modes (sleep or hibernate) and cut down the energy consumption. The comparison-based results of simulation proved that the proposed enhanced SMO-based algorithm significantly outperforms the baseline approaches.

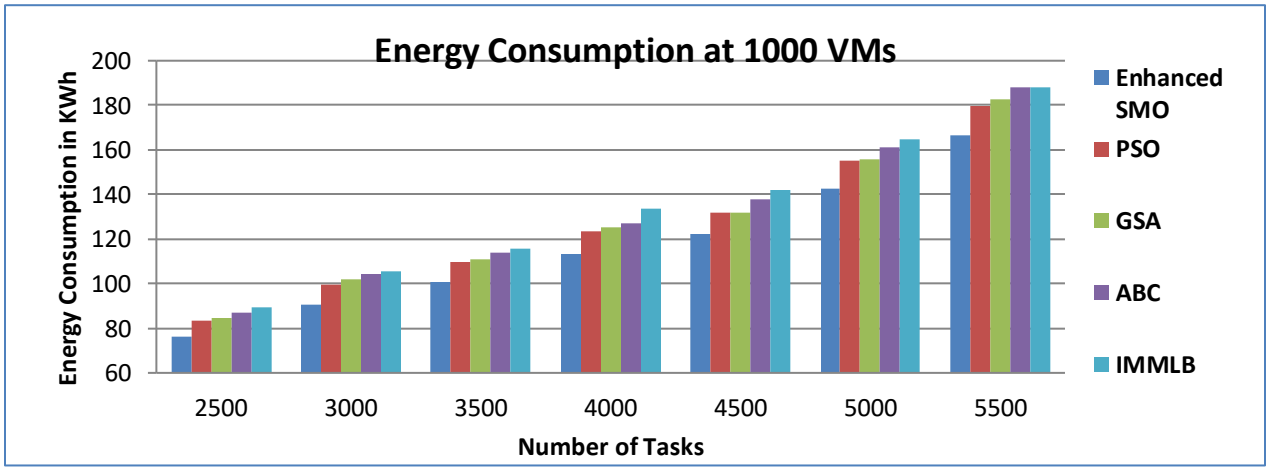

Figure 13 comparison between developed SMO and other algorithms

5.2.7 Trade-off Solution: The simulation environment has been set up to find the compromise solution between time and cost using pareto optimal theory, where 800 virtual machine instances has been taken to execute the 5000 tasks of users. The first computed value of time is approximately 14000 and service cost is 50.83 dollar as represented in the figure 14. Afterthat, more experiment has been performed to validate the results of proposed algorithm by increasing the virtual machines instances from 800 to 2000. The figure 14 depicts the seven set of optimal solution for the conflicting objectives, as the VM instances is increasing to execute the workload, time is reducing and all the solutions have equal priority, no single soluton is best or worst.

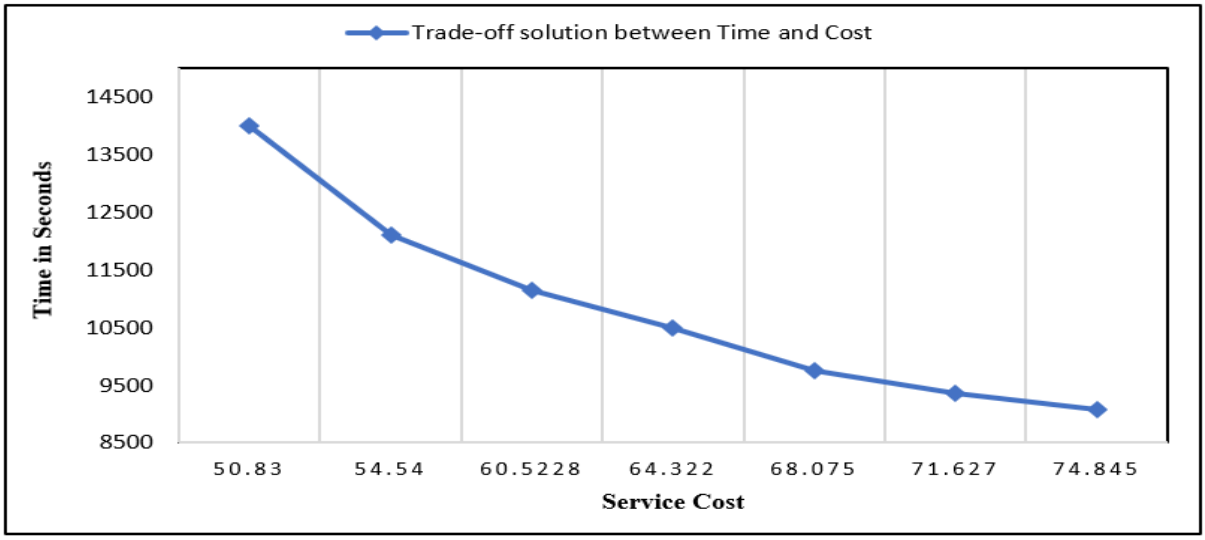

Figure 14 compromise solution between time and cost

6. Conclusion and Future Work

Cloud computing is the most prominent computing paradigm that exploits virtualization, to provision on-demand resources (software and hardware) over the internet using pay as you go model. There are several states of arts approaches that attempted to tackle the mentioned problems, they mainly emphasized some specific type of influential parameters or workload, and do not have the decision-making capability during the execution of the workload. In addition, many traditional security approaches have been implemented to secure the cloud infrastructure, but do not provide a satisfactory solution due to the rapid spread of the threats in virtualized environments. Hence, cloud security is also a concern along with the allocation of heterogeneous resources to upcoming end-user demand. To address the mentioned issue, authors have proposed a secure and adaptive framework that protected the cloud infrastructure by advanced encryption technology and allocates the best resource for processing the user's demand using the decision-making process. All the components of the framework along with their working methodology are discussed deeply in this article. The objective of the proposed framework with integrated enhanced SMO is to search and allocate the best possible resource to the request, and avoid the possibility of an imbalanced workload to improve various significant QoS parameters. The proposed optimization-based algorithm has a high convergence rate due to which it takes a smaller amount of time to search for the optimum resource. The performance of the proposed framework is analyzed by numerous times experiments (test cases) in a simulation environment on diverse workloads and heterogeneous virtual machine instances. The results of simulation experiments show that the proposed an adaptive framework significantly outperforms the baseline techniques for influential parameters in all conditions. In the future, the proposed work can be extended by considering other influential parameters like reliability, fault tolerance etc. We are going to extend this work to a realistic cloud-fog environment with scientific workflow like LIGO, SIPHT, Montage using artificial intelligence-based approaches [38]. Furthermore, we will apply blockchain and reinforcement learning mechnasiam to secure the cloud data.

Conflict of interest: The authors whose names are given in this article certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

References:

[1] M. Kumar et al., “Elastic and flexible deadline constraint load Balancing algorithm for Cloud Computing” in *Procedia Computer Science*, vol. 125, pp. 717-724, 2018.

[2] S. Singh, and I. Chana. "QoS-aware autonomic resource management in cloud computing: a systematic review," in ACM Computing Surveys, vol. 48, no. 3, pp. 1-46, 2015.

[3] Kumar, Mohit, et al. "A comprehensive survey for scheduling techniques in cloud computing." Journal of Network and Computer Applications 143 (2019): 1-33.
[4] Alboaneen, Dabiah, et al. "A metaheuristic method for joint task scheduling and virtual machine placement in cloud data centers." Future Generation Computer Systems 115 (2021): 201-212.
[5] J. Zhao et al., "A Heuristic Clustering-Based Task Deployment Approach for Load Balancing Using Bayes Theorem in Cloud Environment" IEEE transaction on parallel and distributed systems, vol. 27, no. 2, 2016.
[6] H. Chen, F. Wang, N. Helian and G. Akanmu ,"User Priority Guided Min-Min Scheduling Algorithm For Cloud Computing," in *national conference on Parallel Computing Technologies (PARCOMPTECH),* pp. 1-8, Bangalore, India, Oct. 2013.
[7] M.Kumar and S.C. Sharma "Priority Aware Longest Job First (PA-LJF) Algorithm for Utilization of the Resource in Cloud Environment," in *3rd International Conference on Computing for Sustainable Global Development,* pp. 415–420, March, 2016.
[8] Mohit Kumar and SC Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," 7th International Conference on Advances in Computing & Communications, ICACC-2017, 22- 24 August 2017, Cochin, India pp.322-329.
[9] Y. Xin et al., "A load balance-oriented cost-efficient scheduling method for parallel tasks," Journal of Network and Computer Applications, vol. 81, pp. 37-46, 2017.
[10] Azizi, Sadoon, et al. "Grvmp: a greedy randomized algorithm for virtual machine placement in cloud data centers." IEEE Systems Journal (2020).
[11]. D. Babu and P. Venkata, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," Appl Soft Comput, vol.13, no. 5, pp. 2292–2303, May 2013.
[12] Ramezani, Fahimeh, Jie Lu, and Farookh Khadeer Hussain. "Task-based system load balancing in cloud computing using particle swarm optimization." International journal of parallel programming 42.5 (2014): 739-754.
[13] S. Gill et al., "BULLET: Particle Swarm Optimization Based Scheduling Technique for Provisioned Cloud Resources" journal of network and system management, vol. 26, no. 2, pp. 361-400, 2018.
[14] M. Kumar and S.C.Sharma, "PSO-COGENT: Cost and energy efficient scheduling in cloud environment with deadline constraint" in Sustainable Computing: Informatics and Systems, vol. 19, pp. 147-164, 2018.
[15] E. Pacini et al., "Balancing throughput and response time in online scientific Clouds via Ant Colony Optimization" in Adv. Eng. Software vol. 84, pp. 31–47, 2015.
[16] M. Adhikari et al., "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud," *Journal of Network and Computer Applications*, vol. 128, pp. 64-77, 2019.
[17] Abed-alguni, Bilal H., and Noor Aldeen Alawad. "Distributed Grey Wolf Optimizer for scheduling of workflow applications in cloud environments." Applied Soft Computing 102 (2021): 107113.
[18] Chaudhary, Divya, and Bijendra Kumar. "Cloudy GSA for load scheduling in cloud computing." Applied Soft Computing 71 (2018): 861-871.
[19] S. Addya et al., "Simulated annealing based VM placement strategy to maximize the profit for Cloud Service Providers" Int. J. Eng. Sci. Technol, vol. 20, no. , pp. 1249–1259, 2017.
[20] N. Almezeini and A. Hafez, "Task scheduling in cloud computing using lion optimization algorithm *algorithms" in* International Journal of Advanced Computer Science and Applications, vol. 8, no. 11, pp. 1-7, 2017.
[21] J. Li et al., "An Improved Differential Evolution Task Scheduling Algorithm Based on Cloud Computing" in 17th IEEE International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES), pp. 30-35, 2018.
[22] Zhang, Xinqian, et al. "Energy-aware virtual machine allocation for cloud with resource reservation." Journal of Systems and Software 147 (2019): 147-161.
[23] Singh, Sukhpal, and Inderveer Chana. "QRSF: QoS-aware resource scheduling framework in cloud computing." The Journal of Supercomputing 71.1 (2015): 241-292.
[24] J.C. Bansal et al. "Spider monkey optimization algorithm for numerical optimization," Memetic computing, vol. 6, no. 1, pp. 31-47, 2014.
[25] S. Kumar et al.,"Fitness Based Position Update in Spider MonNey Optimization Algorithm." Procedia Computer Science, vol. 62, pp.442-449, 2015.
[26] K. Gupta et al., "Spider monkey optimization algorithm for constrained optimization problems," Soft Computing, vol. 21, no. 23, pp. 6933-6962, 2017.
[27] K. Gupta et al., "Improving the local search ability of spider monkey optimization algorithm using quadratic approximation for unconstrained optimization," Computational Intelligence, vol. 33, no. 2, pp. 210-240, 2017.
[28] Swami, Viren, Sandeep Kumar, and Sanjay Jain. "An improved spider monkey optimization algorithm." Soft Computing: Theories and Applications. Springer, Singapore, 2018. 73-81.
[29] Subramanian, Nalini, and Andrews Jeyaraj. "Recent security challenges in cloud computing." Computers & Electrical Engineering 71 (2018): 28-42.

[30] R. Buyya et al., “Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities,” in IEEE international conference on high performance computing & simulation, pp. 1-11, 2009.
[31] Al-Turjman, Fadi, Mohammed Zaki Hasan, and Hussain Al-Rizzo. "Task Scheduling in Cloud-Based Survivability Applications Using Swarm Optimization in IoT" The Cloud in IoT-enabled Spaces. CRC Press, 2019. 1-32.
[32] Hasan, Mohammed Zaki, and Hussain Al-Rizzo. "Task scheduling in Internet of Things cloud environment using a robust particle swarm optimization." Concurrency and Computation: Practice and Experience 32.2 (2020): e5442.
[33] Luo, Qiang, et al. "Metaheuristic algorithms for a special cutting stock problem with multiple stocks in the transformer manufacturing industry." Expert Systems with Applications 210 (2022): 118578.
[34] Osaba, Eneko, et al. "A tutorial on the design, experimentation and application of metaheuristic algorithms to real-world optimization problems." Swarm and Evolutionary Computation 64 (2021): 100888.
[35] Peres, Fernando, and Mauro Castelli. "Combinatorial optimization problems and metaheuristics: Review, challenges, design, and development." Applied Sciences 11.14 (2021): 6449.
[36] Peres, Fernando, and Mauro Castelli. "Combinatorial optimization problems and metaheuristics: Review, challenges, design, and development." Applied Sciences 11.14 (2021): 6449.
[37] Kumar, Mohit, and Subhash C. Sharma. "PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing." Neural Computing and Applications 32.16 (2020): 12103-12126.
[38] Gill, Sukhpal Singh, et al. "AI for next generation computing: Emerging trends and future directions." Internet of Things 19 (2022): 100514.
[39] Sun, Zaixing, Boyu Zhang, Chonglin Gu, Ruitao Xie, Bin Qian, and Hejiao Huang. "ET2FA: A Hybrid Heuristic Algorithm for Deadline-constrained Workflow Scheduling in Cloud." IEEE Transactions on Services Computing (2022).
[40] Zhang, Fangfang, Yi Mei, Su Nguyen, Mengjie Zhang, and Kay Chen Tan. "Surrogate-assisted evolutionary multitask genetic programming for dynamic flexible job shop scheduling." IEEE Transactions on Evolutionary Computation 25, no. 4 (2021): 651-665.
[41] Luo, Yizhe, Wenrui Ding, and Baochang Zhang. "Optimization of task scheduling and dynamic service strategy for multi-UAV-enabled mobile-edge computing system." IEEE Transactions on Cognitive Communications and Networking 7, no. 3 (2021): 970-984.
[42] Liu, Xuan, Xinning Chen, Qiuying Yang, Shigeng Zhang, Song Guo, Juan Luo, and Kenli Li. "More Than Scheduling: Novel and Efficient Coordination Algorithms for Multiple readers in RFID Systems." IEEE Transactions on Mobile Computing (2022).
[43] Sorbelli, Francesco Betti, Stefano Carpin, Federico Coro, Sajal K. Das, Alfredo Navarra, and Cristina M. Pinotti. "Speeding up routing schedules on aisle graphs with single access." IEEE Transactions on Robotics 38, no. 1 (2021): 433-447.
[44] Rizvi, Naela, Ramesh Dharavath, and Damodar Reddy Edla. "Cost and makespan aware workflow scheduling in IaaS clouds using hybrid spider monkey optimization." Simulation Modelling Practice and Theory 110 (2021): 102328.