ENERGY EFFICIENT RESOURCE MANAGEMENT AND TASK SCHEDULING AT THE CLOUD DATA CENTER

Thesis

Submitted in partial fulfilment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

by

NEERAJ KUMAR SHARMA

DEPARTMENT OF INFORMATION TECHNOLOGY

NATIONAL INSTITUTE OF TECHNOLOGY KARNATAKA

SURATHKAL, MANGALORE 575025

February 27, 2018

DECLARATION

By the Ph.D. Research Scholar

I hereby declare that the Research Thesis entitled **ENERGY EFFICIENT RESOURCE MANAGEMENT AND TASK SCHEDULING AT THE CLOUD DATA CENTER** Which is being submitted to the **National Institute of Technology Karnataka, Surathkal** in partial fulfilment of the requirements for the award of the Degree of **Doctor of Philosophy** in **Information Technology** *is a bonafide report of the research work carried out by me.* The material contained in this Research Thesis has not been submitted to any University or Institution for the award of any degree.

Place: NITK, Surathkal
Date:

Neeraj Kumar Sharma (IT13F04)
Department of Information Technology

CERTIFICATE

This is to *certify* that the Research Thesis entitled **ENERGY EFFICIENT RESOURCE MANAGEMENT AND TASK SCHEDULING AT THE CLOUD DATA CENTER** submitted by **NEERAJ KUMAR SHARMA**, (Registration Number IT13F04) as the record of the research work carried out by him, *is accepted as the Research Thesis submission* in partial fulfilment of the requirements for the award of degree of **Doctor of Philosophy**.

Prof. G. Ram Mohana Reddy
Research Guide

Prof. G. Ram Mohana Reddy
Chairman - DRPC

Acknowledgements

I would like to take this chance to thank those people who have made this thesis possible. First and foremost, I would like to express my deepest gratitude to my research guide, Prof. G. Ram Mohana Reddy, Head of Information Technology Department, for giving me guidance and support throughput my research work. This thesis would not have been possible without his support and suggestions.

I express my thanks to members of my RPAC committee, Prof. K. Chandrasekaran, and Prof. Gangadharan K V for their valuable suggestions during my research work. I would like to thank my brother Dr. Dheeraj Sharma for his valuable suggestions and encouragement.

I would like to thank my parents, and my sister, who gave unconditional, love and inspiration throughout my research work.

Finally, I would like to express my gratitude to my co-researchers who made the past three years more enjoyable and fun. I extend my sincere thanks to all teaching, and supporting staff of the Information Technology Department, for their support.

Last but not least, my gratitude as well as sincere appreciations go towards my wife Mrs. Priyanka Sharma who gave unconditional cooperation, love during my Ph.D. Without them, surely, this research work would not have been possible.

Place: NITK-Surathkal Neeraj Kumar Sharma

Date: February 27, 2018

ABSTRACT

Due to the growing demand for cloud services, allocation of energy efficient resources (CPU, memory, storage, etc.) and utilization of these resources are the major challenging issues of a large cloud data center. To meet the ever increasing demand of the customers, more number of servers are needed at the data center. These data centers require more cooling devices in order to keep the data center at a specified temperature resulting in more energy consumption and CO_2 emission. The user requested on demand virtual machine (VM) allocation problem is widely known as a combinatorial optimization problem. Due to the large number of PMs present in the data center, the specified VM allocation problem is related to the NP-hard/NP-complete complexity class. Finding an optimal solution to the specified VM allocation problem with the multi-objective approach in the polynomial time will thus create a lot of challenges. Further, the networking devices of data center like switches consume 10% to 20% of the total energy consumed by IT devices in the data center. Hence, the network-aware VM allocation algorithm is required to minimize the energy consumption of switches and physical machines (PMs) at the cloud data center. Further, a policy for migrating VMs from underutilized PMs to the energy efficient PMs is required over a period of time without violating the service level agreement (SLA) between the cloud service provider and the customer.

In order to minimize both the energy consumption and resources wastage, this thesis presents multi-objective VM allocation to PM using hybrid bio-inspired algorithms (HGACSO, HGAPSO, and HGAPSOSA) based on GA, CSO, PSO, and SA algorithms. Further, to save the energy consumption of networking switches in the cloud data center, a branch-and-bound based exact algorithm is proposed for VM allocation problem. The proposed branch-and-bound based exact algorithm saves the energy consumption of PMs and networking switches at the cloud data center. Further, the proposed VM migration technique and a

task scheduling technique based on the First-Fit approximation algorithm will not only reduce the energy consumption at the cloud data center but also avoids the SLA.

The experimental results were carried out in both homogeneous and heterogeneous cloud data center environments. Experimental results demonstrated that the proposed VM allocation algorithms outperform the state-of-the-art benchmark and peer research algorithms.

Keywords: Energy efficiency, Data center, Virtual machine, Physical machine, Resources utilization, SLA.

Contents

List of Figures

List of Tables

Abbreviations

$\vec{v_i}$	Velocity of a Particle
$\vec{x_i}$	Position of a Particle
$\vec{x}_{gbesti}$	Global Best Position of a Particle
$\vec{x}_{lbesti}$	Local Best Position of a Particle
A_k	k^{th} Application
b, a_{ij}, f_{ij}, z_k	Binary Variables
B_e	Bandwidth Requested by a VM
B_i^s	Set of PM Types Contain VM_i
C_e	Total Capacity of a Link e
c_i^{mig}	Migration Cost of a VM
chr_i	i^{th} Chromosome in a Generation
f_i^{new}	New Fitness Value of a Chromosome
f_i^{old}	Old Fitness Value of a Chromosome
fit_k	Fitness Value of a Chromosome
L_{mc}	Lower Bound
m_i^o	Migration Overhead of a VM
$n(p)$	Number of Active Ports
P_1^i, P_2^i, P_3^i	Inertia Weight Coefficients
P_k	Probability of a Chromosome to go for Next Generation

P_s^{idle}	Power Consumption of a Switch in Idle Mode
P_s^{port}	Port Power Consumption of a Switch
P_s^{switch}	Power Consumption of a Switch
pm^{MIPS}	Amount of MIPS of a PM
pm^{Pe}	Number of Processing Elements of a PM
pm^{RAM}	Amount of Ram of a PM
PM_s	Set of PM Connected to Switch s
q_1, q_2, q_3	Uncertain Bit Values
sl^t	SLA Penalty
T_{amb}	Ambient Temperature of a PM
T_j	Temperature of pm_j
T_j^{max}	Maximum Temperature of a pm_j
VM^{bw}	Amount of Bandwidth Requested by a VM
VM^{mips}	Amount of mips Requested by a VM
VM^{pe}	Number of Processing Elements Requested by a VM
VM^{ram}	Amount of Ram Requested by a VM
$VM^{storage}$	Amount of Storage Requested by a VM
$VM_{large}(n3)$	Large Number of VMs
$VM_{medium}(n2)$	Medium Number of VMs
$VM_{small}(n1)$	Small Number of VMs
$VM_{x.large}(n4)$	Extra Large Number of VMs

$z_{kk'i}$	Binary Variable
CDC	Count of Dimension
CSO	Cat Swarm Optimization
d	Resource d
DC^u	Data Center Resource Utilization
DVFS	Dynamic Voltage Frequency Scaling
E	Energy Consumption of CPU
f	Minimization Function
f^{max}	Maximum Frequency of CPU
f^{min}	Minimum Frequency of CPU
FF	First Fit
FFD	First Fit Decreasing
GA	Genetic Algorithm
HGACSO	Hybrid Genetic Algorithm and Cat Swarm Optimization
HGAPSO	Hybrid Genetic Algorithm and Particle Swarm Optimization
HGAPSOSA	Hybrid Genetic Algorithm Particle Swarm Optimization and Simulated Annealing
ILP	Integer Linear Programming
l	Level of a Search Tree
LB(N)	Lower Bound
m	Number of PMs at the Data Center
MFD	Modified First Fit Decreasing

MIPS	Millions of Instructions Per Seconds
MR	Maximum Ratio
n	Number of VMs at the Data Center
OPT	Optimal
P^{Idle}	Idle Power Consumption of CPU
P^{max}	Maximum Power Consumption of CPU
P^{min}	Minimum Power Consumption of CPU
PMs	Physical Machines
PSO	Particle Swarm Optimization
R	Thermal Resistance of a PM
S	Set of Switches
s	Number of Chromosomes in a Generation
S'(N)	Set of Unallocated VM
SA	Simulated Annealing
SLA	Service Level Agreement
SMP	Seeking Mode Pool
SRD	Select Range of Dimension
t	Time Instant
u	Rate of CPU Utilization
u_d	Utilization of Resource d of CPU
UB(N)	Upper Bound

VMs	Virtual Machines
w	Learning Coefficient

Chapter 1

Introduction

This chapter introduces the concepts of cloud computing, its characteristics, deployment models, and their technologies. Further, this chapter also gives the overview of energy efficient green cloud computing, and its significance and challenges. In addition to this, challenges in the development of energy efficient resources management at the cloud data center, and motivations for the present research work are discussed.

1.1 Cloud Computing

Cloud computing is an Internet based computing. It provides on demand resources such as processing elements, storage, memory, network, and other devices to the customers on rent basis. The on demand resources from the cloud resource pool are accessed by the customers on shared basis. These shared resources are released and provisioned by the minimal management effort. Further, cloud computing provides the capability to users and enterprises to store and process their data to private or remote data centers. The data center may be located near or far from the city. Hence, cloud computing relies on sharing of resources like electricity grid in an electrical network.

The high bandwidth network, low cost sharing computers, hardware virtualization, service oriented architecture, and utility computing achieve high growth of the cloud computing in the present scenario. It is reported that cloud computing was the highly utilized services due to the high computing capacity of servers, low cost of services, scalability, availability etc. (Gartner 2013). Further, in the near future the growth rate of cloud computing is 50% per year (Kreith 1999). Hence, due to the dependency of large and small scale industries on cloud computing, more resource is required to give the services to the cloud customers. Hence, to match the future requirement of industry, size of the data center is continuously

increasing day by day due to addition of more number of servers to fulfill the customers demand (Bermejo et al. 2016).

The main advantage of cloud computing is, to allow small and large scale enterprises for optimal utilization of modern and updated software and hardware resource, without any additional investment in the infrastructure (e.g. purchasing servers on rent basis). Thus, cloud computing allows customers to concentrate on their core businesses as compared to investment on time and money to improve the computer infrastructure. Cloud computing gives a good environment to enterprises for running their applications faster with less maintenance. Therefore cloud computing enables more friendly environment to the Information Technology (IT) teams, to adjust resources according to the floating business demand. Hence, cloud computing provides the resources to customers based on pay-as-you-go model.

1.2 Cloud Computing Characteristics

Broad Network Access: The cloud computing existing resource can be accessed through the Internet with the help of standard resource access mechanism. Further, this resource access mechanism provides platform independent access through the use of heterogeneous client platform i.e. workstation, laptop, tablet, smart phone etc.

Rapid Elasticity: In cloud computing, the existing resources are provisioned or allocated to customers rapidly and elastically. The cloud resources can go up and down according to the current resources requirement. There are two types of resources scaling options exist at the cloud data center.

- **Horizontal Scaling:** Horizontal scaling involves in adding more number of servers by keeping constant resource capacity of a server and thus resulting in higher resource capacity at the cloud data center.

- **Vertical Scaling:** Vertical scaling involves in improving the resource capacity of individual server and thereby keeping the number of servers constant at the cloud data center.

Resource Pooling: The multi-tenant architecture allows the multiple users served by the same Physical Machine (PM). Further, using the virtualization technique, virtual resources in the form of Virtual Machines (VMs) which run on a single PM are assigned to the users. There are various kinds of virtualization used in the cloud computing namely, para-virtualization and hardware virtualization.

Measured Services: The cloud resources are provided to the users using pay-as-you-go model. The resources utilization is measured and charged based on the specific metric. The specific metric like the amount of CPU cycles used, the amount of storage space used, number of network I/O request generated by the user, etc. are used to calculate the utility charge of cloud resources.

In addition to these essential characteristics of cloud computing other desirable characteristics are required for reducing the cost of cloud computing and the following are the details.

Performance: Cloud computing shows improved performance for the application because of resources provided to the application can be scaled up and down according to the dynamic workload of the application.

Reduced Cost: In the cloud computing, resources are provisioned dynamically based on the computing power, storage, and other resources required for the application. By this way, we can avoid the upfront investment in purchasing the computing assets in the worst-case scenario and thus saving the significant cost for organizations and individuals. Different applications have a large variation in their workload, due to seasonal and other factors. For example, E-Commerce type of applications experience higher workload during holiday season. Therefore, to ensure the market requirements, adequate resources are to be provisioned for the applications to meet the workload demand, and thus satisfying the Service Level Agreement (SLA) between the cloud service provider and the customer.

Reliability: Cloud computing services provider ensures the reliability to the deployed application by managing the IT infrastructure proficiently. Further, the cloud services provider guarantees the reliability and availability of adequate re-

sources in the form of SLA.

Outsourced Management: Cloud computing allows the users to outsource the IT infrastructure required for the external cloud services provider. Thus, the customers can save the huge amount of upfront capital expenditure in selecting the required IT infrastructure and thus paying only for the used resources.

Multi-tenancy: Using the multi-tenancy approach, cloud computing allows multiple users to make use of same shared resources. Nowadays, applications like E-Commerce, Business-to-Business, Banking & Financial, Retail, Social Networking etc. are deployed in multi-tenant cloud computing environment. The different forms of multi-tenancy are as follows.

- **Virtual Multi-tenancy:** In the case of virtual multi-tenancy, the resources like computing power and storage are shared among multiple users. Multiple tenants are served from the different VMs which are executed concurrently on top of the computing and storage resources of the same PM.

- **Organic Multi-tenancy:** In the case of organic multi-tenancy, every component in the system architecture is shared among multiple tenants, such as hardware, OS, database servers, application servers, load balancers etc.

1.3 Types of Cloud Computing Models

There are two types of models exist in the cloud computing such as service model, and deployment model. The details of the of the service and deployment models are described as follows.

1.3.1 Service Models

The cloud services offered by the service provider are in different forms. The National Institute of Standards and Technology (NIST) has defined three service models of cloud computing as shown in Figure 1.1 and the details are as follows.

Infrastructure-as-a-Service (IaaS) Model: IaaS provides the cloud resources (computing, and storage) to the users based on pay-as-you-go model. These resources are provided to the users in the form of VM and virtual storage. In this deployment model, users can start, stop, manage, and configure the

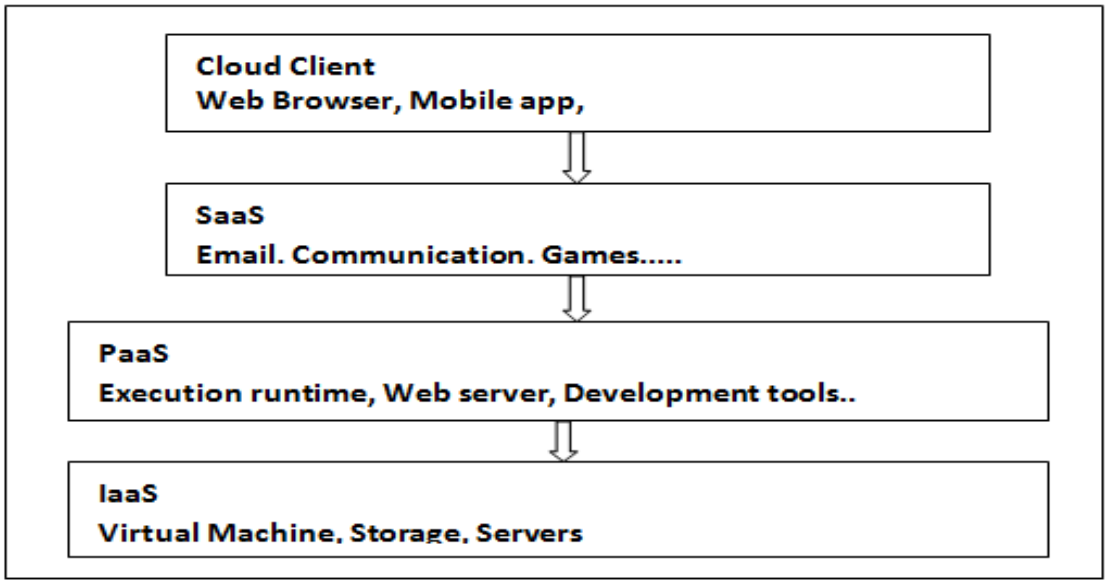

Figure 1.1: Cloud Service Models (Nguyen et al. 2013).

VM and virtual storage according to their requirement. Users can deploy operating systems and applications on the provisioned VM according to their own choice. Further, the cloud services provider manages the cloud underlying infrastructures. The billing of provisioned virtual resources is done by the pay-as-you-go paradigm. The metrics used for the billing purpose are based on the number of VM, types of VM, time duration, and amount of virtual storage provisioned to the users.

Platform-as-a-Service (PaaS) Model: PaaS provides the capability to the users to develop, and deploy their applications in the cloud environment with the help of development tools, application programming interfaces (APIs), software libraries (provided by the cloud services provider). In this model, the cloud services provider will manage the underlying cloud infrastructure such as network, operating systems, and storage etc.

Software-as-a-Service (SaaS) Model: SaaS provides a complete software application or the Graphical User Interface (GUI) to the users. In this deployment model, cloud services provider will manage the underlying cloud infrastructure (servers, network, operating systems, storage, and application software). Users are unaware about the underlying architecture of the cloud. All the SaaS applications are provided to the users through a client interface such as a web browser.

The provided SaaS applications are platform independent, and these can be accessed by different client devices such as workstations, tablets, laptops and smartphones, running on different operating systems. Since, all the services (application and related data) are managed by the service provider, hence the user can access applications from anywhere.

1.3.2 Deployment Models

Cloud computing deployment models define the category of cloud environment, and we can distinguish it on the basis of ownership, size, and access. The cloud deployment models tell about the purpose and nature of cloud. Hence, the NIST defined four cloud deployment models such as Public cloud, Private cloud, Hybrid cloud, and Communicate cloud shown in Figure 1.2 and the details are as follows.

Public Cloud: The existing infrastructure of cloud data center (CPU, Storage, Communication network, etc.) is provisioned for open use by the customers over a network. This type of cloud model is the real representation of cloud as an entity. Further, this type of cloud is owned, managed, and operated by a business, academic, or government organizations. The customer does not have any control on the geographically distributed data centers. Further, public cloud deployment model is the most suitable model for the businesses, because of low capital overhead, and less operational cost. Hence, this model is economical among all the cloud deployment models. In this cloud deployment model, the services provider provides services to customers free of cost or in the form of license. An example of public cloud is Google.

Private Cloud: This type of cloud deployment model is also known as an internal cloud. The platform for cloud computing under this category is deployed in a secure environment using a firewall, and this firewall works under the governance of particular corporate. Further, this type of cloud deployment model permits authorized users to use the cloud services and gives the direct and full control to the organization over the data center. The following conditions such as security alarm, management demands, and up-time requirements are suitable to adopt this type of cloud model.

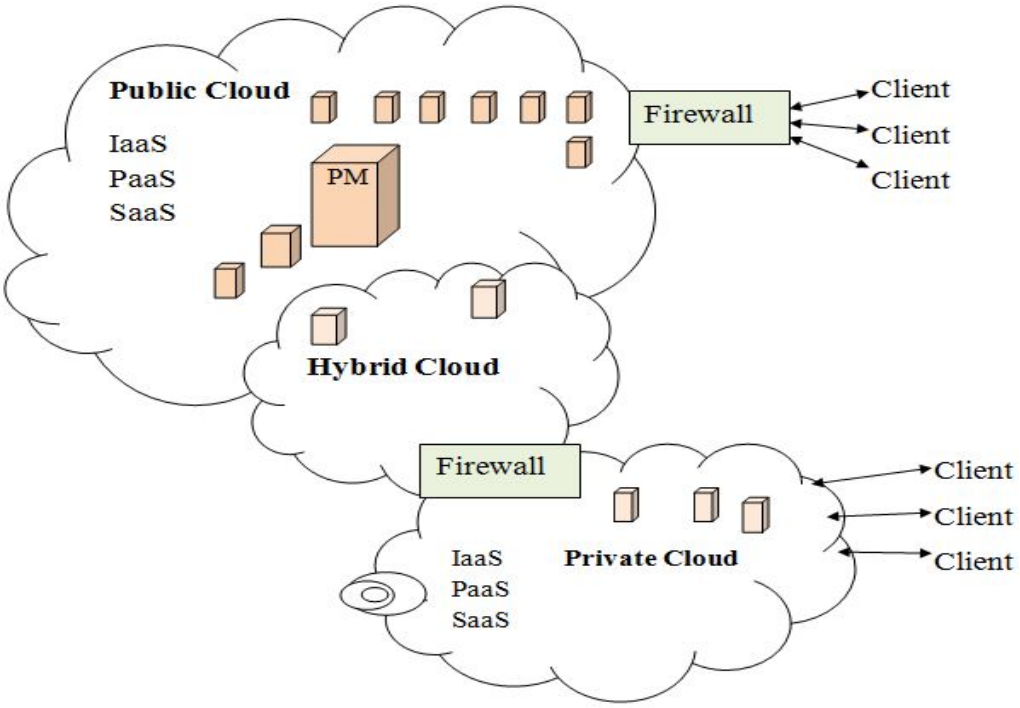

Figure 1.2: Cloud Deployment Models (Gao et al. 2013).

Hybrid Cloud: This is an integrated cloud deployment model of public and private clouds. Hence, in this model, two or more types of services are combined into a single entity, but their services will remain individual from each other. The hybrid cloud model crosses the isolation and the boundaries of the organization. Therefore, it cannot be categorized into private, public, or community cloud. It allows the users to increase the capacity or capability of cloud by using another cloud package/services. In this type of cloud deployment model, resources are allocated or managed by either in-house or external services provider. Further, it is the combination of two platforms where workload is exchanged among public or private clouds according to the demand. Workload or data which is not critical like testing, deployment can be in-house in a public cloud, while the critical or sensitive data can be housed internally. Let us consider an example of an E-commerce website which is hosted on a private cloud that gives the security and scalability, and the brochure is running on a public cloud, therefore it is more economical to private cloud. Hybrid cloud consists of features like scalability, security, and flexibility etc.

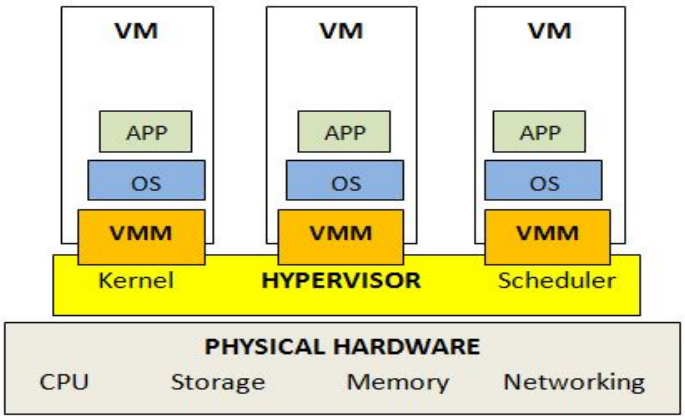

Figure 1.3: Virtualizatrion (Wang et al. 2013).

1.4 Cloud Computing Technologies

1.4.1 Virtualization

In cloud computing, virtualization is defined as a partition of resources (computing, storage, network, and memory) of a PM into a number of VMs. Virtualization is the key technology in the cloud computing, which allows the pooling of different kind of resources. Further, resources are pooled to serve multiple users using multi-tenancy architecture. Figure 1.3 shows the architecture of virtualization technology in cloud computing. Using the virtualization layer, multiple operating systems will run concurrently on VM in the same underlying PM.

Hypervisor: The virtualization is created using a hypervisor or a Virtual Machine Monitor (VMM). The hypervisor acts as a virtual operating platform to a guest operating system (OS). There are two types of hypervisor such as Type-1 hypervisor and Type-2 hypervisor as shown in Figures 1.4 and 1.5 respectively. Further, Type-1 hypervisor runs directly on the host hardware of a PM not only controls the hardware, but also monitors the guest operating systems. Type-2 hypervisor runs on the main/host operating system and monitors only the guest operating systems. In virtualization, a guest operating system is installed on a VM. In virtualization, the guest OS and host OS can be different from each other. There are various forms of virtualization existing in the cloud computing and the details are as follows.

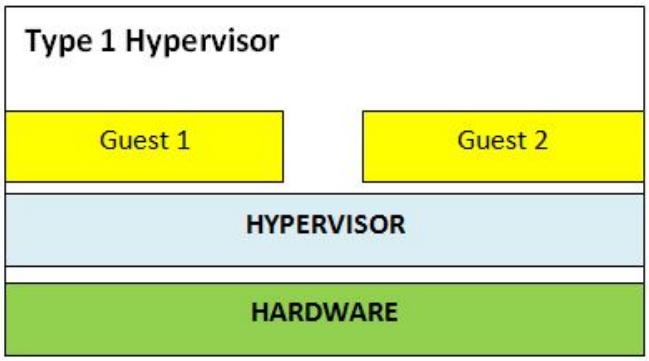

Figure 1.4: Type-1 Hypervisor (Cheng et al. 2016).

Full Virtualization: In this type of virtualization, the virtualization layer decouples the guest OS from the underlying hardware of a PM. Thus, guest OS requires no further modification and guest OS is not aware of what is being virtualized. Full virtualization is enabled by direct execution of user requests and binary translation of OS requests.

Para Virtualization: In this type of virtualization, the guest OS is modified through the communication with a hypervisor, for improving the performance and efficiency of the virtualization. The kernel of the guest OS is modified by replacing non-virtualizable instructions with hypercall, by that way guest OS communicates directly with the virtualization layer (hypervisor).

Hardware Virtualization: In this type of virtualization, the hardware features (Intel's virtualization technology (VT-x) and AMD's AMD-V) are virtualized. In hardware virtualization, privilege and sensitive calls are set automatically trapped inside to the hypervisor. Hence, in this virtualization, there is no need of binary translation or para virtualization.

1.4.2 Load Balancing

Scalability is one of the important feature of cloud computing. To meet the performance requirements of applications, resources can be scaled up at the data center. A load balancing technique is used to distribute the application workload across multiple servers. Hence, by load balancing, we can achieve several benefits

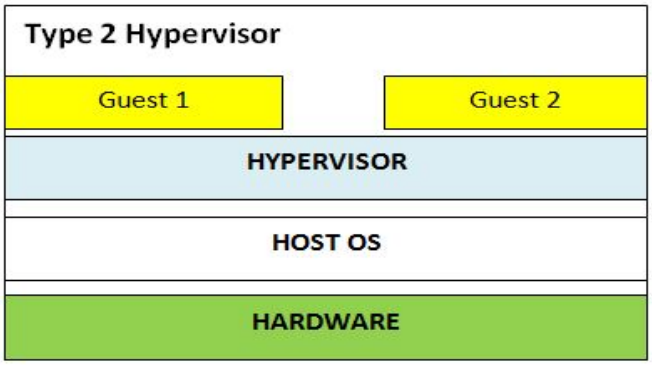

Figure 1.5: Type-2 Hypervisor (Cheng et al. 2016).

such as maximum utilization of resources, minimum user response time, maximum throughput etc. at the cloud data center. Using the load balancing, we can achieve high reliability and availability for cloud based applications. Since, multiple resources work under the load balancer to serve the user requests, hence the load balancer can automatically reroute the user traffic to the healthy resources when one or more resources are failed at the cloud data center. In cloud computing, under load balancer makes a pool of servers for accessing the cloud based application. This load balancer appears as a single server with high computing capacity. Further, routing of users request is determined by the load balancing algorithm. Commonly used load balancing algorithms for routing the user's requests are used in the cloud computing and the details are as follows.

Round Robin: In a round robin load balancing algorithm, existing servers at the cloud data center are selected one by one to serve the incoming user requests in a circular manner. In this algorithm, no priority is assigned to any specific server at the cloud data center.

Weighted Round Robin: In a weighted round robin load balancing algorithm, each server is assigned with some weights. Thus, by this way the incoming user requests are proportionally routed to the servers using a static and dynamic ratio of respective weights.

Low Latency: In a low latency load balancing algorithm, load balancer continuously monitors the latency of each server at the cloud data center. Further, each incoming user requests are routed to the lowest latency server.

Least Connections: In the least connections load balancing algorithm, the incoming user requests are routed to the server which has the least number of connections.

Priority: In priority load balancing algorithm, a priority number is assigned to each server at the data center. Further, the incoming user requests are assigned to the highest priority server, if it is available at the data center. In case of failure of higher priority server, user requests are assigned to lower priority server.

Overflow: The overflow load balancing approach is same as the priority load balancing algorithm. Here, incoming user requests are assigned to higher priority server, and if this higher priority server is in an overflow condition, then requests will be routed to lower priority server.

Sticky Sessions: In this load balancing approach, all the requests belonging to a user session (sticky session) are routed to the same server at the data center. The major advantage of this approach is that it makes session management simple. However, the major disadvantage of this approach is that in case of server failure condition, sessions belonging to that server will be lost since there is no automatic failover mechanism in this approach.

Session Database: In this approach, all the sessions related information is maintained externally in a separate session data base which is replicated to avoid a single point of failure. Thus, this approach involves the extra overhead of storing the session information. However, in place of sticky sessions, this approach provides automatic fail over.

Browser Cookies: In this approach, the user session information is stored on the client side in the form of browser cookies. The advantage of this approach is that, it makes the session management easy and consists of least amount of overheads at the load balancer side.

URL Re-writing: In this approach, an URL re-write engine stores the client session information by updating the URL on the client side. Hence, it avoids the overhead to the load balancer, but the drawback of this approach is that the amount of session information that can be stored is limited. Further, this approach

will not work when applications require large amounts of session information.

The load balancing can be implemented on two levels such as software and hardware. In the case of software based load balancing, load balancer runs on standard operating system, and like other cloud resources balancer is also virtualized. Whereas the hardware load balancer can be implemented in Application Specific Integrated Circuits (ASICs). Further, in a hardware load balancing, the incoming user requests are routed to the underlying servers using some pre-configured load balancing strategy, and the response from the servers is sent back either directly to the user or back to the load balancer where it is being manipulated before being sent back to the user.

1.4.3 Scalability and Elasticity

The multi-tier applications (social networking, e-commerce, business-to-business, etc.) will experience rapid changes in their traffic over the periods of time. Further, each website has a different traffic pattern which can be calculated by number of factors that are generally hard to predict beforehand. Nowadays, applications have multiple tiers of deployment and each tier consists of multiple servers. Thus, the capacity planning is the crucial task for such applications so that it calculates the right size of each tier of the deployment of an application in terms of number of resources and the capacity of each resource required to execute. Capacity planning may also determine the computing resources, storage, memory or networking resources.

Traditional approaches for capacity planning are mainly based on the predicted demands for an application, and account for the worst-case peak load of an application. In case of increasing or decreasing the workload we use the traditional approaches like scale up or scale down. The scale up process enhances the hardware resources (additional computing memory, storage or network resources). Further, scale down process reduces the hardware resources. Traditional scale up and scale down approaches are based on the demand forecast at regular intervals of time. In the case of rapid variation in the workload, traditional approaches are unable to keep track with the demand and lead to either over provisioning or under

Table 1.1: Monitoring Metrics in Cloud Computing

Type	Metrics
CPU	CPU Usage, CPU Idle
Disk	Disk Usage, Bytes/sec (read/write), Operations/sec.
Memory	Memory Used, Memory Free, Page Cache (incoming/outing).
Interface	Octets/sec(incoming/outing).

provisioning of cloud resources. Therefore, over provisioning will lead to higher capital expenditure as compared to required expenditure. On the other hand under provisioning of resources lead to traffic overload resulting in slow response time, low throughput, and resulting in loss to the customers.

1.4.4 Monitoring

Cloud resources are monitored by the cloud services provider. Further, monitoring of cloud resources allows the cloud users to analyze the data variation using various monitoring metrics. During the monitoring of cloud services, users collect the data on various systems and application metrics from the cloud computing instances. Monitoring cloud services provide several pre-defined metrics, users can also define their customized metrics for monitoring the cloud resources. Users can define various actions which are based on the monitoring data. For example, we need auto-scaling of cloud deployment, when the CPU usage of monitoring resources become high. Further, monitoring of cloud services also provides various statistics based on the data collected during monitoring phase. Table 1.1 describes the list of used monitors metrics for cloud computing resources. The cloud resources monitoring is an important phase for allowing the users to keep track of health of applications and services deployed in the cloud. For example, an organization wants to monitor the performance of its website, then it needs to monitor the traffic based on the available data at run time. Hence, monitoring the available data at run time is useful to make correct operational decisions such as scaling up or scaling down the cloud resources according to the traffic.

Table 1.2: Criteria for Defining SLA in the Cloud Computing

Criteria	Details
Availability	Percentage of time the service is guaranteed to be available.
Performance	Response time, Throughput.
Disaster Recovery	Mean time to recover.
Problem resolution	Process to identify problems, support option, resolution expectation.
Security and privacy of data	Mechanism for security of data in storage and transmission.

1.4.5 Service Level Agreement

A Service Level Agreement (SLA) is a contract between the cloud service provider and the cloud customer. SLA specifies the level of service provided by the cloud services provider in the form of minimum level of guaranteed service and a target level. Further, SLA consists of various performance metrics and their corresponding service level objectives. Table 1.2 describes the common criteria to define the SLA in the cloud computing.

1.4.6 Billing Models

A cloud service provider offers a number of billing models for the customers and the details are described as follows.

Elastic Pricing: Elastic pricing is based on a pay-as-you-go model, and accordingly the cloud customers are charged based on the usage of cloud resources. Further, cloud computing provides the benefits to the users based on the provisioning of the cloud resources as per demand. The on demand provisioning of resources and elastic pricing model give the advantage to the customers in terms of saving the cost. The elastic pricing model in the cloud computing is the best option for the customers who are using the cloud services for a short duration of time and thus predicting the usage of resources beforehand.

Fixed Pricing: In case of the fixed pricing model, cloud customers are charged at a fixed rate on weekly, monthly, or yearly basis for the resources. For example, fixed amount has to be charged for running the VM instances irrespective of the actual usage. This model is beneficial for the cloud customer who wants to use cloud resources for longer duration of time and also wants more control over the price of the cloud.

Spot Pricing: In case of spot pricing model, cloud service provider charges variable pricing for the cloud resource used by the customers. When the demand of cloud resources is high, the resources price will increase, and when the demand for cloud resources is low then the cloud resources will decrease. Table 1.3 describes a list of billable resources for the customers in the cloud computing.

1.5 Energy Efficient Green Cloud Computing

Cloud computing is a paradigm shift from computing as a-product to computing as a-service. Due to its several benefits, the cloud computing has a lot of applications in the fields of businesses, educational institutions, governments and individuals in both developed and emerging markets etc. The usages of cloud services are continuously increasing day by day, and, hence this created a huge demand of the data center that supports the cloud. To fulfill the users demand and creating a new service, vendors use large scale data centers which comprise thousands of servers, networking devices with other infrastructure such as cooling, storage and communication networks etc. Thus, due to the growing demand of services the energy consumption at the data centers is also increasing. Then how we can make the cloud data centers greener? Cloud computing is a green solution as the infrastructure of cloud data center embraces two critical elements of a green IT namely, resource efficiency and energy efficiency.

1.5.1 Green Data Center

Geographically distributed cloud data centers are growing in terms of number, capacity, and power consumption due to current technological trends in edge computing. IBM reports (IBM 2014) highlighted that the power consumption of worldwide data centers is currently 100 KWH a year. The carbon footprint of data

Table 1.3: List of Billable Resources

Resource	Details
Virtual Machines	CPU, memory, storage, disk I/O, Network I/O.
Network	Network I/O load balancers, DNS, firewall, VPN.
Storage	Cloud storage, storage volumes, storage gateway.
Data services	Data import/export services, data encryption, data compression, data backup, data redundancy, content delivery.
Security services	Identiy and access management, isolation, compliance.
Support	Level of support, SLA, fault tolerance.
Application services	Queuing service, notification services, work-flow, payment service.
Deployment and management services	Monitoring service, deployment service.

centers is increasing day by day dramatically due to higher energy consumption by IT devices and cooling systems at the data centers. The cloud computing based companies like Google, IBM, Microsoft, etc., invest (tens of millions of dollars) on electricity bill as an operating cost (Asemi et al. 2015). A U.S. Environment Protection Agency (EPA) (Wang et al. 2016) reported, in the year 2006 that U.S data centers consumed 61 billion kilowatt-hours of electrical energy, which is equal to 1.5% of the total energy consumed by the U.S. in the same year.

Continuous usage of Internet and other web applications are causing the rapid growth of data centers and resulting higher energy consumption. Further, to handle more transactions in less period of time, to process and store more transaction, and to automate, the more business related processes, many enterprises are installing more number of servers or expanding the server capacity at the data centers resulting in more computing power to process the data. As energy cost is increasing then the operational cost of the data centers is also increasing. The energy cost of the data center is nearly equivalent to 30% of the total operational cost of a data center. Thus, minimizing the energy consumption of a data center is one of the major challenging issue of IT industry.

The number of new servers are installed 6 times more in the last decade at the worldwide data centers, and modern server's energy consumption is much higher than that of an earlier server model. Hence, providing an uninterrupted power supply is becoming the critical issue of IT industry whose data centers are expanding continuously. Further, electrical energy suppliers require time and money to supply a huge amount of additional electrical power for the data centers. These practical, social, and finical constraints force IT industry towards green data centers with minimum energy consumption.

Therefore, the infrastructure of green data centers in terms of IT equipment, air conditioners, electrical and mechanical systems, and the building that houses the data center should give maximum energy efficiency, low carbon emission, and less adverse impact on environment. Further, modern data centers use advanced cooling, heating, and IT devices. There are different ways to save the energy

consumption at the cloud data center such as (server, storage, and network) virtualizaion, usage of blade servers, clustering and consolidation of servers, energy efficient power supply etc.

1.5.2 Power Consumption at Cloud Data Center

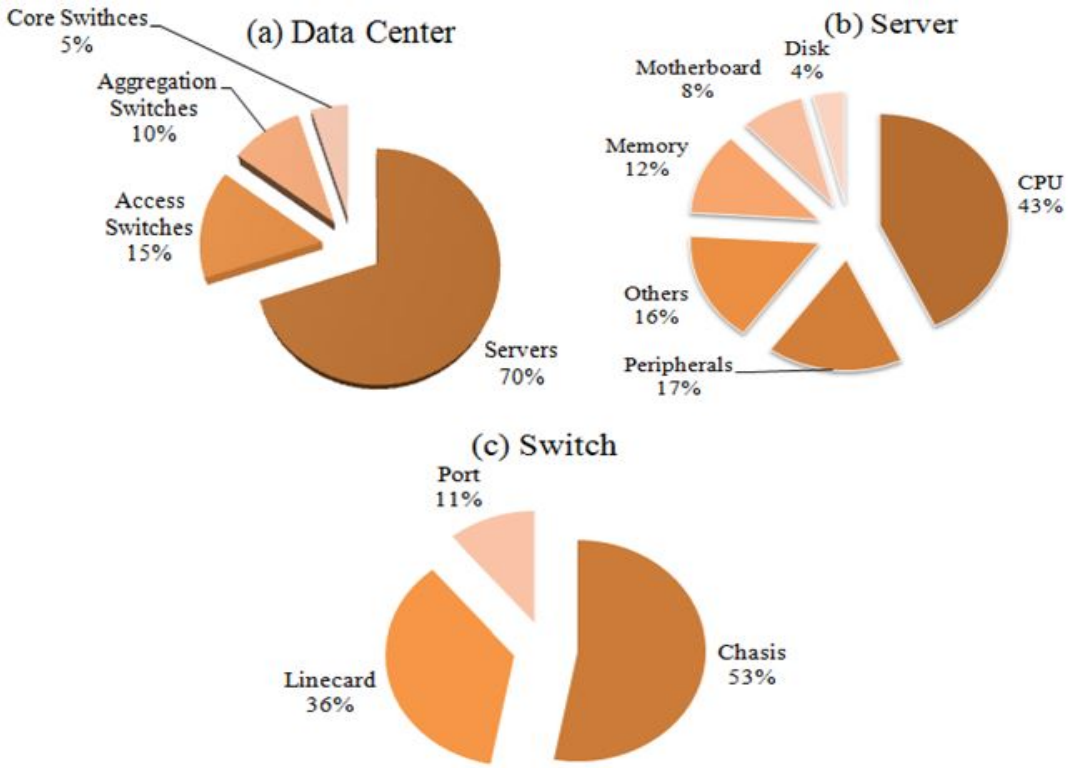

Figure 1.6: Power Consumption at the Data Center (Wang et al. 2017).

According to the report of the Natural Resources Defence Council (NRDC), a worldwide distributed data centers consumed 91 billion KWH of energy in 2013, and further estimated that the energy consumption may reach 139 billions of KWH by 2020 which is 53% more compared to the current energy consumption. Further, out of the total supplied electrical energy to the data center, 10%-15% electricity is consumed by PMs. The main reason for high energy consumption of the PM is due to inefficient usage of these PMs. Even though the PM is in ideal condition, but it still consumes 70% of their peak power. Therefore, energy efficient utilization of PM at the cloud data center is an important issue for saving the electricity cost of the cloud data center. Hence, to resolve this problem, the virtualization

technology is used by the cloud data centers.

The main sources of energy consumption in a data center are Cooling devices (Air Conditioners), Computing resources (PM) and, Networking elements (Switches, Routers). Figure 1.6(a) shows the power consumption of IT devices at the cloud data center.

Power Consumption of a Server (PM): The main components of server which consume the energy are CPU (Processing Element) 43%, Peripheral 17%, Memory 12%, Motherboard 8% and, other components 16%. Hence, among all the components of a sever, the energy consumption of CPU is maximum. Figure 1.6 (b) shows the power consumption of the server at the data center.

Power Consumption of Network Switches: The fat tree architecture based data centers consist of three types of networking switches (access switches, core switches, and aggregation switches). The power consumption of different components for a switch is shown in Figure 1.6 (c).

The use of renewable energy sources (solar, wind energy, etc.) will reduce the energy consumption and CO_2 emission at the data center. The other way of reducing the energy consumption of IT devices is to introduce energy aware resource allocation and migration policy at the cloud data center.

1.6 Challenges

The challenges that must address for the development of an energy efficient green cloud data center are given below:

Energy Efficient VM Allocation: The energy efficient VM allocation problem is widely known as a combinatorial optimization problem. It is also known as a multi-dimensional variable size bin-packing problem, where items are the VMs and bins are the PMs. Due to the large number of PMs present in the data center, the specified VM allocation problem is related to the NP-hard/NP-Complete complexity class. Hence, to resolve the energy efficient VM allocation problem, we need resources allocation policy which not only saves the power consumption but also gives the solution in polynomial time.

Energy Efficient SLA Aware Task Scheduling: The energy consumption of a PM is dependent on the current CPU utilization. Hence, by scheduling the tasks to less number of PMs, we can switch-off idle PMs at the cloud data center, resulting in less energy consumption at the data center. Further, SLA is based on the response time of the end users. Hence, scheduling of more number of tasks to less number of PMs, will result in high response time. Thus, task scheduling is a challenging problem of cloud computing, and hence we need an energy efficient SLA aware task scheduling policy for the cloud data center.

Energy Efficient VM Migration: Since cloud computing is based on a pay-as-you-go model, hence users demand different types of VM instances for a specified period of time. Hence, allocation of new VMs and destruction of time expired VMs is continuous processes in the cloud data center, and thus there may be a chance of a PM is in underutilization (idle) condition. Thus, an energy efficient VM migration policy is required to migrate the VM from the underutilized PM to the energy efficient PM and thereby switching-off underutilized (idle) PMs at the cloud data center.

Reduction of Network Switches' Energy Consumption: Further, the networking devices of data center like switches consume approximately 10% to 20% of the total energy consumed by IT devices at the data center, therefore we need to reduce the energy consumption of network switches. Hence, the network-aware VM allocation algorithms are required to minimize the energy consumption of switches and PM at the cloud data center.

Thermal Aware VM Allocation: The overutilization of PM generates an excessive amount of heat, therefore thermal management of the data center is an important criteria for preventing the overheating of the PM and other damages. Hence, there is a need of an optimal resources allocation policy which not only reduces the energy consumption, but also minimizes the resources wastage, and thereby avoiding overheating of the PM at the cloud data center. Thus, multi-objective VM allocation algorithm is required to minimize the energy

consumption, resources wastage, and thermal temperature at the cloud data center.

1.7 Motivation

The energy efficient VM allocation and migration can reduce the energy consumption at the cloud data center. Further, to schedule the tasks on VM in such a manner that we not only reduce the energy consumption but also avoid the SLA violation at the cloud data center. However, energy efficient thermal aware VM allocation algorithm can reduce the energy consumption, resources wastage, and thermal temperature at the cloud data center. Hence, research work in this thesis focuses on the design and development of VM allocation & migration algorithms, and energy efficient SLA aware task scheduling policy and thus motivated the following issues.

1. Reducing the electricity cost of the data center by energy efficient VM allocation and migration.

2. Improving the corporate image towards Green IT while minimizing the power consumption and CO_2 emission and thereby saving the energy consumption at the cloud data center.

3. Providing sustainability to the data center by optimal utilization of the resources at the data center.

4. Extending the overall hardware life of the data center by reducing the thermal temperature of the data center.

5. Reducing the energy consumption and SLA violation at the data center by energy efficient SLA aware task scheduling.

6. Reducing the networking device energy consumption by network aware energy efficient VM allocation and migration.

1.8 Organization of Thesis

The reminder of this thesis is organized as follows. Chapter 2 reviews the existing works on energy saving at the cloud data center in terms of energy efficient VM allocation, migration, and tasks scheduling. Based on the outcome of the literature survey, the problem statement and research objectives are defined. Finally, highlighting the scope of the proposed research work for VM allocation, VM migration, and task scheduling algorithms at the cloud data center.

In Chapter 3, we describe proposed VM allocation algorithm based on Genetic Algorithm (GA) and Cat Swarm Optimization (CSO) known as the Hybrid of Genetic Algorithm and Cat Swarm Optimization (HGACSO). Further, to check the performance of proposed HGACSO algorithm for VM allocation, we compared our proposed algorithm with other state-of-the-art algorithms such as First-Fit, First Fit Decreasing (FFD), GA, and CSO.

In Chapter 4, we describe our proposed VM allocation algorithm known as Hybrid Genetic Algorithm and Particle Swarm Optimization (HGAPSO). Further, to check the performance of the proposed HGAPSO algorithm, we compared our proposed HGAPSO algorithm with First-Fit, FFD, GA, and Particle Swarm Optimization (PSO). After allocation of VMs to PMs, we proposed energy efficient VM migration and energy efficient tasks scheduling algorithms at the cloud data center. The proposed VM allocation, migration, and task scheduling algorithms not only save the power consumption, but also avoid SLA violation at the cloud data center.

In Chapter 5, we describe the proposed VM allocation algorithm based on the hybrid combination of GA, PSO, and SA known as HGAPSOSA. The proposed HGAPSOSA algorithm reduces the energy consumption, resources wastage, and thermal temperature at the cloud data center. Further, to check the performance of the HGAPSOSA algorithm, we compared HGAPSOSA with other state-of-the-art algorithms such as GA, PSO, and HGAPSO.

In Chapter 6, we describe our proposed exact algorithm for VM allocation in network data center environment. The proposed exact algorithm is based on the

branch-and-bound technique. Further, to select the optimal number of PMs and switches for VM allocation, we proposed lower bound. To migrate the VMs from one PM to another PM, we describe our proposed energy efficient VM migration algorithm for network-aware cloud data center.

In Chapter 7, we describe the performance evaluation of our proposed VM allocation algorithms known as HGAPSO, HGACSO, HGAPSOSA, and branch-and-bound based Exact algorithms.

Finally, Chapter 8 summarizes the contributions of the research work and highlights the possible directions for future research to save the energy consumption at the cloud data center.

1.9 Summary

This chapter described the cloud computing characteristics and types of cloud computing models, such as service model and deployment model. Further, this chapter discussed the cloud computing technologies such as virtualization, load balancing, service level agreement, etc., and energy efficient green cloud computing challenges, and motivation for research.

Chapter 2

Literature Review

This chapter presents the review of the existing power saving techniques at the cloud data center. The chapter also gives the problem statement, objectives, and the outcome of the literature survey from the existing power saving techniques at the cloud data center. Further, the main sources of power consumption in a cloud data center are Cooling devices (A.C.), Computing resources (PM), and Networking elements (Switches, Routers). Further, a PM in the cloud data center consumes two types of power, such as static power, and dynamic power. The power consumption of networking switch is dependant on the number of switched-on ports. Hence, to reduce the power consumption of the data center, we use different techniques such as energy efficient VM allocation, migration, and task scheduling etc. at the cloud data center.

Further, the energy efficient resources allocation in the form of VMs to PMs not only saves the energy consumption, but also reduces the resources wastage at the cloud data center. The migration of VMs from underutilized PM to energy efficient PM will switch-off underutilized or idle PM at the cloud data center. Hence, energy efficient VM migration not only saves the power consumption, but also reduces the SLA violation in terms of response time at the cloud data center. The energy efficient task scheduling allocates the task to the VM which not only saves the energy consumption, but also reduces the SLA violation at the cloud data center. Following are the details of the various state-of-the-art approaches for reducing the power consumption at the cloud data center.

2.1 Data Center Power Management Techniques

The cloud data center consists of two types of devices such as IT devices (eg. PM, Switches, Routers, Cables, etc.), and non IT devices (eg. Cooling devices, Pump room, Lights, Switch gear, etc.). Hence, the power management techniques at the

cloud data center are broadly classified into two categories such as IT devices, and non IT devices power management techniques. Figure 2.1 shows the classification of power management techniques at the cloud data center. Further, IT devices, power management techniques are again classified into two categories such as PM power management techniques and switch power management techniques. The PM power management techniques are again classified into two categories such as static and dynamic power management techniques.

The static power management techniques deal with different levels, such as circuit level, logic level, and architecture level of the PM. On the other hand, dynamic power management techniques are classified into two categories such as hardware and software level power management techniques. Further, we can reduce the power consumption of non IT devices by using energy efficient cooling devices, designing energy efficient cloud architecture, and appropriate site selection for the construction of cloud data centers. The power management of non IT devices at the cloud data center is beyond the scope of this thesis. Hence, the detailed description of IT devices, power management techniques are described as follows. Further, to reduce the power consumption of PM at the cloud data center, we need to discuss the static and dynamic power management techniques at the cloud data center. Hence, the existing works on PM power consumption using static and dynamic power management techniques are described as follows.

2.1.1 Static Power Management Techniques

The static power is consumed by the system components of a PM. The static power consumption is due to the leakage current of active circuits which are under the switched-on condition of a PM. Hence, this type of power consumption is independent of the clock rates of the CPU. Thus, it does not rely on the utilization of CPU. Further, we can specify this power consumption by the type of transistor used, and the technology applied to the processor of the PM. Thus, we can reduce the static power consumption by reducing the leakage current of the PM. The leakage current of the PM is reduced by three ways (reducing the supplied voltage, reducing the size of the circuit in the system, and cooling the system

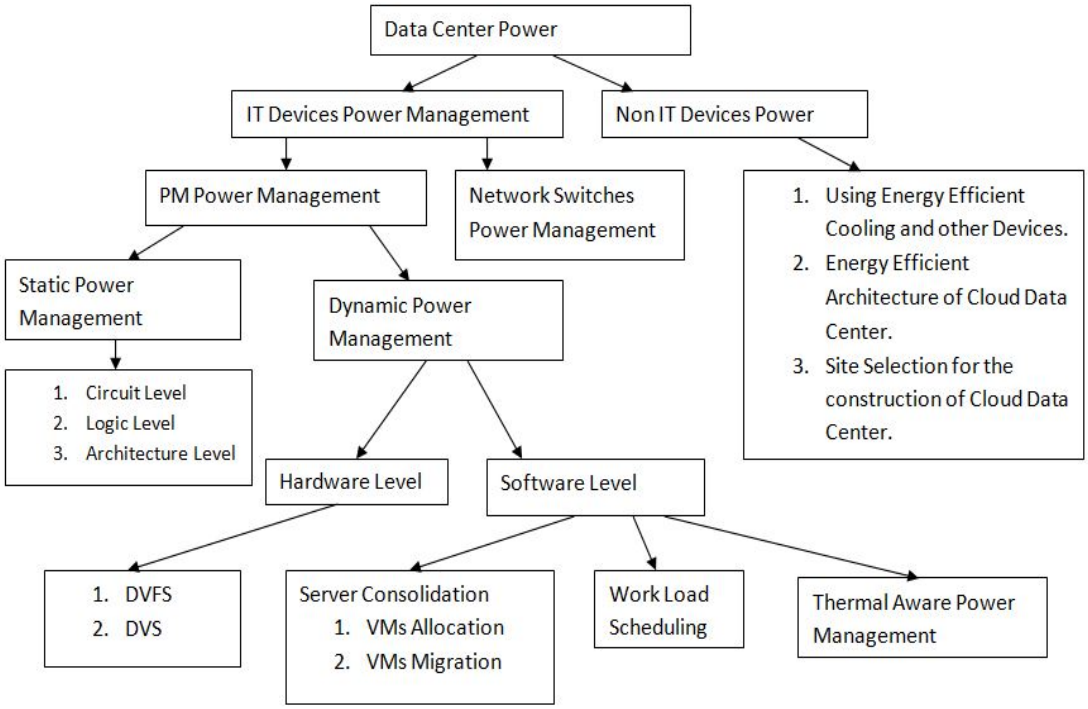

Figure 2.1: Classification of Power Management Techniques (Guyon et al. 2017).

by applying cooling technologies). Further, all three static power management techniques require a low level system design, and thus reduce the static power consumption of the PM at the cloud data center.

Many state-of-the-art works on static power management techniques of a PM have been developed. Further, these static power management techniques include all the optimization methods for efficient design of logic, circuit, and architecture of the PM. Andersen et al. (2009) designed a novel system architecture referred to as a fast array of wimpy nodes (FAWN). Further, they reduced static power consumption by using FAWN low power data intensive computing. In their proposed architecture, authors combined the low power CPU with small amount of local flash storage and thus resulting in an efficient parallel data access in the system.

Further, to reduce the static power consumption, Vasudevan et al. (2010) conducted the experiment on FAWN by applying various types of workloads. The experimental results demonstrated that low power CPU nodes were more energy efficient as compared to the conventional high performance CPU nodes in terms of energy efficiency. Caulfield et al. (2009) designed a novel power efficient architec-

ture referred to as Gordon architecture for reducing the static power consumption of cloud data centers. Further, authors reduced the static power consumption of the cloud data center by utilizing the low power processor and flash memory of the Gordon architecture. Further, Valentini et al. (2013) discussed more details about the static power consumption technique of a PM.

Thus, to save the static power consumption of a PM, we need an energy efficient hardware level system design. In addition to energy efficient hardware design, it is also important to consider the implementation of programs that are to be executed on the system. Hence, in efficient software design we can create the adverse effect on static power saving of the PM, and this may lead to power loss. Thus, code generation, instructions used in the code, and their order of execution are also important for the execution of an application at the CPU.

Tiwari et al. (1993), Su et al. (1994) analyzed the power consumption caused by the software at the hardware level. Further, the details of hardware level power management solution is beyond the scope of this thesis.

2.1.2 Dynamic Power Management Techniques

The dynamic power consumption of a PM relies on the current utilization of PM. Hence, dynamic power consumption depends on the clock rates, I/O activity, and usage scenario of CPU. Further, there are two sources of dynamic power consumption, such as switched capacitance, and short circuit current. The switched capacitance is the major source of dynamic power consumption and, it is due to the charging and discharging of the capacitor. Further, short circuit current is the minor source of dynamic power consumption. Hence, the dynamic power consumption of a PM is described by Eq. 2.1.

$$P_{Dynamic} = \alpha C V^2 f \tag{2.1}$$

Where, α represents the switching capacity of the PM; C, and V are the physical capacitance and voltage of the PM respectively; f is the frequency of the PM.

Thus, dynamic power consumption is reduced by reducing the switching activity of a PM, reducing the physical capacitance (by designing low level parameter such as transistor size), reducing the supply voltage of the PM, and reducing the

clock frequency of the PM. Further, to reduce the dynamic power consumption of the PM, we can classify this dynamic power management technique on the basis of applied levels such as (Hardware, Software).

2.1.2.1 Hardware Level Solution

The hardware level dynamic power management technique of a PM works on the basis of designing a methodology for providing the requested services generated by an application using minimum number of components. Further, the dynamic power management technique at the hardware level will switch-off the idle components of the system, and varies the frequency of CPU between two modes such as (minimum frequency (f^{min}), and maximum frequency (f^{max})). Hence, to reduce the dynamic power consumption of a PM, Benini et al. (2000), Kuo & Lu (2014) designed a technique by switching the frequency of CPU between two modes.

Further, the well known hardware level dynamic power management techniques like Dynamic Voltage Frequency Scaling (DVFS) and Dynamic Voltage Scaling (DVS) are widely used in today's modern PM (Snowdon et al. 2005). To reduce the power consumption of a PM, the voltage frequency of the CPU will be varied on the basis of its current workload. Thus, CPU works in between two modes of frequency, such as minimum frequency and maximum frequency. When the CPU is in idle condition (no workload) then CPU works on minimum frequency mode by switching-off the server components which are not useful.

Hence, using DVFS we can save up to 30% power consummation of a PM at the cloud data center. In the case of full workload, the CPU works in the full frequency mode. To reduce the dynamic power consumption at the hardware level, Ge et al. (2005) and Hsu & Feng (2005) suggested DVFS based approaches for energy efficiency of a single PM by restricting the use of CPU utilization well below the upper threshold value of CPU utilization.

The time-out based approach proposed by Kveton et al. (2007) kept the CPU in the low power state under no workload condition beyond a certain time-out threshold. The main limitation of this approach is that it does not keep the CPU in the lower state until the time-out period has passed; hence, it will not save the

energy during this time period.

Further, to save the dynamic power consumption of a PM at the cloud data center, Wu et al. (2014) proposed a DVFS based dynamic power management technique for reducing the energy consumption of the cloud data center. Their approach was based on DVFS that takes the task on priority basis and scheduled on the minimum amount of resources. By this way, the overall utilization of the data center is increasing and resulting in lower energy consumption at the data center. The disadvantages of their work are lower priority task suffered by lower response time and there may be a chance of SLA violation. The key limitation of dynamic power management techniques at the cloud data center is that these techniques are very sophisticated in nature but deal with single PM only and it is very difficult to change its policy. Hence, in order to resolve this issue, the software level power management techniques are described below.

2.1.2.2 Software Level Solution

Since, hardware level dynamic power saving techniques are very sophisticated, and at this level, it is very difficult to implement and modify this technique. But, DVFS based hardware level power saving technique provides an efficient direction to reduce the power consumption of a PM. The power consumption of the PM in the idle condition is approximately 70% power of its peak power consumption. Hence, multiple PM power saving techniques by switching-off the idle PM is the only solution to save the power consumption at the cloud data center. Thus, there is a need to provide the solution for saving the power consumption in the cloud data center environment. Further, the dynamic power consumption at software level in the cloud data center environment is reduced using energy efficient VM allocation and migration, energy efficient tasks scheduling, network aware VM allocation, and migration, etc. The detailed description of dynamic power saving techniques at the software level are described as follows.

A. Energy Efficient VM Allocation and Migration

The main focus in this approach is to minimize the energy consumption of the data center by using the minimum number of energy efficient PMs for the VM

allocation while switching-off unused PMs at the cloud data center. To solve VM allocation problem, Ajiro & Tanaka (2007), and Coffman et al. (1997) used approximation approaches, such as First-Fit, Best-Fit respectively. But approximation approaches will not provide a global optimum solution and Best-Fit approximation approach is not scalable in nature due to the long convergence time, and the other limitation of their suggested work is based on a single objective function.

Further, Beloglazov & Buyya (2013) suggested a Modified Best Fit Decreasing (MBFD) algorithm by first sorting the VMs in the decreasing order and PM in the increasing order on the basis of their processing capacity. After sorting of VMs and PMs, allocation of VM on PM is done by using First Fit Decreasing (FFD). The limitations of their work are: single objective based VM allocation, and MBFD which is not scalable in nature when large number of requested VMs are arriving at the cloud data center.

To save the energy consumption at the cloud data center by VM allocation, and migration, Chowdhury et al. (2015) designed number of VM allocation techniques based on the approximation algorithms. In their suggested work, authors placed the VMs energy efficiently on the PMs at the cloud data center. Further, authors migrated VMs from one PM to another PM by making clustering of the VMs at the cloud data center. Further, they used the CloudSim simulator (Calheiros et al. 2011) to check the performance of their proposed VM allocation and migration algorithm at the cloud data center. The limitation of their work is that the same types of VMs and PMs are considered for creating the homogeneous cloud data center environment. But, the authors did not consider the heterogeneous cloud data center environment.

Kim et al. (2014) designed a model for calculating the energy consumed by the VM without measurement of hardware at the cloud data center. Further, their suggested model estimated the energy consumption by the VM based on in-processor events generated by the VM. Based on this energy measurement model of VM, the authors suggested a VM scheduling algorithm for reducing the energy consumption at the cloud data center.

For energy efficient VM allocation in multi-tenant cloud data center environment, Dai et al. (2016) suggested two algorithms. Further, in their approach, authors considered the approximation algorithm and then explored two greedy approximation algorithms known as minimum energy VM scheduling algorithm (MinES), and minimum communication VM scheduling algorithm (MinCS). Further, these algorithms not only save the energy consumption, but also avoid the SLA violation at the cloud data center. But, the authors did not consider the heterogeneous cloud data center environment.

In addition to the existing approximation algorithms for the allocation of VMs at the cloud data center, some other works used different algorithms for similar type of VM allocation problem. Other researchers used bio-inspired and nature-inspired algorithms, such as GA, PSO, CSO, etc. for VM allocation at the cloud data center. Xiong & Xu (2014) designed energy efficient algorithm for VM allocation problem using PSO. The limitation of this work is that authors considered a single type of VM. Gao et al. (2013) suggested a multi-objective ant colony based VM allocation at the cloud data center. The limitation of this work is that authors considered the homogeneous data center in the form of VMs and PMs.

Wang et al. (2013) suggested the energy efficient VM placement in the cloud data center using PSO. The limitation of their work is that non-energy aware random reallocation of VM after velocity changes takes a lot of iterations and gave a non-optimal solution.

Further, in order to reduce the energy consumption and resources wastage, Xu & Fortes (2010) developed the multi-objective based VM placement in the cloud data center, and thereby minimizing both the energy consumption and the resources wastage using GA and fuzzy logic techniques. The problem with this approach is that some infeasible VMs are reallocated randomly to the PMs after crossover and mutation operations and this has resulted in an inferior quality solution for the next generation of chromosomes.

In addition to the existing approximation and bio-inspired algorithmic approaches, several predictive frameworks were proposed by Tighe & Bauer (2014),

Prevost et al. (2011) and thus reduced the number of switched-on PM. Rather than reserving the VM for each application all the time, Nguyen et al. (2013) dynamically adjusted the number of switched-on PM by predicting the workload of the cloud data center.

Xie et al. (2013) suggested a heuristic based VM allocation algorithm at the cloud data center. In their proposed VM allocation work, authors considered only the heterogeneous cloud data center environment. Further, different types of resources (CPU, Ram, Storage) are requested by the VM at the cloud data center. But the key limitation of their proposed work is that authors did not consider the energy efficient VM migration.

Further, Congestion-aware VM placement at the cloud data center is proposed by Yan et al. (2017). In their work, authors formulated a root assignment problem to VM by minimizing the maximum link utilization at the cloud data center with controlling splitting paths. Further, they proposed a polynomial time heuristic known as perturbation algorithm for the VM placement at the cloud data center. The key limitation of their work is that authors considered only the bandwidth requirement of the VM but did not consider other resources requirements such as (MIPS, Ram, and Storage etc.) at the cloud data center.

In order to resolve the problem of energy consumption at the cloud data center, Liu et al. (2017) proposed decision making models for participants in cloud energy storage. The proposed decision making models are developed for both users and Cloud Energy Storage (CES) operator. In the case of first decision model, authors assumed that consumers and CES operator consist of perfect forecast of loads and prices. In the case of second decision model, authors assumed that consumers use a more rudimentary control strategy and CES has only imperfect information about what is going to happen. Further, CES allows the consumers to charge and discharge their energy cost in the same way as they would with their self-owned distributed devices. Hence, by this way, we can save the energy consumption at the cloud data center. However, authors considered homogeneous cloud data center environment and their proposed work is not dealing with any

energy efficient VM allocation algorithm. Further, the energy consumption at the cloud data center is reduced by energy efficient task scheduling and the details are as follows.

B. Energy Efficient Task Scheduling

In order to reduce the energy consumption of the cloud data center, Shu et al. (2014) designed an improved clonal selection algorithm (CSA) for energy efficient workload scheduling based on cost and energy consumption model. Further, authors reduced the response time, and SLA violation at the cloud data center using CSA. The key limitation of this work is that the authors considered only the homogeneous cloud data center environment and further they directly applied task to the PM without considering the virtualized data center environment.

Wang & Zhang (2011) developed a green algorithm for the cloud data center network. The proposed algorithm not only reduced the completion time of task, but also reduced the energy consumption at the cloud data center. Further, in their proposed algorithm, authors effectively assigned the tasks via partial task shuffling, and thus adjusted the cloud servers speed. The key limitation of their proposed work is that authors directly applied the task to the PM without considering the virtualized cloud data center environment.

In order to reduce the limitations of the above works, Dabbagh et al. (2015) considered the virtualized cloud data center environment, and designed an energy efficient resource allocation and provisioning framework in virtualized cloud data center environment. In their proposed work, authors used the machine learning and stochastic approaches to predict the workload and thus predicted the VM as requested by the users. Using the predicted VM, authors switched-on those PMs which are used for the allocation of VMs at the cloud data center. Thus, by this way, authors switched-off those PMs which are not used for the VM allocation (idle PM) at the cloud data center, and thus reduced the energy consumption at the cloud data center. After allocation of VMs to PMs, authors allocated the task energy efficiently on VM at the cloud data center. The key limitation of their proposed work is that inaccurate work load prediction will lead to the SLA violation

at the cloud data center.

A novel task scheduling and server provisioning approach for reducing the energy consumption at the cloud data center is proposed by Liu et al. (2013). In their proposed task scheduling and sever provisioning approach, authors used greedy approach for task scheduling. Further, they proposed most efficient server first task scheduling scheme to minimize the energy consumption at the cloud data center. But, the key limitation of their proposed algorithm is that authors aggressively reduced the energy consumption by scheduling the task to the least number of servers at the cloud data center thus resulting in overutilization condition of PM, and there may be a chance of SLA violation at the cloud data center.

Further, to resolve the problem of energy consumption at the geographically distributed cloud data centers, Cheng et al. (2016) proposed an elastic power aware resource provisioning policy for heterogeneous workloads in self sustainable cloud data center. In their proposed work, authors scheduled workload on geographically distributed cloud data centers by powering the data center using renewable energy sources. Further, the core idea of their proposed work for searching an optimal energy efficient resources allocation is based on the simulated annealing and fuzzy logic. Their proposed resource provisioning approach (ePower) mainly dependant on renewable energy for powering the world-wide distributed cloud data centers to compute the heterogeneous workload. In their proposed work, authors reduced the energy consumption and SLA violation at the cloud data centers. But the key limitation of their proposed work is that the energy efficient workload scheduling within a data center is not consider by the authors.

Further, Beaumont et al. (2013) suggested several approaches to assign a set of servers to the client's at the cloud data center using the greedy heuristic. The goal of their proposed work was to maximize the overall throughput at the cloud data center. Further, authors considered both cases such as offline (clients known before), and the online (clients can join and leave the system at any instance) versions of resource allocation at the cloud data center. The key limitation of their proposed work is that authors maximized the overall throughput, but they

did not compute the energy consumption of the cloud data center.

In order to solve the energy consumption problem for an enterprise cloud, Pantazoglou et al. (2016) proposed the decentralized workload management technique. In their proposed work, authors designed a technique for energy efficient management of VMs instances which are provided by the enterprises. Further, authors maintained a hypercube to scale up and scale down the cloud when workload is varied over the period of time. Further, authors proposed VM migration technique by migrating the VMs from the underutilized cube and thereby switching-off the underutilized cubes, resulting in higher energy efficiency of the cloud data centers.

Dalvandi et al. (2017) proposed a workload scheduling model at the cloud data center. In their work, authors proposed a Sliding-Scheduling tenant request model that enables the tenants to specify the required time duration of their application and resources requirement within a certain window. Further, authors proposed a heuristic algorithm that not only reduced the power consumption, but also allocated the required resources by considering the request duration and current shut down time of the PM. Hence, by their proposed approach, authors reduced the power consumption and thus provided the resources guarantee to the application in the cloud computing. But the key limitation of their suggested work is that authors considered the calculation of resources request in advance, but it is a difficult task in real time scenario and further, authors did not minimize the energy consumption at the cloud data center.

Further, PSO based Adaptive Multi-objective Task Scheduling (AMTS) policy for the cloud data center is proposed by He et al. (2016). In their proposed approach, authors not only maximized the resources utilization but also minimized the task completion time at the cloud data center. The convergence time of their proposed PSO based AMTS algorithm is high, and there may be a possibility of SLA violation at the cloud data center.

In order to reduce the energy consumption at the cloud data center, a fast energy aware VM allocation, and task scheduling algorithms are proposed by Li et al. (2017). Further, in their proposed iterative algorithm, authors divided the

resources provisioning and task scheduling into multiple steps, and thereby reducing the energy consumption, time complexity, and execution of the proposed algorithms. Further, in their proposed energy efficient VM allocation and tasks scheduling algorithm, authors allocated the task to the energy efficient VM at the cloud data center. The key limitation of this approach is that it saved a very less amount of energy at the cloud data center, but authors did not consider the heterogeneous cloud data center environment.

Further, by applying aggressive energy efficient workload scheduling, and VM allocation policy using minimum number of PMs for task scheduling and VM allocation at the cloud data center, there may be a chance of increased thermal temperature of PM (due to overutilization of PM) at the cloud data center. We need a thermal aware VM allocation algorithm at the cloud data center. Thus, the detailed description of existing energy efficient thermal aware VM allocation, and task scheduling algorithms at the cloud data center are described below.

C. Thermal Aware VM Allocation, and Task Scheduling

Energy efficient resources allocation in the form of VMs to PMs and utilization of resources at the cloud data center for avoiding thermal hotspots (due to PM overheating) are conflicting objectives. Since by maximizing resources utilization, we can minimize the energy consumption of the cloud data center, but overutilization of resources will increase the thermal temperature, and thus resulting in thermal hotspots. Hence, for resources allocation at the cloud data center, we need to consider both of the aforementioned objectives.

Thus, to minimize the energy consumption and thermal hot spots, Lee et al. (2012) proposed a proactive cross-layer approach for providing an automatic management of the cloud data center. Further, in their proposed approach, authors required a continuous processing and analysis of real time feedback from multiple layers at the cloud data center. But the key limitation of their proposed work is that authors required information about the previous status of the data center to manage the thermal hot spots at the cloud data center, but the authors did not consider any thermal aware VM allocation algorithm at the cloud data center.

In order to resolve the problem of needing information about the previous status of cloud data center, Polverini et al. (2014) designed an online multi-objective batch job scheduling algorithm known as GreFar for the distributed cloud data centers. In their work, authors satisfied the energy cost, queuing delay, and thermal temperature of the cloud data centers. Further, using GreFar jobs scheduling algorithm, jobs are processed when the length of the queue is sufficiently large, or electricity prices are sufficiently low. The limitation of their work is that authors did not consider the thermal aware VM allocation, and task scheduling at a single cloud data center.

Further, energy efficient CPU utilization based thermal aware VM migration algorithm is suggested by Kinger & Goyal (2013). In their proposed work, authors suggested VM live migration approach on the basis of CPU utilization. Further, to migrate the VMs from one PM to another PM, authors set an upper and lower thresholds for CPU utilization and kept the CPU utilization between these threshold utilization values. Hence, by this way, authors reduced the energy consumption and thermal temperature of the cloud data center. The limitation of this work is that authors did not consider the thermal aware VM allocation algorithm at the cloud data center.

The thermal aware workload load balancing approach is proposed by Yao et al. (2015) for the cloud data center. In their proposed approach, authors suggested an adaptive power control policy by using the correlation between power consumption of servers and CRACs. Further, authors solved the power control problem by exploiting the Recursive-Least Square based Predicative Control (RPC). Hence, in their proposed method, the tasks were assigned uniformly to each of the servers (PMs) at the cloud data center and resulting in no overutilization of PM and thus kept the thermal temperature low. Their work is mainly focused on thermal aware task scheduling but did not focus on the thermal aware VM allocation and migration at the cloud data center.

Oxley et al. (2014) suggested a multi-objective (thermal, power, and co-location aware) resources allocation in heterogeneous cloud data center environment. In

their proposed work, authors suggested the most efficient task assigning to CPU core techniques using greedy heuristic, genetic algorithm, and nonlinear programming. Hence, using their proposed work, authors reduced the power consumption, and thermal temperature of a PM at the cloud data center. But, their work is focused on the energy efficient thermal aware task assigning but not focused on the VM allocation at the cloud data center. Further, genetic algorithm takes huge amount of time for the convergence, hence their algorithm did not give the optimal solution in terms of energy efficient thermal aware task assigning to the CPU core at the cloud data center.

Hence, in order to resolve the problem of minimizing the energy consumption, and thermal temperature at the cloud data center, Li et al. (2014) suggested an energy efficient thermal aware VM allocation algorithm at the cloud data center. In their proposed work, authors suggested an optimal trade-off between load consolidation and balance. Further, authors defined two values of CPU utilization (lower, and normal) and found the optimal PM for the VM allocation using GA at the cloud data center. The limitation of their proposed work is that, this approach will not give the optimal solution in the case of large search space and further it takes a lot of convergence time.

In addition to the existing energy efficient thermal aware VM allocation algorithm, Chen et al. (2015) designed a VLSI floor-planning algorithm for the cloud computing platform and thus reduced the thermal temperature of a PM. Further, authors suggested a parallel floor-planning algorithm using advanced adjacency probability cross entropy optimization and a new integer linear programming based resources provisioning for efficiently using the computing resources, and thus handled the thermal temperature of the PM at the cloud data center. But the limitation of their proposed work is that it deals with a hardware level thermal temperature reduction but it does not deal with software level, thermal temperature reduction at the cloud data center.

In order to reduce the energy consumption and thermal temperature of the cloud data center, Wang et al. (2015) proposed a power and thermal aware VM mi-

gration technique at the cloud data center. Further, authors divided the proposed VM migration technique into three steps such as (i) identification of overloaded and underloaded PM, (ii) selection of VMs for the migration, (iii) migration of VMs to the underutilized PM at the cloud data center. Hence, by their proposed thermal aware VM migration technique authors migrated the VMs from the overutilized PM to underutilized PM and thereby reducing power consumption, thermal temperature, and SLA violation at the cloud data center. Their proposed work dealt with VM migration only, but not dealt with energy efficient thermal aware VM allocation policy at the cloud data center.

Further, PM level temperature prediction at the cloud data center is proposed by Wu et al. (2016). In their proposed work, authors predicted the thermal temperature of CPU using Virtual Machine Manager (VMM), and temperature sensor. Hence, a Support Vector Machine (SVM) model was trained using the collected data and deployed this thermal temperature prediction model to real cloud data center environment. But the limitation of this work is that this model will not accurately predict the thermal temperature of CPU and wrong thermal temperature prediction may lead to SLA violation at the cloud data center.

The GA based energy efficient, thermal aware job scheduling algorithm known as (PETs) is proposed by Alsubaihi & Gaudiot (2016). Authors combined job mapping, core scaling, and thread allocation into one scheduler. Further, this scheduler does the job scheduling in terms of less (energy consumption, thermal temperature, and execution time) under peak power and peak temperature constraints at the cloud data center.

In order to resolve the energy consumption and thermal temperature problem at the cloud data center, Li et al. (2017) proposed a GRANITE holistic VM scheduling algorithm for energy efficient thermal aware VM allocation at the cloud data center. Further, the proposed GRANITE is based on the Greedy approach and thus minimized the total energy consumption of the cloud data center in two stages such as initial VM placement, and dynamic live migration. Meanwhile the CRACs capacity at the cloud data center is dynamically updated for achieving

better cooling efficiency. But, the authors did not consider the SLA violation during VM allocation and migration at the cloud data center.

All the above mentioned existing works with reference to reducing energy consumption, and thermal temperature dealt with non-network aware cloud data center environment, and saved only PM power consumption. Further, the existing works on networking device power consumption are described below.

D. Network Devices Power Consumption

The networking devices consume a significant amount of power (30%) of the IT device power consumption at the cloud data center. Hence, to reduce the networking devices power consumption at the cloud data center, Lawey et al. (2014) designed an energy efficient framework for multiple cloud services over non-bypass IP/WDM core networks. In their approach, authors reduced the energy consumption of cloud network by migrating the content on the basis of its access frequency. Further, they developed a heuristic approach for the placement of VMs, and thus saved the power consumption of networking devices at the data center. In their work, authors gave less emphasis on reducing the server energy consumption when compared to the network devices energy consumption. Thus, reducing the server energy consumption at the data center will play a vital role in minimizing the overall energy consumption of the cloud data center.

In order to reduce the power consumption at network aware data center environment by applying renewable energy sources, Gattulli et al. (2014) suggested a dynamic routing of on-demand optical circuit to power the data center using renewable energy. Hence, in their approach, authors reduced the CO_2 emission and energy consumption at the cloud data centers. Their work dealt with geographically distributed data centers in IP-over-WDM networks and thus energy saving within a single data center was not consider by them. Further, to reduce the power consumption of network switches, Li et al. (2015) suggested greedy approach for power saving at the data center. In their approach, authors showed that the power consumption of network switches at the cloud data center is dependent on both the number of active switches and its working duration. But authors did

not consider the PM power consumption at the cloud data center.

Gunaratne et al. (2008) studied the adaptive link rate for wired networks. Using the adaptive link rate method, authors dynamically adjusted the data rate of links in the wired network on the basis of their traffic requirements. By this way the idle components of switches in the network will be switched-off, however the remaining active components of the switches will fulfill the future network requirements. Hence, while saving the power consumption of the networking devices, the quality of service (QoS) should be satisfied by using the adaptive link rate. But the limitation of their proposed work is that authors did not consider the PM power consumption at the cloud data center.

Further, to reduce the networking switches power consumption, Nedevschi et al. (2008) designed and evaluated the performance of two methods of power management techniques of wired networks. In the first work, authors kept the switch components in sleep mode when they were in idle condition, whereas in the second case authors adjusted the rate of network operation on the basis of their offered workload. Further, their experimental results demonstrated that there was 50% energy saving in the case of lightly utilized wired networks. But authors did not consider the PM power consumption at the cloud data center.

A dynamic power management technique known as an Elastic Tree for the data center network is suggested by Heller et al. (2010). On the basis of current workload at the data center, the Elastic Tree technique determines the set of active networking elements such as switches, and links dynamically. In their work, authors proposed a greedy bin packing algorithm and a heuristic using the topology of data center network. But authors did not consider the PM power consumption at the cloud data center.

An energy aware algorithm for the management of data center network is proposed by Mahadevan et al. (2009). In their work, authors considered the sleep mode of idle components of networking switch in place of keeping the networking switches into sleep mode. Their proposed data center network power saving algorithm performed following operations such as (i) disabling the switch port in

the case of not forwarding any traffic, (ii) dynamically setting the data forwarding capacity of the port. (iii) switching-off those line cards when they do not have any active port. (iv) switching-off networking switches that are not in use.

Hence, in case of network aware cloud data center environment, the authors mainly focused on reducing the switch power consumption, but they did not take into account the PM power consumption at the cloud data center.

Table 2.1 summarises the existing power saving techniques of PM and their limitations, Table 2.2 summarises the existing power saving techniques of networking elements and their limitations, and Table 2.3 summarises the reduction of power consumption, thermal temperature, and their limitations at the cloud data data center, respectively.

2.2 Outcome of Literature Review

After an extensive literature review of the existing power management techniques as detailed in Table 2.1, Table 2.1, and Table 2.3, we identified the following open issues and research gaps for further reduction of the power consumption at the cloud data center.

- Most of the existing works on energy efficient resources allocation at the cloud data center considered the homogeneous cloud data center environment. Hence, we need a VM allocation technique for both homogeneous and heterogeneous cloud data center environments.

- In most of the existing works on energy efficient VM allocation, authors considered only one resource (CPU) requested by the VM. Hence, we need a multi-constraints (CPU, RAM, Storage, bandwidth) VM allocation algorithm at the cloud data center.

- To reduce the energy consumption, resources utilization, and thermal temperature at the cloud data center, we need a muli-objective VM allocation technique at the cloud data center.

- Most of the authors did not consider the SLA violation during VM migration, and task scheduling. Hence, we need an energy efficient SLA aware VM

Table 2.1: Existing Works on Power Saving Techniques at the Data Center

Authors	Methodolgy	Limitations
Liu et al. (2017)	SW based model, VM consolidation using DVFS.	Single PM power management in homogeneous data center environment.
Nathuji & Schwan (2007)	SW based model, VM consolidation using DVFS.	Single PM power management in heterogeneous cloud data center environment.
Nathuji et al. (2007)	SW based model, VM consolidation using DVFS.	Single PM power management in homogeneous cloud data center environment.
Raghavendra et al. (2008)	SW based model, VM consolidation using DVFS.	Single PM power management in homogeneous cloud data center environment.
(Verma et al. 2008)	SW based model VM consolidation using DVFS.	Single PM power management in heterogeneous cloud data center environment.
Kusic et al. (2008)	SW-based Model, On/Off Switching.	Single PM power management in homogeneous cloud data center environment.
Gmach et al. (2009)	VM consolidation using DVFS, on-off switching.	Single PM power management in homogeneous cloud data center environment.
Cardosa et al. (2009)	DVFS based SW model.	Single PM power management in homogeneous cloud data center environment.
Kumar et al. (2009)	SW model, VM consolidation.	Single PM power management in homogeneous cloud data center environment.
Jung et al. (2009)	SW based model.	Single PM power management in homogeneous cloud data center environment.
Song et al. (2009)	SW-based Model, VM Consolidation	Single PM power management in heterogeneous cloud data center environment.
Stillwell et al. (2009)	SW-based Model, On/Off Switching.	Single PM power management in homogeneous cloud data center environment.
Lefevre & Orgerie (2010)	SW-based Model, VM Consolidation.	Single PM power management in homogeneous cloud data center environment.
Gulati et al. (2012)	SW-based Model, VM Consolidation.	Single PM power management in homogeneous cloud data center environment.

Table 2.2: Existing Works on Power Saving at Network Aware Data Center

Authors	Methodology	Limitations
Son et al. (2017)	A SLA and energy efficient dynamic overbooking of resources for VM in SDN-based cloud data center.	Not dealing with energy efficient network aware VM allocation, and migration problem.
Wang et al. (2017)	Energy efficient bandwidth allocation using artificial intelligence based abstraction Model.	Not dealing with energy efficient physical resources allocation at the cloud data center.
Fioccola et al. (2016)	Energy aware resource framework for distributed cloud infrastructures to manage the network and IT infrastructure in network aware cloud data center.	Not dealing with energy efficient network aware resources allocation at the cloud data center.
Abdelaal et al. (2016)	Cost aware VM allocation technique using Software Defined Network (SDN) resource allocation strategy at the cloud data center.	Not consider the minimization of energy consumption in network cloud data center environment.
Duggan et al. (2016)	Network aware VM migration based on artificial intelligence technique known as Reinforcement Learning (RL).	Not dealing with energy efficiency and VM allocation at the cloud data center.
Li et al. (2016)	Energy aware workload scheduling based on heuristic algorithm at cloud data center.	Not dealing with VM allocation and migration at network aware cloud data center environment.

Table 2.3: Existing Works on Reducing Power and Temperature at Data Center

Authors	Methodology	Limitations
Chen et al. (2014)	Reduced data center thermal temperature using cyber physical approach with Computational Fluid Dynamics (CFD) model.	Adjusting transient temperature distribution and calibrate it using sensor feedback, not dealing with VM allocation and migration at the cloud data center.
Chaudhry et al. (2012)	Compared the inactive and proactive thermal aware scheduling and monitoring techniques at the cloud data center.	Did not consider energy efficient thermal aware VM allocation and migration at the cloud data center.
Song et al. (2015)	A muti-tiers thermal intelligent workload placement algorithm at the cloud data center.	Not dealing thermal-aware VM allocation and migration at the cloud data center.
He et al. (2015)	High Temperature Ambient (HTA) corrosion resistant technology for cooling data center temperature.	Not dealing with energy efficient thermal aware resources allocation at cloud data center.
Aransay et al. (2015)	Energy efficient thermal aware VM allocation using Trust-and-Reputation System (TRS).	Not dealing with VM migration policy, and considered homogeneous cloud data center environment.
Karn & Elfadel (2016)	Multi-objective (power, thermal, and performance) aware VM auto scaling policy at the cloud data center.	Not considered energy efficient thermal-aware VM allocation at the cloud data center.

migration, and task scheduling at the cloud data center.

- Most of the authors did not consider the PM power consumption in the direction of minimizing the energy consumption in network aware cloud data center environment. Hence, in order to reduce the power consumption of networking devices, and PM at the cloud data center, a network aware VM allocation and migration technique is required at the network aware cloud data center environment.

- Most of the authors did not consider the SLA violation during task scheduling, and VM migration at the network aware cloud data center. Hence, we need a SLA aware energy efficient task scheduling, and VM migration algorithms in network aware cloud data center environment.

2.3 Problem Statement

In order to solve the open issues and research gaps discussed in the previous section, we need to design and develop multi-objective (minimizing energy consumption, resources wastage, and thermal temperature), multi-constraint (CPU, RAM, Storage, etc.) VM allocation algorithm in both non-network and network aware cloud data center environments. Further, we need to design and develop an energy efficient SLA aware task scheduling, and VM migration policy for both non-network and network aware cloud data center environments. Further, the performance evaluation is to be carried out in both heterogeneous and homogeneous cloud data center environments. Hence, the research problem is stated as follows:

"To design and develop an energy efficient network/SLA aware VM allocation, migration, and tasks scheduling algorithms for the cloud data center".

2.4 Research Objectives

The objectives of this research work are as follows:

- To design and develop non-network aware energy efficient VM allocation algorithms at cloud data center using HGACSO, HGAPSO, and HGAPSOSA.

- To design and develop network aware energy efficient VM allocation algorithm at cloud data center using Branch-and-Bound based Exact algorithm.
- To design and develop energy efficient task scheduling policy for non-network and network aware data center using First-Fit approximation algorithm.
- To design and develop energy efficient VM migration policy for non-network and network aware data center using First-Fit approximation algorithm.

In order to accomplish the aforementioned objectives, we designed number of energy efficient VM allocation algorithms (HGACSO, HGAPSO, and HGAPSOSA) in non-network aware cloud data center environment, and branch-and-bound based Exact algorithm in network aware cloud data center environment. Further, we developed energy efficient SLA aware First-Fit approximation based task scheduling, and VM migration policies for both non-network and network aware cloud data center environments. The experimental results are carried out in both homogeneous and heterogeneous cloud data center environments.

2.5 Summary

This chapter provided a review of the classification of power management techniques at the cloud data center. Further, we discussed various existing works, such as multi-objective VM allocation and migration, energy efficient task scheduling, minimizing energy consumption of networking devices, etc. at the cloud data center. The research problem statement and objectives were defined based on the outcome of the literature review.

The following chapters discuss the issues and solution of multi-objective, multi-constraints VM allocation, migration, and task scheduling problems in both non-network, and network aware cloud data center environments. Further, the performance evaluation of proposed methods is carried out for both homogeneous and heterogeneous cloud data center environments.

Chapter 3

Energy Efficient VM Allocation, Migration Using HGACSO

The VM allocation, and migration are the crucial technologies in improving the efficiency of the cloud data center in terms of reducing the energy consumption, resources wastage, etc. This chapter deals with the multi-objective VM allocation to the PM using the proposed HGACSO algorithm at the cloud data center. The proposed HGACSO is the hybrid combination of Genetic Algorithm (GA) and Cat Swarm Optimization (CSO) known as HGACSO. The multi-objective based VM allocation to PM not only minimizes the energy consumption, but also maximizes the resource utilization at the cloud data center. Hence, to solve the Pareto optimal VM allocation problem (minimizing the energy consumption, and maximizing the resources utilization), we make an Euclidean distance based objective function to select the optimal number of PMs for the allocation of VMs.

Further, allocation and destruction of on demand VMs on commercial cloud data center are the common practices. Hence, we need an energy efficient SLA aware VM migration policy at the cloud data center. Thus, consolidates the VMs to less number of energy efficient PMs in such a way that we not only reduce the energy consumption but also avoid the SLA violation at the cloud data center. The research contributions towards designing an energy efficient cloud data center by VM allocation, and migration are described as follows.

1. To design and develop a multi-objective energy efficient VM allocation technique using HGACSO.

2. To design and develop an energy efficient VM migration policy using the First-Fit approximation.

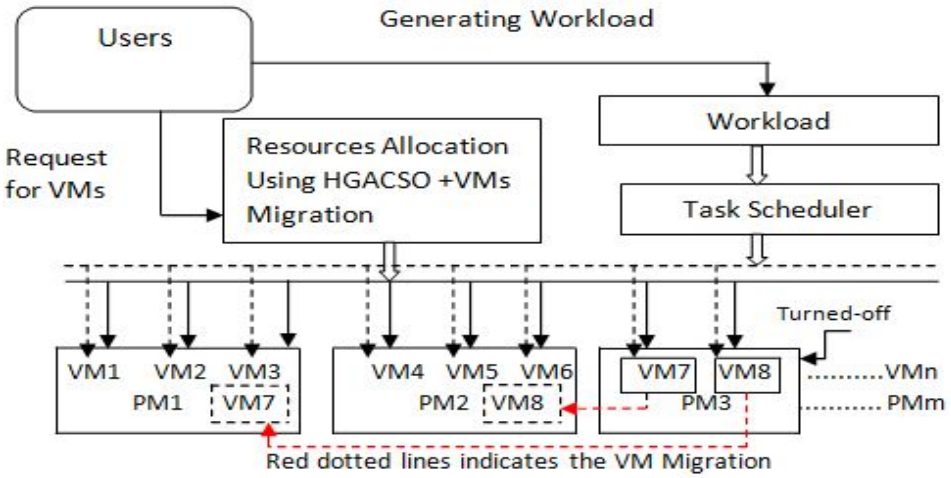

Figure 3.1: Flow Chart of VM Allocation, and Migration.

3.1 Proposed Work

The details of the proposed VM allocation, and migration techniques at the cloud data center are given in the following sections. Figure 3.1 shows the flow chart of the proposed VM allocation, and migration at the cloud data center. The proposed VM allocation, and migration techniques follow a systematic approach to resolve the energy consumption problem at the cloud data center. First we allocate the users requested VMs to PMs at the cloud data center using the proposed HGACSO. After allocation of VMs to PMs, we applied the proposed VM migration policy to migrate the VMs from underutilized PM to the energy efficient PM at the cloud data center.

Further, each user's request consists of information about of number of VMs, type of VMs, and time duration of the VM at the cloud data center. We considered four different types of VMs in our proposed work. Hence, the user requested query for time duration (t) is defined in terms of R_i=$\{VM_{small}(n1),\ VM_{medium}(n2),\ VM_{large}(n3),\ VM_{x-large}(n4)\}$, *where* $VM_{small}(n1)$, $VM_{medium}(n2)$, $VM_{large}(n3)$ $VM_{x-large}(n4)$ *are the number of small, medium, large, and extra large types of VMs requested by a user at the cloud data center, respectively.*

3.1.1 VM Allocation Using Proposed HGACSO

Before applying the HGACSO for the allocation of VMs to PMs at the cloud data center, we need to define a power consumption model of a PM at the cloud data

center, and design a mathematical model for multi-objective multi-constraints VM allocation problem at the cloud data center. The details of the power consumption model, a mathematical model for VM allocation problem, and description of proposed HGACSO are described as follows.

A. Energy Consumption Model of a PM

A dynamic power management technique for reducing the power consumption of a PM such as Dynamic Voltage Frequency Scaling (DVFS) is based on two states of the CPU (idle state, workload state). During idle (no workload) condition of the CPU, Operating System (OS) will switch-off those components of the PM which are not useful and thereby reducing the power consumption of the PM. Hence, CPU works on the minimum frequency mode (f^{min}). Further, in case of workload condition, the power consumption of the PM is dependant on the amount of workload applied to the CPU, and utilization rate of CPU.

The CPU works on the maximum frequency mode (f^{max}) under the condition of full CPU utilization. Thus, CPU works on two frequency modes such as (f^{min}, f^{max}). Further, in our proposed work, we consider all the PMs based on inbuilt DVFS technique and the power and energy consumption of a PM based on the approach of (Minas & Ellison 2009). The power consumption (P_j) and the energy consumption (E_j)(during the time duration of t_1 to t_2) of (pm_j) are described by Eqs. 3.1 and 3.2 respectively.

$$P_j = ([P_j^{max} - P_j^{min}])u + P_j^{idle} \tag{3.1}$$

$$E_j = \int_{t=t_1}^{t=t_2} P_j dt \tag{3.2}$$

Where, P_j^{max} and P_j^{min} are the maximum and minimum power consumptions of pm_j, respectively; u is the rate of CPU utilization between 0 and 1; E_j is total energy consumption of pm_j during time duration of t_1 to t_2.

If a data center has m number of PMs, then the total energy consumption of a data center (DC^{energy}) is defined by Eq. 3.3. The resources (mips, ram, storage) utilization of pm_j (u_j^d) is defined by Eq. 3.4. The average resource utilization of

the data center (DC^u) over time duration of t_1 to t_2 is defined by Eq. 3.5.

$$DC^{energy} = \sum_{j=1}^{m} E_j \tag{3.3}$$

$$u_j^d = \frac{\sum_{i=1}^{n} a_{ij} * VM_i^d}{pm_j^d} \quad \forall d \in \{mips, ram, storage\} \tag{3.4}$$

$$a_{ij} = \begin{cases} 1 & if \quad VM_i \quad allocated \quad to \quad pm_j \\ 0 & else \end{cases}$$

$$DC^u = \int_{t=t_1}^{t=t_2} \frac{\sum_{j=1}^{m} u_j^{mips} + \sum_{j=1}^{ram} u_j^{ram} + \sum_{j=1}^{m} u_j^{storage}}{|d| \sum_{j=1}^{m} b_j} dt \tag{3.5}$$

$\sum_{j=1}^{m} b_j > 0 \quad \& \quad |d| = 3$

Where, b_j is a binary variable indicating whether pm_j is used for VM allocation or not. The value of $b_j = 1$ if pm_j is used for the VM allocation otherwise it is 0; $|d|$=3 is defined for the number of resources such as mips, ram, storage.

B. Multi-Objective VM Allocation Problem Formulation

Let us consider a data center with n number of VMs and m number of PMs. We need to allocate VMs to PMs at the cloud data center in such a manner that it provides the global optimal solution in terms of minimizing the energy consumption ($p = min \sum_{j=1}^{m} P_j$) and maximizing the resources utilization ($u = max \sum_{j=1}^{m} u_j^d$) at the cloud data center. Thus, multi-objective VM allocation problem is referred to as Pareto optimal problem. Therefore, to calculate the global optimal solution for the VM allocation with two conflict objectives (minimizing energy consumption, maximizing resources utilization), we designed an objective function based on an Euclidean distance. The Euclidean distance based objective function gives a global optimal solution for both minimizing the energy consumption and maximizing the resources utilization is described by the following example.

Further, the other distance metrics such as the Hamming distance (it measures the distance between two strings bit by bit), Mahalanobis distance (it measures the distance between a point P and distribution D), Haversine distance (it gives the distance between two points on a sphere) etc. are not useful and feasible for making the objective function of a Pareto optimal optimization problem. Hence,

the considered Euclidean distance gives the minimum distance between two points in d dimension-space.

Further, Euclidean distance in this proposed work is not indicating the distance between the servers in the cloud data center, but Euclidean distance based objective function is used for selecting the best PM for the VM allocation. This can be achieved by calculating an optimal point between energy consumption and resource utilization. Therefore Euclidean distance is the feasible and the best choice for our proposed work. Further, the current resources utilization of PM_j (u_j^d) is in between 0 and 1. Thus, we use the normalized value of power consumption between 0 and 1 by dividing the current power consumption of PM_j (P_j) by their maximum power consumption value (P_{jmax}).

Example:- Let us consider two different types of PMs at the data center. The type1 PM consists of one processing element of 1000 mips and maximum power consumption of (P_{type1}^{max}). The type2 PM has two processing elements of 2500 mips each and maximum power consumption of (P_{type2}^{max}). From the specification of Intel Xeon processors available on Intel's web site (Minas & Ellison 2009), the power consumption is about 30% more on advance CPUs. Hence, on the basis of power model, the inequality conditions $P_{type1}^{max} < P_{type2}^{max}$ and $2 * P_{type2}^{max} > P_{type1}^{max}$ are satisfied.

Further, we consider 4 VMs requested at the data center with one processing element of 500 mips each. Hence, there are 16 different VM allocations possible at the data center. Out of these 16 combinations, we consider the most power efficient feasible allocations of VMs. The allocation of VMs at the data center is done using three cases and the details are shown in Figure 3.2. The function value of case two (f_2) is less as compared to the function value of case three (f_3) such as ($f_2 < f_3$) due to more wastage of mips in case 3 as compared to case 2. Further, the function value of case 1 (f_1) is more as compared to case 2 even though the wastage of mips in case 1 is 0% because of power inequality condition. Hence, the minimum optimization function value f_2 gives the optimal placement of VMs at the data center.

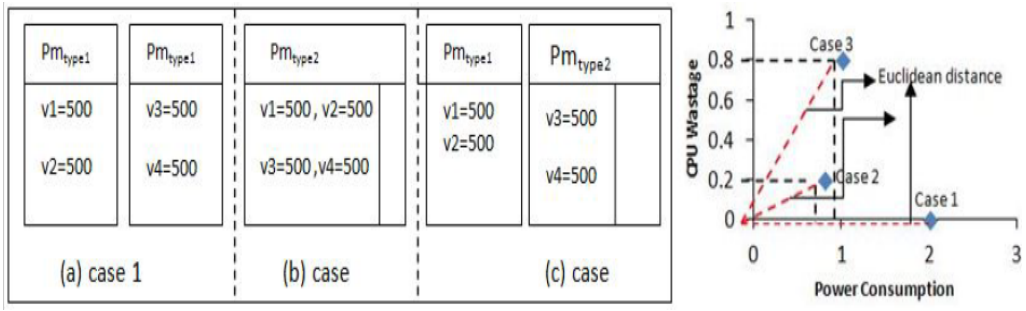

Figure 3.2: VM Allocation Scenario.

$$min \quad f = \sum_{j=1}^{m} \sqrt{(\frac{P_j}{P_j^{max}})^2 + (u_j^d - 1)^2} \tag{3.6}$$

$$VM_i^{pe} * a_{ij} \leq pm_j^{pe} \quad \forall j \in \{1, 2, 3, .., m\} \tag{3.7}$$

$$\sum_{i=1}^{n} VM_i^{mips} * a_{ij} \leq pm_j^{mips} \quad \forall j \in \{1, 2, 3, .., m\} \tag{3.8}$$

$$\sum_{i=1}^{n} VM_i^{ram} * a_{ij} \leq pm_j^{ram} \quad \forall j \in \{1, 2, 3, .., m\} \tag{3.9}$$

$$\sum_{i=1}^{n} VM_i^{storage} * a_{ij} \leq pm_j^{storage} \quad \forall j \in \{1, 2, 3, .., m\} \tag{3.10}$$

The multi-objective minimization function is described by Eq. 3.6. The constraints satisfaction of different resources such as processing elements, MIPS, RAM, and Storage, are defined by Eqs. 3.7 to 3.10 respectively.

C. Basics of GA, and CSO

Since the proposed HGACSO algorithm for multi-objective VM allocation is the hybrid combination of GA, CSO, hence to understand the functionally of the proposed HGACSO, we need to know the basic concepts of GA, and CSO.

(i) GA: The searching of solution to a problem is based on the random searching techniques in the search space. Further, each chromosome in the GA gives the possible solution in the search space. In the beginning we need to generate the number of chromosomes for the initial population. Further, we need to apply the crossover and mutation operations on the chromosomes. Hence, based on the fitness function value of chromosome, the new population is generated iteratively using crossover and mutation operations.

(ii) CSO: The CSO works on a random population of cats to search the solution of a problem. Further, each cat in the population of cats gives the possible solution to a problem. In the first iteration, we need to generate the number of cats for the initial population of cats. Further, we need to divide the cats into two modes such as seeking mode and tracing mode. In the seeking mode operation, we need to generate the number of copies of each cat which belongs to the seeking mode, and further to change the position of the cats randomly. Hence, after generating the copies of cats and changing the position of the cats randomly, we need to select the fittest cat position among the copies of the cats.

Further, in the tracing mode operation, we need to change the position of the cat by applying the velocity to the current position of the cat. The velocity of the cat is calculated by using the global best cat position. Hence, after applying seeking and tracing modes of operations, we need to select the fittest cats for the next generation. At each time step t using the global best cat position $\vec{x}_{lbesti}$, both velocity and position for time step (t+1) are calculated by Eqs. 3.11 and 3.12 respectively.

$$V_i^{t+1} = wV_i^t + cr(X_{gbest}^t - X_i^t) \tag{3.11}$$

$$X_i^{t+1} = X_i^t + V_i^{t+1} \tag{3.12}$$

D. Description of HGACSO

GA is based on the operations such as selection, crossover and mutation for the generation of new chromosomes. In GA, chromosomes share information with each other resulting in whole population moves in a group towards near global optimal solution. However, GA takes more convergence time if the solution space is large. In CSO, each of the cats deal with one of two operating modes such as seeking mode or tracing mode.

Cats go from one position to another position through the problem space by following the current optimum cat by changing its velocity and position. Hence, CSO has fast convergence, but due to dependency on the current optimum cat, sometimes the solution falls in local optima. Therefore, CSO and GA will complement each other from solution and convergence point of view. Initially generated

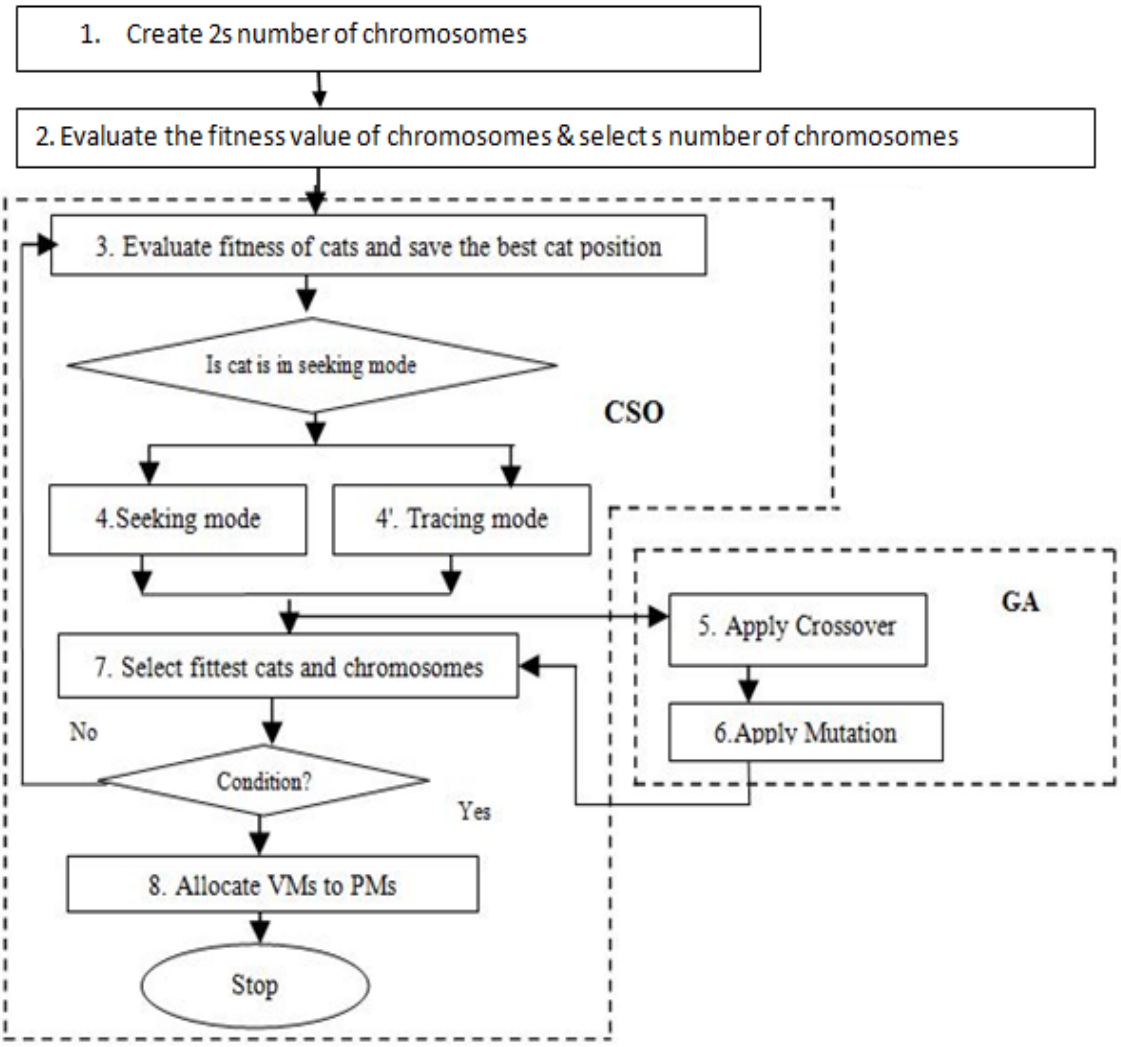

Figure 3.3: Flow Chart of HGACSO.

chromosomes are considered as cats in the applied CSO for the improvement of chromosomes. Crossover and mutation operations are applied for finding the best solution in successive iterations. Hence, by this way, we can allocate VMs on minimum number of energy efficient PMs.

The detailed flow chart of the proposed HGACSO algorithm is shown in Figure 3.3. In HGACSO, finite number of chromosomes is generated in the initial iteration. After the generation of chromosomes for the initial population, sort the chromosomes in the decreasing order on the basis of their fitness function. Discard half of the chromosomes from the population and remaining half chromosomes will go for the further operations. The best selected chromosomes will treat initial cats

for the CSO. Further, apply CSO algorithm for the treatment of initial solution by randomly dividing the cats in the swarm into two groups. One group of cats is treated in seeking mode and the other group of cats in tracing mode. After applying the operation of seeking mode and tracing mode, calculate the best cats among two groups of cats. Create a copy of the best cats and consider this created copy as chromosomes for the crossover and mutation operations.

After applying the crossover and mutation operations, arrange cats and chromosomes in the decreasing order on the basis of their fitness value. Discard half of the cats or chromosomes with least fitness value and remaining half cats or chromosomes will go for the new generation. The proposed HGACSO consists of three major operations namely, enhancement, crossover, and mutation. Details of these operations, chromosome encoding and population are presented below.

Chromosome Encoding: Let us consider a generation has '2s' number of chromosomes then out of theses 2s chromosomes generate '2s-1' chromosomes randomly i.e. Map VM to PM randomly like map_i=(VM_i map to PM_j) and the coded chromosome is shown in Figure 3.4. The remaining one chromosome is generated by using FFD technique (sorting the VMs in decreasing order on the basis of their resource capacity, and then allocate VM to PM using First Fit). The size of the chromosome is equal to the number of PMs in a cloud data center.

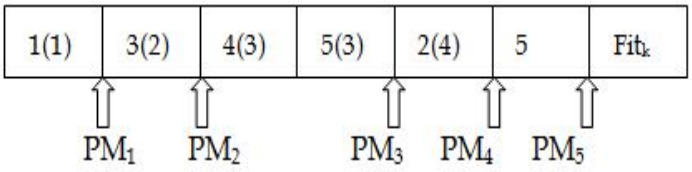

Figure 3.4: Chromosome Representation.

Fitness Function: The fitness of the chromosome is carried out by applying Euclidean distance based objective function f to each chromosome; the fitness value and the probability of the chromosome selection are defined by Eqs (3.13) and (3.14) respectively. The chromosomes with maximum Fit_k value will be con-

sidered for the next generation.

$$fit_k = \frac{1}{\sum_{j=1}^{m} \sqrt{(\frac{pm_j}{pm_j^{max}})^2 + (u_j^d - 1)^2}} \tag{3.13}$$

$$p_k = \frac{fit_k}{\sum_{k=1}^{s} fit_k}. \tag{3.14}$$

Cat Position and Encoding: The cat position is defined by m bit vector where m represents the number of PMs in a data center. The position of i^{th} cat in a m-bit vector at iteration t is given as X_i^t=$\{x_{i1}^t, x_{i2}^t, ..., x_{im}^t\}$ and each bit of the position vector is either 0 or 1. In CSO, each chromosome is treated as a cat. Hence, we need to convert the cat into binary representation like those PMs contain VMs by assigning 1 otherwise 0 and the details are shown in Figure 3.5.

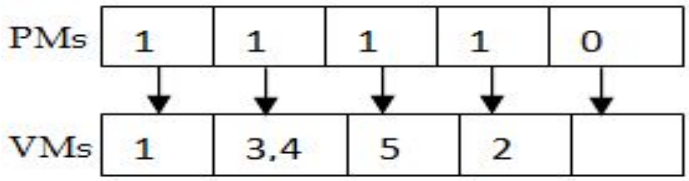

Figure 3.5: Cat Representation.

Seeking Mode: The steps performed in seeking mode are as follows:

(1) Select Maximum Ratio (MR) randomly such as (n_c) fraction of cats are in seeking mode and the remaining cats are in tracing mode.

(2) Produce Seeking Memory Pool (SMP) copies of the selected seeking cat.

(3) On the basis of Count of Dimensions (CDC) update the position of each copy by randomly adding or subtracting the Seeking Range of selected Dimensions (SRD) fraction of the current position value of copy cats with dimension=d.

(4) Evaluate the fitness value of all the copies.

(5) Select the best cat from all copies and place at selected seeking cat.

(6) Repeat Step 2 for all seeking cats.

Tracing Mode: The tracing mode operation changes the position of VM from one PM to another PM by applying the velocity to the cat. Further, the tracing mode operation consists of cat velocity, subtraction operator, addition operator, multiplication operator. The detailed description of cat velocity, and

these operators are described as follows:

Cat Velocity: The i^{th} cat velocity at iteration t is defined as m bit velocity vector V_i^t=$\{v_{i1}^t, v_{i2}^t, ..., v_{im}^t\}$. This velocity vector changes the cats position and changing the cats position by velocity is meant for changing the position of a VM from one PM to another PM. Each bit of the velocity is either 0 or 1. In the first iteration, we apply the random velocity to the cats. The random velocity is the m (number of PM) length binary stream of 0s and 1s.

Subtraction Operator: The subtraction operator is defined as the position difference between two VM position vectors. In CSO, the subtraction operator is represented by symbol Θ. The resultant bit value of the subtraction of two position vectors (X_i^t Θ X_i^t) is 1 if both have same bit value; otherwise it is 0 (Ex: (1,1,1,0,1) Θ (1,0,1,0,0)=(1, 0, 1, 1,0)).

Addition Operator: The addition operator is defined as the velocity change of the cat. In CSO, it is represented by symbol $\oplus$ and is defined as $P_1V_1^t \oplus P_2V_2^t\oplus, ...P_nV_n^t$. This represents the velocity change using V_1^t with probability P_1, and using V_n^t with probability P_n. In CSO, each probability $P_i, \sum_{i=1}^{n} P_i = 1$ is defined as an inertia coefficient. The results of the addition operation are explained as follows. For example, 0.3(1,1,0,1,0)$\oplus$0.7(1,1,1,0,0)=(1,1, $\neq$, $\neq$, 0). In this case, the probability of occurrence of 1 at 1^{st} and 2^{nd} bit positions is 1 and the probability of occurrence of 0 at the 5^{th} bit position is 1. So the bit values at 1^{st}, 2^{nd}, and 5^{th} positions are 1, 1, and 0 respectively, and the remaining positions are uncertain (denoted by $\neq$). To calculate the uncertain bit, we need two inertia weight coefficients P_1^i, P_2^i, and these are defined as follows:

$$f(X_i^t) = \sum_{j=1}^{m} \sqrt{(\frac{P_j}{P_j^{max}})^2 + (u_j^d - 1)^2} \tag{3.15}$$

$$P_{1i} = \frac{f(X_i^t)}{f(X_i^t) + f(X_{gbest}^t)} \tag{3.16}$$

$$P_{2i} = \frac{f(X_{gbesti}^t)}{f(X_i^t) + f(X_{gbest}^t)} \tag{3.17}$$

Where, $f(X_i^t)$ denotes the fitness of a cat i for the solution X_i^t; X_{gbest}^t represents the global best cat position. For calculation of inertia weight coefficients, we need

to consider high energy efficiency and maximum utilization with high probability.

$$\begin{cases} uncertain \quad bit = q_1, \quad if \quad rand \leq P_{1i} \\ uncertain \quad bit = q_2, \quad if \quad P_{1i} < rand \leq P_{2i} \end{cases}$$

Where, q_1 is the respective bit of the cats velocity vector before updating; q_2 is the corresponding bit value of the global best cat vector.

Multiplication Operator: The multiplication operator is represented by operator $\otimes$. It is defined for updating the cat position ($X_i^t \otimes V_i^{t+1}$) that represents the i^{th} cat current position X_i^t and it is updated by velocity V_i^{t+1}. The computation rule for the new cat position is as follows:

(i) If the corresponding bit value of the velocity vector is 1, then the corresponding bit value of the new position vector is not adjusted; (ii) If the corresponding bit value of the velocity vector is 0, then it will be adjusted. For example (1,1,1,0,1)$\otimes$(1,0,1,0,1), where (1,0,1,1,1) is the position vector and (1,0,1,0,1) is the velocity vector. The 2^{nd} and 4^{th} bits of the velocity vector are 0s and hence VMs are allocated to PMs when the 2^{nd} and 4^{th} positions of VM are updated. Finally, the CSO is used for computing the velocity and updated position as per the following details.

$$V_i^{t+1} = wV_{i,d}^t \oplus c * r(X_{gbest,i}^t \Theta X_i^t) \tag{3.18}$$

$$X_i^{t+1} = X_i^t \otimes V_i^{t+1} \tag{3.19}$$

Crossover Operation: After improving the fitness quality of the chromosomes using CSO, we need to apply GA operations (crossover, and mutation). Further, before applying the crossover and mutation operations, we need to store one copy of the cats temporarily, and another copy will go for the crossover operation as shown in Figure 3.3. The CSO generated cats are considered as chromosomes for applying the crossover, and mutation operations. In the proposed HGACSO algorithm, we used single point crossover operation on the chromosomes since multi-point crossover operation not only migrates more number of VMs from one chromosome to another chromosome, but also degrades the fitness value of the chromosomes and further generates more number of infeasible VMs

in the chromosomes. Therefore, reallocating these infeasible VMs, we need an extra computational time and hence, we did not consider multi-point crossover operation. In the proposed HGACSO, we randomly selected a map_i location on each chromosome and after this location, we exchanged all the VMs between two chromosomes in the current generation. The details of single-point crossover operation are shown in Figure 3.6.

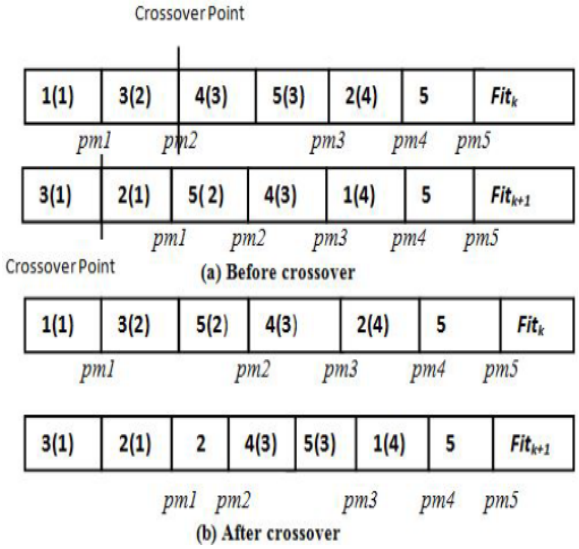

Figure 3.6: Crossover Operation.

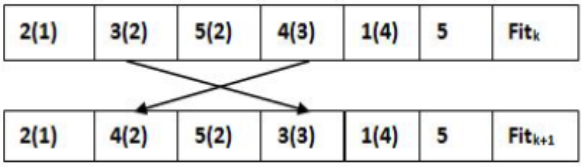

Figure 3.7: Mutation Operation.

Mutation Operation: After completion of the crossover operation, we performed a mutation operation on the chromosomes by randomly selecting two PMs in a chromosome and the details are shown in Figure 3.7. We select one VM on each of the selected PM and then exchange the selected VM.

Reallocation of Infeasible VMs: After applying the CSO, crossover, and mutation operations to the chromosomes, there is a possibility that some cats and chromosomes will be in infeasible condition because of migration of VMs from one

PM to another PM. Hence, we need to reallocate the infeasible VMs in the cats and chromosomes after each operation. The feasibility of the cats and chromosomes is calculated by using constraints defined by Eqs. 3.7 to 3.10.

The reallocation of infeasible VMs in the cats and chromosomes is carried out by First-Fit based approximation. At the end of each iteration (after applying CSO, and GA), we have one set of cats and another set of chromosomes. Hence, by using these sets of cats and chromosomes, we need to select the fittest chromosomes and cats for the next iteration. The selection of buffer cats and chromosomes is done by using roulette wheel selection policy as defined by Eq. 3.14.

Termination Conditions: The proposed HGACSO will be terminated when one of the following conditions is satisfied. (i) When the number of iterations is greater than the maximum set value of iterations. (ii) When there is no improvement in successive iterations or the repetition of chromosomes and cats value.

The steps of the proposed HGACSO are described by Algorithm 3.1. Steps 1-11 describe the initialization of (s-1) number of random chromosomes. Step 12 describes the generation of one chromosome using the FFD technique. Step 13 describes the termination conditions of the proposed HGACSO. Steps 14 to 16 describe the mapping of chromosomes to cats. Step 17 describes the calling of Procedure 3.1 for applying CSO operations. Step 18 describes the calling of crossover operation. Step 19 describes the calling of Procedure 3.2 for back-filling infeasible VMs. Steps 20 to 21 describe the calling of mutation operation. Step 22 describes the back-filling of infeasible VMs. Steps 23 and 24 describe the selection of fittest (cats and chromosomes), and increment the iteration value respectively. Finally, Step 25 generates the final mapping list of VMs to PMs at the data enter.

The detailed steps of CSO operation are described by Procedure 3.1. Further, the CSO consists of two operations such as (seeking and tracing). Steps 1-4 describe the calculation of position and initialization of velocity in the case of initial iteration. Position and velocity reassignment are described by Steps 5-8. Random flag values (0 or 1) assignment for division of cats in seeking and tracing modes are described by Steps 9-10. Step 11 describes the calculation of best cat

Algorithm 3.1 HGACSO

Initialize: t←1, t^{max}, s

Input: VMs list (VM[]), PM list (PM[])

Output: Mapping of VM to PM

1: **for** $i \leftarrow 1$ to (s-1) **do**
2: **while** $VM[] \neq empty$ **do**
3: Randomly select VM_i from VM[]
4: Randomly select pm_j from PM[]
5: **for** Each d **do**
6: **if** $VM_i^d \leq pm_j^d$ **then**
7: Allocate $VM_i to pm_j$
8: Update $pm_j^d = pm_j^d - VM_i^d$
9: Remove VM_i form VM[].
10: **else**
11: go to 4
12: Call FirstFit(Chr_{2s})
13: **while** Termination condition==false **do**
14: Select s fittest Chromosomes.
15: **for** $i \leftarrow 1$ to s **do**
16: $Cat_i \leftarrow Chr_i$
17: Call ***Procedure 3.1***
18: Call Crossover($Parent_1, Parent_2$)
19: Call ***Procedure 3.2***
20: **for** $i \leftarrow 1$ to s **do**
21: Call Mutation(Chr_i)
22: Call ***Procedure 3.2***
23: Select s fittest cats or chromosomes.
24: t← t+1
25: Output: Mapping list of VM tp PM.

Procedure 3.1: Applying CSO

if $t == 1$ **then**
    for $k \leftarrow 1tos$ **do**
        $X_i^t \leftarrow X_k^0$;
        $V_i^t \leftarrow V_K^0$;
Else
for k← 1 to s **do**
    $X_i^t \leftarrow X_k^{t+1}$;
    $V_i^t \leftarrow V_K^{t+1}$;
for k← 1 to s **do**
    $Cat_k^{flag} = 0$ or 1
    Calculate(X_{best}^t)
for $k \leftarrow 1tos$ **do**
    if $Cat_k^{flag} == 1$ **then**
        Cat is in seeking mode
        for l← 1 to SMP **do**
            Cat_{lk}
            Change dimension d
        Calculate best Cat
        $Cat_k \leftarrow Cat_{best}$
for k← 1 to s **do**
    if $Cat_k^{flag} == 0$ **then**
        Cat is in tracing mode
        Calculate V_k^{t+1} Using Eq. 3.18
        $X_k^{t+1} \leftarrow X_k^t + V_k^{t+1}$

Procedure 3.2: Removing/Refilling of Infeasible VMs

1: **for** i← 1 to s/2 do **do**
2: $VM'[i] == empty$
3: **for** i←1 to s/2 do **do**
4: **for** j←1 to m do **do**
5: **for** k← 1 to d do **do**
6: **if** $\sum_{i=1}^{n} a_{ij} VM_i^d > pm_j^d$ **then**
7: Remove VM_i from pm_j
8: $VM'[i].add(VM_i)$
9: **for** i← 1 to s/2 **do**
10: Backfill VM from $VM'[i]$ to Chr_i or $Part_i$ using FFD

position. Steps 12-19 describe the seeking mode operation. Steps 20-24 describe the tracing mode operation.

Further, the detailed steps of removing and backfilling infeasible VMs to PMs are described by Procedure 3.2. Steps 1 to 2 describe the initialization of empty VM list for all individuals. Checking of infeasible VMs using different dimensions of the PM, and further adding these infeasible VMs (by applying FFD technique) is described by Step 10.

3.1.2 VM Migration Using First-Fit

After allocation of VMs to PMs at the cloud data center, we need to check the lifetime of the VM as specified by the users in their request. Thus, we need to destroy the time expired VMs at the cloud data center. Further, destroying the time expired VMs from the PM may lead to an underutilized condition of PM at the cloud data center. Hence, we need to migrate the VMs from underutilized PMs to the energy efficient PMs and thereby switching-off underutilized/idle PMs at the cloud data center. This will result in saving energy consumption at the cloud data center. Further, the migration of VMs from underutilized PM to energy efficient PM not only saves the energy consumption, but also avoids the SLA violation at the cloud data center. Hence, we need a VM migration algorithm which migrates

the VM from underutilized PM to energy efficient PM in a quick period of time. Hence, we designed a First-Fit approximation based energy efficient SLA aware VM migration policy for a cloud data center.

Each PM consists of two different states such as overutilization and underutilization. The underutilization and overutilization states are related to energy saving and SLA violation respectively. Underutilization state is related to the wastage of energy for PM due to the idle or underutilized state of PM which consumes 70% of the peak energy of a PM. In the case of overutilized state the utilization of the PM is more compared to specified maximum utilization. Further, when the CPU utilization of the PM is less compared to specified minimum CPU utilization value then the migration of VM takes place from the underutilized PM and the details are described as follows.

Let us consider the minimum threshold value of CPU utilization for a PM_i is pm_i^{min} and the current CPU utilization at instance t is pm_i^u. The migration of VM from PM_i will take place when $pm_i^u < pm_i^{min}$. After selection of underutilized PM, we need to remove all the VMs form the selected underutilized PM and reallocate the removed VMs using FFD (sort the PMs on the basis of their resources utilization in decreasing order and reallocate VMs using First-Fit technique). In the proposed work we used less than 30% and greater than 75% of CPU utilization as the underutilized and overutilized states of a PM respectively.

3.2 Experimental Setup, Results and Analysis

To check the performance of proposed HGACSO, and VM migration technique, we consider different types of VMs and PMs at the cloud data center. Further, we conducted the experiment in both homogeneous and heterogeneous cloud data center environments. The details of the experimental setup and results & analysis are discussed as follows:

A. Experimental Setup: The proposed VM allocation and migration techniques deal with large number of PMs and VMs at a single data center. Therefore, we used CloudSim simulator (Calheiros et al. 2011) for checking the performance of the proposed work. In the simulator, we changed the default VMs instances

Table 3.1: Configuration of PMs

PM Type	PE	MIPS	RAM(GB)	STORAGE(GB)
ProLiantM110G5XEON3075	2	2660	4	160
IBMX3250Xeonx3480	4	3067	8	250
IBM3550Xeonx5675	12	3067	16	500

Table 3.2: Configuration of VMs

VM Type	PE	MIPS	RAM (GB)	STORAGE (GB)
Small	1	500	0.5	40
Medium	2	1000	1	60
Large	3	1500	2	80
X.large	4	2000	3	100

by Amazon EC2 VMs instances (Amazon-Website 2014) and three different configured PMs such as Dell (Wu et al. 2014), and IBM (IBM-Switch-Model 2014) for creating the real homogeneous and heterogeneous data center environments in the form of PMs and VMs. The overall configuration of the PMs and VMs are described in Table 3.1 and Table 3.2 respectively.

B. Results and Analysis: To evaluate the performance of our proposed HGACSO algorithm with other state-of-the-art algorithms such as First-Fit, First Fit Decreasing (FFD), GA, and CSO in terms of energy consumption and resources utilization. The experimental results are carried out in two different cases. In Case-1, we consider different combinations of PMs and VMs (PMs are fixed at 60% of VMs) in two different data center environments (homogeneous and heterogeneous). Further, in Case-2, we check the performance of our proposed HGACSO algorithm by keeping number of PMs constant and varying number of VMs in both cloud data center environments (homogeneous and heterogeneous). The number and types of PMs used for different amount of VM allocation in heterogeneous and homogeneous data center environment is shown in Table 3.3 (Case-1). The HGACSO algorithm utilizes less number of PMs for VMs allocation as compared

Table 3.3: Number of PMs Used for VM Allocation at the Cloud Data Center

VMs(PMs)	Homogeneous					Heterogeneous				
	First Fit	**FFD**	**GA**	**CSO**	**HGACSO**	**First Fit**	**FFD**	**GA**	**CSO**	**HGACSO**
100(60)	48	46	44	41	37	(14, 18, 8)	(13,16,8)	(11, 15, 8)	(10, 14,7)	(10,12,6)
200(120)	96	92	88	82	74	(28, 36, 16)	(26,32,16)	(22,30,16)	(20,28,14)	(20,24,12)
400(240)	192	184	176	164	148	(56,72, 32)	(52,64,32)	(44,60,32)	(40,56,28)	(40,48,24)
600(360)	288	276	264	246	222	(84,108,48)	(78,96,8)	(66,90,48)	(60,84,42)	(60,72,36)
800(480)	384	368	352	328	296	(112,144,64)	(104,128,64)	(88,120,64)	(80,112,56)	(80,96,48)
1000(600)	480	460	440	410	370	(140,180,80)	(130,160,80)	(110,150,80)	(100,140,70)	(100,120,60)

to the First-Fit, FFD, GA, and CSO. Since, a hybrid combination of GA, and CSO has improved the fitness of the chromosomes by applying CSO before applying the crossover and mutation operations. Thus, the crossover and mutation operations produced good quality children, and thereby achieving the near global optimal solution for VM allocation.

Case-1

Figure 3.8, Figure 3.9 and Figure 3.10 show the CPU, RAM, and Storage utilization respectively in heterogeneous cloud data center environment while taking different combinations of VMs and PMs using First-Fit, FFD, GA, CSO, and HGACSO algorithms. The CPU utilization, RAM utilization, and Storage utilization of the proposed HGACSO are high as compared to the First-Fit, FFD, GA, and CSO. Since, HGACSO used near optimal combination of (Type 1, Type 2, Type 3) PMs for VMs allocation and hence, HGACSO results in very less CPU wastage as compared to the First-Fit, FFD, GA, and CSO. The CPU utilization in the case of HGACSO is more as compared to First-Fit, FFD, GA, and CSO since HGACSO uses less number of PMs used for VMs allocation. The utlilization of CPU, RAM, and Storage for the proposed HGACSO is improving when more number of requested VMs are allocated to the PM. The number of used PMs for the VMs allocation in HGACSO is continually decreasing as compared to the other VM allocation techniques.

Further, to check the performance of proposed HGACSO based VM allocation algorithm in a homogeneous environment, we consider only a single type of PM (Type 2 PM) in the cloud data center. The configuration details of Type 2 PM

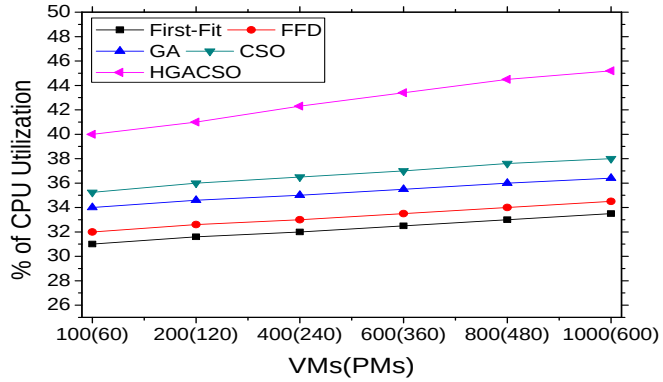

Figure 3.8: % of CPU Utilization in Heterogeneous Data Center.

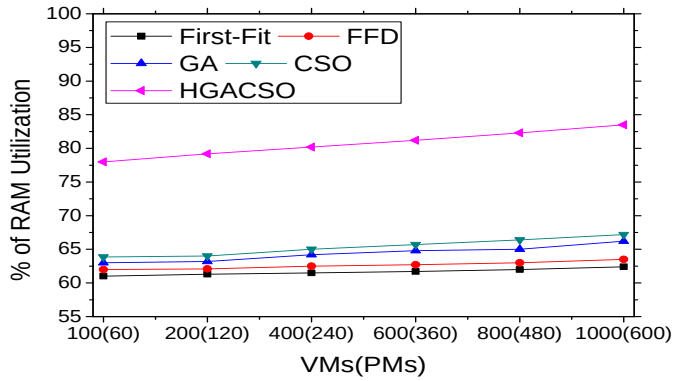

Figure 3.9: % of RAM Utilization in Heterogeneous Data Center.

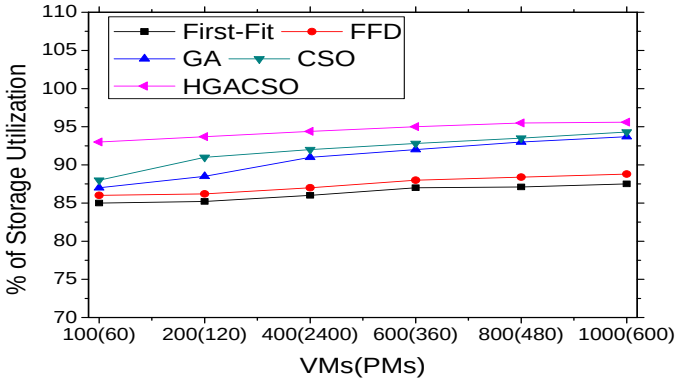

Figure 3.10: % of Storage Utilization in Heterogeneous Data Center.

is given in Table 3.1. The number of PMs used by the different VM allocation techniques for different combinations of VMs(PMs) is shown in Table 3.3. The HGACSO used less number of PMs for the VM allocation due to the global optimal placement of VMs to PMs. The experimental results for the homogeneous

environment show that HGACSO algorithm performs better than other VM allocation techniques in terms of energy consumption and resources utilization.

Figure 3.11, Figure 3.12, and Figure 3.13 show the CPU, RAM, and Storage utilization of the homogeneous data center for different VM allocation techniques using different combinations of VMs(PMs) respectively. The CPU, RAM, and Storage utilization of the data center is maximum in the case of HGACSO when compared to other VM allocation techniques. This is due to the less number of energy efficient PMs used for the VM allocation. Since, more number of PMs are switched-off and thus, minimizes the resources wastage. The resources utilization graph is almost straight line in all VM allocation techniques in both homogeneous and heterogeneous cloud data center environments because of number of used PMs are increasing in the same proportion as number of VMs are increasing resulting in less variation in resources utilization.

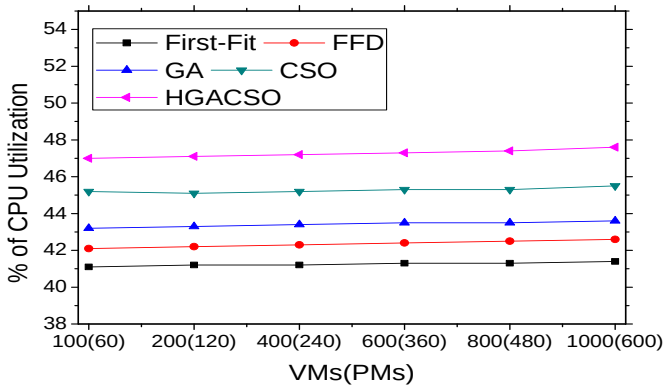

Figure 3.11: % of CPU Utilization in Homogeneous Data Center.

Figure 3.14 and Figure 3.15 show the power consumption and multi-objective function value of the cloud data center for different VM allocation techniques using different VMs(PMs) combinations in homogeneous data center environment. The power consumption of the data center in the case of HGACSO is low when compared to the First-Fit, FFD, GA, and CSO. Since, HGACSO has used less number of PMs for VM allocation since, idle PMs (which are not used for VM allocation) are switched-off and hence HGACSO consumes less power consumption. Further, the multi-objective function value is low in the case of HGACSO when

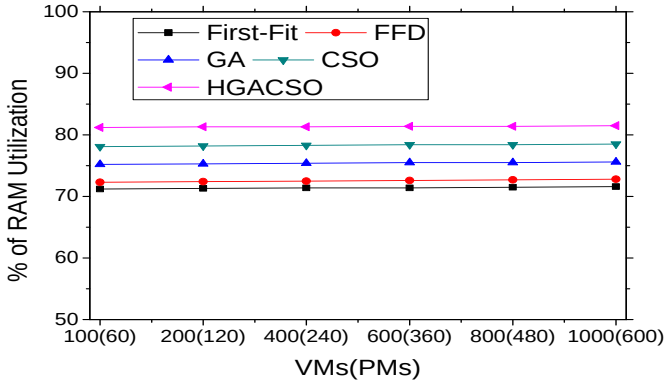

Figure 3.12: % of RAM Utilization in Homogeneous Data Center.

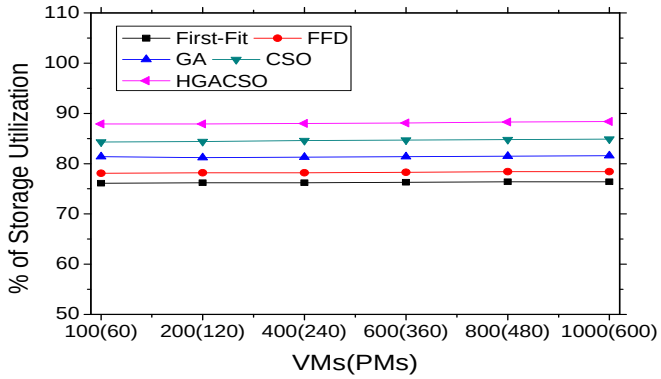

Figure 3.13: % of Storage Utilization in Homogeneous Data Center.

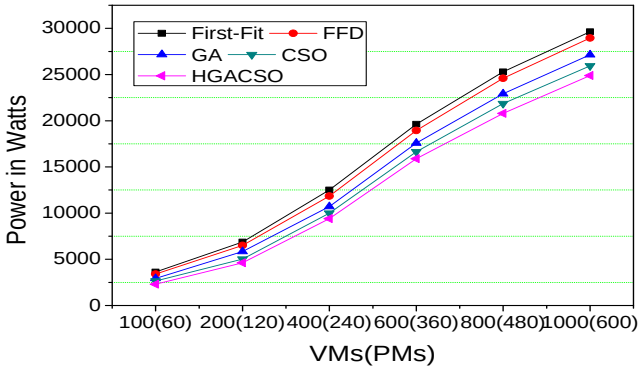

Figure 3.14: Power Consumption in Homogeneous Data Center Environment.

compared to other First-Fit, FFD, GA, CSO due to less resources wastage and low power consumption at the cloud data center.

Figure 3.16 and Figure 3.17 show the power consumption and resources utilization based on multi-objective function for different combinations of VMs and

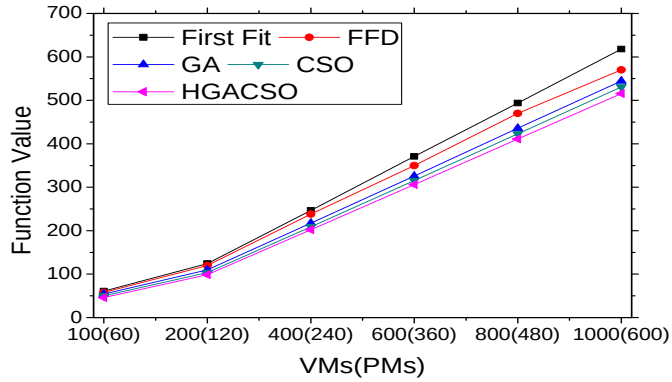

Figure 3.15: Function Value in Homogeneous Data Center Environment.

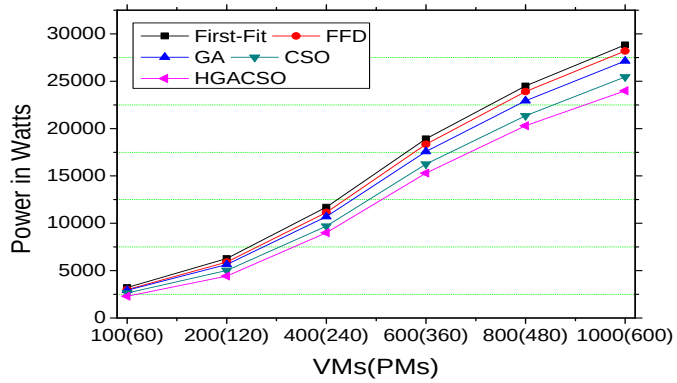

Figure 3.16: Power Consumption in Heterogeneous Data Center Environment.

PMs using First-Fit, FFD, GA, CSO, and HGACSO algorithms in heterogeneous data center environment. The multi-objective function value is low in HGACSO when compared to the other VM allocation techniques since HGACSO gives the global optimal solution by minimizing both the resource wastage and power consumption for different combinations of VMs(PMs). The difference between power consumption and multi-objective function value of HGACSO as compared to other VM allocation techniques is increasing when more number of VMs are allocated to the data center. Further, more number of PMs are switched-off in the case of HGACSO as compared to the First-Fit, FFD, GA, and CSO techniques.

Case-2

Table 3.4 shows the number of PMs used for VM allocation while taking different number of VMs and constant number of PMs at the cloud data center. Further, we check the performance of cloud data center in terms of CPU, RAM,

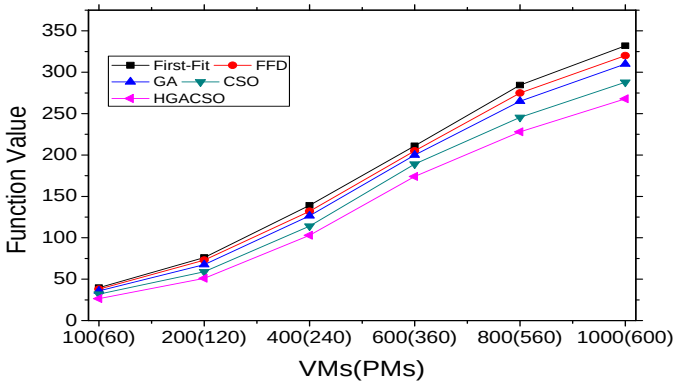

Figure 3.17: Function Value in Heterogeneous Data Center Environment.

and Storage utilization by taking different number of VMs and constant number of PMs in both (homogeneous, heterogeneous) cloud data center environments. Figures. 3.18, 3.19, and 3.20 show the % of CPU, RAM, and Storage utilization while keeping number of PMs constant at the heterogeneous cloud data center environment. The CPU, RAM, and Storage utilization of the data center is high in the case of HGACSO when compared to First-Fit, FFD, GA, and CSO techniques. Figures 3.21, 3.22, and 3.23 show the % of CPU, RAM, and Storage utilization while keeping number of PMs constant at the homogeneous cloud data center environment. The % of CPU, RAM, and Storage utilization is high in the case of proposed HGACSO when compared to that of other state-of-the-art VM allocation algorithms. Further, performance of CPU, RAM, and Storage utilization is varying while taking different number of VMs and keeping number of PMs constant in both homogeneous and heterogeneous cloud data center environments. Since, size of the chromosome is large in proportion to the number of VMs as requested by the user, hence there is a high probability that bio-inspired and hybrid bio-inspired algorithms such as GA, CSO, HGACSO select idle PMs in the chromosome for the allocation of VMs at the cloud data center.

Figures 3.24 and 3.25 show the power consumption while taking variable number of VMs and constant number of PMs at the cloud data center in homogeneous and heterogeneous data center environments respectively. The power consumption of the proposed HGACSO algorithm is low when compared to that of other VM

Table 3.4: Number of PMs Used for VM Allocation at the Constant Data Center

VMs(PMs)	Homogeneous					Heterogeneous				
	First Fit	**FFD**	**GA**	**CSO**	**HGACSO**	**First Fit**	**FFD**	**GA**	**CSO**	**HGACSO**
100(600)	48	46	45	42	38	(14, 18, 8)	(13,16,8)	(12, 15, 8)	(11, 14,7)	(10,13,6)
200(600)	96	92	90	85	77	(28, 36, 16)	(26,32,16)	(24,31,16)	(22,30,14)	(22,26,12)
400(600)	192	184	175	165	149	(56,72, 32)	(52,64,32)	(45,60,32	(41,56,28)	(41,48,24)
600(600)	288	276	265	247	223	(84,108,48)	(78,96,8)	(67,90,48)	(61,84,42)	(61,72,36)
800(600)	384	368	356	332	299	(112,144,64)	(104,128,64)	(90,121,66)	(82,113,57)	(84,97,49)
1000(600)	480	460	442	412	371	(140,180,80)	(130,160,80)	(112,150,80)	(101,140,70)	(101,120,60)

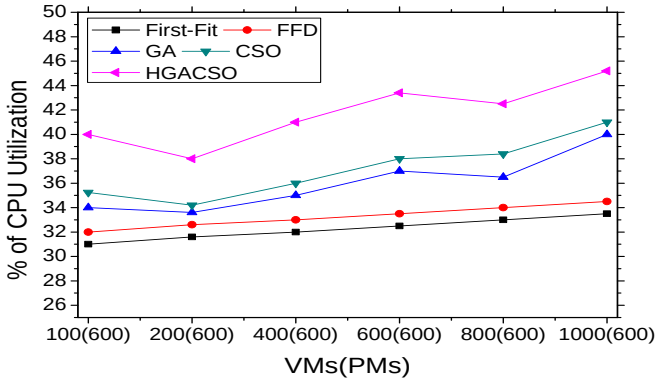

Figure 3.18: % of CPU Utilization in Constant Heterogeneous Data Center.

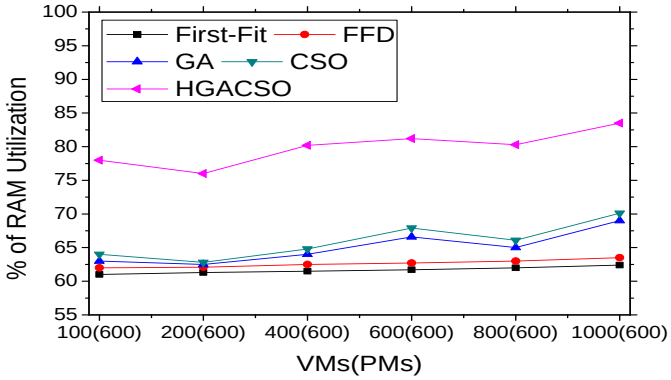

Figure 3.19: % of RAM Utilization in Constant Heterogeneous Data Center.

allocation algorithms since our proposed HGACSO algorithm is dealing with less number of switched-on PMs.

Further, to check the performance of the proposed energy efficient VM migration policy with HGACSO, we need to allocate the users requested VMs in time varying manner at the cloud data center. Hence, in our proposed work we considered total number of 50 users with their different VM requests for a pre-

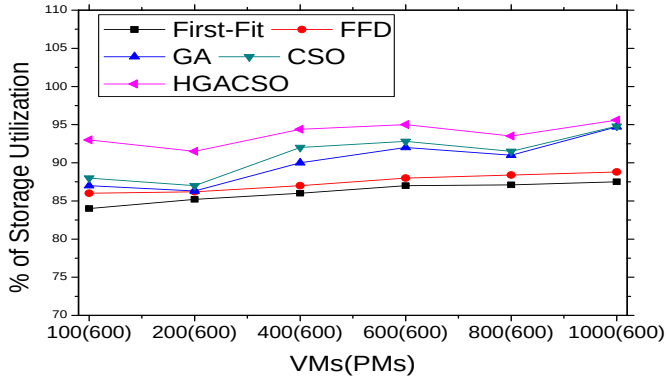

Figure 3.20: % of Storage Utilization in Constant Heterogeneous Data Center.

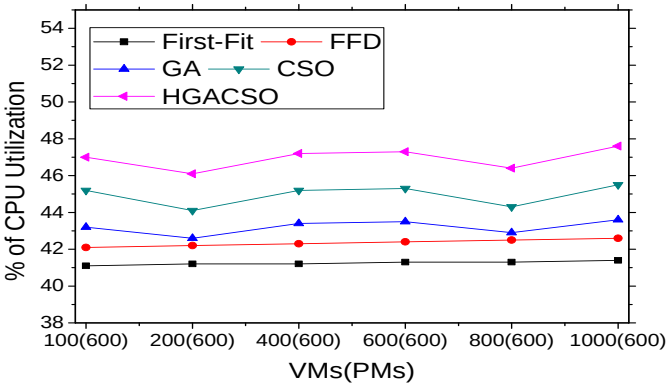

Figure 3.21: % of CPU Utilization in Constant Homogeneous Data Center.

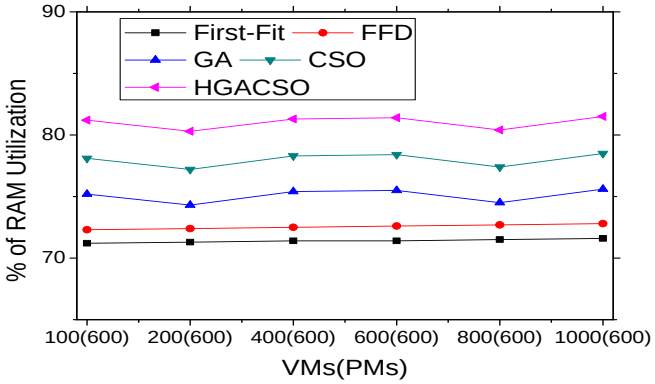

Figure 3.22: % of RAM Utilization in Constant Homogeneous Data Center.

defined period of time at the cloud data center. Figure 3.26 shows the number of VMs (small, medium, large) requested by the 50 users. Figure 3.27 shows the duration of VMs requested by different users during different time intervals. The total time duration of a user requested VMs allocated to the cloud data center is

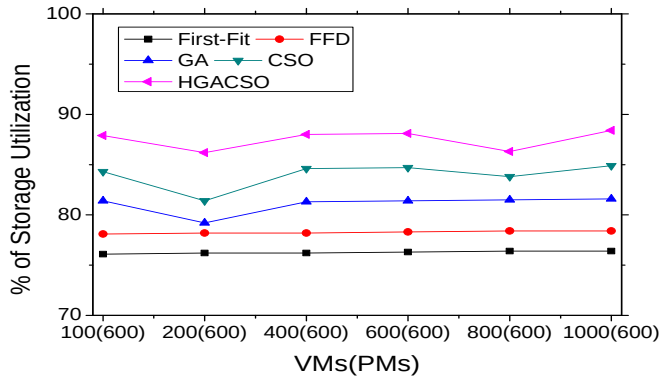

Figure 3.23: % of Storage Utilization in Constant Homogeneous Data Center.

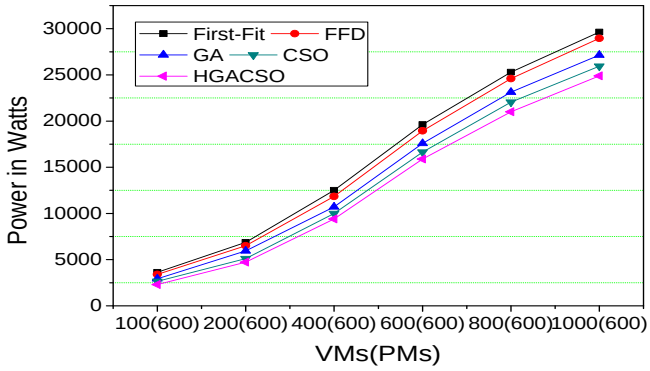

Figure 3.24: Power Consumption in Constant Homogeneous Data Center.

calculated by subtracting the allocation time from the destroy time, i.e. (destroy time-allocation time) and the details are shown in Figure 3.27.

Figure 3.28 shows the number of PMs used by the different techniques during different time instances for the allocation of users requested VMs. The proposed HGACSO takes less number of PMs for the allocation of VMs as compared to other VM allocation techniques. Since, HGACSO gives near optimal solution in terms of VM allocation to PM, hence HGACSO uses less number of PMs for the VM allocation. Initially in Figure 3.28, the graph is going upward due to continuous arrival of users requests at the cloud data center. After 50 time instances the graph is going down due to the destroying of those VMs which completed their specified time slots and also due to the switching-off idle PMs. Further, migration of VMs from the underutilized PM to the energy efficient PM using First-Fit technique results in more number of switched-off PMs.

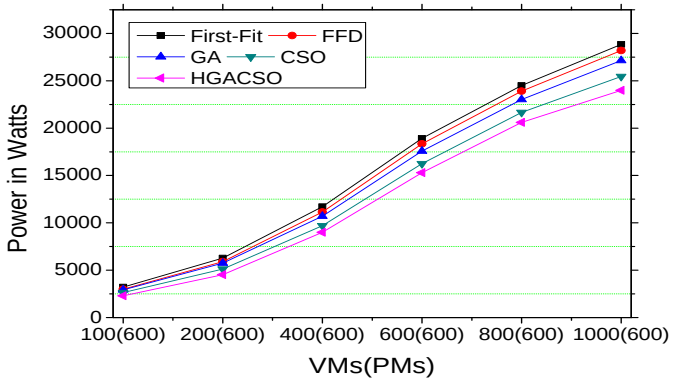

Figure 3.25: Power Consumption in Constant Heterogeneous Data Center.

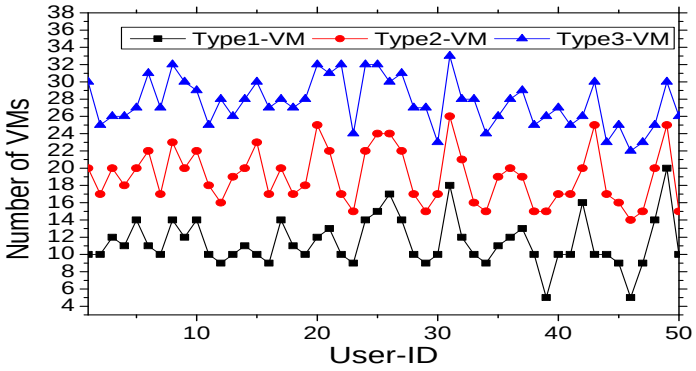

Figure 3.26: Number of VMs Requested.

The energy consumption and resources utilization of the data center during different period of time is shown in Figures 3.29 and 3.30, respectively. The energy consumption of the data center is low in the case of the proposed HGACSO approach when compared to the other VM allocation techniques due to less number of energy efficient PMs used by HGACSO, further migration of VM from the underutilized PM to the energy efficient PM and thus switching-off the underutilized PM will result in more energy saving at the cloud data center.

The energy consumption graph as shown in Figure 3.29, at the starting, is going upward due to the continuous increment of number of PMs used by the cloud data center for the allocation of user requested VMs. After 50 seconds this graph is going down because there are no further requests coming from the user side and those VMs completed their time slots will be destroyed from the data center and this will result in switching-off more number of idle PMs. Further migration

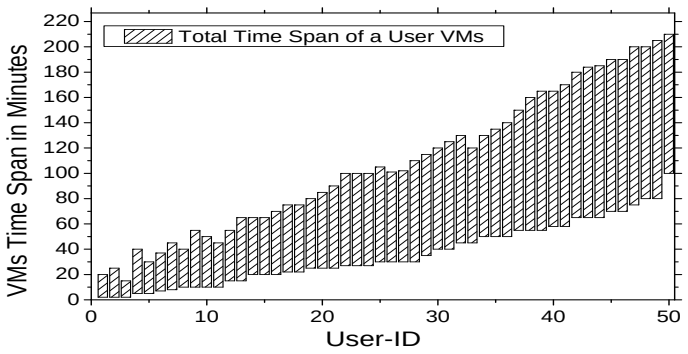

Figure 3.27: Time Duration of Requested VM.

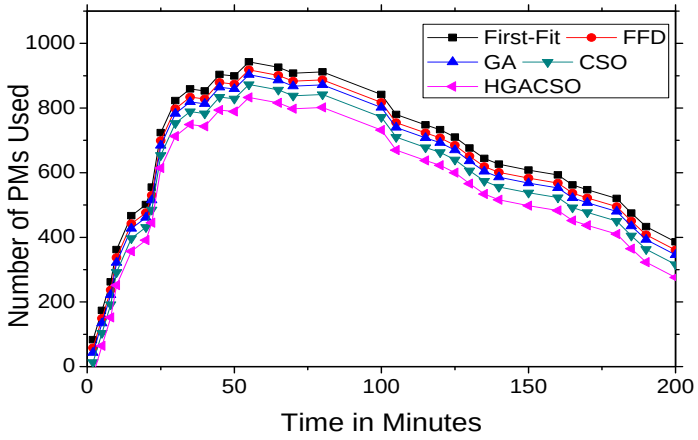

Figure 3.28: Number of PM Used to Allocate VM.

of VMs from underutilized PM to the energy efficient PM will result in further reduction of energy consumption of the data center.

Figure 3.30 shows the % of CPU utilization of the data center during different time instances. The CPU utilization of the proposed HGACSO algorithm is more when compared to the other VM allocation techniques, since HGACSO uses less number of PMs for the VM allocation. Hence, the resource wastage in the case of HGACSO is less when compared to other VM allocation techniques.

Figure 3.31 shows the total energy consumption of the data center during the specified time period. The energy consumption of the data center by the proposed HGACSO algorithm is (20%, 17%, 14%, and 8%) less when compared to First-Fit, FFD, GA, and CSO respectively. Figure 3.32 shows the average % of CPU utilization. The CPU utilization is (37%, 30%, 27%, and 15%) more in the case

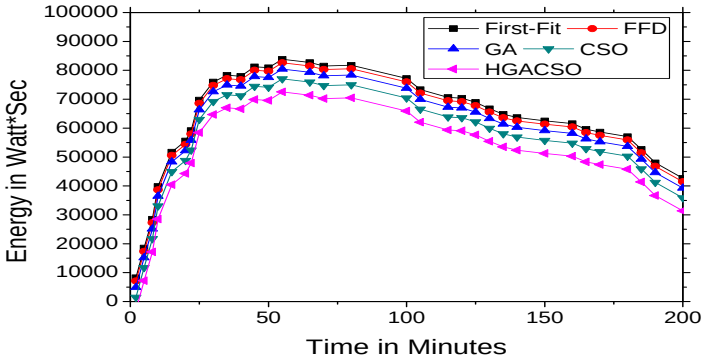

Figure 3.29: Energy Consumption at the Data Center.

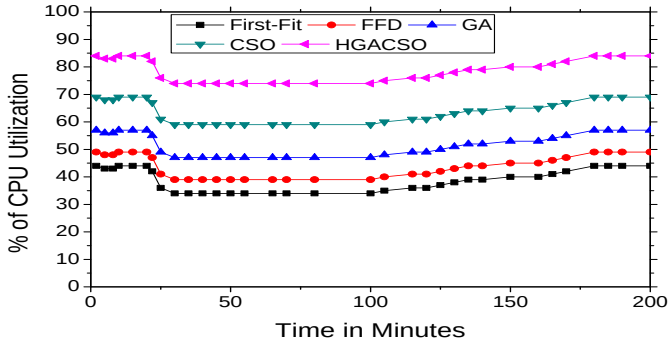

Figure 3.30: % of CPU at the Data Center.

of HGACSO as compared to the First-Fit, FFD, GA, and CSO respectively.

To check the scalability of the proposed algorithm, we conducted the experiment on HP Compaq LE1902x with 8 GB RAM, 3.40 GHz i-7 Processor and calculated the execution time of the proposed HGACSO algorithm. The execution time of the proposed HGACSO algorithm is mainly dependant on the crossover operation, mutation operation, and CSO. Hence, total execution time of the proposed HGACSO algorithm is shown in Figure 3.33. The total execution time for the placement of 1000 VMs at the data center is approximately 4.5 minutes which is scalable in nature for the large data center.

The minimization of power consumption on each iteration is considered for three evolutionary algorithms such as GA, CSO, and HGACSO and its convergence is shown in Figure 3.34. The convergence of HGACSO is fast as compared to that of GA and CSO, since we improved the fitness of the chromosomes by applying the CSO just before using the crossover and mutation operations of GA.

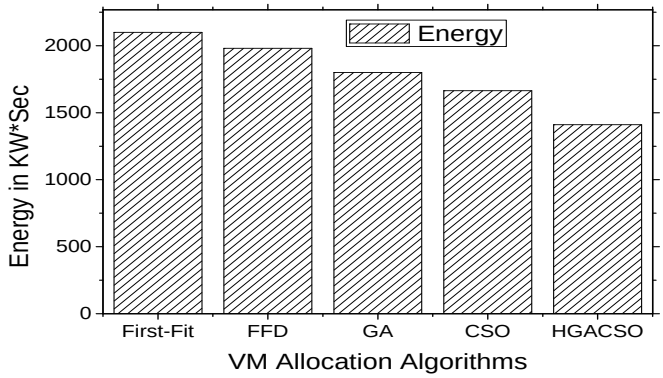

Figure 3.31: Total Energy Consumption at the Data Center.

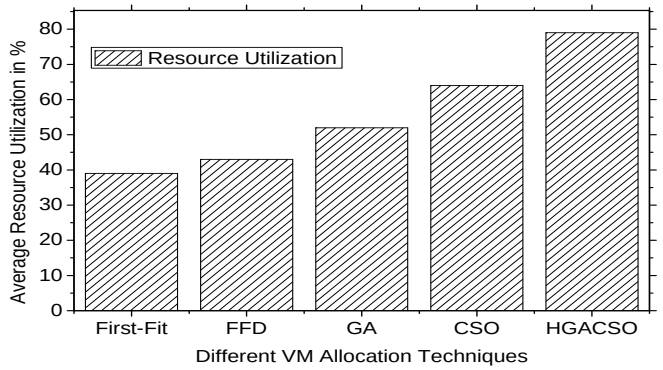

Figure 3.32: Average CPU Utilization at the Data Center.

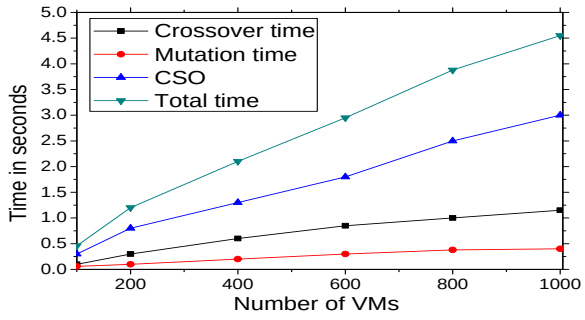

Figure 3.33: Execution Time of HGACSO in Minutes.

Further, this will result in less number of iterations required for the convergence of our proposed HGACSO algorithm.

The generation size (s), number of iterations (t), crossover rate (CR), mutation rate (MR) are the important parameters for the performance of proposed HGACSO algorithm. Figure 3.35 shows the performance of the HGACSO over dif-

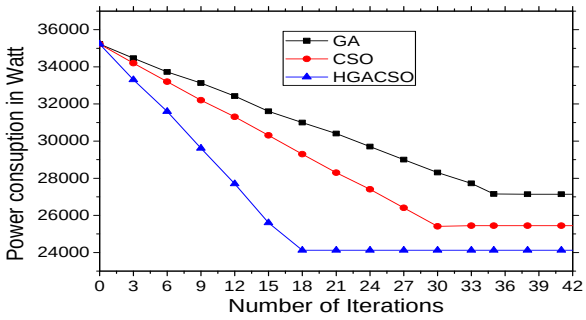

Figure 3.34: Minimization of Power Consumption on Each Iteration.

ferent crossover rates. When the crossover rate is low, then the objective function value is high; thus the HGACSO converges to non-global optimal point and takes more number of iterations. Further, by increasing the crossover rate, the objective function value is continuously going down, and an optimal crossover point of 0.6 is considered for the proposed HGACSO. Further, increasing the crossover rate from 0.6 increases the objective function value due to the early stagnation of the chromosomes resulting in sub-optimal value of objective function.

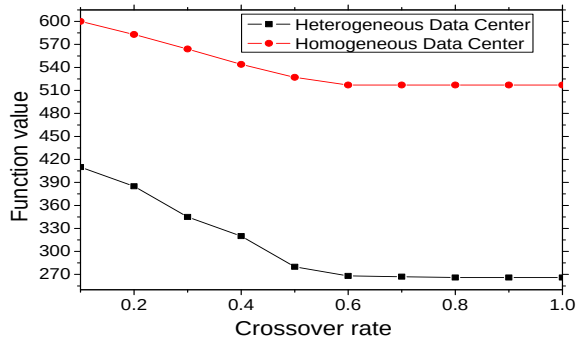

Figure 3.35: Performance of HGACSO Over Crossover Rate.

The performance of HGACSO on mutation rate is shown in Figure 3.36. The mutation rate of 0.03 gives the near global optimal allocation of VMs to the PMs by the proposed HGACSO. The value of mutation rate less than 0.03 gives very little divergence in the solution space resulting in sub-optimal solution. Mutation rate greater than 0.03 gives higher divergence in the solution space thus, HGACSO diverges from the final near global optimal solution. To check the performance of proposed HGACSO on solution size, we set different generation sizes (5 to 25).

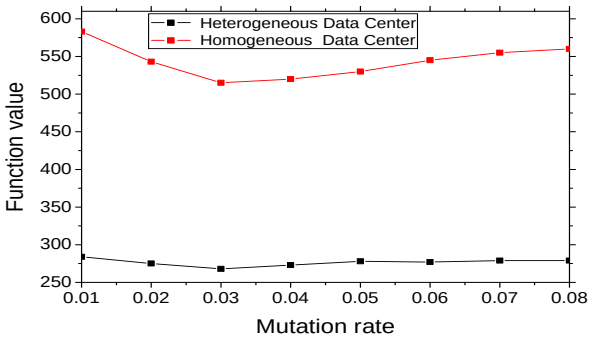

Figure 3.36: Performance of HGACSO Over Mutation Rate.

Table 3.5: HGACSO Parameters

Parameters	**GA**	**CSO**	**HGACSO**
Population size	10	10	10
Max. Iteration	50	50	50
Crossover rate	0.6	-	0.6
Mutation rate	0.03	-	0.03
Selection	Roulette Wheel	-	Roulette Wheel
(c1, c2,w)	-	(0.5, 0.5,0.5)	(0.5, 0.5,0.5)

The performance of HGACSO is high for generation size 10. The solution size less than 10 takes more number of iterations and gives sub-optimal solution due to less diversity in the solution space. The solution size more than 10 gives the same solution as that of 10 but the simulation running time is increased due to high of computation time of the proposed HGACSO.

The best values of the parameters for our proposed HGACSO algorithm are given in Table 3.5. The best values of crossover and mutation parameters are calculated for checking the performance of HGACSO on each crossover point between 0.1 and 1 by increasing the crossover rate from 0.01 to 0.08. In this process, we kept all other parameters such as population size, constants ($c1$, $c2$) and weight w constant. After getting best values of crossover and mutation operations, we increased the population size and fixed the minimum generation size for optimal allocation of VMs to PMs. The best constant values $c1$, $c2$ and weight value w are

calculated by taking initial values $c1$=0.1, $c2$=0.1, and w=0.1 and increase these values by 0.1 till it will not reach to 0.9. Let us consider the minimum values of $c1$, $c2$, and w so that there will not be any further performance improvement in the proposed HGACSO.

Time Complexity Analysis The time complexity of proposed HGACSO with VM migration technique is based on GA, and CSO. Let us consider maximum iteration size is $'k'$, individual size is $'m'$(number of PM), Number of VM is $'n'$, generation size is $'s'$, and crossover point for each generation is $'p'$. The time complexity of GA is dependent on the individual generation, crossover, and mutation operations. The time complexity of CSO is dependent on (Seeking mode, and Tracing mode). Further, the First Fit Decreasing (FFD) based VM migration requires O($nlogn$) time complexity. HGACSO=O(O(Individual generation)+ Total iterations*(O(Fitness calculation*Size of generation)+ O(Crossover*Size of generation)+O(Mutation*Size of generation))+ O(CSO)+O(VM migration). Hence, resultant time complexity of HGAPSO=O(O($n*s$)+O(k*(O($m*s$)+O($n*m*s-p*m*s$)+O(s))) + O(k*(O($s*m$)+O($s*m$)+O(s)+ O(s)+ O($n*m*s$))+O($nlogn$)) which is equal to O($n.*m*s*k$+O($nlogn$) polynomial in nature.

3.3 Summary

This chapter discussed the proposed HGACSO based VM allocation and energy efficient SLA aware VM migration policy at the cloud data center. Further, the experiments are conducted in both (homogeneous and heterogeneous) cloud data center environment. The experimental results demonstrated that proposed HGACSO with VM migration policy consumed (20%, 17%, 14%, and 8%) less energy over First-Fit, FFD, GA, and CSO respectively. Further, the resources utilization by proposed HGACSO was (37%, 30%, 27%, and 15%) more when compared to that of First-Fit, FFD, GA, and CSO respectively at the cloud data center. Hence, multi-objective based VM allocation technique is more efficient when compared to the existing state-of-the-art VM allocation. The proposed VM allocation and migration algorithm takes about 4.5 minutes to allocate 1000 number of VMs at the cloud data center. Hence, these techniques are useful for the

allocation and migration of VM at the commercial cloud data center.

But the key limitations of the proposed work in this Chapter 3 are: we did not consider the energy efficient task scheduling policy, and the proposed HGACSO algorithm takes more convergence time to give the solution of multi-objective VM allocation problem. Thus, in order to solve these issues we propose HGAPSO algorithm for multi-objective VM allocation to PM, and energy efficient SLA aware task scheduling policy at the cloud data centre. The detailed description of HGAPSO, and task scheduling policy are discussed in Chapter 4.

Chapter 4

Energy Efficient VM Allocation, Migration, and Task Scheduling Using HGAPSO

The allocation of different instances of VMs to PMs is known as a combinatorial optimization problem of the cloud computing. Further, it is also referred to as a multi-dimensional variable size bin-packing problem where items are the VMs and the bins are the PMs (Kim et al. 2014). The time complexity of the multi-dimensional VM allocation problem is NP-hard/NP-complete class (Gao et al. 2013). Hence, finding an optimal solution of the multi-dimensional VM allocation problem with multi-objective approach will thus create a lot of challenges for the researcher. Further, the complexity of the multi-objective, multi-dimensional VM allocation problem is generally defined in two different ways.

(i) When the similar types of VMs and PMs are considered in the data center (homogeneous environment), then the reduction of resources wastage can also reduce the energy consumption.

(ii) When different types of VMs and PMs are considered in the data center (heterogeneous environment), then the reduction of resources wastage may not guarantee similar reduction of energy consumption (Whitley et al. 1994).

Thus, we need a global optimal solution in the polynomial time and, thus we can achieve both the energy efficiency and minimization of resources wastage in homogeneous and heterogeneous data center environments. Further, we need an energy efficient SLA aware task scheduling policy for not only reducing the energy consumption, but also for avoiding the SLA violation at the cloud data center. After allocation of VMs to PMs, VM will start executing the tasks at the cloud data center. Further, we need an energy efficient SLA aware VM migration policy for migrating the VM from the underutilized PM to the energy efficient PM at

the cloud data center. Thus, resulting in switching-off underutilized/idle PM and saves the energy consumption with no SLA violation at the cloud data center.

In Chapter 3, we discussed multi-objective VM allocation to PM problem using HGACSO at the cloud data center. But, the limitation of HGACSO algorithm is, it takes more convergence to give the solution for the VM allocation problem. Further, in Chapter 3 we did not consider energy efficient SLA aware task scheduling policy at the cloud date center.

Thus, to resolve these issues, in this chapter, we describe the multi-objective VM allocation on PM, task scheduling, and VM migration policy at a cloud data center in terms of energy efficiency, minimization of resources wastage, and avoiding SLA violation. We propose an hybrid algorithm referred to as a HGAPSO based on Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Euclidean distance and thus achieving the optimal energy efficiency and resource utilization. Further, we design a First-Fit approximation based task scheduling and VM migration policy for reducing the energy consumption and thereby avoiding the SLA violation at the cloud data center.

The key research contributions towards energy efficient VM allocation, migration and energy efficient task scheduling techniques are described as follows:

1. To design, a multi-objective VM allocation policy for the cloud data center using proposed HGAPSO algorithm.

2. To design an energy efficient SLA aware task scheduling algorithm at the cloud data center using the First-Fit approximation algorithm.

3. To design an energy efficient SLA aware VM migration policy.

Figure 4.1 shows the flow chart of the VM allocation, task scheduling, and VM migration at the cloud data center. The proposed VM allocation, task scheduling, and VM migration techniques follow a systematic approach to resolve the energy consumption problem at the cloud data center. First, we allocate the users requested VMs to PMs at the cloud data center using the proposed HGAPSO. After allocation of VMs to PMs, we applied our proposed energy efficient work-

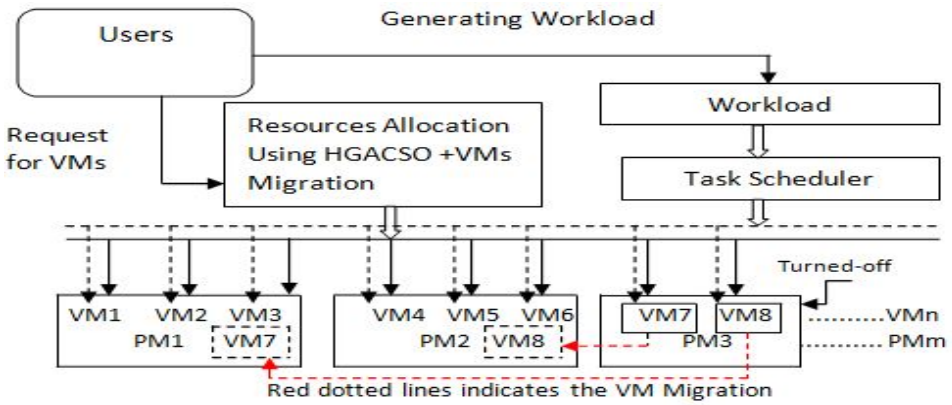

Figure 4.1: Flow Chart of VM Allocation, Task Scheduling, and VM Migration.

load policy to schedule the tasks to VM, and then after, we applied our proposed VM migration policy to migrate the VM from underutilized PM to energy efficient PM at the cloud data center. Further, each user's request consists of information about the number of VMs, type of VMs, and time duration of the VM at the cloud data center. We considered four different types of VMs in our proposed work. Hence, the user's requested query for time duration (t) is defined in terms of R_i=$\{VM_{small}(n1), VM_{medium}(n2), VM_{large}(n3), VM_{x.large}(n4),\}$, *where* $VM_{small}(n1)$, $VM_{medium}(n2)$, $VM_{large}(n3)$, $VM_{x.large}(n4)$ *are the numbers of small, medium, large, and extra large types of VMs requested by a user at the cloud data center respectively.*

4.1 VM Allocation Using Proposed HGAPSO Algorithm

Before applying the HGAPSO for the allocation of VMs to PMs at the cloud data center, we need to define a power consumption model of a PM and then we need to design a mathematical model for multi-objective multi-constraints VM allocation problem at the cloud data center.

Hence, in this chapter, we used the same power consumption model of a PM, and mathematical model for multi-objective VM allocation problem as discussed in Section 3.1. Thus, the power consumption (P_j) and the energy consumption (E_j)(during time duration of t_1 to t_2) of (pm_j) are described by Eqs. 4.1 and 4.2 respectively.

$$P_j = ([P_j^{max} - P_j^{min}])u + P_j^{idle} \tag{4.1}$$

$$E_j = \int_{t=t_1}^{t=t_2} P_j dt \tag{4.2}$$

Where, P_j^{max} and P_j^{min} are the maximum and minimum power consumption of pm_j respectively; u is the rate of CPU utilization between 0 and 1; E_j is the total energy consumption of pm_j during time duration of t_1 to t_2.

If a data center has m number of PMs, then the total energy consumption of a data center (DC^{energy}) is defined by Eq. 4.3. The resource (mips, ram, storage) utilization of pm_j (u_j^d) is defined by Eq. 4.4. The average resource utilization of the data center (DC^u) over time duration of t_1 to t_2 is defined by Eq. 4.5.

$$DC^{energy} = \sum_{j=1}^{m} E_j \tag{4.3}$$

$$u_j^d = \frac{\sum_{i=1}^{n} a_{ij} * VM_i^d}{pm_j^d} \quad \forall d \in \{mips, ram, storage\} \tag{4.4}$$

$$a_{ij} = \begin{cases} 1 & if \quad VM_i \quad allocated \quad to \quad pm_j \\ 0 & else \end{cases}$$

$$DC^u = \int_{t=t_1}^{t=t_2} \frac{\sum_{j=1}^{m} u_j^{mips} + \sum_{j=1}^{ram} u_j^{ram} + \sum_{j=1}^{m} u_j^{storage}}{|d| \sum_{j=1}^{m} b_j} dt \tag{4.5}$$

$\sum_{j=1}^{m} b_j > 0 \quad \& \quad |d| = 3$

Where, $'b_j'$ is a binary variable indicating whether pm_j is used for VM allocation or not. The value of $b_j = 1$ if pm_j is used for the VM allocation otherwise it is 0; $|d|$=3 is defined for the number of resources such as mips, ram, storage.

Further, Euclidean distance based objective function for multi-objective VM allocation problem is defined as:

$$min \quad f = \sum_{j=1}^{m} \sqrt{(\frac{P_j}{P_j^{max}})^2 + (u_j^d - 1)^2} \tag{4.6}$$

$$VM_i^{pe} * a_{ij} \leq pm_j^{pe} \quad \forall j \in \{1, 2, 3, .., m\} \tag{4.7}$$

$$\sum_{i=1}^{n} VM_i^{mips} * a_{ij} \leq pm_j^{mips} \quad \forall j \in \{1, 2, 3, .., m\} \tag{4.8}$$

$$\sum_{i=1}^{n} VM_i^{ram} * a_{ij} \leq pm_j^{ram} \quad \forall j \in \{1, 2, 3, .., m\} \tag{4.9}$$

$$\sum_{i=1}^{n} VM_i^{storage} * a_{ij} \leq pm_j^{storage} \quad \forall j \in \{1, 2, 3, .., m\} \tag{4.10}$$

The multi-objective minimization function is described by Eq. 4.6. The constraints satisfaction of different resources such as processing elements, MIPS, RAM, and STORAGE, are defined by Eqs. 4.7 to 4.10 respectively.

A. Basic Concepts of GA, and PSO

The proposed HGAPSO algorithm is based on the hybrid combination of GA and PSO. Hence, we need to understand the basic terminology and operations of these algorithms. Further, we already discussed the basics of GA in Section 3.1. Hence, the detailed description of PSO is described as follows.

The PSO starts searching a solution for the problem by using a population of particles. Further, the initial population of the particles are generated randomly by random allocation of VMs to PMs at the cloud data center. Each particle in the population of particles gives the feasible solution of the VM allocation problem with reference to objective function. Each particle in the population consists of a position vector such as ($\vec{x_i}$). Further, a swarm (group of particles) moves in the solution space by applying the velocity to the particle. This particle velocity is represented by a velocity vector ($\vec{v_i}$). Further, at each time step $'t'$, using the local best particle position $\vec{x}_{lbesti}$ and the global best particle position $\vec{x}_{gbest}$; both velocity vector and position of the particle for time step $(t+1)$ are calculated by Eqs. 4.11 and 4.12 respectively.

$$\vec{v}_i(t+1) = w\vec{v}_i(t) + c_1 r_1(\vec{x}_{lbesti}(t) - \vec{x}_{gbest}(t)) + c_2 r_2(\vec{x}_{gbest}(t) - \vec{x}_i(t)) \tag{4.11}$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1). \tag{4.12}$$

Where, w is a learning coefficient between 0 and 1; c_1 and c_2 are the constants between 0 and 1; r_1 and r_2 are the uniformly distributed random numbers between 0 and 1.

D. Description of HGAPSO

The flow chart of proposed HGAPSO is shown in Figure 4.2. Further, the motivation to use the hybrid combination of GA and PSO is described as follows.

Both GA, and PSO techniques work on a random population in order to

search the solution for discrete optimization problem. In the proposed HGAPSO, the crossover operation of GA changes the placement of VM from one chromosome to another chromosome by migrating the VM from one chromosome to another chromosome. However, the performance of GA is poor, and it takes more convergence time when the fitness values of the parent chromosomes are low and the searching space is large. Further, the other reason for giving the poor quality of solution is due to the random migration of VMs from one PM to another PM in the chromosome during crossover and mutation operations. The other reason for generating the poor quality solution in GA is due to the random migration of VM in the chromosome by selecting the random crossover point during crossover operation. But due to the random migration of VMs from one PM to another PM, the probability of searching for a optimal solution in GA is high.

On the other hand in PSO, particles move their position from one place to another place in the search space using the migration of VMs from one PM to another PM by following the global and local best positions of the particle. Hence, this will result in migration of VMs from non-efficient PM to energy efficient PM. Thus, PSO has fast convergence, but due to the dependency on the local best particle position, sometimes the final solution falls in local optima. Therefore, before applying the crossover and mutation operations of GA, we need to apply PSO for improving the fitness value of the parent chromosomes, and thus obtaining the global optimal allocation of VMs to the PMs in less time. Hence, the proposed HGAPSO consists of GA, and PSO operations and the details are as follows.

First of all we need to generate the finite number of chromosomes in the initial population. Further, each chromosome 's' represented by (Chr.s) consists of a fitness value (Fit.s) as shown in Figure 4.2. Further, in proposed HGAPSO, the total s number of chromosomes (Chr.1, Chr.2,....Chr.s) are generated in the initial iteration and these chromosomes have fitness values (Fit.1, Fit.2,....Fit.s). Hence, after the generation of chromosomes in the initial population, we need to sort the chromosomes in the decreasing order on the basis of their fitness value. Discard half of the chromosomes (less fittest) from the population and remaining

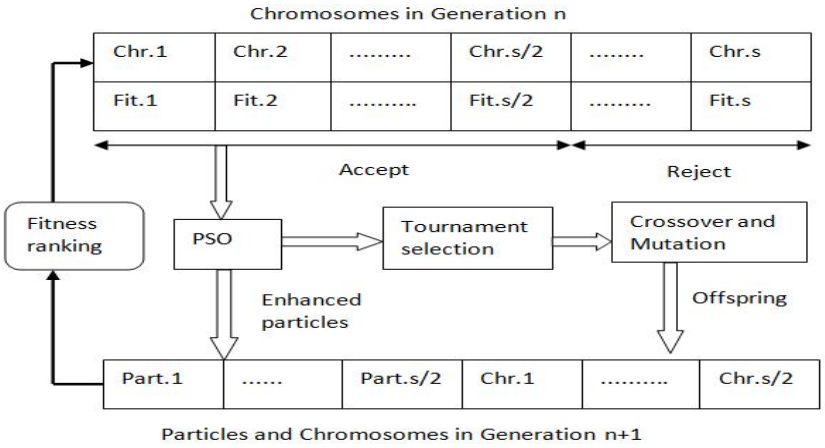

Figure 4.2: Flow Chart of the Proposed HGAPSO Algorithm.

half chromosomes (more fittest) will be considered for further operations. The best selected chromosomes are treated by the initial particles using PSO. Further, in the proposed HGAPSO, the quality of the best selected chromosomes can be improved by using the PSO and then apply the GA operations (crossover and mutation operations respectively) to the said fine quality chromosomes. At the end of GA operations, we need to select the fittest particles or chromosomes for the the next iteration. The HGAPSO algorithm terminates the iteration and thus allocates VMs to PMs when one of the termination conditions is satisfied.

Hence, the detailed description of generation of chromosomes using GA, enhancement of chromosomes using PSO, crossover and mutation operations using GA are defined as follows:

Further, in this chapter, we used the same approaches for the generation of chromosomes, and selection of fittest chromosomes as discussed in Section 3.1. Hence, the fitness value of the chromosome k, and their probability to go for the next computation are defined by Eq. 4.13, and Eq. 4.14 respectively. The fitness value of the chromosome is in between 0 and 1.

$$fit_k = \frac{1}{\sum_{j=1}^{m} \sqrt{(\frac{pm_j}{pm_j^{max}})^2 + (u_j^d - 1)^2}} \tag{4.13}$$

$$p_k = \frac{fit_k}{\sum_{k=1}^{s} fit_k}. \tag{4.14}$$

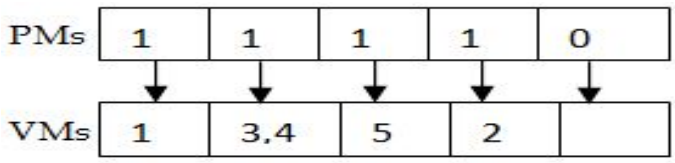

Figure 4.3: Binary Representation of a Particle.

To select the fittest chromosomes from the population, we need to calculate the fitness value of chromosomes. Let s be the size of a generation then half of the fittest chromosomes, such as $s/2$ number of chromosomes will be considered for further computation and thereby destroying remaining half of the chromosomes in the population. The selected fittest chromosomes are considered for PSO, and GA operations (crossover, mutation).

Particle Encoding and Position: After selecting the fittest chromosomes from the initial population, we need to apply the PSO to improve the fitness quality of the chromosomes. Hence, mapping of a chromosome to a particle is described as follows: The current position of a particle is defined in terms of m-bit vector, where m is the number of PMs at the data center. Thus, position of i^{th} particle is defined as $X_i^t = \{x_1^t, x_2^t, ..., x_m^t\}$, where each bit of the position vector is either 0 or 1. Hence, we assign binary digit 1 to those PM which contain VM, otherwise, the value of PM is 0. Thus, m bit binary vector describes the position of a particle. The binary representation of a particle is shown in Figure 4.3.

Particle Velocity: The velocity of the i^{th} particle at iteration t is defined by m bit velocity vector $V_i^t = \{v_1^t, v_2^t, ..., v_m^t\}$. By applying the velocity to the current position of the particle, we changed the position of the particle. Hence, by changing the position of the particle, we changed the position of the VM from one PM to another PM. Each bit of the velocity vector is either 0 or 1. We, applied the random velocity in the first iteration to change the position of the particle.

Subtraction Operator: The subtraction operator calculates the difference between two position vectors of VM. The subtraction operator is defined by Θ in the proposed HGAPSO. After applying the subtraction operator on two position vectors such as $(X_i^t \ \Theta \ X_i^t)$, we got binary 1 if both position vectors have same

bit value, otherwise the outcome value is 0. For example (Ex: (1,1,1,0,1) Θ (1,0,1,0,0)=(1, 0, 1, 1,0)).

Addition Operator: The addition operator calculates the changes in velocity of the particle. In the proposed HGAPSO algorithm, the addition operator is defined by symbol $\oplus$. By applying the addition operator, we calculated the changes in the velocity of the particle such that $P_1V_1^t \oplus P_2V_2^t \oplus, ...P_nV_n^t$ represents the velocity change using V_1^t with probability P_1, and using V_n^t with probability P_n. In the proposed HGAPSO algorithm, each probability $P_i, (\sum_{i=1}^{n} P_i = 1)$ is defined by an inertia coefficient. The detailed description of the addition operator is defined by the following example: Let us consider, 0.3(1,1,0,1,0)$\oplus 0.7$(1,1,1,0,0)=(1,1, $\neq$, $\neq$, 0), where the probability of occurrence of 1 at 1^{st} and 2^{nd} bit positions is 1 and the probability of occurrence of 0 at the 5^{th} bit position is 1. Thus, bit values at 1^{st}, 2^{nd}, and 5^{th} positions are 1, 1, and 0 respectively, and the remaining positions are uncertain (denoted by $\neq$). Further, to calculate the uncertain bit value, we need to define three inertia weight coefficients such as P_1^i, P_2^i, P_3^i, and these inertia weight coefficients are defined as follows:

$$f(X_i^t) = \sum_{j=1}^{m} \sqrt{(\frac{P_j}{P_j^{max}})^2 + (u_j^d - 1)^2} \tag{4.15}$$

$$P_{1i} = \frac{f(X_i^t)}{f(X_i^t) + f(X_{lbest,i}^t) + f(X_{gbest}^t)} \tag{4.16}$$

$$P_{2i} = \frac{f(X_{lbest,i}^t)}{f(X_i^t) + f(X_{lbest,i}^t) + f(X_{gbest}^t)} \tag{4.17}$$

$$P_{3i} = \frac{f(X_{gbesti}^t)}{f(X_i^t) + f(X_{lbest,i}^t) + f(X_{gbest}^t)} \tag{4.18}$$

Where, $f(X_i^t)$ represents the fitness value of i^{th} particle for the solution X_i^t; X_{lbesi}^t, and X_{gbesi}^t represent the local best position and global best position of i^{th} particle respectively. To calculate the inertia weight coefficients, we need to consider the high energy efficiency and the maximum utilization with high probability.

$$\begin{cases} uncertain \quad bit = q_1, \quad if \quad rand \leq P_{1i} \\ uncertain \quad bit = q_2, \quad if \quad P_{1i} < rand \leq P_{2i} \\ uncertain \quad bit = q_3, \quad if \quad P_{2i} < rand \leq p_{3i} \end{cases}$$

Where, q_1 is the corresponding bit of the particle velocity vector before updating; q_2 is the corresponding bit value of the local best particle vector; and q_3 is the corresponding bit of the best global particle velocity vector.

Multiplication Operator: To update the position of the particle, we need to apply a multiplication operator in the proposed HGAPSO. The multiplication operator is represented by $\otimes$ in the proposed HGAPSO. The multiplication operator is applied between particle position X_i^t and updated velocity, and thereby calculating the new position of the particle such that $(X_i^t \otimes V_i^{t+1})$ gives X_i^{t+1}. The proposed computation rule for calculating the new position of the particle is described as follows:
(i) If the corresponding bit value of the velocity vector is 1, then the corresponding bit value of the new position vector is not adjusted; (ii) If the corresponding bit value of the velocity vector is 0, then it will be adjusted. For example, (1,1,1,0,1)$\otimes$(1,0,1,0,1), where (1,0,1,1,1) is the position vector and (1,0,1,0,1) is the velocity vector. The 2^{nd} and 4^{th} bits of the velocity vector are 0s and hence VMs are allocated to PMs when the 2^{nd} and 4^{th} positions of VMs are updated.

Finally, the PSO is used for computing the velocity and updated position as per the following details.

$$V_i^{t+1} = P_1 V_i^t \oplus P_2(X_{lbesti}^t \Theta X_i^t) \oplus P_3(X_{gbesti}^t \Theta X_i^t) \tag{4.19}$$

$$X_i^{t+1} = X_i^t \otimes V_i^{t+1} \tag{4.20}$$

Crossover Operation: After improving the fitness quality of the chromosomes using PSO, we need to apply GA operations (crossover, and mutation). Further, before applying the crossover and mutation operations, we need to store one copy of the particles temporarily, and another copy will go for the crossover operation as shown in Figure 4.2. The PSO generated particles are considered as chromosomes for applying the crossover, and mutation operations. In the proposed HGAPSO algorithm, we used single point crossover operation on the chromosomes since, multi-point crossover operation not only migrates more number of VMs from one chromosome to another chromosome, but also degrades the fitness

value of the chromosomes and further generates more number of infeasible VMs in the chromosomes. Therefore, reallocating these infeasible VMs, we need an extra computational time and hence, we did not consider multi-point crossover operation. Hence, we applied the single point crossover operation in the same manner as discussed in Section 3.1.

Mutation Operation: After completion of the crossover operation, we performed a mutation operation on the chromosomes, and the detailed description of mutation operation is already discussed in Section 3.1.

Reallocation of Infeasible VMs: In the case of PSO, when addition and subtraction operators are applied there is a chance of some particles reaching infeasible condition. The same infeasible condition arises when applying crossover and mutation operations on these chromosomes. Thus, to remove and refill the infeasible VMs, we used FFD approximation policy as discussed in Section 3.1.

After completion of GA and PSO, we have one set of particles and another set of chromosomes. Using these sets of particles and chromosomes, we need to select the best particles or chromosomes using the roulette wheel selection operation as defined by Eq. 4.14 for the next generation.

Termination Conditions: The proposed HGAPSO will be terminated when one of the following conditions is satisfied. (i) When the number of iterations is greater than the maximum set value of iterations. (ii) When there is no improvement in successive iterations or the repetition of chromosomes and particles value.

Algorithm 4.1 describes the allocation of VMs to PMs using the proposed HGAPSO. Where t is the iteration, t^{max} is the maximum number of iterations, and s is the generation size initialized by an integer number. Steps 1 to 11 generate the s-1 number of random chromosomes. Step 12 describes the generation of one chromosome using FFD technique. Step 13 describes the termination condition of the proposed HGAPSO algorithm when specified termination condition such as ($t = t^{max}$ or repetition of the same type of chromosomes and particles) is satisfied. Step 14 selects the $s/2$ number of fittest chromosomes. Steps 15 to 16

Algorithm 4.1 Proposed HGAPSO

Initialize: t←1, t^{max}, s

Input: VMs list (VM[]), PM list (PM[])

Output: Mapping of VM to PM

1: **for** i ← 1 to (s-1) **do**
2: **while** $VM[] \neq empty$ **do**
3: Randomly select VM_i from VM[]
4: Randomly select pm_j from PM[]
5: **for** Each d **do**
6: **if** $VM_i^d \leq pm_j^d$ **then**
7: Allocate VM_i to pm_j
8: Update $pm_j^d = pm_j^d - VM_i^d$
9: Remove VM_i form VM[].
10: **else**
11: go to 4
12: Call FirstFit(Chr_s)
13: **while** Termination condition==false **do**
14: Select s/2 fittest Chromosomes.
15: **for** $i \leftarrow 1$ to s/2 **do**
16: $Part_i \leftarrow Chr_i$
17: Call ***Procedure 4.1***.
18: Call Crossover($Parent_1, Parent_2$)
19: Call ***Procedure 3.2***
20: **for** $i \leftarrow 1$ to $s/2$ **do**
21: Call Mutation(Chr_i)
22: Call ***Procedure 3.2***
23: $t \leftarrow t + 1$
24: Allocate VM to PM

Procedure 4.1: Applying PSO

1: **if** $t == 1$ **then**
2: $X^t_{gbest} \leftarrow null$
3: **for** i$\leftarrow$ 1 to s/2 **do**
4: $X^t_i \leftarrow RandomPosition()$;
5: $V^t_i \leftarrow RandomVelocity()$;
6: $X^t_{lbesti} \leftarrow X^t_i$
7: **if** $f(X^t_{xlbesti} > f(X^t_{gbest})$ **then**
8: $X^t_{gbest} \leftarrow X^t_{lbesti}$
9: **for** i$\leftarrow$ 1 to s/2 **do**
10: Update V^{t+1}_i using Eq.4.19
11: Update X^{t+1}_i using Eq. 4.20
12: **if** $f(X^{t+1}_i > f(X^t_{lbesti}))$ **then**
13: $X^{t+1}_{lbest} \leftarrow X^{t+1}_i$
14: **if** $f(X^{t+1}_{lbesti} > f(X^t_{gbest})$ **then**
15: $X^{t+1}_{gbest} \leftarrow X^{t+1}_{lbesti}$
16: Call ***Procedure 3.2***

describe the mapping of Chromosomes to Particles. Step 17 calls the Procedure 4.1 for performing the PSO operation. Step 18 calls the Crossover operation, and Step 19 calls the Procedure 3.2 (as already described in Section 3.1) for removing and refilling the infeasible VMs. Steps 20 to 22 describe the mutation operation and also applying the Procedure 3.2 on each chromosome. Step 23 increments the iteration value.

For enhancing the fitness value of the chromosomes (by applying the PSO) is described by Procedure 4.1. Random generated particle position and velocity in the initial iteration are described by Steps 1 to 5. Current particle position to local particle position are described by Steps 6 to 8. Steps 9 to 11 describe the updating the velocity of particle, and its current position. Steps 12 to 15 describe the computation of local best and global best particle positions. Removing and refilling of infeasible VMs by calling Procedure 3.2 is described by Step 16. Further, the detailed description of Procedure 3.2 is described in Section 3.1.

4.1.1 Energy Efficient SLA Aware Task Scheduling

After allocation of user requested VMs to the PMs at the cloud data center, now VMs are ready for processing the user's generated applications. Hence, we need to define an energy efficient SLA aware task scheduling policy at the cloud data center. The SLA violation is measured in terms of response time, availability of VM, and security level of the VM at the cloud data center. Hence, to avoid the SLA violation, we need to reduce the response time, maximize the availability of the purchased VMs, and minimize the security threats at the user end. Thus, if we schedule the maximum number tasks to the less number of VMs then by this way, we can switch-off idle VMs at the data center, this will result in reducing the energy consumption at the data center. But the response time of this type of task scheduling approach will be high at the cloud users, and this will result in SLA violation at the cloud data center. On the other hand scheduling the small number of tasks to the maximum number of VMs will result in more number of PMs in switched-on condition, and thus leading to higher energy consumption but there is no SLA violation at the cloud data center.

Further, response time in terms of SLA violation also depends on the execution time of the task scheduling algorithm. Hence, we design an energy efficient SLA aware task scheduling policy based on the First-Fit approximation with our defined constraints. The reason behind using the First- Fit approximation is due to its fast execution time and resulting in higher response time for an application. Further, to reduce the SLA violation, we defined the (task, VM) constraint for scheduling the task to the VM at the cloud data center. The details of the proposed energy efficient SLA aware task scheduling policy are described as follows.

In our proposed work we considered user application as a stream of tasks for the execution. Hence, we can divide an application into number of tasks of different lengths. Further, execution of tasks requires any kind of resources, such as CPU, RAM, Storage, or I/O devices. Hence, if a data center has k number of users requested VMs allocated to the PM then k^{th} user generated application A_k is represented as a stream of tasks like $A_k = \{t_0^k, t_1^k, t_2^k, .t_l^k\}$. Where t_l^k is the l^{th} task of an application A_k. Like VM and PM, a task can also be defined as a vector unit.

The dimension of the task vector is based on the type and amount of resources required to execute, such as CPU, RAM, Storage, etc. Further, the user requires the security levels such as low, medium, or high, and these levels of security is specified at the time of VMs requested by the user. Hence, in our proposed work, we considered the requested amount of security of a VM in between 0 and 1, where 0 represents the lowest security and 1 represents the highest security demand of a VM. To provide the requested level of security to a VM, we assigned a trust level to each PM in between 0 and 1, where 0 shows the least secured PM and 1 shows the highest secured PM at the cloud data center. For the security and billing reason, the k^{th} user generated application, A_k tasks are applied to their own purchased VM. Further, in the HGAPSO, we consider all the tasks of CPU intensive. Therefore, before task scheduling, task $t_i^k(mis)$ must satisfy the

following constraints, and then this task will go to the respective VM (VM, task).

$$\begin{cases} If \quad t_i^k(mis) \leq VM_{small}^{mips} \quad go \quad to \quad VM_{small} \\ ElseIf \quad VM_{small}^{mips} < t_i^k(mis) \leq VM_{medium}^{mips} goto \quad VM_{medium} \\ ElseIf \quad VM_{medium}^{mips} < t_i^k(mis) \leq VM_{large}^{mips} \quad goto \quad VM_{large} \\ Else \quad VM_{x.large}^{mips} < t_i^k(mis) \leq VM_{x.large}^{mips} \quad goto \quad VM_{x.large} \end{cases}$$

If more than one task comes at the same time instance t which falls in the same conditional range such as $VM_{large} < t_i^k(mis), t_{i+1}^k(mis) \leq VM_{x.large}^{mips}$, then one task will wait in the queue of $VM_{x.large}$ till previously assigned task is not completed its execution. Hence, in our proposed work, we consider the sequential task dependency in an application A_k such as task t_{i+1}^k, which will be executed after the execution of task t_i^k.

The Service Level Agreement (SLA) is a contractual document between the customer and the cloud service provider. The SLA violation in the cloud computing depends on the type and amount of infrastructure purchased by the customer and the quality of service provided by the service provider. If there is a quality mismatch from the contractual document, then the service provider will pay the amount of money to the customer based on their contractual document. If the application running time is longer, then in that case, in place of defined deadline for each task, the average response time or total execution time of the application does matter. Hence, user defines the application deadline in terms of execution time, and this execution time depends on the amount of resources in terms of VMs purchased by the user. Hence, on the basis of purchased VMs configuration, the customer, thus defines the deadline time of each application. Then the formulation of SLA violation is defined as follows:

Let us consider a customer purchased different type of VMs in different quantity such as $VM_{small}(n1)$, $VM_{medium}(n2)$, $VM_{large}(n3)$, $VM_{x.large}(n4)$ and application A_k is running on these purchased VMs then deadline D_k, for the application A_k is defined as follows:

$$D_k = \frac{\sum_{k=1}^{l} t_k^{mis}}{\sum_{i=1}^{n1+n2+n3+n4} VM_i^{mips}} \tag{4.21}$$

If waiting time ($w_i^k(t)$), and execution time ($ex_i^k(t)$) of a task t_i^k for application A_k, are considered then the response time $r_i^k(t) = w_i^k(t) + ex_i^k(t)$. Therefore the response time of an application A_k ($R_k(t)$) having $'l'$ number of tasks is defined as

$$R_k(t) = \sum_{i=1}^{l} w_i^k(t) + r_i^k(t) \tag{4.22}$$

Further, let us consider b_i a binary variable which represents the availability of the VM_i. If VM_i is available at the data center, then value of b_i=1 otherwise b_i=0. Hence, SLA violation condition of an application A_k is defined as

$$\begin{cases} No \quad SLA \quad violation \quad if \quad (R_k(t) \leq D_k(t)) \quad and \\ \sum_{i=1}^{n1+n2+n3+n4} b_i \geq [vm_{small}^t(n1) + vm_{medium}^t(n2) vm_{large}^t(n3) + vm_{x-large}^t(n3)] \\ and \quad VM_i^{security} \geq PM_j^{trust} \\ SLA \quad violation \quad otherwise. \end{cases}$$

The amount of SLA violation of a given application A_k is dependent on the time difference between the response time ($R_k(t)$) and the deadline time ($D_k(t)$), such as ($R_k - D_k$). Since the commercial cloud service provider, such as Amazon provides EC_2 VMs instances on the rent basis by creating the different types of user requested VMs to the data center during a specified period of time. Hence, each type of VMs such as small, medium, large, and x.large, have different amount of rent such as \$0.0065 per Hour for small VM instance (Amazon-Website 2014). Hence, we consider α, β, γ, and δ are the rents of small, medium, large, and x.large types of VMs for unit time duration respectively. Further, if SLA violation occurs for a user then the cloud service provider will pay the penalty to the user, and this penalty is defined in the case of different type of VM instances given by

$$\begin{cases} VM_{small}(SLA) = (R_k(t) - D_k(t)) * \alpha \\ VM_{medium}(SLA) = (R_k(t) - D_i(t)) * \beta \\ VM_{large}(SLA) = (R_k(t) - D_k(t)) * \gamma \\ VM_{x.large}(SLA) = (R_k(t) - D_k(t)) * \delta \end{cases}$$

Where, $VM_{small}(SLA)$, $VM_{medium}(SLA)$, $VM_{large}(SLA)$, $VM_{x.large}(SLA)$ are the penalty amounts to be paid by the service provider to the customer in the case of small, medium, large, and x.large VMs instances respectively.

4.1.2 Energy Efficient SLA Aware VM Migration

After scheduling of tasks to the VM at the cloud data center, we need to check the lifetime of VM requested by the users on regular time intervals. Thus, we need to terminate the VMs which have completed their life time or executed all the tasks which are associated with these VM. Hence, after termination of time expired VMs from the PMs at the cloud data center, there is a chance that some PMs will be in underutilized/idle condition. Thus, we need to migrate the VMs from underutilized PMs to the energy efficient PMs at the cloud data center, and thus we can switch-off idle PMs at the cloud data center. Hence, the termination of time expired VMs from the PMs at the cloud data center gives the opportunity to the cloud data center manager to re-optimize the placement position of VMs to PMs by migrating the VMs from underutilized PM to the energy efficient PM at the cloud data center.

Hence, using the VM migration techniques, we can consolidate the VMs into less number of energy efficient PMs and thereby shutting down the idle PMs at the cloud data center. Further, migration of VMs from one PM to another PM will create two issues such as taking an extra period of time (SLA violation during VM migration time) and requiring additional power consumption. Hence, our proposed VM migration policy not only reduces the enegry consumption, but also avoids the SLA violation at the cloud data center.

The objective function for optimum VM migration is to migrate the VMs from the source node to the destination node so that filling the maximum number of VMs at the destination node within their capacity, and thus switching-off source node after VM migration. Hence, by this way we can maximize the number of idle PMs to be turned off at the data center.

Let s be the set of non idle PMs such that $|s| < m$. The objective function for

the optimal VM migration and consolidation is defined by

$$max \quad y = \sum_{i \in s} P_i^{min} y_i - \sum_{i \in s} \sum_{j \in s} \sum_{k=1}^{n_i} P'_K z_{ijk}. \tag{4.23}$$

Where, y_i=1 if pm_i is used for the VM allocation, otherwise $y_i = 0$; Variable z_{ijk}=1 if VM_k is migrating from pm_i to pm_j otherwise it is 0; P'_K is the power consumption of VM_k; n_i is the set of VMs to be migrated from pm_i. The objective function described by Eq. 4.23 should satisfy the following migration constraints during the VM migration.

$$\sum_{i \in s} \sum_{k=1}^{n_i} VM_k^d z_{ijk} \leq pm_j^d (1 - y_j) \forall d \in \{mips, ram, storage\} \tag{4.24}$$

$$\sum_{j \in s} \sum_{k=1}^{n_i} z_{ijk} = n_i y_i \quad \forall i \in s, i \neq j \tag{4.25}$$

The constraints satisfaction for different resources is defined by Eq. 4.24. Migration of a set of VMs (n_i) from pm_i to pm_j is defined by Eq. 4.25. The proposed migration policy consists of two stages for calculating when and where to migrate VM at the data center. Further, to avoid SLA violation and also to minimize the energy consumption at the data center, we set the lower threshold value of CPU utilization (u_l) for each PM at the data center. In the first stage, we decide when to migrate the VM (selection of source node) and in the second stage, we decide where to migrate the VM (selection of destination node). The migration of VMs from pm_i will take place if the current CPU utilization of pm_i is less than the lower threshold value u_l such as ($u_j^{cpu} < u_l$), and the cost of migration for all the VMs allocated to pm_i should be less than the rent of VMs as decided by the cloud service provider during the left over time t_l. Hence, the selection of a set of VMs n_i from the source node pm_i is defined by

$$n_i = \begin{cases} migration \quad of \quad n_i \quad if \quad u_i^{cpu} < u_l \quad and \\ c_{mig}^t * n_i < n_i r * t_l, \quad r \in \{\alpha, \beta, \gamma, \delta\}, \quad i \in s \\ No \quad migration \quad otherwise. \end{cases} \tag{4.26}$$

The left over time (t_l) is the difference between total life time (t_t) of VM_i and current time (t_c) of VM_i such as $t_l = t_t - t_c$. The total migration cost (c_{mig}^t) for the

set of VMs n_i is dependant on two factors such as SLA penalty during shutdown time s_t and migration overhead cost c^o_{mig}. Therefore, the total migration cost is defined by

$$c^t_{mig} = \sum_{k=1}^{n_i} s_t * r + \sum_{k=1}^{n_i} c^o_{mig} \tag{4.27}$$

The amount of SLA penalty during the VM shutdown time is equal to the VM rent paid by the user to the service provider during that period of time. Further, the migration overhead cost c^o_{mig} is dependent on the network bandwidth, size of memory content to be copied from the source node to the destination node etc. Hence, the fixed power consumption P'_k is assigned to each VM instance on the basis of its type such as small, medium, large, and x.large as a migration overhead cost. In the second stage, we arrange all the VMs collected by the source nodes in the decreasing order on the basis of their resource requirement and then allocate the VMs to the PMs using First-Fit technique.

4.2 Experimental Setup, Results and Analysis

To check the performance of proposed HGAPSO, VM migration policy, and task scheduling policy, we consider different types of VMs and PMs at the cloud data center. Further, we conducted experiment in both homogeneous and heterogeneous cloud data center environments. The details of the experimental setup and results & analysis are discussed as follows.

Experimental Setup: To evaluate the performance of the proposed HGAPSO algorithm, we used CloudSim (Calheiros et al. 2011) simulator with certain modifications. The main reason for using a CloudSim simulator is due to the proposed multi-objective VM allocation to the PM is a NP-hard problem, hence for checking the performance of the proposed algorithm, we required a large number of VMs(PMs) combinations. Therefore, conducting the experiment on a simulator in place of real cloud is more feasible and beneficial.

The other reason for using the CloudSim simulator as compared to other existing simulators is that CloudSim provides heterogeneous environment in the form of different configured PMs and VMs. Thus, we can conduct the experiment using

Table 4.1: Configuration of PMs

PM Type	PE	MIPS	RAM(GB)	STORAGE(GB)
ProLiantM110G5XEON3075	2	2660	4	160
IBMX3250Xeonx3480	4	3067	8	250
IBM3550Xeonx5675	12	3067	16	500

Table 4.2: Configuration of VMs

VM Type	PE	MIPS	RAM (GB)	STORAGE (GB)
Small	1	500	0.5	40
Medium	2	1000	1	60
Large	3	1500	2	80
X.large	4	2000	3	100

large amount of VMs and PMs with different combinations of VMs(PMs). We can write our own VMs allocation policy and further, we can dynamically apply the task to the VM.

In CloudSim simulator, we consider three different types of PMs and four different types of VMs for the creation of heterogeneous data center environment in the form of VMs and PMs. The PM power consumption and other resources characterises of PMs are downloaded from the IBM (IBM-Switch-Model 2014) and Dell (Wu et al. 2014) websites. Further, the Amazon based VMs instances used for the experiment is downloaded from Amazon website (Amazon-Website 2014). The configurations of PMs and VMs used for the simulation are shown in Tables 4.1 and 4.2 respectively. Further, for all VMs(PMs) combinations, we consider the same number of VMs and PMs (all types). For example, 100 VM and 60 PM combination of data center consists of 25 small, 25 medium, and 25 x.large types of VM; 20 Proliant, 20 IBM3250, and 20 IBM3550 types of PM. The experimental results are carried out in two different cases. In Case-1, we considered different combinations of PMs and VMs (PMs are fixed at 60% of VMs) in two different data center environments (homogeneous and heterogeneous). Further, in Case-2,

Table 4.3: Number of PMs used for VM Allocation at Data Center

VMs(PMs)	Homogeneous					Heterogeneous				
	First Fit	**FFD**	**GA**	**PSO**	**HGAPSO**	**First Fit**	**FFD**	**GA**	**PSO**	**HGAPSO**
100(60)	48	46	44	42	38	(14, 18, 8)	(13,16,8)	(11, 15, 8)	(10, 15,7)	(10,13,6)
200(120)	96	92	88	83	75	(28, 36, 16)	(26,32,16)	(22,30,16)	(20,29,14)	(20,25,12)
400(240)	192	184	176	166	150	(56,72, 32)	(52,64,32)	(44,60,32)	(40,58,28)	(40,50,24)
600(360)	288	276	264	248	224	(84,108,48)	(78,96,8)	(66,90,48)	(60,86,42)	(60,74,36)
800(480)	384	368	352	330	298	(112,144,64)	(104,128,64)	(88,120,64)	(80,114,56)	(80,98,48)
1000(600)	480	460	440	412	372	(140,180,80)	(130,160,80)	(110,150,80)	(100,142,70)	(100,122,60)

we check the performance of our proposed HGAPSO algorithm by keeping the number of PMs constant and varying number of VMs in both cloud data center environments (homogeneous and heterogeneous).

Results and Analysis: For evaluating the performance of the HGAPSO algorithm as compared to the other state-of-the-art VM allocation algorithm such as First-Fit, FFD, GA, and PSO, in terms of energy consumption and resources utilization, we considered different combinations of VMs and PMs in two different data center environments (Homogeneous and Heterogeneous).

Case-1

The number and types of PMs used for different amount of VM allocation in heterogeneous and homogeneous data center environment is shown in Table 4.3 (Case-1). The HGAPSO algorithm utilizes less number of PMs as compared to the First-Fit, FFD, GA, and PSO. Since, a hybrid combination of GA, and PSO has improved the fitness of the chromosomes by applying PSO before applying the crossover and mutation operations of GA. Thus, the crossover and mutation operations produced good quality children, and thereby achieving the near global optimal solution for VM allocation. Thus, resulting in less number of PMs used for the VM allocation in the case of HGAPSO.

Figure 4.4, Figure 4.5 and Figure 4.6 show the CPU utilization, RAM utilization, and Storage utilization respectively, of the heterogeneous data center for different VMs(PMs) combinations using First-Fit, FFD, GA, PSO, and HGAPSO algorithms. The CPU utilization, RAM utilization, and Storage utilization of the proposed HGAPSO are slightly higher as compared to the First-Fit, FFD, GA,

and PSO. Since, the HGAPSO used near optimal combination of (Type 1, Type 2, Type 3) PMs for VM allocation and hence HGAPSO results in very less CPU wastage as compared to the First-Fit, FFD, GA, and PSO. The CPU utilization in HGAPSO is more as compared to First-Fit, FFD, GA, and PSO due to the less number of PMs used for VM allocation in the case of HGAPSO. The CPU, RAM, and Storage utilization of the proposed HGAPSO is slightly improving when more number of requested VMs are allocated to the PMs. The number of used PMs for the VM allocation in HGAPSO is continually decreasing as compared to the other VM allocation techniques.

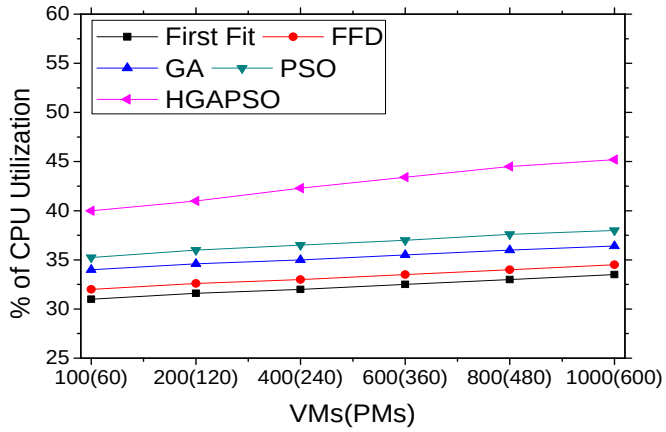

Figure 4.4: % of CPU Utilization in Heterogeneous Data Center.

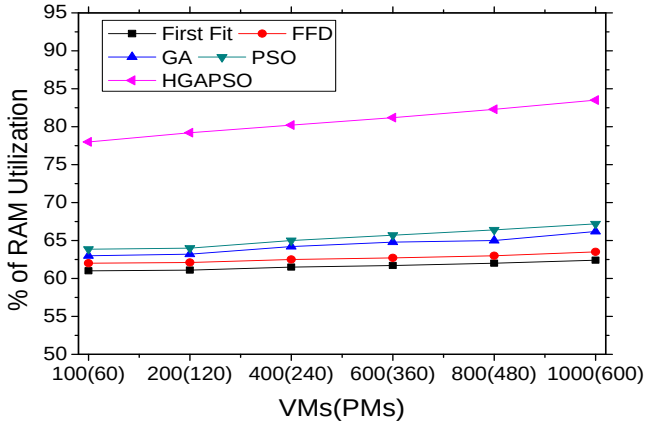

Figure 4.5: % of RAM Utilization in Heterogeneous Data Center.

To check the performance of HGAPSO based VM allocation algorithm in homogeneous environment, we consider only single type of PM (Type 2 PM) in the

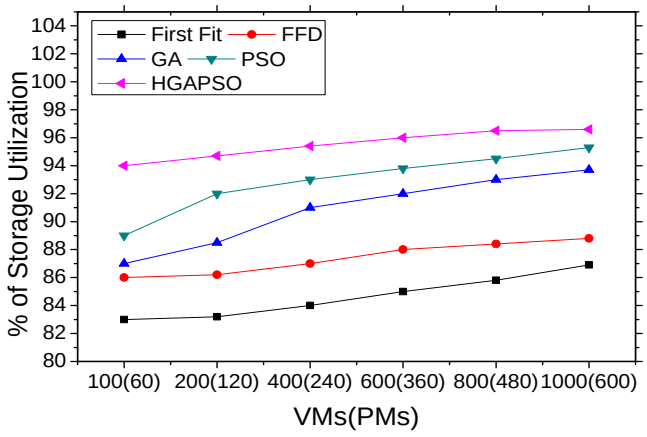

Figure 4.6: % of Storage Utilization in Heterogeneous Data Center.

data center. The detailed configuration of Type 2 PM is given in Table 4.1. The number of PMs used by the different VM allocation techniques by taking different combinations of VMs(PMs) is shown in Table 4.3. The HGAPSO used less number of PMs for the VM allocation due to the global optimal placement of VMs to PMs. The simulation results for the homogeneous environment show that HGAPSO algorithm performs better than other VM allocation techniques in terms of energy consumption and resources utilization.

Figure 4.7, Figure 4.8, and Figure 4.9 show the CPU, RAM, and Storage utilization of the homogeneous data center for different VM allocation techniques using different combinations of VMs(PMs) respectively. The CPU, RAM, and Storage utilization of the data center is maximum in the case of HGAPSO when compared to other VM allocation techniques. This is due to the less number of energy efficient PMs used for the VM allocation since more number of PMs are switched-off and, thus minimizes the resources wastage. The resources utilization graph is almost straight line for all VM allocation techniques since used PMs are increasing in the same proportion as number of VMs are increasing and thus resulting in less variation in resources utilization.

Figure 4.10 shows the power consumption for different VM allocation techniques using different VM(PM) combinations in homogeneous data center environment. The power consumption of the data center in the case of HGAPSO is

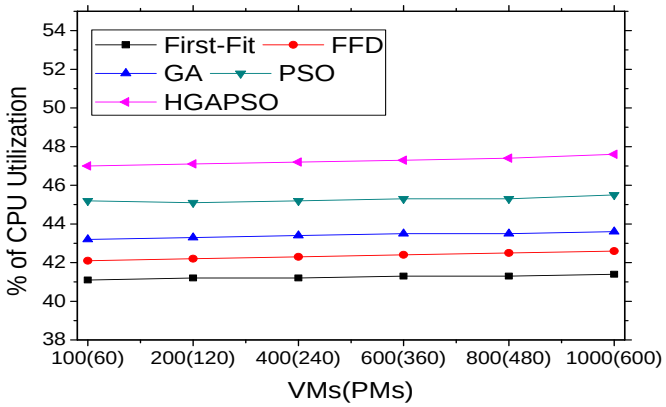

Figure 4.7: % of CPU Utilization in Homogeneous Data Center.

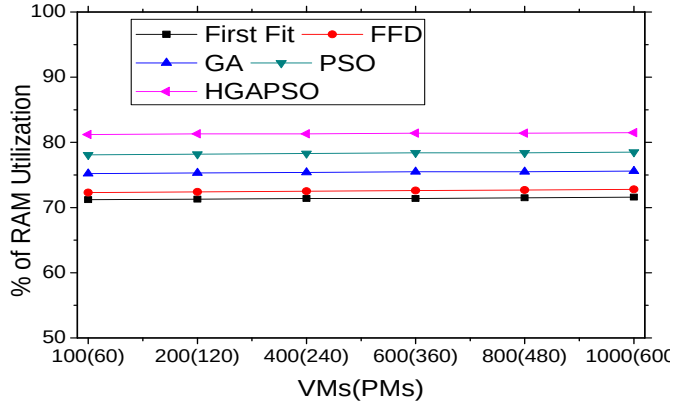

Figure 4.8: % of RAM Utilization in Homogeneous Data Center.

low when compared to the First-Fit, FFD, GA, and PSO. HGAPSO has used less number of PM for VM allocation since idle PM (which are not used for VM allocation) are switched-off and hence HGAPSO consumes less power consumption.

Figure 4.11 shows the power consumption and resources utilization based on multi-objective function for different combinations of VMs and PMs using HGAPSO, GA, PSO, FFD, and First-Fit algorithms in the heterogeneous data center environment. The multi-objective function value is low in HGAPSO when compared to the other VM allocation techniques since HGAPSO gives the global optimal solution by minimizing both the resource wastage and power consumption for different combinations of VMs(PMs). The difference between power consumption and function value of HGAPSO as compared to other VM allocation techniques is increasing when more number of VMs are allocated to the data cen-

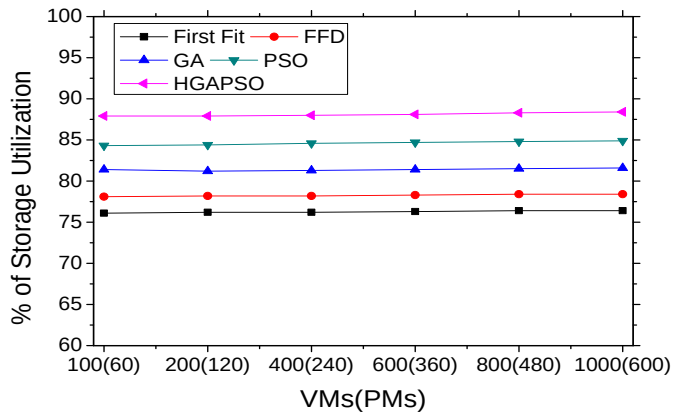

Figure 4.9: % of Storage Utilization in Homogeneous Data Center.

ter. Further, more number of PMs are switched-off in the case of HGAPSO as compared to First-Fit, FFD, GA, and PSO.

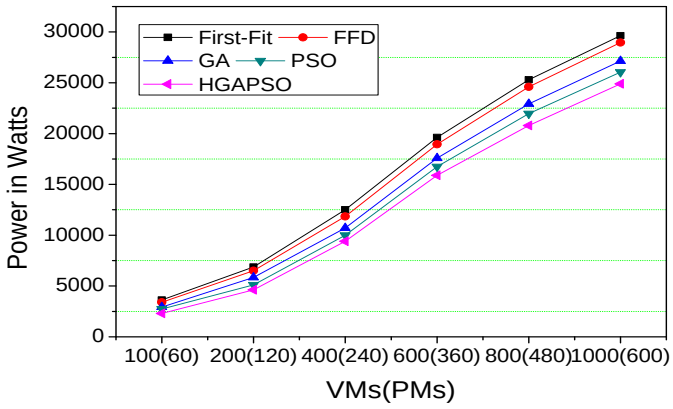

Figure 4.10: Power Consumption in Homogeneous Data Center Environment.

Figure 4.12 and Figure 4.13 show the objective function value based on resources utilization and power consumption for different VM allocation techniques over different combinations of VMs(PMs) in homogeneous and heterogeneous data center environment respectively. The power consumption and resources wastage are less in the case of our proposed HGAPSO algorithm when compared to other VM allocation techniques. This is due to the global optimal placement of VMs to PMs that results in lower overall function value in the case of HGAPSO.

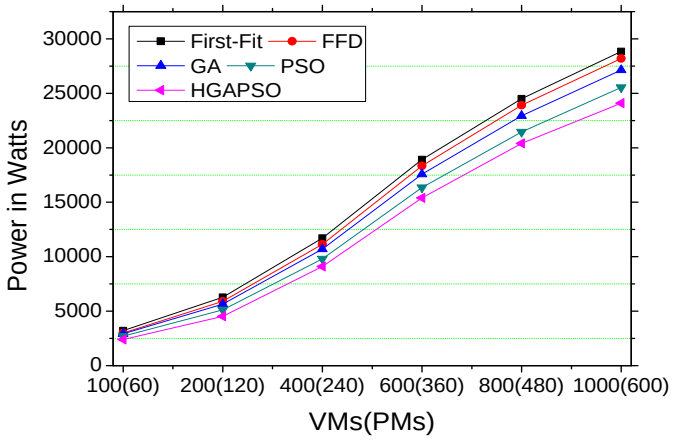

Figure 4.11: Power Consumption in Heterogeneous Data Center Environment.

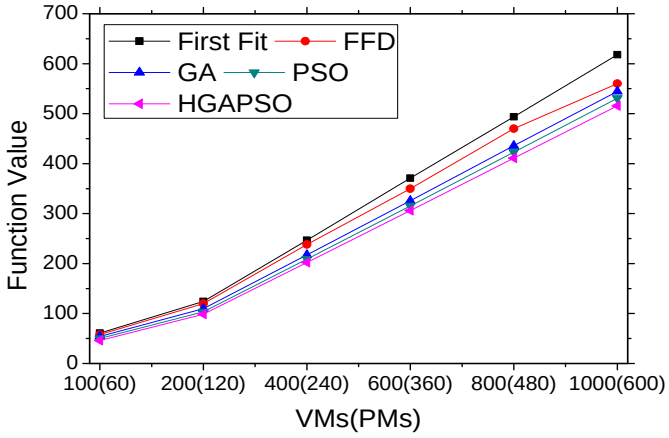

Figure 4.12: Objective Function Value in Homogeneous Data Center.

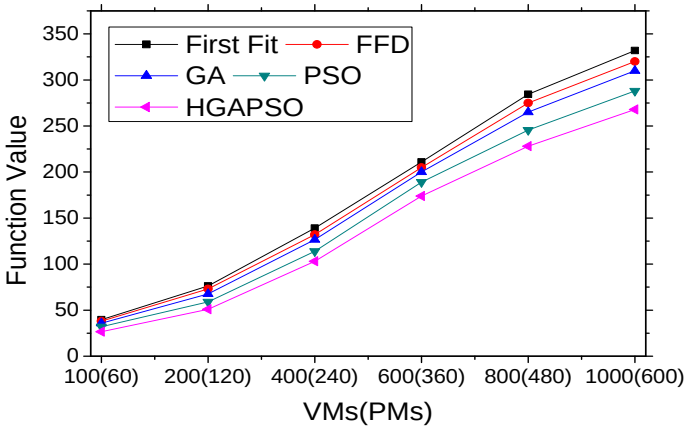

Figure 4.13: Objective Function Value in Heterogeneous Data Center.

Table 4.4: Number of PMs used for VM Allocation at Constant Data Center

VMs(PMs)	Homogeneous					Heterogeneous				
	First Fit	**FFD**	**GA**	**PSO**	**HGAPSO**	**First Fit**	**FFD**	**GA**	**PSO**	**HGAPSO**
100(600)	48	46	45	43	40	(14, 18, 8)	(13,16,8)	(12, 15, 8)	(11, 15,7)	(10,13,6)
200(600)	96	92	90	86	78	(28, 36, 16)	(26,32,16)	(24,31,16)	(22,31,14)	(22,26,12)
400(600)	192	184	175	167	150	(56,72, 32)	(52,64,32)	(45,60,32	(41,58,28)	(41,48,24)
600(600)	288	276	265	249	224	(84,108,48)	(78,96,8)	(67,90,48)	(62,84,42)	(61,72,36)
800(600)	384	368	356	334	300	(112,144,64)	(104,128,64)	(90,121,66)	(83,114,57)	(84,97,49)
1000(600)	480	460	442	414	373	(140,180,80)	(130,160,80)	(112,150,80)	(103,141,70)	(101,120,60)

Case-2

The number of PMs used for the allocation of different number of VMs while keeping constant number of PMs at the cloud data center is described in Table 4.4. Further, we keep the number of PMs constant and varying number of VMs to check the performance of cloud data center in terms of CPU, RAM, and Storage utilization in both (homogeneous, heterogeneous) cloud data center environments. Figures 4.14, 4.15, and 4.16 show the % of CPU, RAM, and Storage utilization while keeping number of PMs constant at the heterogeneous cloud data center environment. The CPU, RAM, and Storage utilization of the data center is high in the case of HGAPSO when compared to First-First, FFD, GA, and PSO based VM allocation techniques.

Figures 4.17, 4.18, and 4.19 show the % of CPU, RAM, and Storage utilization while keeping the number of PMs constant at the homogeneous cloud data center environment. The % of CPU, RAM, and Storage utilization is high in the case of proposed HGAPSO when compared to that of other state-of-the-art VM allocation algorithms. Further, performance of CPU, RAM, and Storage utilization is varying while taking different number of VMs and keeping number of PMs constant in both homogeneous and heterogeneous cloud data center environments. Since, size of the chromosome is large in proportion of number of VMs requested by the user, so there is a high probability that HGAPSO selects idle PM in the chromosome for the allocation of VMs.

Figures 4.20 and 4.21 show the power consumption while varying the number of VMs, and constant number of PMs in homogeneous and heterogeneous cloud

data center environments respectively. The power consumption of the proposed HGAPSO algorithm is low when compared to that of other VM allocation algorithm since HGAPSO is dealing with less number of switched-on PMs.

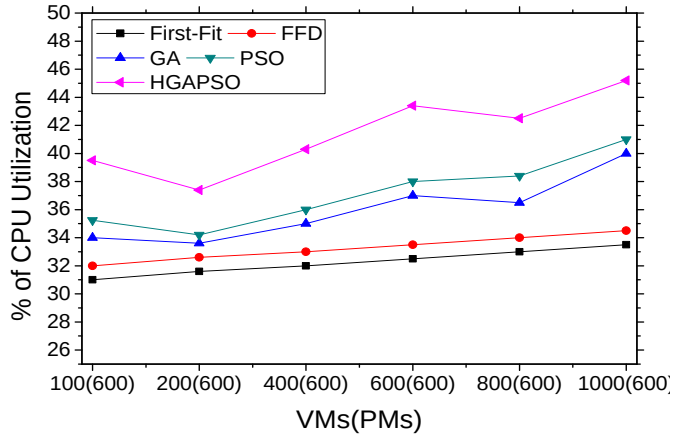

Figure 4.14: % of CPU Utilization in Constant Heterogeneous Data Center.

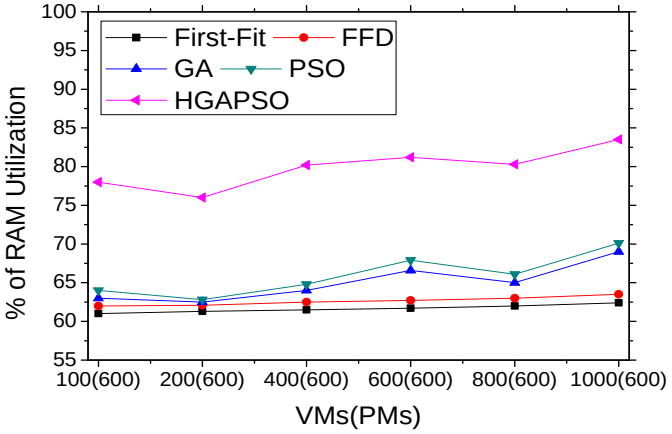

Figure 4.15: % of RAM Utilization in Constant Heterogeneous Data Center.

After allocation of VMs to the PMs, we applied the random tasks on a regular time interval (1 second) for the calculation of energy consumed by the data center during specified duration of time. Hence, in our simulation, we generated four different tasks on every 1 second during the period of 200 seconds. The task scheduling on VM is done by using the VM(task) conditions specified in phase 2 (task scheduling). The random allocated tasks to Type1 VM, Type2 VM, Type3

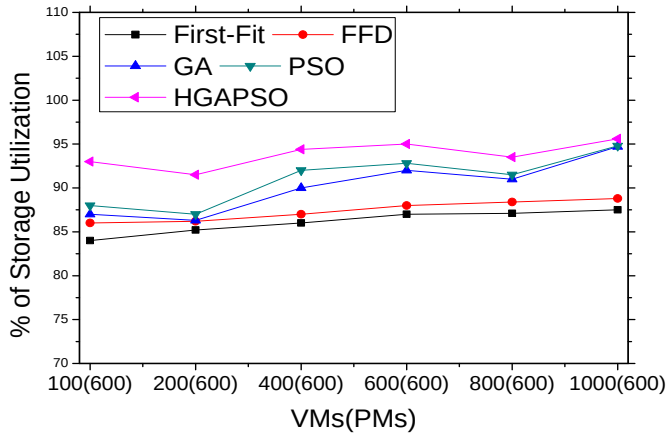

Figure 4.16: % of Storage Utilization in Constant Heterogeneous Data Center.

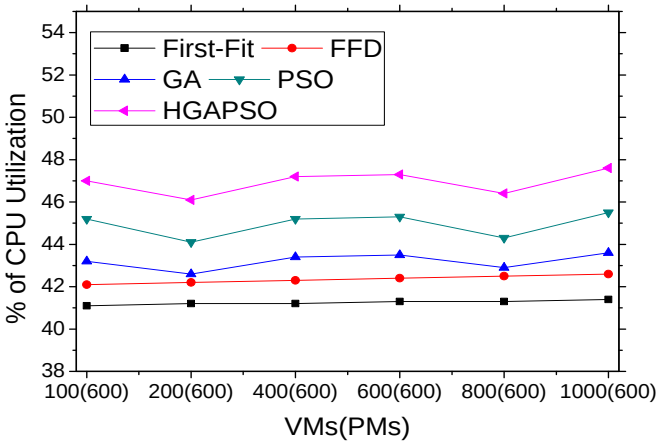

Figure 4.17: % of CPU Utilization in Constant Homogeneous Data Center.

VM, and Type4 VM during 200 seconds are shown in Figure 4.22, Figure 4.23, Figure 4.24, Figure 4.25 respectively.

We considered the processor intensive tasks in millions of instructions throughout our simulation. The amount of energy consumed by the data center using different VM allocation algorithms when 1000 VM and 600 PM combination used at the heterogeneous and homogeneous data center environments are shown in Figure 4.26 and Figure 4.27 respectively. The energy consumption in both homogeneous and heterogeneous data center environment is less in HGAPSO when compared to the other VM allocation techniques. This is due to the global optimal placement of VMs to PMs by our proposed HGAPSO resulting in switching-off more number of PMs at the data center. Hence, energy consumed by the data center is low in

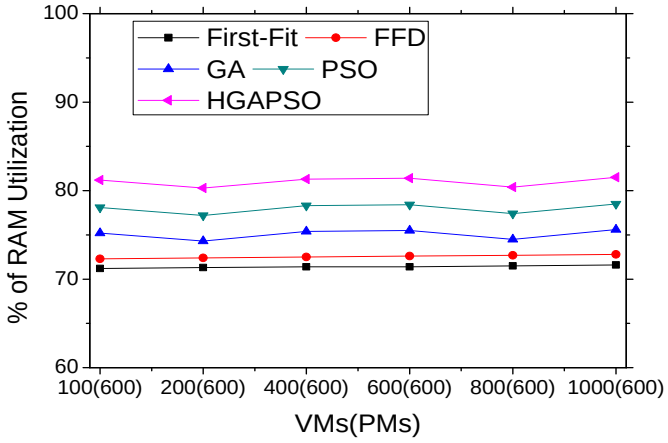

Figure 4.18: % of RAM Utilization in Constant Homogeneous Data Center.

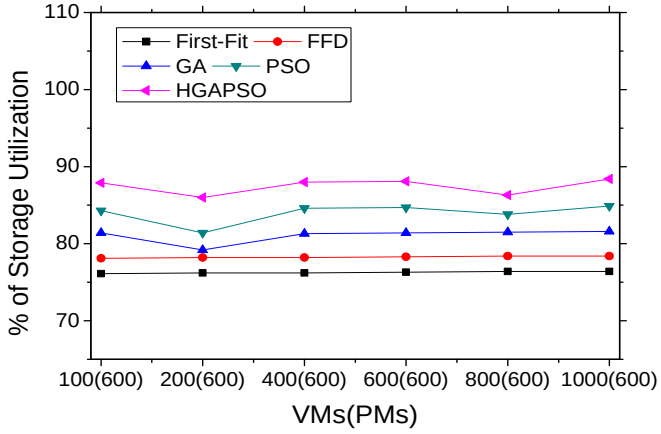

Figure 4.19: % of Storage Utilization in Constant Homogeneous Data Center.

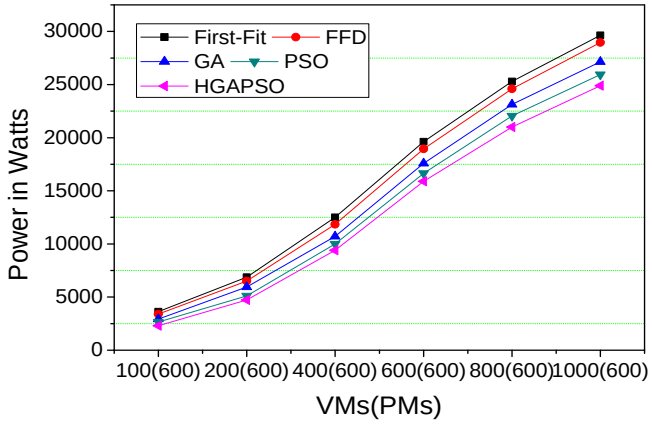

Figure 4.20: Power Consumption in Constant Homogeneous Data Center.

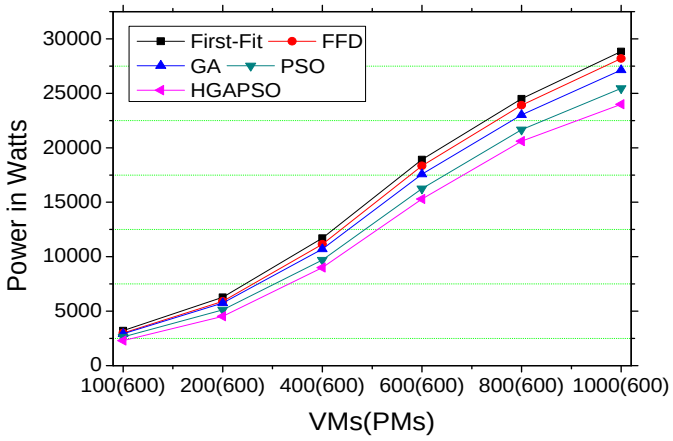

Figure 4.21: Power Consumption in Constant Heterogeneous Data Center.

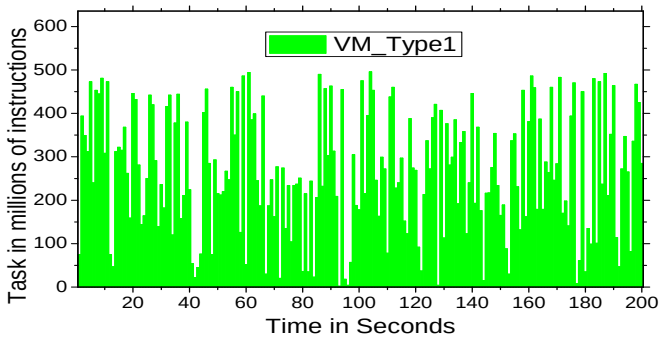

Figure 4.22: Task Allocation on Type1 VM.

HGAPSO as compared to other VM allocation techniques.

To check the performance of our proposed VM migration policy, we generated the user requested VMs in discrete time intervals at the data center. In this experiment, we considered 600 PMs at the data center (e.g. 200 type1 , 200 type2, 200 type3). Further, to derive the system model, each user has to send the request to the service provider and thereby getting the required VMs and the type of instances (small, medium, large, x.large). The range of VMs requested by the user is in between [1,100]. The life time of the VM is uniform in the range of [30s to 200s]. To migrate a VM from source node to the destination node, we used a constant migration overhead cost c^o_{mig} in terms of power consumption for each type of VMs such as 10 watt (small), 20 watt (medium), 30 watt (large), and 40 watt (x.large). On the arrival of each new user's request at the data center, we

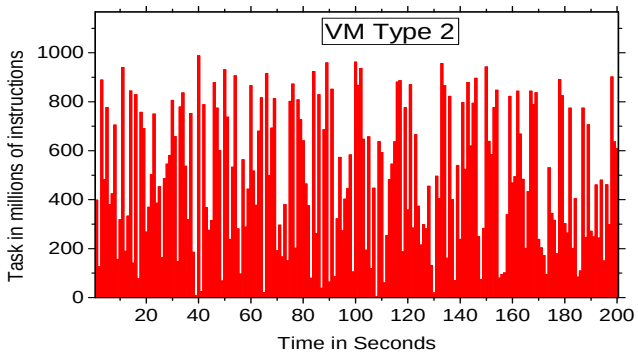

Figure 4.23: Task Allocation on Type2 VM.

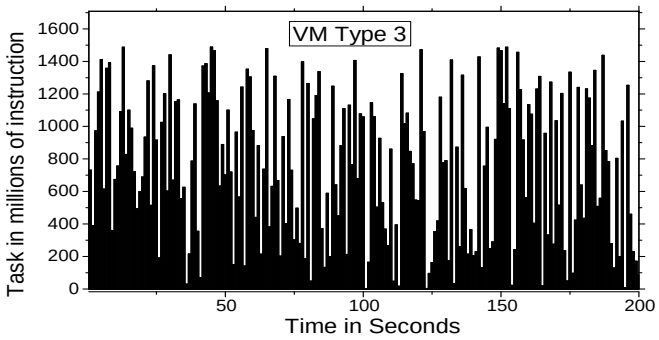

Figure 4.24: Task Allocation on Type3 VM.

applied our proposed HGAPSO for VM allocation. After completion of task by the VM, we terminate the VM from the PM, and further applied the proposed VM migration policy on the PM which are under utilized. Here, we considered the lower utilization threshold of (u_l=50%) for all the PMs at the data center.

Figure 4.28 shows the combined performance of proposed HGAPSO with VM migration policy known as (HGAPSO+VM Migration) in comparison with both (GA+VM Migration) and (PSO+VM Migration). The proposed HGAPSO+VM Migration saves 15% and 10% energy at the data center when compared to (GA+VM Migration) and (PSO+VM Migration) respectively. The combined approach gives better results in terms of energy saving over GA and PSO due to switching-off underutilized PM at the data center using VM migration technique. Hence, by this way, we can increase the effective CPU utilization of PMs at the

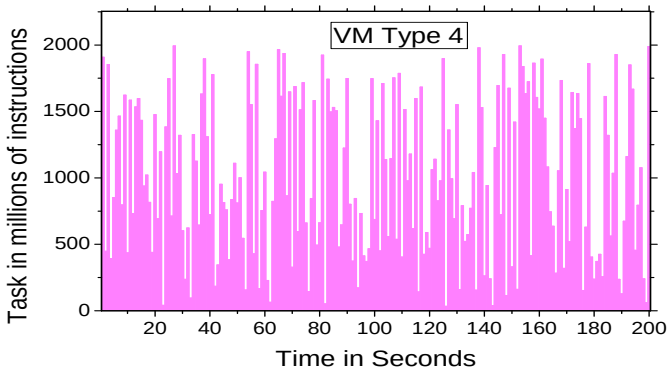

Figure 4.25: Task Allocation on Type4 VM.

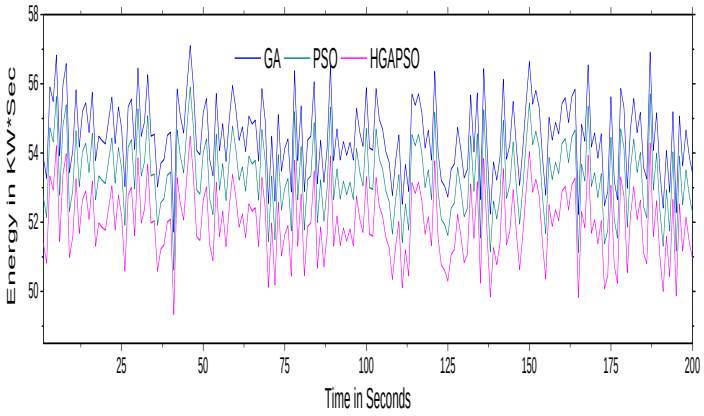

Figure 4.26: Energy Consumption in Homogeneous Data Center.

data center and consolidate the VMs to less number of energy efficient PMs at the data center. The migration of VMs at the data center is based on a regular time interval. In our experiment, we used the migration time interval of 30 seconds. After each 30 seconds, we gathered the utilization status of each used PMs at the data center and made a list of VMs which are allocated to underutilized PMs. Further, we migrate all the VMs in the list using migration policy as defined in Phase 3 and then we can switch-off underutilized PMs at the data center.

Figures 4.29 and 4.30 show the average resource utilization and overall energy consumption using 600 VM and 360 PM in homogeneous and heterogeneous data center environments respectively. The average resource utilization as shown in Fig. 4.29 for our proposed (HGAPSO+VM) Migration algorithm is 12% and 14% more when compared to that of GA, and PSO in heterogeneous and homogeneous

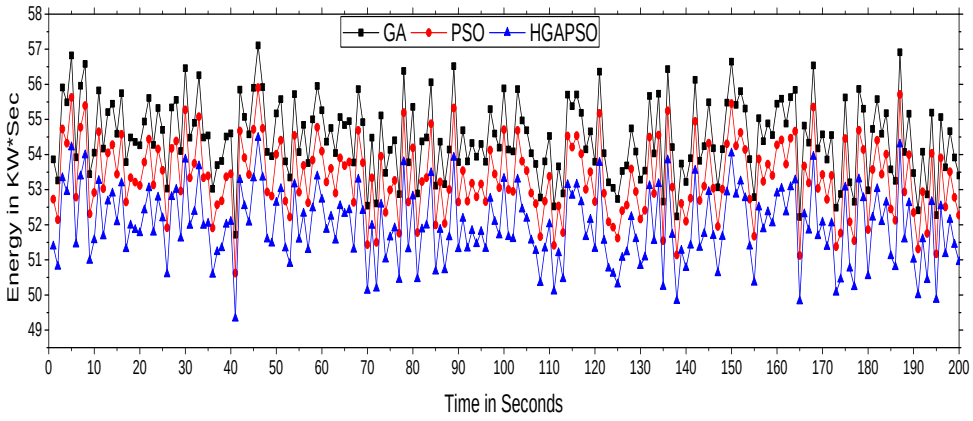

Figure 4.27: Energy Consumption in Heterogeneous Data Center.

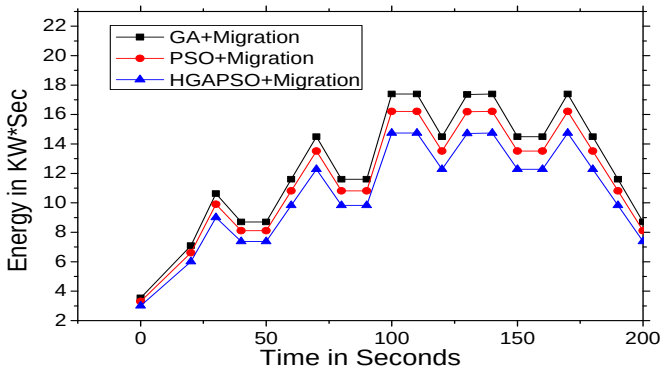

Figure 4.28: Energy Consumption with VM Migration Policy at Data Center.

environments respectively. Further, the overall energy consumption is (9.8%) less as compared to that of exact algorithm in both heterogeneous and homogeneous environments.

To check the scalability of the proposed algorithm, we conducted the experiment on HP Compaq LE1902x with 8 GB RAM, 3.40 GHz i-7 Processor and calculated the execution time of the proposed HGAPSO algorithm. The execution time of the proposed HGAPSO algorithm is mainly dependant on the GA operations (crossover and mutation) and PSO. Hence, total execution time of proposed HGAPSO algorithm is shown in Figure 4.31. The total execution time for the placement of 1000 VMs to the data center is approximately 3.5 minutes in the case of HGAPSO which is 1.55 minutes low as compared to HGACSO at the cloud data center. Hence, for the large data centers HGAPSO is more scalable in

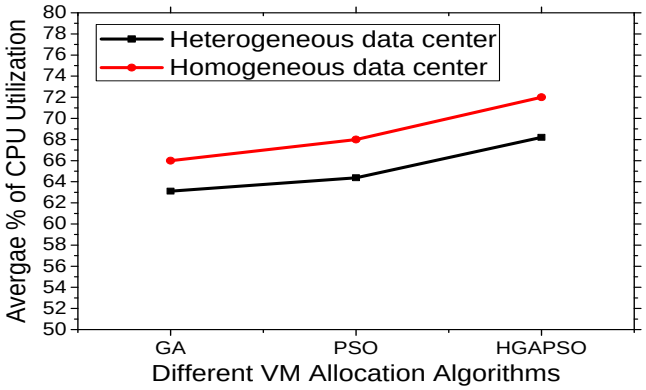

Figure 4.29: Average Resource Utilization.

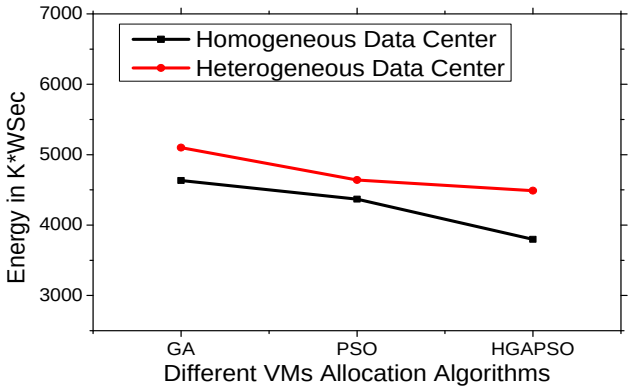

Figure 4.30: Total Energy Consumption.

nature as compared to HGACSO. Further, the detailed performance evaluation of HGAPSO, and HGACSO in terms of energy consumption, resources utilization is discussed in Chapter 7.

The minimization of power consumption on each iteration is considered for three evolutionary algorithms such as GA, PSO, and HGAPSO and its convergence is shown in Figure 4.32. The convergence of HGAPSO is fast as compared to that of GA and PSO since we improved the fitness of the chromosomes by applying the PSO, just before using the crossover and mutation operations. Further, this will result in less number of iterations required for the convergence of our proposed HGAPSO algorithm. For checking the reliability of the proposed HGAPSO algorithm, we conducted algorithm 15 times on each VMs(PMs) combinations and calculated the frequency of the same solution. The final allocation

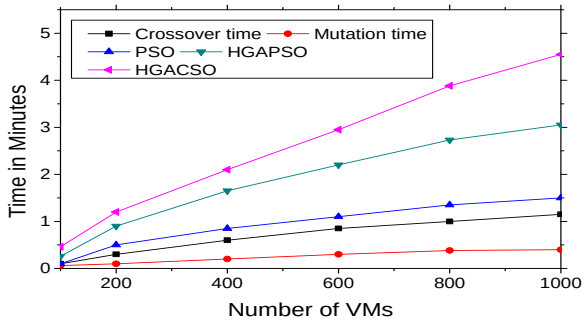

Figure 4.31: Execution Time of HGAPSO in Minutes.

of VMs to PMs at the data center is based on the solution which has the highest frequency. Hence, by this way, we calculated the confidence interval ($> 90\%$) of the highest frequent solution for the HGAPSO.

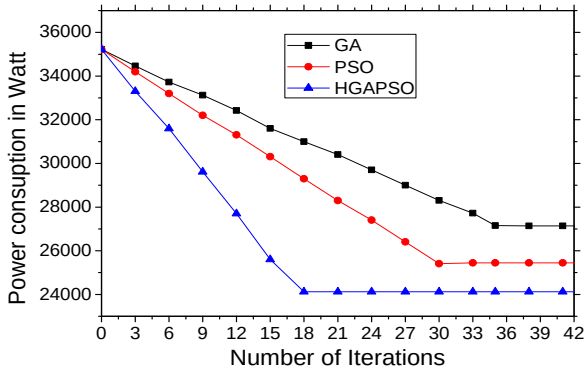

Figure 4.32: Minimization of Power Consumption on Each Iteration.

The generation size (s), number of iterations (t), crossover rate (CR), mutation rate (MR) are the important parameters for the performance of proposed HGAPSO algorithm. Figure 4.33 shows the performance of the proposed HGAPSO over different crossover rates. When the crossover rate is low then the objective function value is high; thus the proposed HGAPSO converged to non-global optimal point and takes more number of iterations. Further, by increasing the crossover rate, the objective function value is continuously going down, and an optimal crossover point of 0.6 is considered for the proposed HGAPSO. Further, increasing the crossover rate from 0.6 increases the objective function value due

to the early stagnation of the chromosomes and this will lead to sub-optimal value of objective function.

The performance of HGAPSO on mutation rate is shown in Figure 4.34. The mutation rate of 0.03 gives the near global optimum allocation of VMs to PMs by the proposed HGAPSO. The value of mutation rate less than 0.03 gives very little divergence in the solution space resulting in sub-optimal solution. Mutation rate greater than 0.03 gives higher diversity in the solution, thus HGAPSO moves away from the final near global optimum solution. To check the performance of proposed HGAPSO on solution size, we set different generation sizes (5 to 25). The performance of HGAPSO is high for generation size 10. The solution size less than 10 takes more number of iterations and gives sub-optimal solution due to less diversity in the solution space. The solution size more than 10 gives the same solution as that of 10 but the simulation running time is increased due to the high computation time of the proposed HGAPSO.

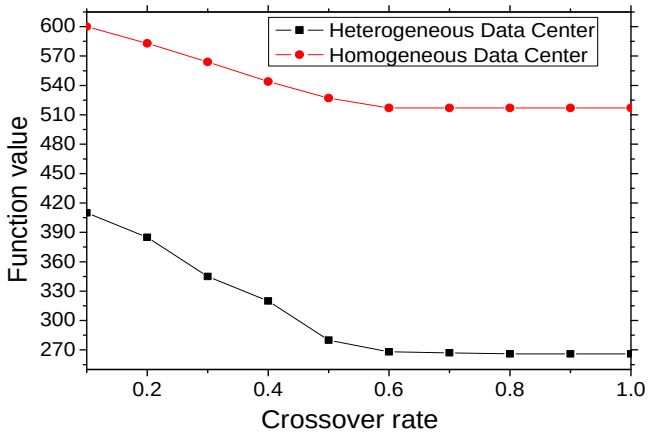

Figure 4.33: Performance of HGAPSO Over Crossover Rate.

The best values of the parameters for our proposed HGAPSO algorithm are given in Table 4.5. The best values of crossover and mutation parameters are calculated by checking the performance of HGAPSO on each crossover point between 0.1 and 1 through increasing the crossover rate from 0.01 to 0.08. In this process, we kept all other parameters such as population size, constants ($c1$, $c2$) and weight w are constant. After getting best values of crossover and mutation

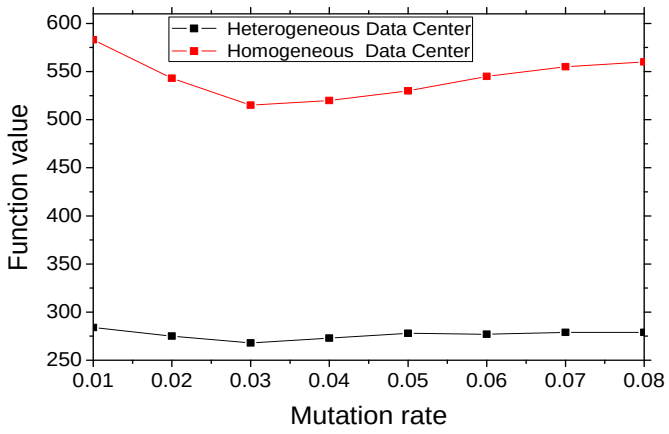

Figure 4.34: Performance of HGAPSO Over Mutation Rate.

Table 4.5: HGAPSO Parameters

Parameters	**GA**	**PSO**	**HGAPSO**
Population size	10	10	10
Max. Iteration	50	50	50
Crossover rate	0.6	-	0.6
Mutation rate	0.03	-	0.03
Selection	Roulette Wheel	-	Roulette Wheel
(c1, c2,w)	-	(0.5, 0.5,0.5)	(0.5, 0.5,0.5)

operations, we increased the population size and considered the minimum generation size which gives optimal allocation of VMs to PMs. The best constant values $c1$, $c2$ and weight value w are calculated by taking initial values $c1$=0.1, $c2$=0.1, and w=0.1 and increase these values by 0.1 till it will not reach 0.9. Let us consider the minimum values of $c1$, $c2$, and w so that there will not be any further performance improvement in the proposed HGAPSO.

Time Complexity Analysis: The time complexity of proposed HGAPSO with VM migration technique is based on GA, and PSO. Let us consider maximum iteration size k, individual size m (number of PMs). Number of VMs n, generation size s, and crossover point for each generation p. The time complexity of GA is dependent on the individual generation, crossover, and muta-

tion operations. The time complexity of PSO is dependent on (Particle generation, Initial velocity setting, Position calculation, global fitness calculation, local fitness calculation, updating in velocity, position updating, back fill of missed VMs, and removing of duplicate VMs for specified iterations). Further, the First-Fit Decreasing (FFD) based VM migration requires O($nlogn$) time complexity. HGAPSO=O(O(Individual generation)+ Total iterations*(O(Fitness calculation*Size of generation)+ O(Crossover*Size of generation)+O(Mutation*Size of generation))+ O(PSO)+O(VM migration) Hence, resultant time complexity of HGAPSO=O(O($n * s$)+O(k*(O($m * s$)+O($n * m * s - p * m * s$)+O(s))) + O(k*(O($s * m$)+O($s * m$)+O(s)+ O(s)+ O($n * m * s$))+O($nlogn$)) which is equal to O($n. * m * s * k$)+$O(nlogn)$ polynomial in nature.

4.3 Summary

This chapter highlights the hybrid approach for multi-objective VM allocation to the PMs at the data center in both homogeneous and heterogeneous data center environments up to 1000 VM and 600s PM. The hybrid approach of HGAPSO allocates the VMs to PMs using both GA and PSO. The hybrid algorithm saves the energy consumption by (26%, 22%, 18%, and 15%) in heterogeneous environment and (20%, 16%, 12%, and 9%) in homogeneous environment over First-Fit, FFD, GA,and PSO respectively. The average resources utilization of the data center is also increased by the HGAPSO algorithm. The execution time of HGAPSO is further, reduced by replacing CSO of HGACSO with PSO in the case of HGAPSO. Further, the customer based SLA is defined for IaaS cloud model.

But the key limitations of the proposed HGAPSO is that we randomly migrate the VMs from one PM to another PM in mutation operation. Hence, there is a chance to lose the fitness value of the chromosomes. Further, in this chapter, we did not consider the thermal temperature of the cloud data center. Hence, to resolve these issues we propose energy efficient thermal aware VM allocation using HGAPSOSA at the cloud data center. The details of HGAPSOSA are discussed in Chapter 5.

Chapter 5

Energy Efficient Thermal Aware VM Allocation and Migration Using HGAPSOSA

The multi-objective energy efficient thermal aware VM allocation to PMs at the cloud data center is an important and challenging problem. In Chapter 3, and Chapter 4, we discussed the multi-objective VM allocation problem by using HGACSO, and HGAPSO algorithms respectively. Further, we considered two objectives such as minimizing both the energy consumption, and resources wastage at the cloud data center using proposed HGACSO, and HGAPSO.

However, minimizing the energy consumption of the data center by allocating the large number of VMs to small number of PMs will lead to overutilization of PM (higher temperature of PM) and thus, there is a chance of failure of hardware resources at the cloud data center. Hence, in this chapter, we propose an energy efficient, thermal and resources aware VM allocation algorithm at the cloud data center. The proposed VM allocation algorithm is the hybrid combination of GA, PSO, and Simulated Annealing (SA) known as HGAPSOSA. Further, we propose a First-Fit approximation based thermal aware VM migration policy at the cloud data center. The research contributions towards the development of efficient VM allocation, and migration techniques are as follows:

1. To design an Euclidean distance based multi-objective optimization function for minimizing the energy consumption, thermal temperature, and resources wastage at the cloud data center.

2. To design a novel hybrid bio-inspired HGAPSOSA algorithm for VM allocation using GA, PSO, and SA and thus reducing the energy consumption, thermal temperature, and resources wastage at the cloud data center.

5.1 Proposed Work

The energy efficient thermal aware VM allocation and migration consists of two phases such as allocation of users requested VMs to PMs over the period of time, and application of VM migration policy to migrate the VMs from underutilized PM to energy efficient PM at the cloud data center. The flow chart of the proposed VM allocation and migration policy is already discussed in Section 3.1. The details of VM allocation and migration policies are described below.

5.1.1 VM Allocation Using Proposed HGAPSOSA

Before designing a mathematical model of VM allocation problem, we need to know the power consumption of a PM at the cloud data center. Hence, in this chapter, we used the same power consumption model of a PM as discussed in Section 3.1. Further, before applying the VM allocation algorithm to the data center, we need to know the thermal temperature dissipation model of a PM, and a mathematical model of VM allocation problem which consists of multi-objective function such as minimizing the energy consumption, thermal temperature, and resources wastage. Hence, thermal temperature model, multi-objective VM allocation problem formulation, and description of HGAPSOSA are discussed below.

A. Thermal Temperature Model of a PM

Thermal performance is the critical parameter to evaluate the performance of the cloud data center. Since, hot spots are generated due to the overutilization of a PM resulting in disruptive downtime of a PM at the data center. Further, there is a well known duality between heat transfer and register capacitor (RC) circuit (Kim et al. 2014). Thus, we used a thermal RC circuit to model the steady state temperature (T) of pm_j as described by Eq. 5.1.

$$T = PR + T_{amb} \tag{5.1}$$

Since, there exists the linear relationship between CPU utilization and power consumption, hence the temperature of a PM in terms of the CPU utilization is described by Eq. 5.2.

$$T_j = P_j^{min} + (P_j^{max} - P_j^{min})R + T_{amb} \tag{5.2}$$

Where, T_j is the thermal temperature of a pm_j; R is the thermal resistance between power and CPU utilization; T_{amb} is the ambient temperature of a PM.

B. VM Allocation Problem Formulation

To solve the Pareto optimal multi-objective VM allocation problem at the cloud data center, we used an Euclidean distance to calculate an optimal value of all objectives. Let us consider the resources utilization of pm_j as (u_j^d), $\forall d \in \{MIPS, RAM, Storage\}$. Since, all the individual resources are available in different quantity at the PM, hence we used the normalized value of resource utilization, power consumption, and, thermal temperature.

Hence, the normalized values of resource utilization, power consumption, and thermal temperature are given by $(u_j^d/\ u_j^{max})$, (P_j/P_j^{max}), and (T_j/T_j^{max}) respectively. Further, the normalized values of resource utilization, power consumption, and thermal temperature are in between 0 and 1. Hence, the objective function to minimize the resource wastage, power consumption, and thermal temperature is described by Eq. 5.3.

$$min \quad f = \sum_{j=1}^{m} \sqrt{(\frac{P_j}{P_j^{max}})^2 + (u_j^d - 1)^2 + (\frac{T_j}{T_j^{max}})^2} \tag{5.3}$$

$$VM_i^{pe} * a_{ij} \leq pm_j^{pe} \quad \forall j \in \{1, 2, 3, .., m\} \tag{5.4}$$

$$\sum_{i=1}^{n} VM_i^{mips} * a_{ij} \leq pm_j^{mips} \quad \forall j \in \{1, 2, 3, .., m\} \tag{5.5}$$

$$\sum_{i=1}^{n} VM_i^{ram} * a_{ij} \leq pm_j^{ram} \quad \forall j \in \{1, 2, 3, .., m\} \tag{5.6}$$

$$\sum_{i=1}^{n} VM_i^{storage} * a_{ij} \leq pm_j^{storage} \quad \forall j \in \{1, 2, 3, .., m\} \tag{5.7}$$

Eqs. 5.4 to 5.7 describe the constraints of (MIPS, RAM, and Storage) respectively.

C. Basics of GA, PSO, and, SA

The proposed HGAPSOSA algorithm is the hybrid combination of GA, PSO, and SA. Hence, to understand the working of the proposed HGAPSOSA algorithm, we need to consider the basic principles of GA, PSO, and SA. Further, basics of GA, and PSO are already discussed in Section 3.1, and Section 4.1 respectively.

Hence, the detailed description of SA is given as follows.

Generally SA is used to solve the larger nonlinear convex combinatorial optimization problem. Further, SA is generally used when the search space is discrete and large. It involves heating and controlled cooling of a material to increase the size of the material and thereby removing the defects of the material. At each iteration SA considers some neighboring state s'' of the current state s' and probabilistically decides the moving of system into state s'' or remains in the same state. Hence, this probability will lead the system to move to a lower energy state. Ultimately this step is repeated until the system reaches in the state which is good enough for the objective of application.

D. Description of HGAPSOSA

Figure 5.1 shows the flowchart of the proposed HGAPSOSA algorithm. The motivation behind the hybrid combination of GA, PSO, and Simulated Annealing (SA) techniques are explained as follows. Both GA and PSO techniques work on a random population for finding a solution in the given search space to solve the discrete optimization problem. In the proposed HGAPSOSA algorithm, the crossover operation of GA changes the positions of VMs from one chromosome to another chromosome by migrating the VMs from one chromosome to another chromosome. However, GA takes more convergence time using two conditions (i) When the search space is large and (ii) When the fitness value of the parent chromosomes is low. The other important reason to generate the low quality of solution is due to random migration of VMs from one PM to another PM. But the migration of VMs from one PM (non-energy efficient) to another PM (energy efficient) will ensure high probability of getting the global optimal solution for VM allocation problem at the cloud data center.

In the case of PSO, all the particles move their position by moving the VMs from one PM to another PM using the global and local best positions of the particles. Thus, resulting in migration of VMs from non-energy efficient PMs to the energy efficient PMs at the cloud data center. Hence, PSO has fast convergence speed as compared to GA because of dependency on the local best particle posi-

tion. But, sometimes PSO falls in local optima. Hence, to minimize the drawbacks of both GA, and PSO, HGAPSO is designed based on the hybrid combination of GA, and PSO. But, the crossover and mutation operations change the position of VMs randomly, resulting in poor solution quality of the chromosomes in the case of hybrid approach based on GA and PSO.

Hence, we modify the mutation operation of HGAPSO by applying SA to the chromosomes, and thus we migrate the VMs from the least energy efficient PM to the energy efficient PM. Hence, using the hybrid combination of GA, PSO, and, SA techniques, we obtained the global best optimal solution for multi-objective VM allocation problem at the cloud data center.

The complete operations of the proposed HGAPSOSA algorithm are described in 18 steps as shown in Figure 5.1. In the first iteration of HGAPSOSA, we generate a finite number of chromosomes as discussed in Section 3.1. After the generation of chromosomes, we sort the chromosomes in the decreasing order based on their fitness value.

Further, we discarded half of the least fittest chromosomes from the population, and the remaining fittest chromosomes go for the further operations. Hence, after selecting the fittest chromosomes we need to apply the PSO operation for improving the fitness value of the chromosomes as discussed in Section 4.1. Thus, by applying the PSO we improved the fitness value of the chromosomes and then we applied GA operations (crossover, and mutation) on the chromosomes as discussed in Section 3.1. In the proposed HGAPSOSA algorithm, we modified mutation operation by applying simulated annealing to the chromosomes. After compilation of all the operations of the proposed HGAPSOSA algorithm, we need to select the fittest individual (particles or chromosomes) for performing the next iteration. The execution of the proposed HGAPSOSA algorithm will be terminated when one of the defined termination conditions is satisfied as discussed in Section 4.1.

Fitness Function: To select the best chromosomes, and particles for the next iteration, we need to define the fitness function. Hence, the Euclidean distance based fitness value of the chromosome, and particles k is defined by Eq. 5.8.

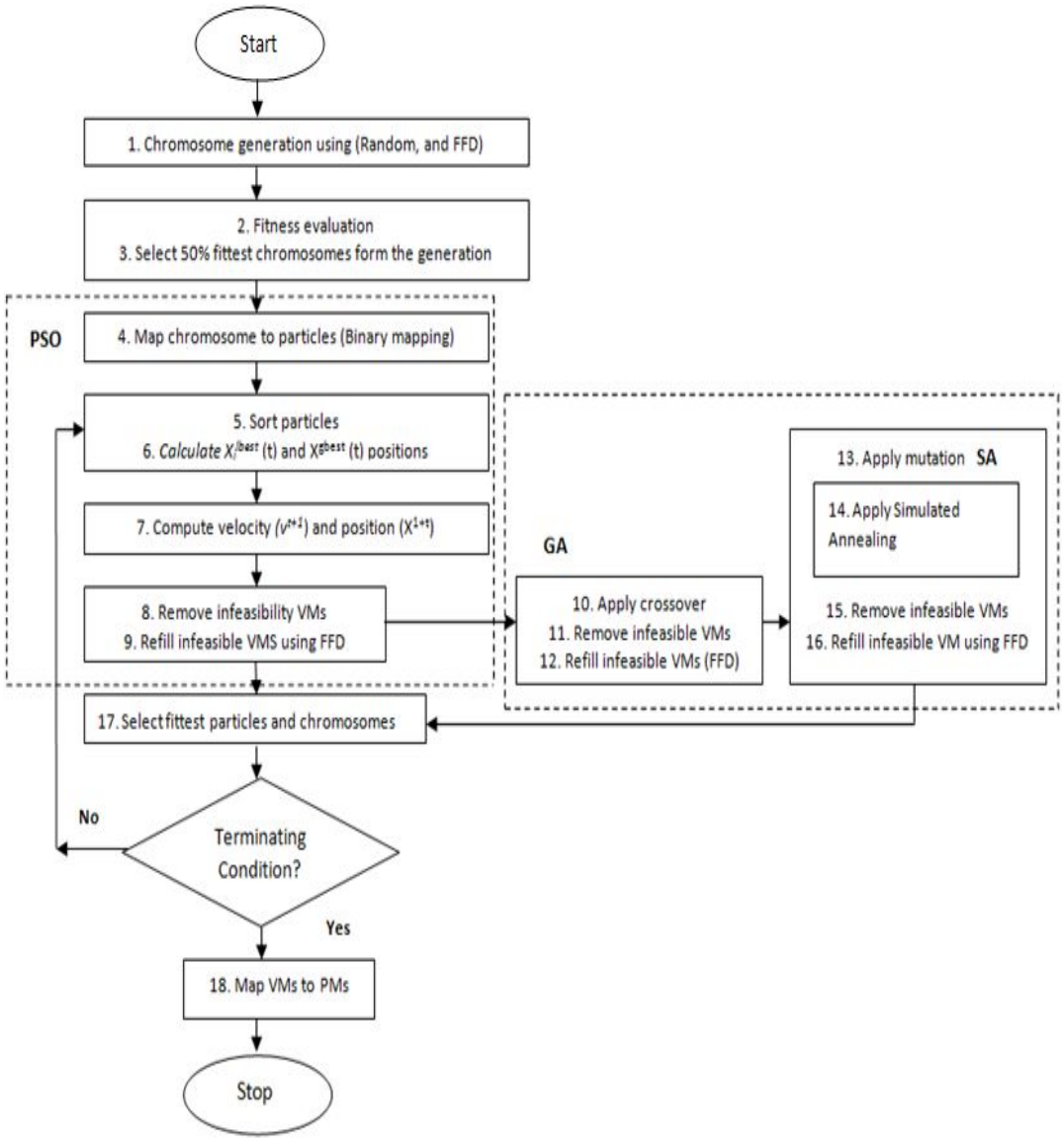

Figure 5.1: Flow Chart of the Proposed HGAPSOSA Algorithm.

Further, the fitness function selects the optimal number of PMs for the allocation of VMs and thereby reducing energy consumption, thermal temperature, and resources wastage at the cloud data center. The fitness value of the chromosome is in between 0 and 1.

$$fit_k = \frac{1}{\sum_{j=1}^{m} \sqrt{(\frac{pm_j}{pm_j^{max}})^2 + (u_j^d - 1)^2 + (\frac{T_j}{T_j^{max}})^2}} \tag{5.8}$$

To select the fittest chromosomes from the population, we need to calculate the fitness value of chromosomes. Let s be the size of a generation then half of the fittest chromosomes, such as $s/2$ number of chromosomes will be considered for further computation and thereby destroying remaining half of the chromosomes in the population. The selected fittest chromosomes are considered for PSO, GA operations (crossover, mutation), and simulated annealing. Hence, the probability of the chromosomes to go for further operations is defined in Section 4.1.

After selecting the best chromosomes on the basis of fitness function from the population, we need to apply the PSO operation to improve the fitness quality of the chromosomes. Thus, we used the same chromosome to particle mapping technique as discussed in Section 4.1. To change the current position of the particle in the proposed HGAPSOSA, we need to apply the initial velocity to the particle. Hence, we applied m bit binary velocity to the current position of the particle as discussed in Section 4.1. To calculate the difference between two position vectors of the VMs, we need to apply the subtraction operation as discussed in Section 4.1 of Chapter 4.

Addition Operator: The addition operator calculates the changes in velocity of the particle. In the proposed HGAPSOSA algorithm, the addition operator is defined by symbol $\oplus$. By applying the addition operator, we calculated the changes in the velocity of the particle, such as $P_1V_1^t \oplus P_2V_2^t \oplus, ...P_nV_n^t$ represents the velocity change using V_1^t with probability P_1, and using V_n^t with probability P_n. In the proposed HGAPSOSA algorithm, each probability $P_i, (\sum_{i=1}^{n} P_i = 1)$ is defined by an inertia coefficient. The detailed description of the addition operator is defined by the following example: Let us consider, $0.3(1,1,0,1,0) \oplus 0.7(1,1,1,0,0) = (1,1, \neq,$

$\neq$, 0), where the probability of occurrence of 1 at 1^{st} and 2^{nd} bit positions is 1 and the probability of occurrence of 0 at the 5^{th} bit position is 1. Thus, bit values at 1^{st}, 2^{nd}, and 5^{th} positions are 1, 1, and 0 respectively, and the remaining positions are uncertain (denoted by $\neq$). Further, to calculate the uncertain bit value, we need to define three inertia weight coefficients such as P_1^i, P_2^i, P_3^i, and these defined inertia weight coefficients are as follows:

$$f(X_i^t) = \sum_{j=1}^{m} \sqrt{(\frac{P_j}{P_j^{max}})^2 + (u_j^d - 1)^2 + (\frac{T_j}{T_j^{max}})^2} \tag{5.9}$$

$$P_{1i} = \frac{f(X_i^t)}{f(X_i^t) + f(X_{lbesti}^t) + f(X_{gbest}^t)} \tag{5.10}$$

$$P_{2i} = \frac{f(X_{lbesti}^t)}{f(X_i^t) + f(X_{lbesti}^t) + f(X_{gbest}^t)} \tag{5.11}$$

$$P_{3i} = \frac{f(X_{gbesti}^t)}{f(X_i^t) + f(X_{lbesti}^t) + f(X_{gbest}^t)} \tag{5.12}$$

Where, $f(X_i^t)$ represents the fitness value of i^{th} particle for the solution X_i^t; X_{lbesi}^t, and X_{gbesi}^t represent the local best position and global best position of i^{th} particle respectively. To calculate the inertia weight coefficients, we need to consider the high energy efficiency and the maximum utilization with high probability.

$$\begin{cases} uncertain \quad bit = q_1, \quad if \quad rand \leq P_{1i} \\ uncertain \quad bit = q_2, \quad if \quad P_{1i} < rand \leq P_{2i} \\ uncertain \quad bit = q_3, \quad if \quad P_{2i} < rand \leq p_{3i} \end{cases}$$

Where, q_1 is the corresponding bit of the particle velocity vector before updating; q_2 is the corresponding bit value of the local best particle vector; and q_3 is the corresponding bit of the best global particle velocity vector.

To update the position of the particles, we need to apply the multiplication operator in the proposed HGAPSOSA. Hence, we applied the multiplication operator as discussed in Section 4.1. At the end of PSO operation, we need to apply the crossover operation in the proposed HGAPSOSA algorithm. Thus, we applied crossover operation between two chromosomes as discussed in Section 4.1. At the end, we need to apply the mutation operation in the proposed HGAPSOSA by

applying the SA on the chromosomes. The detailed description of the mutation operation is described below.

Mutation Operation: In the proposed HGAPSOSA, we performed mutation operation by applying simulated annealing on the chromosomes. Thus, in the simulated annealing, first we arranged the PMs in the decreasing order on the basis of their fitness value in the chromosome, and then select the least fittest PM in the chromosome. After selecting the least fittest PM, we reallocated all the VMs of least fittest PM to the energy efficient PM and calculated the new fitness value of chromosome. If ($fit_i^{new} > fit_i^{old}$) then change the VM position permanently otherwise select the second least fittest PM in the chromosome and perform the same procedure, and compare the new and old fitness values of the chromosome. The details of the mutation operation using SA are shown in Figure 5.2.

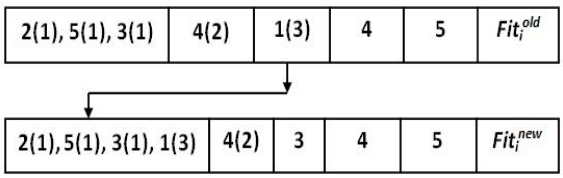

Figure 5.2: Mutation Operation.

After applying PSO, crossover, and SA we reallocate infeasible VMs in the particles and chromosomes using the reallocation policy as discussed in Section 4.1. Further, the termination condition of the proposed HGAPSOSA is same as discussed in Section 4.1.

The detailed steps of the proposed HGAPSOSA for the allocation of VMs to PMs are described in Algorithm 5.1. Where t is the current iteration initialized by 1. t^{max}, and s are the maximum number of iterations and generation size respectively and these values are set at the beginning of the iteration. Steps 1 to 11 generate the $s-1$ number of chromosomes by applying random VM allocation policy. Step 12 describes the generation of chromosomes using FFD approximation technique. Termination condition of the proposed HGAPSOSA algorithm is described by Step 13. Step 14 describes the selection of half of the fittest chromosomes from the generation. Mapping of particles from the chromosomes is

Algorithm 5.1 Proposed HGAPSOSA

Initialize: t←1, t^{max}, s

Input: VM-List, PM-List

Output: Final Allocated List of VM to PM

1: **for** i ← 1 to (s-1) **do**
2: **while** $VM[] \neq empty$ **do**
3: Select randomly VM_i from VM-List
4: Select randomly pm_j from PM-List
5: **for** Each d **do**
6: **if** $(VM_i^d \leq pm_j^d)$ **then**
7: Allocate VM_i to pm_j
8: Update dimension of pm_j
9: Remove VM_i from VM-List.
10: **else**
11: go to 4
12: Generate (Chr_s) using First Fit
13: **while** Termination condition==false **do**
14: Select s/2 Number of Fittest Chromosomes.
15: **for** i← 1 to s/2 **do**
16: $Part_i \leftarrow Chr_i$
17: Calling ***Procedure 4.1***.
18: Calling Crossover$(Parent_1, Parent_2)$ Operation
19: Calling ***Procedure 3.2***
20: **for** i← 1 to $s/2$ **do**
21: Calling ***Procedure 5.1*** (Chr_i)
22: $t \leftarrow t + 1$
23: Final Allocation of VM to PM

Procedure 5.1 SA

1: **for** i←1 to s/2 do **do**

2: Arrange PM in the decreasing order based on the fitness value and generate Chr_i^{new}

3: Select all VM from the least fittest PM and reallocate VM to the energy efficient PM.

4: **if** $(fit_i^{new} > fit_i^{old})$ **then**

5: $Chr_i^{old} = Chr_i^{new}$

described by Steps 15 to 16. Improvement in the fitness value due to the PSO is described by Step 17. The details of the PSO operation are discussed in Section 4.1. Step 18 calls the crossover operation on chromosomes. Removing and Refilling of infeasible VMs is described by calling Algorithm 4.2 as discussed in Section 4.1 (Step 19). Calls of Procedure 5.1 and mutation operation are described by Steps 20 to 22. Step 23 increments the iteration value by 1 after completion of each iteration.

Procedure 5.1 describes the mutation operation by applying simulated annealing. Steps 1 to 2 describe sorting the PMs of all chromosomes in the decreasing order based on their fitness value. Step 1 describes the selection of all the VMs which are allocated to the least energy efficient PMs. Steps 4 to 5 describe reallocation of VMs from the least fittest PM to the energy efficient PM at the cloud data center.

5.1.2 Energy Efficient Thermal Aware VM Migration

To migrate the VM from underutilized PM to energy efficient PM at the cloud data center, we used the same First-Fit approximation based approach as discussed in Section 4.1.2. The proposed First-Fit approximation based VM migration technique will switch-off underutilized/idle PM at the cloud data center and resulting in reducing the energy consumption, thermal temperature, and resources wastage at the cloud data center.

5.2 Experimental Setup, Results and Analysis

To evaluate the performance of the proposed HGAPSOSA algorithm for multi-objective resources allocation at the cloud data center, we implemented our proposed HGAPSOSA algorithm using Java 1.7. Further, the performance of proposed multi-objective HGAPSOSA is compared with other state-of-the-art VM allocation algorithms in terms of energy efficiency, resources utilization, and thermal temperature by taking different number of VMs at the cloud data center. The experimental setup and results & discussion are described below.

Experimental Setup: In order to calculate the performance of power consumption and thermal temperature of the proposed HGAPSOSA algorithm, we used power, and thermal temperature models of IBM Blade server (IBM-Switch-Model 2014). The IBM Blade center server (PM) consists of 14 HS21 blades each having two Xeon (Dual-Core), 2.33 GHz processors with 4 MB L2 cache, 8 GB RAM, and 120 GB disk storage. Figure 5.3 shows the power consumption of the blade server. It is clear from Figure 5.3 that the power consumption is linearly dependent on the CPU utilization (%) of the server. Further, CPU temperature is approximately linear function of the CPU utilization (%) as shown in Fig. 5.4.

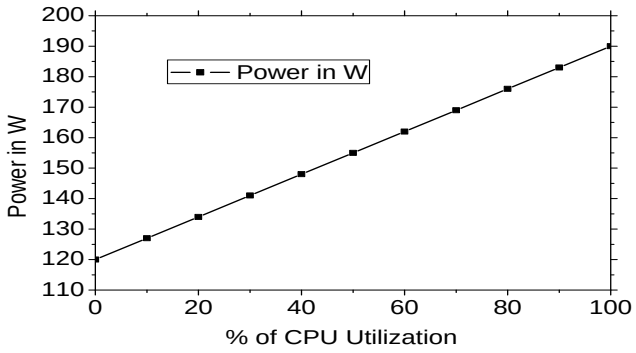

Figure 5.3: Power Consumption.

In our proposed work, we considered the total temperature of a CPU as an average temperature of 4 cores. Figure 5.4 shows the CPU temperature of 4 cores for different % of CPU utilization. To create the different types of VMs on the PMs at the data center, we considered Amazon based three different types of VMs instances, such as small-VM, medium-VM, large-VM (Amazon-Website

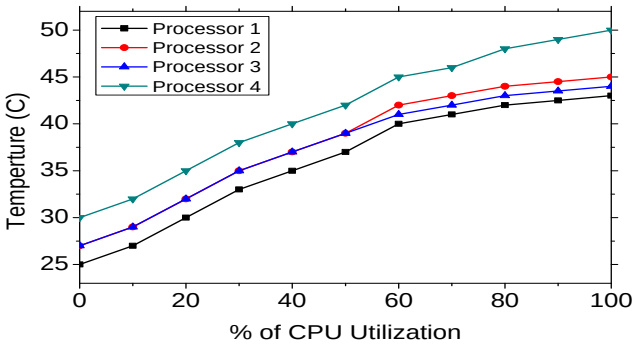

Figure 5.4: Temperature Variation.

2014). The overall configuration of small-VM (number of processing elements is 1, mips is 500, ram 0.5 GB, storage 40 GB), medium-VM (number of processing elements is 2, mips is 1000, ram 1 GB, storage 60 GB), and large-VM (number of processing elements is 3, mips is 1500, ram 2 GB, storage 80 GB).

Results and Discussion: To check the performance of the proposed HGAPSOSA algorithm, experimental results are carried out in two different cases. In Case-1, we considered different combinations of VMs and PMs (PMs are fixed at 60% of VMs) in two different data center environments (homogeneous and heterogeneous). Further, in Case-2, we check the performance of our proposed HGAPSOSA algorithm by keeping number of PMs constant and taking variable number of VMs at the cloud data center.

Case-1

The number of switched-on PMs at the cloud data center for the allocation of different number of VMs is shown in Table 5.1. Hence, number of switched-on PMs for the VMs allocation (based on different VMs(PMs) combinations) is less in the case of our proposed HGAPSOSA, due to modified crossover operation using simulated annealing. Thus, we switched-off underutilized PMs by migrating the VM from underutilized PM to the energy efficient PM at the cloud data center. This will result in less number of switched-on PMs in the case of proposed HGAPSOSA when compared to that of GA, PSO, and HGAPSO. After allocation of VMs to PMs, we checked the resources (CPU, RAM, and Storage) utilization on different VMs(PMs) combinations at the cloud data center. The

Table 5.1: Number of Switched-on PMs at the Data Center

VM(PM)	GA	PSO	HGAPSO	HGAPSOSA
100(60)	50	48	42	39
200(120)	98	95	83	77
400(240)	196	191	166	153
600(360)	294	287	249	230
800(480)	390	380	332	307
1000(600)	487	477	415	384

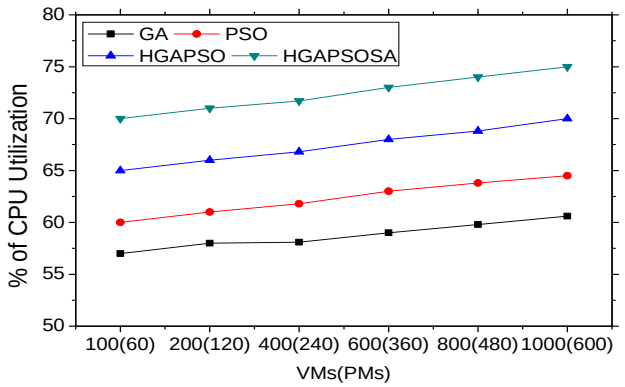

Figure 5.5: % of CPU Utilization at the Data Center.

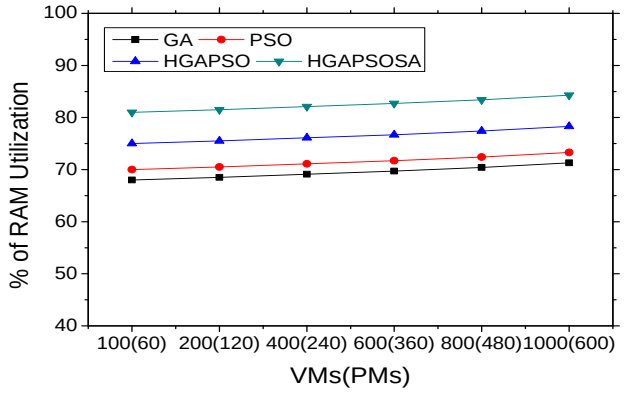

Figure 5.6: % of RAM Utilization at the Data Center.

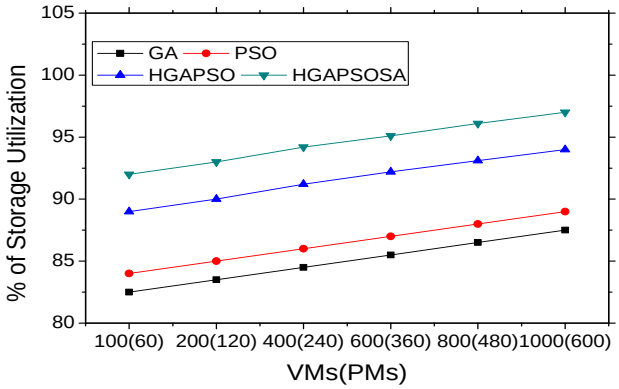

Figure 5.7: % of Storage Utilization at the Data Center.

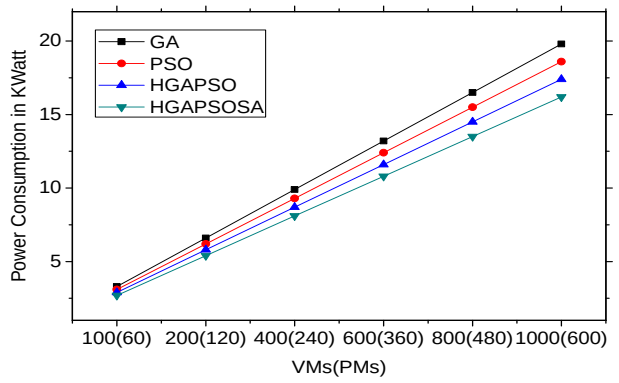

Figure 5.8: Power Consumption at the Data Center.

resources (CPU, RAM, and Storage) utilization (%) at the cloud data center are shown in Figure 5.5, Figure 5.6, and Figure 5.7 respectively. The % of resources utilization in the case of our proposed HGAPSOSA algorithm is high when compared to the other VM allocation techniques, since HGAPSOSA takes less number of PMs for the allocation of VMs. Thus, the resources wastage is less resulting in higher % of resources utilization in the case of HGAPSOSA.

Further, the power consumption at the data center for different VMs(PMs) combinations is shown in Figure 5.8. The power consumption of the data center in the case of our proposed HGAPSOSA algorithm is less, since our proposed algorithm gives an optimal solution for the multi-objective VM allocation problem. Hence, more number of idle PMs are switched-off, resulting in less power consumption at the cloud data center.

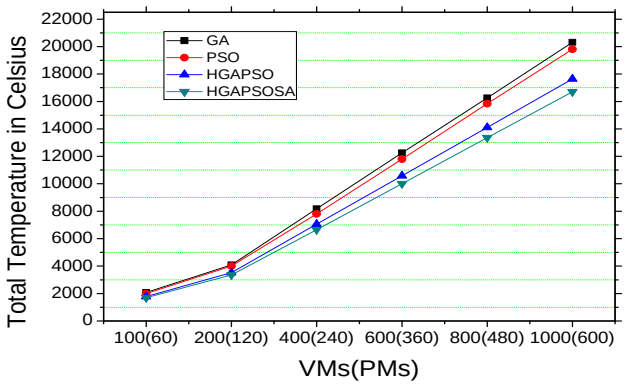

Figure 5.9: Temperature of the Data Center.

Table 5.2: Number of Switched-on PMs at the Constant Data Center

VM(PM)	GA	PSO	HGAPSO	HGAPSOSA
100(600)	51	49	44	40
200(600)	105	102	91	84
400(600)	197	192	168	154
600(600)	302	295	260	234
800(600)	391	383	335	309
1000(600)	488	479	417	385

Figure 5.9 shows the temperature of the data center for different VMs(PMs) combinations. The temperature of the data center is low when compared to the other VM allocation algorithms, due to the optimal utilization of PM resources, and more number of PMs are in switched-off condition.

Case-2

Table 5.2 shows the number of switched-on PMs at the cloud data center for the allocation of different number of VMs while keeping the number of PMs constant. Further, we keep the number of PMs constant and vary the number of VMs to check the performance of our proposed HGAPSOSA algorithm in terms of CPU, RAM, and Storage utilization at the cloud data center. Figures 5.10, 5.11, and 5.12 show the % of CPU, RAM, Storage utilization respectively, using different VM allocation techniques at the cloud data center. The % of

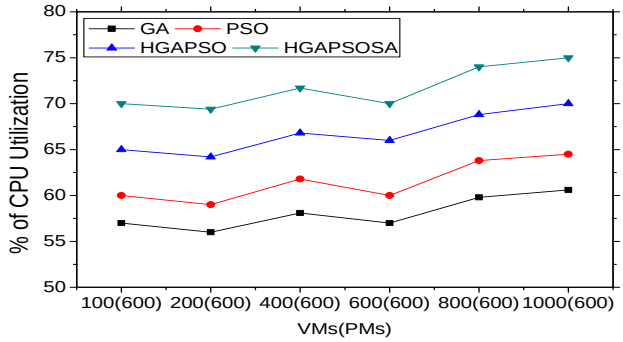

Figure 5.10: % of CPU Utilization at Constant Data Center.

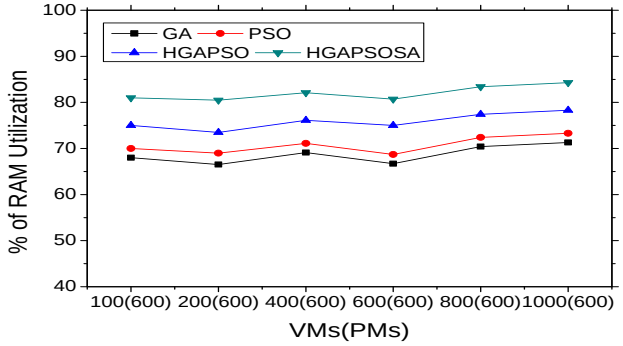

Figure 5.11: % of RAM Utilization at Constant Data Center.

CPU, RAM, and Storage utilization of the data center is high in the case of HGAPSOSA when compared to GA, PSO, and HGAPSO. Further, performance of CPU, RAM, and Storage utilization is varying while taking different number of VMs and keeping number of PMs constant at cloud data center. Since, size of the chromosome is large in proportion to the number of VMs, so there is a high probability that HGAPSOSA selects idle PM in the chromosome for the allocation of VMs at the cloud data center.

Figures 5.13 and 5.14 show the power consumption and thermal temperature while taking variable number of VMs and constant number of PMs at the cloud data center. The power consumption and overall temperature of the cloud data center in case of proposed HGAPSOSA algorithm is low due less number of switched-on PMs.

To check the performance of the proposed HGAPSOSA in terms of the power consumption, resources utilization, and thermal temperature, we fix the size of

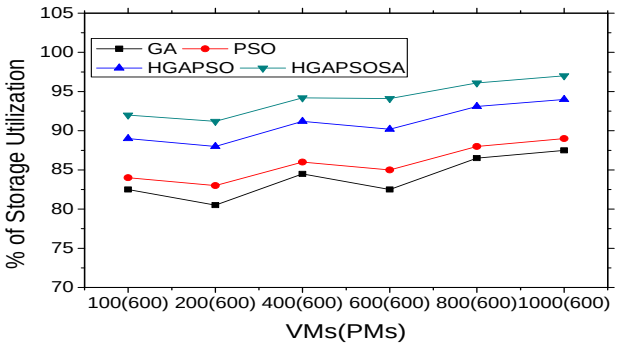

Figure 5.12: % of Storage Utilization at Constant Data Center.

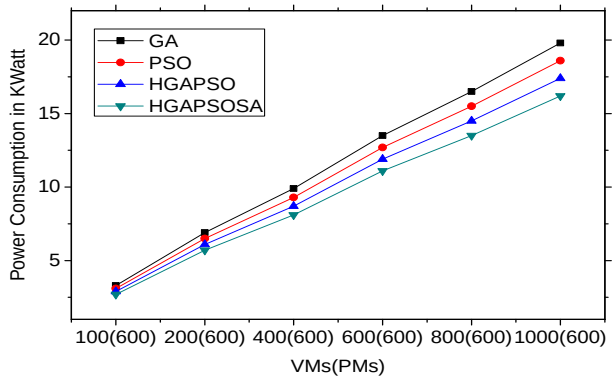

Figure 5.13: Power Consumption at the Constant Data Center.

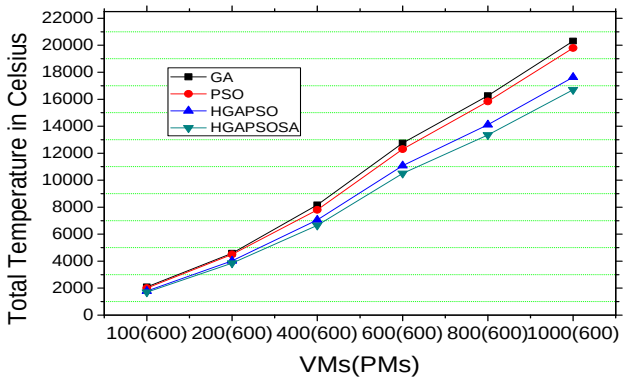

Figure 5.14: Temperature of the Constant Data Center.

the data center by taking 1024 PMs at the data center. Further, we generated a different number of VMs requests for different users in discrete time intervals. The requests arrive from different users at the cloud services provider in the form of small-VM, medium-VM, and large-VM. In the proposed work, we considered users requested VMs between 1 and 100 at the data center. We set a time duration or life time of all the users requested VMs from 30 minutes to 200 minutes.

The proposed HGAPSOSA algorithm is applied after arrival of each users request at the data center. Further, for switching-off the underutilized PMs, we removed the time expired VMs from the data center, and then the VMs are migrated from underutilized PM to energy efficient PM using First-Fit approximation policy. Hence, during the VM migration, the migration process required extra overhead in terms of additional power consumption. Therefore, we used constant migration cost in terms of power consumption during migration time for each type of VMs, such as 10 watts (small), 20 watts (medium), 30 watts (large), and 40 watts (x.large) in our experiment.

We considered total 100 users to conduct the experiment for VM allocation and migration at the data center by assigning a user-id to each user. The details of number of VMs requested by each user are shown in Figure 5.15. Further, the time duration or life time of VMs requested by users is shown in Figure 5.16. The life time of the VMs is calculated by subtracting the allocation time from the destroy time.

The number of PMs (of different types) are used for the allocation of users requested VMs at the data center as shown in Figure 5.17. Further, in our proposed HGAPSOSA algorithm, number of PMs used on each time instances at the data center is less as compared to the other VM allocation algorithms. The reason behind in using less number of PMs for the allocation VMs is due to optimal allocation by our proposed HGAPSOSA in terms of resources utilization. Hence, this will result in low resources wastage and thus selects less number of PMs for the allocation of VMs at the data center.

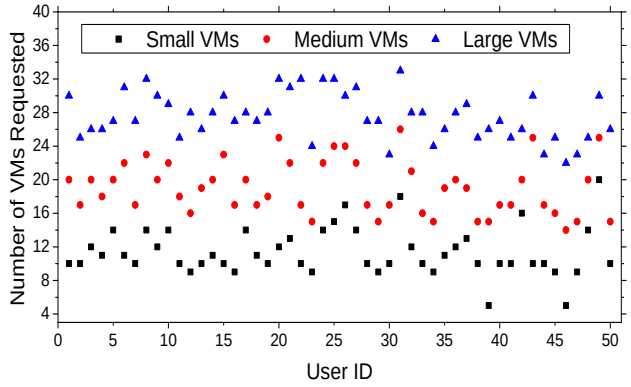

Figure 5.15: Users Requested VMs at the Data Center.

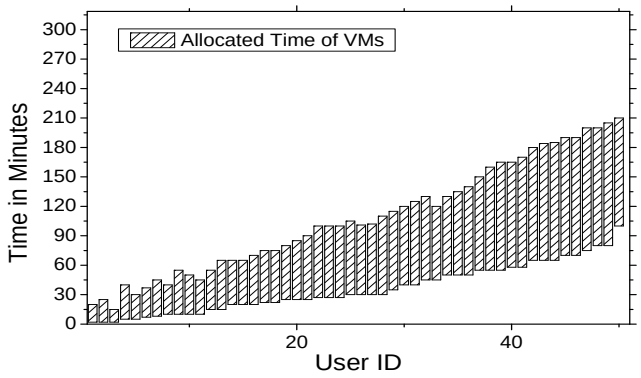

Figure 5.16: Time Duration of Users Requested VMs at the Data Center.

The energy consumption, and CPU utilization of the data center during different periods of time instances are shown in Figure 5.18 and Figure 5.19, respectively. The energy consumption of the data center is low in the case of proposed HGAPSOSA algorithm when compared to that of the other VM allocation techniques. This is due to the less number of energy efficient PMs used by our proposed HGAPSOSA algorithm, and migration of VMs from the underutilized PM to the energy efficient PM. In the beginning, the energy graph is going upward because of the continuous arrival of users' requests at the data center as shown in Figure 5.18. Further, after 80 minutes, no users' requests are generated resulting in no new switched-on PMs at the data center.

Further, the underutilized (idle) PMs will be switched-off using the proposed VM migration policy. This will result in the downward trend in the energy graph

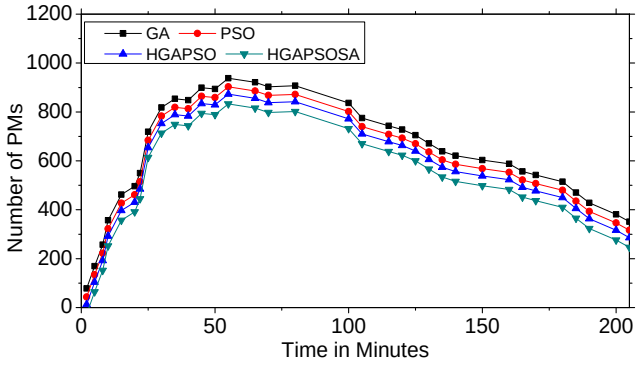

Figure 5.17: Switched-on PMs at the Data Center.

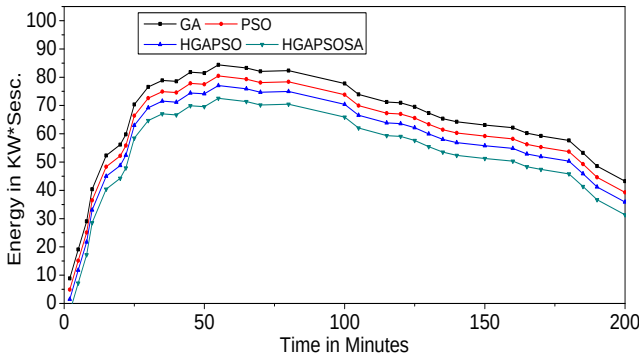

Figure 5.18: Energy Consumption at the Data Center.

as shown in Figure 5.18. The % of CPU utilization at the data center over different intervals of time is shown in Figure 5.19. The CPU utilization in the case of proposed HGAPSOSA algorithm is high when compared to that of GA, PSO, and HGAPSO, since Euclidean distance based multi-objective function gives an optimal number of PMs for the allocation of VMs at the data center. Thus, resulting in less wastage of resources at the cloud data center in the case of HGAPSOSA.

Further, to check the scalability of the proposed HGAPSOSA algorithm for a large data center, we conducted the experiment on the HP Compaq LE1902X machine (i3 processor with 2 GB RAM). The running time of the proposed HGAPSOSA algorithm is dependent on the crossover operation, PSO, and simulated annealing based modified mutation operations. Hence, less number of iterations are required to converge our proposed HGAPSOSA algorithm. The

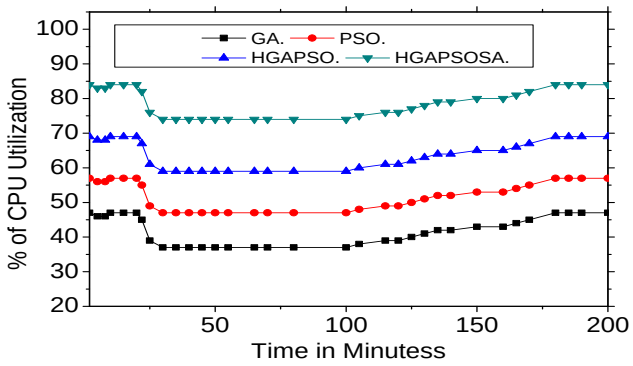

Figure 5.19: Resources Utilization at the Data Center.

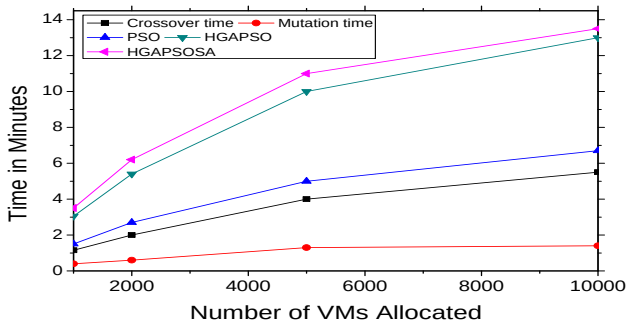

Figure 5.20: Execution Time of VM Allocation Algorithms.

execution time of the proposed HGAPSOSA algorithm in terms of GA, PSO, and SA operations is shown in Figure 5.20. The execution time of the proposed HGAPSOSA is calculated by taking different number of VMs in between [1000, 10000] at the cloud data center. The execution time of HGAPSOSA is slightly higher when compared to that of HGAPSO, but due to the better performance in terms of reducing energy consumption, resources wastage, and thermal temperature, the execution time will be manageable for the large data center. Further, the execution time of the proposed HGAPSOSA algorithm is approximately 17 minutes while taking 10000 VMs.

Time Complexity Analysis: The time complexity of proposed HGAPSOSA technique is based on GA, PSO, and SA. Let us consider k, m, n, s, and p are the iteration size, individual size, number of VMs, generation size, crossover point respectively. Hence, the time complexity of the proposed HGAPSOSA=O(O(Individual generation)+ Total iterations*(O(Fitness calculation*Size

of generation)+
O(Crossover*Generation Size)+O(Mutation*Generation Size))+ O(PSO)+O(SA). Thus, total time complexity of HGAPSOSA=O(O($n * s$)+O(k*(O(m*s)+O(n*m*s−p*m*s)+O(s))) + O(k*(O(s*m)+O(s*m)+O(s)+ O(s)+ O($n * m * s$))+O($mlogm$)) which is equal to O($n * m * s * k$+$mlogm$).

5.3 Summary

This chapter discussed the proposed energy efficient thermal aware VM allocation and migration policy using HGAPSOSA at the cloud data center. The proposed HGAPSOSA algorithm is the hybrid combination of GA, PSO, and, SA. In the proposed HGAPSOSA algorithm we improved the performance of HGAPSO algorithm by modifying mutation operation using SA. The experimental results demonstrated that proposed HGAPSOSA algorithm consumed (19%, 13%, and 5%) less energy over GA, PSO, and HGAPSO respectively.

Further, the resources utilization by proposed HGAPSOSA was (33%, 21%, and 6%) more when compared to that of GA, PSO, and HGAPSO respectively at the cloud data center. The average temperature of the data center in the case of HGAPSOSA was reduced by (10 degree, 6 degree, 3 degree) when compared to that of GA, PSO, and HGAPSO respectively. Hence, multi-objective based VM allocation technique is more efficient when compared to the existing state-of-the-art VM allocation algorithms. The proposed VM allocation algorithm takes about 3.10 minutes to allocate 1000 number of VMs at the cloud data center. Hence, these proposed techniques are useful for reducing the energy consumption, thermal temperature, and resources wastage at the cloud data center.

But the key limitation of the proposed work is that we did not consider the network elements power consumption at the cloud data center. Hence, to resolve this issue, we propose a branch-and-bound based exact algorithm for VM allocation in network-aware cloud data center environment. Further, we also propose a VM migration policy in network-aware cloud data center. The detailed description of proposed branch-and-bound based exact algorithm, and VM migration policy in network-aware cloud data center are discussed in Chapter 6.

Chapter 6

Energy Efficient Network-Aware Resource Management Using Exact Algorithm

Network-aware VM allocation and migration at the cloud data center are the challenging and important problems of energy efficient cloud computing. In Chapter 3, and Chapter 4, we discussed multi-objective (minimizing energy consumption, and resources wastage) VM allocation problem using HGACSO, and HGAPSO algorithms respectively. In Chapter 5, we discussed the minimization of thermal temperature, energy consumption, and resources wastage of a cloud data center using HGAPSOSA algorithm. Further, in Chapter 3, Chapter 4, and Chapter 5, we allocate VMs to PMs in non-network aware cloud data center environment using hybrid bio-inspired algorithms: HGACSO, HGAPSO, and HGAPSOSA, respectively. Hence, there is a need to allocate the VMs to PMs in network-aware cloud data center environment.

Thus, in this chapter we discuss an energy efficient network-aware resources allocation in the form of virtual machines (VM) by finding the minimum number of energy efficient physical machines (PMs) at the cloud data center. We formulate this VM allocation problem as Integer Linear Programming (ILP) problem, then we allocate VMs to PMs using proposed branch-and-bound based exact algorithm. Further, to reduce the execution time of proposed branch-and-bound algorithm for VM allocation, we propose some dominance rules for reducing the space of the search tree. In addition, we investigate lower bounds for VM allocation problem in network-aware data center environment. Further, we propose network and SLA-aware VM migration algorithm to save the energy consumption by switching-off both idle PMs and switches at the cloud data center.

The research contributions towards developing an exact energy efficient and network-aware VM allocation and migration algorithm are as follows:

1. Development of a mathematical model for calculating the lower bounds to select the optimal number of PMs in network-aware data center.

2. Design and Develop energy efficient network-aware VM allocation using branch-and-bound based exact algorithm.

3. Design and Develop energy efficient network and SLA-aware VM migration policy using First-Fit approximation algorithm.

4. Evaluation the performance of proposed energy efficient network and SLA-aware VM allocation with migration policy at the data center.

6.1 Proposed Work

The network-aware VM allocation using branch-and-bound based exact algorithm and First-Fit approximation based VM migration policy are described below:

6.1.1 VM Allocation Using Branch-and-Bound Based Exact Algorithm

Before applying the proposed VM allocation and migration algorithms we need to know the power consumption model of (PM, networking-switch), and the network topology of a cloud data center.

A. Power Consumption and Network Topology of Data Center

The power consumption at the cloud data center is dependent on IT devices and non-IT devices (Air Conditioners, Motor Pumps, UPS, etc.). The PMs and networking devices consume 45% and 15%, respectively of the total power consumption at the data center (Xu et al. 2013). The reduction of power consumption by PMs, and networking devices will also decrease the heat dissipation resulting in reduction of power consumed by the cooling devices. In this chapter, we used the same power consumption model of a PM as discussed in Section 4.1. Hence, the power consumption of pm_k at time instant t is described by Eq. 6.1.

$$P_k^{pm}(t) = ([P_k^{max} - P_k^{min}])u + P_k^{idle} \tag{6.1}$$

Where, P_k^{max} is the maximum power consumption of pm_k, and P_k^{min} is the minimum power consumption of a pm_k; u is the utilization rate of a CPU ($0 \leq u \leq 1$).

Power consumption of a Network Switch: The power consumption of a networking device (switch) is mainly dependent on the flow of traffic through the device, and its hardware configuration (processor board, memory, cooling system etc.) (Xiong & Xu 2014). The reduction of switch's power consumption can be achieved by (Fang et al. 2013) : (i) Switching-off a network device port saves 0.5% of the total power consumption of a switch; (ii) Switching-off a line card saves maximum 30% of the total power; iii) Switching-off the port to a lower bandwidth (from 1Gbps to 100 Mbps) saves around 0.005% to 0.4% of the total power consumption of a switch. Hence, to save the power consumption of the switch, we need to switch-off line card or unused ports. Further, most of the switches are based on Top of Rack (ToR) switches with a single line card. Therefore, we can model the power consumption of a network switch at time instant t using the following Eq. 6.2.

$$P_s^{switch}(t) = P_s^{idle} + P_s^{port} * n(p) \tag{6.2}$$

Where, P_s^{idle} is the power consumed by the switch s in an idle condition (all ports disabled); P_s^{port} is the power consumption of one port of switch s; $n(p)$ is the number of active ports of switch s. CISCO single line card switch consumes 1 kW in full load condition, each of the ports consumes 5W (CISCO 2014).

Network Topology of a Data Center: Almost all the world-wide data centers use the Fat-tree architecture as a network topology. Figure 6.1 shows the Fat-tree network architecture of the data center. The Fat-tree architecture consists of three level of switches such as core switches, aggregation switches, access or ToR switches. Each of these switches and PMs are connected through other switches by a link. The power consumption of switches at the data center is dependent upon the number of switches which are in active mode (switched-on) condition and the working duration of the switches. Two different VMs which are allocated to different PMs in the same rack can communicate through ToR switch. For example, VM_1 allocated on pm_1 and VM_2 allocated on pm_2 communicate

through s1 switch. On the other hand, if two different VMs are allocated to two different racks then the communication among these VMs is dependent upon the bandwidth of links. For example, VM_1 allocated on pm_1 and VM_5 allocated on pm_5 are communicated through path s1-s9-s17-s11-s3.

The inter-connected racks of servers using a Fat-tree data center topology consist k array three layer topology (edge, aggregation, core). Each pod (rack) consists of $((k/2)^2)$ number of servers and 2 layers having $k/2$ number of switches and each switch consists of k number of ports. Each edge switch is connected to $k/2$ servers and $k/2$ aggregation switches. Further, each of aggregation switches are connected to $k/2$ edges and $k/2$ core switches. $(k/2)^2$ core switches (each of the core switches) are connected to k pods. Figure 6.1 shows the data center network architecture of $k = 4$ port switches.

To save the energy consumption at the cloud data center, we consolidate the VMs to minimum number of energy efficient PMs and switches. Further, the migration of VMs from underutilized PM to energy efficient PM is carried out in such a way that both idle PMs and idle switches will be switched-off. For example, migration of VM_7 from pm_5 to pm_1 is done by switching-off idle PM_5 and idle switch s3 in the data center.

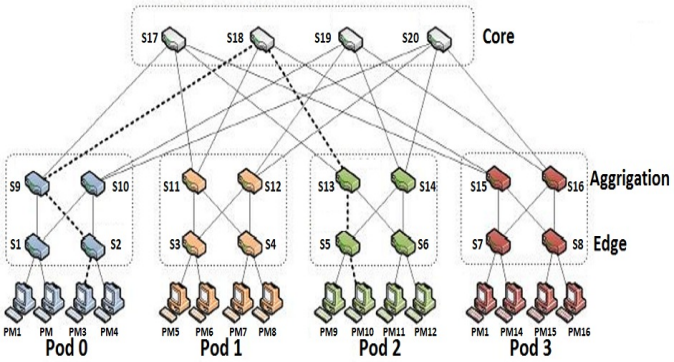

Figure 6.1: Fat-tree Architecture of Data Center (Son et al. 2017).

B. ILP-based VM Allocation Problem Formulation

Before applying the branch-and-bound based exact algorithm for VM allocation problem, we need to design a mathematical model of VM allocation problem.

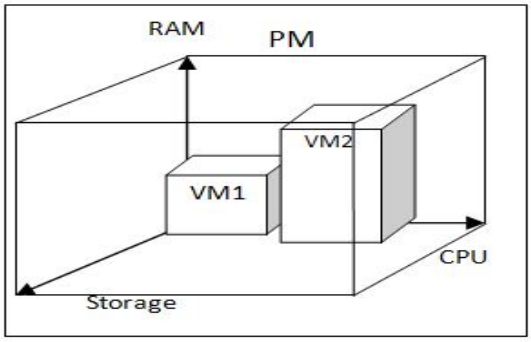

Figure 6.2: Allocation of VMs to a PM.

Hence, the Integer Linear Programming (ILP) based mathematical model for the VM allocation problem is described as follows:

ILP is a mathematical optimization technique consists of some or all variables restricted to integer values (Pisinger & Sigurd 2005). Figure 6.2 shows the multi-dimensional allocation of VMs to PMs, where two VMs are allocated to one PM. There are multiple resources (CPU, RAM, Storage, etc.) which are requested by the VMs to a PM. The remaining resources available on a PM after VMs allocation is dependent on the VM allocation policy. Further, allocation of future requested VMs to a PM, requires sufficient amount of resources at the PM. The blank area as shown in Figure 6.2 represents the remaining total capacity of CPU, RAM, and Storage. The two diagonals (RAM, Storage) have a lot of remaining capacity, but with very less remaining capacity of CPU, hence the PM is unable to allocate new requested VMs because of lack of CPU.

In our proposed work we consider multi-tenant cloud data center environment. Thus, a PM in the data center can provide service to number of users by creating a number of VMs at the cloud data center. Further, let us consider a data center with m different types of PMs and n number of PMs of each type, hence the total number of separate PMs in a data center is $m' = mn$. Thus, the data center has a set of PMs i.e. PM=$\{pm_1, pm_2, ...pm_{m'}\}$ and n' number of user's requested VMs, i.e. $VM = \{VM_1, VM_2, ..., VM_{n'}\}$. The resources capacity of a pm_k i.e. CPU, RAM, Storage, and number of processing elements are (pm_k^{MIPS}, pm_k^{RAM}, $pm_k^{Storage}$) respectively. Further the resources i.e. cpu, ram, number

of processing elements, and storage requested by a VM_i are (VM_i^{mips}, VM_i^{ram}, VM_i^{pe}, $VM_i^{storage}$) respectively.

To balance the allocation of resources among VMs for different PM dimensions, we need to minimize the resource wastage of a PM. The different resources of a PM are expressed by different units such as CPU in mips, RAM, and Storage in Gbps, Bandwidth in Hertz etc. Hence, the normalized remaining capacity (the ratio of remaining resources to total resources of a PM) is required to allocate VMs in a balanced way. The utilization of a pm_j at time instant t for d number of resources is defined as:

$$u_j^d(t) = \sum_{i=1}^{n'} \frac{f_{ij} VM_i^d}{pm_j^d} \quad d \in \{MIPS, RAM, Storage\}, \\ i \in \{1, 2, ...n'\}, j \in \{1, 2, ..m'\} \quad (6.3)$$

The utilization of a resource is in between 0 and 1. Further, we consider the utilization of data center on discrete time interval. Hence, the average resources utilization for a data center over time duration t_1 to t_2 is given by:

$$dc_r^u = \sum_{t=t_1}^{t_2} \frac{\sum_{k=1}^{m'} (u_k^{MIPS}(t) + u_k^{RAM}(t) + u_k^{Storage}(t))}{|d| \sum_{k=1}^{m'} z_k} \quad (6.4)$$

Where, f_{ik} is a binary variable with value of 1 when VM_i allocated to pm_k; z_k is a binary variable, the value of $z_k = 1$ when pm_k is used for the VM allocation otherwise its value is 0. Further, let us consider a data center with a set of networking switches S. Each PM in the data center is associated with a ToR switch in the rack. The relation function of pm_k and ToR switch $s \in S$ is defined by a function $r : k \leftarrow s$. Let us consider a Boolean variable y_s^t that represents the use of switch s at time slot t. The value of y_s^t=1 if switch is in use, otherwise its value is 0. Since, the switching-off core switches is not feasible for the data center due to continues arrival of VMs request at the cloud data center, hence we need to select the optimal number of PMs for the allocation of VMs. Thus, we keep the core switches always be in switched-on condition, and we only switch-off idle ToR switches at the cloud data center. The set of PMs within a given rack connected to ToR switch s is PM_s. Hence the number of active ports at switch s at time slot

t is expressed as $\sum_{k \in PM_s} z_k$. The power consumption of switch s at time instant t is described by Eq. 6.5.

$$P_s^{switch}(t) = P_s^{idle} + P_s^{port} \sum_{k \in PM_s} z_k \tag{6.5}$$

Where, P_s^{idle} is the power consumption of a switch in an idle condition (all ports are disabled); P_s^{port} is the power consumption of an enabled port.

Let us consider c be the customer and C be the set of customers in the data center at time instant t. Further, a set of VMs requested by customer c is (VM_c), and $VM_i, VM_j \in VM_c$ exist at the same time instance t in the data center. In this thesis, we consider the network communication between VMs of the same customer, then the allocation of bandwidth to VM is the maximum requirement i.e. $\max\{VM_i^{bw}, VM_j^{bw}\}$. The set of all the links is denoted by E and individual link is represented by e. In the proposed work, we are not focusing on any kind of interconnection network topology or routing protocol in the data center. We defined a matrix Z as a communication matrix that related to end point PM to links; for example, $z_{kk'}^e$ represents the pm_k and $pm_{k'}$ and it is communicated by networking link e. In the tree topology of the data center there exists only one path from one server to another server.

Let us consider c_e as the total capacity of link e. The bandwidth requirement of a link e (B_e) is described by Eq. 6.6.

$$B_e = \sum_{c \in C} \sum_{i,j \in VM_c} \sum_{k,k' \in PM} f_{ik} f_{j,k'} max\{VM_i^b, VM_j^b\} z_{kk'}^e,$$

$$Where \quad i \neq j, \quad and \quad k \neq k' \tag{6.6}$$

In our proposed VM allocation to PM problem, we assume all the VMs fit into at least one PM, otherwise no solution exists for the VM allocation problem. Hence, it is obvious that for the allocation of n' number of VMs, we can not use more than n' number of PMs of any type where $(n' \geq m')$. Thus, we can transform the VM allocation to the PM problem in a binary version where, we have $m' = mn$ separate PMs at the data center. Each user requested VMs is related with life time span VM_i^t represented by start time (t_i^s) and destruction time (t_i^d),

$VM_i^t = \{t_i^s, t_i^d\}$.

Further, the allocation of user requested VMs in network aware cloud data center environment required different types of resources such as (mips, ram, storage, and bandwidth). Hence, if we consider each requested resource as a dimension then allocation of VMs to PMs looks like a multi-dimensional bin packing problem. Thus, To formulate the multidimensional VM allocation to PM problem at the data center in binary form as an ILP model, we used the following decision variables: if VM_i is allocated left to VM_j then the binary variable l_{ij} is 1; if VM_j is allocated left to VM_i then the binary variable l_{ji} is 1; if VM_i is below to VM_j then b_{ij} is 1; if VM_j is below to VM_i then b_{ji} is 1; If VM_i is allocated to pm_k then the binary variable f_{ik} is 1; If pm_k is used for VM allocation then the binary variable z_k is 1 otherwise its value is 0; At the end, (x_i, y_i) are the lower left coordinates of VM_i. To ensure that no two VMs overlap in terms of dimensions, then the following inequality conditions must hold good:

$$l_{ij} + l_{ji} + b_{ij} + b_{ji} + (1 - f_{ik}) + (1 - f_{jk}) \geq 1, i < j, \\ i, j \in \{1, 2, ..n'\} \quad k \in \{1, 2, ...m'\} \quad (6.7)$$

If VM_i is located left to VM_j, then $l_{ij} = 1$; hence we have $x_i + VM_i^{mips} \leq x_j$. In general, we defined the following constraints defined by Eq. 6.8.

$$l_{ij} = 1 \rightarrow x_i + VM_i^{mips} \leq x_j, \quad b_{ij} = 1 \rightarrow y_i + VM_i^{ram} \leq y_j \quad (6.8)$$

No dimension of the VM must exceed that of the PM dimensions, then $0 \leq x_i \leq pm_k^{MIPS} - VM_i^{mips}$ and $0 \leq y_i \leq pm_k^{RAM} - VM_i^{ram}$. Further if VM_i has to be placed in some pm_k then $\sum_{k=1}^{m'} f_{ik} \geq 1$. pm_k is used if at least one VM is allocated in pm_k, then $\sum_{i \in 1,2,..,n'} f_{ik} > 0 \rightarrow z_k = 1$.

Let us consider the following parameters of a PM: $pm_{max}^{MIPS} = max_{k \in \{1,2,...m'\}} pm_k^{MIPS}$, $pm_{max}^{RAM} = max_{K \in \{1,2,...m'\}} pm_k^{RAM}$, and $pm_{max}^{STORAGE} = max_{K \in \{1,2,...m'\}} pm_{max}^{Storage}$ be the maximum PM MIPS, RAM, and Storage respectively.

Since, the computational complexity of the ILP based VM allocation model becomes high as we consider more number of dimensions of the VM. Hence, in the

proposed VM allocation model we consider two dimensions of VMs such as (mips, and ram), and we did not take into account the storage requirement of VMs at the cloud data center.

Further, if PM is in idle condition then we switched-off this PM and takes the value of z_k=0. Hence, in both the conditions of PM (idle, not utilized) the value of z_k=0. Thus, to minimize the power consumption of networking switches and PM at the data center, using standard modeling techniques for ILP model, the multi-dimensional VM allocation to PM problem in network-aware cloud data center environment can be formulated as follows:

$$f = min\{\sum_{k \in PM} z_k P_k^{pm}(t) + \sum_{s \in S} y_s P_s^{switch}(t)\},$$
$$Where k \in \{1, 2, ...m'\}, s \in S \quad (6.9)$$

Subject to the following constraints:

$$l_{ij} + l_{ji} + b_{ij} + b_{ji} + (1 - f_{ik}) + (1 - f_{jk}) \geq 1, \quad i, j \in \{1, 2, .., n'\}$$
$$k \in \{1, 2, .., m'\}, i < j \quad (6.10)$$

$$x_i - x_j + pm_{max}^{MIPS} l_{ij} \leq pm_{max}^{MIPS} - VM_i^{mips},$$
$$i, j \in \{1, 2, .., n'\}, k \in \{1, 2, ..m'\} \quad (6.11)$$

$$y_i - y_j + pm_{max}^{RAM} b_{ij} \leq pm_{max}^{RAM} - VM_i^{ram},$$
$$i, j \in \{1, 2, .., n'\}, k \in \{1, 2, ..m'\} \quad (6.12)$$

$$x_i \leq pm_k^{MIPS} - VM_i^{mips} + (1 - f_{ik}) pm_{max}^{MIPS},$$
$$i, j \in \{1, 2, .., n'\}, k \in \{1, 2, ..m'\} \quad (6.13)$$

$$y_i \leq pm_k^{RAM} - VM_i^{ram} + (1 - f_{ik}) pm_{max}^{RAM},$$
$$i, j \in \{1, 2, ..n'\} k \in \{1, 2, .., m'\} \quad (6.14)$$

$$\sum_{k=1}^{m'} f_{ik} \geq 1, \quad i \in \{1, 2, .., n'\}, k \in \{1, 2, ..m'\} \tag{6.15}$$

$$f_{ik} \leq z_k, \quad i \in \{1, 2, .., n'\}, k \in \{1, 2, ..m'\} \tag{6.16}$$

$$B_e \leq c_e \tag{6.17}$$

Where, Eq. 6.9 describes the objective function for VM allocation problem to save power consumption at the data center. Eqs. 6.10 to 6.17 describe the constraints satisfaction in the form of different dimensions (mips, ram, and bandwidth) of the PMs and switches. The binary variables $l_{ij}, b_{ij} \in \{1, 0\}$; $f_{ik} \in \{0, 1\}$; $(x_i, y_i) \geq 0$; $z_k \in \{0, 1\}$. Since, the proposed ILP based multi-objective VM allocation model allocate the VMs in non-overlapping manner in the network-aware cloud data center environment, hence we can allocate the exact amount of resources as requested by the VMs. Further, our proposed ILP based VM allocation model consists of number of binary and continues variables. Hence, the computational time complexity shows that the ILP model is very difficult to solve the problem of VM allocation. Hence, lower bounds are required in different scenario in order to reduce the time complexity of the problem.

C. Lower Bounds for VM Allocation

The lower bound R_L for VM allocation to PM problem is given by the worst-case performance ratio.

$$R_L = inf_{I \in p}\{\frac{L(I)}{OPT(I)}\} \tag{6.18}$$

Where, p is the set of all instances, $L(I)$ is the objective value given by Eq. 6.10 for instance I, $OPT(I)$ is the optimal objective value for instance I. Since, we need the minimum number of power efficient PMs for the allocation of of VMs, hence we used the power consumption model of PM as described by Eq. 6.1. Further, to make the lower bound for the multi-dimensional VM allocation problem, we consider the power dependent ceiling function $\lceil . \rceil_P$. The power dependent ceiling function gives the minimum number of power efficient PMs for the allocation of VMs, and the proposed lower bound is fully dependent on the power consumption model of PM.

We can make the lower bounds from the one dimensional VM allocation problem by relaxing some of the constraints in the proposed multi-dimensional VM allocation problem. In the one dimensional VM allocation problem, let us assume n' VM with mips dimension (VM_i^{mips}) for $i = \{1, 2, ...n'\}$, and m types of PM at the data center with sizes pm_k^{MIPS}, and the power consumption P_k for $k = \{1, 2, ..m\}$. The one dimensional VM allocation problem with arbitrary power consumption of PM in the binary form can be formulated as follows:

$$min \sum_{k=1}^{m'} P_k z_k \tag{6.19}$$

Subject to the following constraints:

$$\sum_{k=1}^{m'} f_{ik} = 1, \quad i = \{1, 2, ..n'\} \tag{6.20}$$

$$\sum_{i=1}^{n'} VM_i^{mips} f_{ik} \leq pm_k^{MIPS} z_k, \quad k = \{1, 2, ..m'\}$$

$$f_{ik} \in \{0, 1\}, \quad i \in \{1, 2, ..n'\}, k \in \{1, 2, ..m'\} \tag{6.21}$$

Where, $m' = mn$ is an upper bound on the total number of PMs used for the VM allocation. In the one dimensional VM allocation to the PM problem, since all the VMs have to be allocated to some PMs at the data center, hence our proposed solution must contain a PM that can hold the largest VM by satisfying the following valid lower bound:

$$L_{mips} = \lceil min_{k=\{1,2,..m\}} \{P_k : pm_k^{MIPS} \geq VM_i^{mips}, \quad i = 1, 2, ..n'\} \rceil_P \tag{6.22}$$

In the case of multi-dimensional VM allocation problem, we consider the sets B_i for $i \in \{1, 2, ..n'\}$ with a set of PMs types that can contain the allocated VM_i such as

$$B_i = \{k \in \{1, 2, .., m\} : pm_k^{MIPS} \geq VM_i^{mips} \wedge pm_k^{RAM} \geq$$

$$VM_i^{ram} \wedge pm_k^{Storage} \geq VM_i^{storage}\} \tag{6.23}$$

Any feasible solution for the allocation of VMs to PMs should satisfy at least one PM from each of the sets of PMs types B_i. Let $P_{k_i} = min_{k \in B_i} P_k$ be the power consumption of the least power of a PM in B_i. Then the lower bound for VM allocation problem is given by

$$L_{MIPS} = \lceil maxP_{k_i} \rceil_P \quad where \quad i \in \{1, 2, ...n'\} \tag{6.24}$$

This lower bound value can be calculated in polynomial time $O(n'm)$. $L_{mips \wedge ram \wedge storage} = \lceil max_{i=1,2,..n} P_{k_i} \rceil$ is a lower bound on the solution value, and this solution can be evaluated in polynomial time $O(n'm)$. The continuous lower bound in one dimensional case with PM power consumption is equal to size of PM. Hence, $L_c^e = \sum_{i=1}^{n'} VM_i^{mips}$ is a lower bound on the optimal value of VM allocation problem. In the case of multi-dimensional VM allocation problem, we consider different dimensions of VM space. Thus, if the power consumption of a PM is dependant on the MIPS of a PM, then $L_c^e = \sum_{i=1}^{n} VM_i^{mips}$ is a lower bound on the optimal value of VM allocation in one-dimensional case. Hence, in the case of multi-dimensional VM allocation problem, the lower bound on the optimal value is described as $L_{mc}^e = \sum_{i=1}^{n'} VM_i^{mips} VM_i^{ram} VM_i^{storage}$. Thus, for the general case with arbitrary power consumption of PM, the continuous lower bound for multi-dimensional VM allocation problem is described as follows:

$$L_{mc} = \lceil \frac{P_{k_0}}{pm_{k_0}^{MIPS} pm_{k_0}^{RAM} pm_{k_0}^{Storage}} \sum_{i=1}^{n'} VM_i^{mips} VM_i^{ram} VM_i^{storage} \rceil_P \tag{6.25}$$

The worst-case possibility of VM allocation problem is defined when all the VMs are not fit into the cheapest PM. Hence, in this case, we can calculate a better bound by modifying the continuous lower bound (L_{mc}) so that the power consumption of allocating each of the VM is a fraction of the power consumption of a PM in which the VMs fits. Let l_j be the PM with cheapest power to VM space ratio of $P_{l_i}/pm_{l_j}^{MIPS} pm_{l_j}^{RAM} pm_{l_j}^{Storage}$ in which VM_i fits. Thus, the modified continuous lower bound (L'_{mc}) is as follows:

$$L'_{mc} = \sum_{i=1}^{n} \frac{VM_i^{mips} VM_i^{ram} VM_i^{storage} P_{l_i}}{pm_{l_i}^{MIPS} pm_{l_i}^{RAM} pm_{l_i}^{Storage}} \tag{6.26}$$

Hence, the maximum value of the lower bound, i.e. $\max\{L_{mc}, L'_{mc}\}$ is also a valid lower bound on the optimal solution for the muli-dimensional VM allocation to PM problem in non network-aware data center environment.

$$L_{mv} = \lceil max\{L_{mc}, L'_{mc}\}\rceil$$
$$= \lceil max\{\sum_{i=1}^{n} \frac{VM_i^{mips} VM_i^{ram} VM_i^{storage}}{pm_{l_j} pm_{l_j}^{MIPS} pm_{l_j}^{RAM} pm_{l_j}^{Storage}}, P_{k_1}, .., P_{k_n}\}\rceil_P \quad (6.27)$$

Thus, the non-network aware lower bounds are used to select the optimal number of PMs for allocation of VMs at the data center using branch-and-bound based exact algorithm. But in real network-aware environment, we need to select the optimal number of both PMs and switches for the allocation of VMs at the data center. Hence, the VMs requested bandwidth is also taken into account. Further, the bandwidth of a link e is shared by VMs which are allocated to different PMs, and these PMs are connected by switch s through link e. Hence, to calculate the lower bound for network-aware one dimensional VM allocation problem, let B_i^s and B_j^s be the sets of PMs types that can contain the VM_i and VM_j respectively. Further, these sets of PMs are connected to switch $s \in S$ by link e in the case of one-dimensional VM allocation problem and contain the allocated VM_i and VM_j such as:

$$B_i^s \cup B_j^s = \{k, k' \in \{1, 2, ..m\} : \{pm_k^{MIPS} \geq VM_i^{mips}$$
$$\wedge \{pm_{k'}^{MIPS} \geq VM_i^{mips} \wedge max\{VM_i^{bw}, VM_j^{bw}\}\} \quad (6.28)$$

Hence, the valid lower bound for multi-dimensional network-aware VM allocation to PM problem $(L_m^n)$is given by

$$L_m^n = \lceil minP_{k_i} \cup minP_{k'_j}\rceil_P \quad where \quad i, j \in \{1, 2, ...n'\} \quad (6.29)$$

$$B_i^s \cup B_j^s = \{k, k' \in \{1, 2, ..m\} : \{pm_k^{MIPS} \geq VM_i^{mips} \wedge$$
$$pm_k^{RAM} \geq VM_i^{ram} \wedge pm_k^{Storage} \geq VM_i^{storage}\} \wedge \{pm_{k'}^{MIPS} \geq$$
$$VM_i^{mips} \wedge pm_{k'}^{RAM} \geq VM_i^{ram} \wedge pm_{k'}^{Storage} \geq VM_i^{storage}\} \wedge$$
$$max\{VM_i^{bw}, VM_j^{bw}\}\} \quad (6.30)$$

Any feasible solution for network-aware VM allocation to PM should satisfy at least one PM from the set of type B_i^s and another PM from the set of B_j^s respectively. Let P_k^s and $P_{k'}^s$ be the power consumptions of the least power of pm_k and $pm_{k'}$ in B_i^s and B_j^s respectively. Then, the valid lower bound for the multi-dimensional network-aware VM allocation problem is given by:

$$L_m^n = \lceil maxP_{k_i} \cup maxP_{k'_j} \rceil_P \quad where \quad i,j \in \{1,2,...n'\} \quad (6.31)$$

Thus, these valid lower bounds are used to select both optimal number of PMs and switches for allocation of VMs in network-aware cloud data center environment using branch-and-bound based exact algorithm.

D. Branch-and-Bound Based Exact Algorithm

The branch-and-bound based exact algorithm for solving the VM allocation problem at the cloud data center is described as follows.

1. Notations: To calculate the optimal allocation of VMs to PMs, the following data is associated with each node N at the level l of the search tree such as: a lower bound $LB(N)$, an upper bound $UB(N)$, and the partial solution.

Definition of symbols for an exact algorithm are as follows:
$S'(N)$ is a subset of unallocated user requested VMs such as $(|S'(N)| = n' - l)$; $v_k(N)$ is the number of PMs of type k that have been already used at the data center for the allocation of VMs; $v(N)$ is the total number of PMs of all types that have been already used for the allocation of VMs at the data center such as $(v(N) = \sum_{k=1}^{m} v_k(N))$; $p(N)$ is the sum of power consumption of all the PMs already used for the VM allocation such as $(p(N) = \sum_{k=1}^{m} v_k(N)p_k)$.

In the proposed exact algorithm for VM allocation, the branching strategy for calculating the optimal solution in the search tree is implemented similar to that of (Martello & Toth 1990), (Haouari & Serairi 2011). Further, the root node N_0 in the search tree contains no solution (empty solution) and other nodes at different level $l \geq 1$ consists of a partial solution (a subset of VM allocated to PM) at the cloud data center. A child node in the search tree is created by allocating the largest VM in the VM list to an already initialized PM (used PM) or to an empty PM (unused PM) of type k. Therefore, a node N in the search tree may consist

of maximum number of child nodes equal to $m + v(N)$. A child node N^+ of a given node N in the search tree space corresponding to allocating VM_i into a PM of type k is created if and only if the following conditions are satisfied:
(i) (C_1) $v(N) \leq v^u$ (v^u is an upper bound for the total number of PMs that are used in finding an optimal solution for the VM allocation at the data center);
(ii) (C_2) $v_k(N) < v_k^u$ (v_k^u is an upper bound for the number of PMs of type k that are used in finding an optimal solution for the VM allocation);
(iii) (C_3) $P(N) + P_K < ub(N)$;
(vi) (C_4) $d_i \leq D_k$, $\forall d \in \{mips, ram, storage\}$ (or $d_i \leq D_K$ if k is initialized PM;

Hence, if the above defined conditions are met, then the lower bound value $lb(N^+)$ is computed. Further, if $lb(N^+) \geq ub(N)$ then the child node is pruned from the search tree.

2. An Improved Heuristic: Let us consider a node N of the search tree for which a lower bound value $LB(N)$ can be calculated after satisfying one of the previously described relaxation conditions. Let n_k (k=1,2,...,m) be the total number of PMs of type k (pm_k) that have been used in the optimal solution for VM allocation. Then, we use a heuristic approach for identifying the possibility to pack all the VMs from $S'(N)$ into these PMs. The overall idea of the heuristic approach is similar to that of the sub-set sum heuristic approach. The PMs at the data center are first sorted in non-decreasing order on the basis of their power consumption, then the PMs are filled one by one by allocating a subset of unallocated VMs from the VM list in each PM. The allocation of a subset of VMs in each of the PM should satisfy the resources constraints as defined in the proposed ILP model for VM allocation discussed in Section 3. If no extra PM is required for allocating the VM, then one of the feasible solution for the VM allocation problem with a cost equal to $LB(N)$ will be determined. Hence, a child node N is pruned from the search tree and the present solution is accordingly updated.

3. Dominance Rules: To reduce the power consumption, and size of the search tree, we defined the power aware dominance rules in order to reduce the power consumption, and size of a search tree and thereby finding an optimal so-

lution for VM allocation problem. Further, the dominance rules fully dependent on the power consumption a PM as described by Eq. 6.1.

Rule 1 Let us consider $pm_i^{MIPS} + pm_k^{MIPS} \leq pm_h^{MIPS}$ and $P_i + P_k \geq P_h$. Then PM of type i, and k are mutually exclusive from each other. Thus, an optimal solution for the VM allocation contains the value of $x_i = 0$ or $x_k = 0$. Further, if the node N of a search tree uses a PM of type i (pm_i), then we will create the child node in the search tree corresponding to initializing a type k PM (pm_k).

Rule 2 Let us consider $2pm_i^{MIPS} \leq pm_k^{MIPS}$ and $2P_i \geq P_k$ for $i < k$. Then, an optimal solution for VM allocation should include at most one PM of type i.

Rule 3 Let us consider $pm_i^{MIPS} + pm_k^{MIPS} \geq pm_h^{MIPS}$ and $P_i + P_k \leq P_h$ for $i < k < h$. Then, for an optimal solution of VM allocation, a PM of type h should not include an VM_j such that $pm_h^{MIPS} - pm_k^{MIPS} \leq VM_j^{mips} \leq pm_i^{MIPS}$ or $pm_h^{MIPS} - pm_i^{MIPS} \leq VM_j^{mips} \leq pm_k^{MIPS}$. Further, if a PM of type h contains a VM such that there is a possibility to get a better solution for VM allocation by reallocating the VM of this PM into two different PM of type i and k respectively.

Rule 4 The rule 4 is the special case of a Rule 3. Let us consider $2pm_i^{MIPS} \geq pm_h^{MIPS}$ and $2P_i \leq P_h$ for $i < h$. Then, for an optimal solution for VM allocation, a PM of type h should not contain VM_j such that $pm_h^{MIPS} - pm_i^{MIPS} \leq VM_j^{mips} \leq pm_i^{MIPS}$.

Except these aforementioned rules, three symmetric breaking rules are also defined for reducing the space of search tree.

Rule 5 If $VM_i^{mips} = VM_j^{mips}$ and $i < j$ then an unallocated VM_j is allocated to a PM if an index value is not smaller than the index value of a PM, where VM_i is allocated previously. (The index value to each of the PM is assigned on the basis of their order in which they are initialized).

Rule 6 If two already initialized PMs in the data center consist the same remaining resource capacity, then an unallocated VM_j is allocated to the PM that consists of the smallest index value.

Rule 7 If the remaining capacity of the resources for an used PM is equal to the resource capacity of an empty PM, then in this case, an unallocated VM_j is

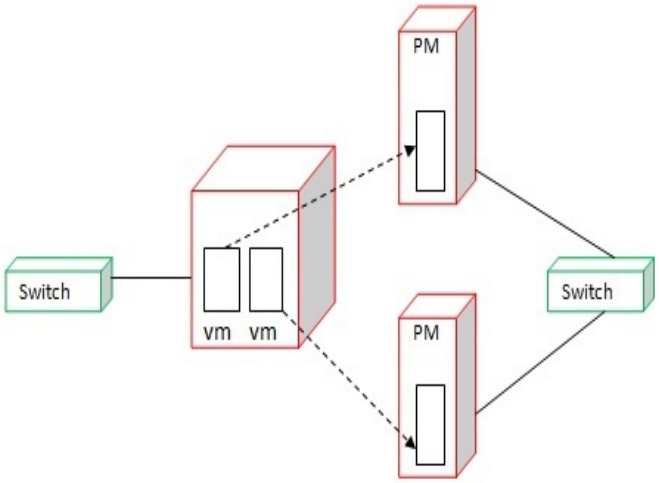

Figure 6.3: VM Migration at the Cloud Data Center.

allocated to the previously used PM.

6.1.2 VM Migration Policy

The previously allocated VM in the data center will be destroyed from the PM at the data center since their allotted time is over. The destruction of VMs from the PMs thus creates the underutilized/idle PMs at the data center. Therefore, it gives the opportunity to the data center manager to reallocate the VMs to PMs by using the VM migration policy. Hence, with the help of VM migration policy, we can migrate the VMs from the underutilized PM, and thus we can re-allocate the VMs onto already initialized energy efficient PMs.

Further, switching-off the idle PM will result in saving unwanted energy consumed by underutilized/idle PMs at the cloud data center. Hence, the objective function for VM migration is defined in terms of migrating the VM from one PM (source node) to the another PM (destination node) so that allocating the maximum number of VMs at the destination node within their physical resources capacity. Further, we can switch-off maximum number of idle PMs at the data center. Figure 6.3 shows the migration of VM from underutilized PM to energy efficient PM. Consider a data center consists of already used PM (PM'), $|PM'| < m'$. Thus, the objective function for VM migration in a cloud data center is defined as:

$$max \quad y = (\sum_{k} P_k^{min} z_k + \sum_{s \in S} P_s^{switch} z_k^s)$$
$$- (\sum_{k} (\sum_{k'} \sum_{i \in n_k} P_i' z_{kk'i} + \sum_{s \in S} P_s^{switch} z_{k'}^s)).$$
$$i \in \{1, 2, ..n'\}, k, k' \in \{1, 2, ..m'\} \quad (6.32)$$

Where, z_k=1 if pm_k is used for allocation of VM, otherwise $z_k = 0$; The binary variable $z_{kk'i}$=1, if VM_i migrates from pm_k to $pm_{k'}$ otherwise $z_{kk'i}$=0; z_k^s is one if $pm_{k'}$ is connected to switch s otherwise it is 0; P_i' is the power consumption of VM_i; n_k is a set of VMs to be migrated from pm_k to $pm_{k'}$.

Further, the $pm_k, pm_{k'} \in PM'$ and these PMs are connected to switch s by link e, and the link capacity of link e is (c_e). Hence the VM migration constraints are defined as:

$$\sum_{k'} \sum_{i \in n_k} VM_i^d z_{kk'} \leq pm_{k'}^d (1 - z_k) \forall d \in \{mips, ram, storage\},$$
$$i \in \{1, 2, ..n'\} \quad k, k' \in \{1, 2, ..m'\} \quad (6.33)$$

$$VM_i^{bw} \leq c_e, \quad i \in \{1, 2, ...n'\}, \quad e \in \{E\} \quad (6.34)$$

$$\sum_{k'} \sum_{i \in n_k} z_{kk'i} = n_k z_k \quad \forall k, k' \in \{1, 2..m'\}, \quad i \in \{1, 2, ..n'\}, \quad k \neq k' \quad (6.35)$$

The constraints satisfaction for resources (MIPS, RAM, Storage) are defined by Eq. 6.33. The bandwidth requirement constraints are defined by Eq. 6.34. Eq. 6.35 defines the migration of a set of VMs (n_k) from the source node pm_k to the destination node $pm_{k'}$. In the proposed VM migration policy, we need to decide when to migrate the VMs and where to migrate the VMs at the data center. Hence, to decide when to migrate the VMs from the PM, we set the lower threshold value (u_l) of all the PMs at the data center. The value of (u_l) will play the critical role not only in avoiding the Service Level Agreement (SLA) violation but also in minimizing the energy consumption at the cloud data center. Hence, the migration of all the VMs from the source node pm_k will take place if the

following conditions are satisfied :

(i) The present CPU utilization of pm_k should be less than the defined lower threshold value u_l of pm_k be $(u_k^{cpu} < u_l)$.

(ii) The migration cost for all the VMs allocated to pm_k should be less than the rent of VM as decided by the cloud service provider during the left over time t_l.

Hence the mathematical expression for a set of VMs (n_k) for migration from the source node pm_k is defined by

$$n_k = \begin{cases} If \quad u_k^{cpu} < u_l \quad and \quad \sum_{i \in n_k} c_i^{mig} < \sum_{i \in n_k} t_l^i * r, \\ r \in \{\alpha, \beta, \gamma, \delta\}, VMmigration \\ Otherwise \quad no \quad VM \quad migration. \end{cases} \tag{6.36}$$

Where α, β, γ, δ are the rent of small, medium, large, and x.large VMs for the unit time duration respectively.

The left over time (t_l^i) of a VM_i is the difference between total life time (t_t^i) of VM_i and current time (t_c^i) of VM_i such as $t_l^i = t_t^i - t_c^i$. The migration cost (c_{mig}^t) for a VM_i to be migrated is calculated by two factors: SLA penalty during shut down time (s_i^t), and migration overhead cost (m_i^o). Hence, the total migration cost is defined by Eq. 6.37.

$$c_i^{mig} = s_i^t * r + m_i^o \tag{6.37}$$

Algorithm 6.1 describes steps of branch-and-bound based exact algorithm for VM allocation. Where *Opt* is the optimal solution in the form of VM allocated to PM. Step 1 describes the root node of the search tree which contains the empty solution. Step 2 describes the creation of child node of the search tree. Steps 3 to 10 describe the allocation of VMs to PMs at the child node if constraints and lower bound conditions are satisfied by the PM at the child node. Here, the child node gives the partial solution in the form of a few VM allocated to the PM. Step 11 describes the updating of PM dimensions at the child node after allocating the VMs to PMs. The deletion of VMs from the user requested list after allocation to the PM is described in Step 12. Steps 13 to 15 describe the mutual exclusive

Algorithm 6.1 Branch-and-Bound Based Exact Algorithm

Input: VM list (VM), PM list (PM), Link set (E)

Output: Opt=Optimal solution

N^0=empty (Root node)

Create child node N^+

if $(pm(N) < v^u \quad \& \quad pm_k(N) < v_k^u \quad \& \quad P(N) + P_k < ub(N))$ **then**

  for Each d **do**

    if $(VM_i^d \leq pm_k^d \quad \& \quad VM_i^{bw} \leq c_e)$ **then**

      Compute lb(N^+)

if $(lb_(N^+) \geq ub(N))$ **then**

  pruned N^+

if $(lb(N^+) < ub(N))$ **then**

  Allocate $VM_i \quad to \quad pm_k$

  Update $pm_k^d = pm_k^d - VM_i^d$

  Remove VM_i from VM

if $(pm_i + pm_k \leq pm_h \quad \& \quad P_i + P_k \geq P_h)$, $i < k < h$ **then** , $i, k, h \in \{1, 2, ..m\}$

  pm_i and pm_k are mutually exclusive

  $z_i = 0 || z_k = 0, \quad z_i, z_k \in \{0, 1\}$

if In Node (N) pm_i, and pm_k are already initialized **then**

  Not create child node

if $(2pm_i \leq pm_h$ & $2P_i \geq P_h$, $i < h)$ **then**

  At most one $pm_i \in Opt$

if $(pm_i + pm_k \geq pm_h$ & $P_i + P_k \leq P_h$,$i < k < h)$, $\quad i, h \in \{1, 2, ..m\})$ **then**

  $pm_h \in opt$ not allocated VM_j

  $pm_h - pm_k \leq VM_j \leq pm_i$

  $pm_h - pm_k \leq VM_j \leq pm_i$ or $pm_h - pm_i \leq VM_j \leq pm_k$

if $(2pm_k \geq pm_h \quad \& \quad 2P_i \leq P_h, i < h, \quad i, h \in \{1, 2, ..m\})$ **then**

  $pm_h \in opt$ not included VM_j

  Allocate VM_j to higher index pm where VM_j is allocated.

if $(pm_i$,$pm_j \in V(N) \quad \& \quad pm_i^d = pm_j^d, \quad i, j \in \{1, 2, ..m'\})$ **then**

  VM_j is allocate to pm_i

if $(pm_i \in V(N) \quad \& \quad pm_j \notin V(N) \quad \& \quad pm_i^d = pm_j^d)$ **then**

  Allocate VM_j to pm_i

Algorithm 6.2 VM Migration Policy using First-Fit

Input: $PM' \subset PM$

Output: Migration of VM

1: VM'=empty
2: **for** $k \leftarrow m'$ **do**
3: **if** $u_k^{CPU} < u_l \quad \& \quad c_{mig}^t * n_k < n_k * r * t_l$ **then**
4: Put n_k into VM'
5: Sort PM in non decreasing order using p_j
6: **for** $i \leftarrow 1 \quad to \quad m'$ **do**
7: **if** $(k \neq k') \quad and \quad VM_k^d z_{kk'i} \leq pm_k^d(1 - z_{k'}) \quad \forall d$ **then**
8: Allocate $VM \in VM' \quad topm_j$
9: Update $pm_{k'}^d = pm_{k'}^d - VM_i^d \quad \forall d$
10: Destroy VM_i from the n_i
11: Remove VM_k from VM'

condition between PMs. Steps 16 to 17 describe the selection of a PM for creating the child node. Steps 18 to 19 describe the maximum use of a single type of PM for the creation of child node. Steps 20 to 23 describe the allocation of a VM to a PM over power consumption criteria of different PM. Steps 24 to 26 describe the allocation of a VM to a higher index PM. Steps 27 to 28 describe the allocation of VM on already initialized PM at the data center.

Algorithm 6.2 describes the exact migration of VM from underutilized PM to the energy efficient PM using the First-Fit approximation technique. Step 1 creates the empty list of VM. Steps 2 to 4 collect the VMs from the underutilized PMs. Step 5 describes the sorting of PMs using power consumption (p_k) function. Steps 6 to 11 describe the reallocation of VMs from underutilized PM to another energy efficient PM at the cloud data center.

6.2 PERFORMANCE EVALUATION

Experimental Setup: Since, proposed ILP model, branch-and-bound based exact algorithm, and network aware VM migration policy consist of number of

Table 6.1: Configuration of PMs

PM Type	PE	MIPS	RAM(GB)	STORAGE(GB)
ProLiantM110G5XEON3075	2	2660	4	160
IBMX3250Xeonx3480	4	3067	8	250
IBM3550Xeonx5675	12	3067	16	500

variables, and methods. Thus, we did not use existing simulators for checking the performance of the proposed work. Further, the experiment is conducted by taking different VMs and set the size of the data center by taking constant number of PMs. Thus, we used Java 1.7 version to create network-aware cloud data center environment, and implementation of proposed VM allocation and migration algorithms. To check the solution quality of our proposed exact algorithm, we compared our proposed algorithm with the Best-Fit algorithm as a benchmark solution. For heterogeneous environment, we created three different kinds of PMs; further we considered four different kinds of VMs in terms of resource allocation at the data center. The configurations of the PMs and VMs are given in Table 6.1 and Table 6.2 respectively. The different kinds of PMs configuration are downloaded from the (IBM-Power-Model 2014) and (Dell-Power-Model 2014) websites. The Amazon EC-2 based VMs instances are used for the creation of VMs to PMs. The configuration of VMs instances is downloaded from Amazon website (Amazon-Website 2014). Since, Amazon EC-2 based VMs configuration has not specified any bandwidth requirement of VM; hence in this work, we randomly selected the bandwidth requirement for each type of VMs (50 Mbps small, 60 Mbps medium, 70 Mbps large, and 80 Mbps x.large).

The power consumption of each of a 16 ports switch is 80W in full load condition (Fang et al. 2013). Each switch consists of 16 ports and each port of a switch consumes 5W. In our experiment, we set the in-rack and out-rack link capacity of 1 Gbps respectively. For creating the Fat-tree architecture based data center, we used 16 port switches, and the details of a Fat-tree architecture based data center are given in Table 6.3.

Table 6.2: Configuration of VMs

VM Type	PE	MIPS	RAM (GB)	STORAGE (GB)
Small	1	500	0.5	40
Medium	2	1000	1	60
Large	3	1500	2	80
X.large	4	2000	3	100

Table 6.3: Data Center Configuration

Type of switches	Edge	Aggregation	Core	Total Switches	Total PM	Total Pods
16 ports	128	128	64	320	1024(341,341,342)	16

Results and Analysis: To handle the condition of a steady stream of new VMs creation, we applied our proposed branch-and-bound based exact algorithm on the arrival of each user requests at the cloud data center. Further, to remove the time expired VMs from the PMs, we applied our proposed algorithm on a regular time interval and migrate the VMs from underutilized PM to energy efficient PM at the cloud data center. The performance of the proposed branch-and-bound based exact algorithm in terms of energy consumption and resources utilization is compared with the Best-Fit algorithm as it is the benchmark for both Homogeneous and Heterogeneous data center environments. Further, the experiment is conducted by taking different number of VMs and keeping size of the data center constant by taking 1024 number of PMs in the data center. For each user request, we set the same number of VMs of all types. For example, 400 VMs consist of (100 small, 100 medium, 100 large, 100 x.large type of VM). Further, we divided the total PMs of the data center into number of clusters, and each cluster consist of the same number of PMs all the types. For example 90 PMs in a cluster consist of (30 Type1, 30 Type2, and 30 Type3).

Table 6.4 shows the number of PMs of different types used for the VM allocation in two different data center environments (homogeneous and heterogeneous). In the case of an exact algorithm, we need less number of PMs when compared

Table 6.4: Number of PMs Used at the Cloud Data Center

VM	Best Fit		Exact Algorithm	
	Heterogeneous	**Homogeneous**	**Heterogeneous**	**Homogeneous**
200	(16,42,12)	74	(14,34,10)	64
400	(32,84,24)	148	(26,67,19)	128
600	(48,126,36)	222	(38,98,27)	192
800	(64,168,48)	296	50,130,35	256
1000	(80,210,60)	370	(61,160,42)	320

Table 6.5: Number of Switches Used for VM Allocation

VM	Best Fit								Exact Algorithm							
	Heterogeneous				Homogeneous				Heterogeneous				Homogeneous			
	Pod	Edge	Agg.	Core	Pod	Edge	Agg.	Core	Pod	Edge	Agg.	Core	Pod	Edge	Agg.	Core
200	2	9	9	1	2	10	10	1	1	8	8	1	1	1	8	1
400	3	18	18	1	3	19	12	1	2	14	14	1	2	24	24	1
600	4	26	26	1	4	28	28	1	3	21	21	1	3	24	24	1
800	5	35	35	1	5	37	37	1	4	27	27	1	4	32	32	1
1000	6	44	44	1	6	47	47	1	5	33	33	1	5	40	40	1

to the Best-Fit algorithm for the allocation of different number of VMs. The reason behind taking less number of PMs by the branch-and-bound based exact algorithm is due to an optimal selection of PM for the allocation of VMs using search tree in the exact algorithm. Hence, more number of VMs are allocated to less number of PMs in the case of an exact algorithm as compared to that of the Best-Fit algorithm. Thus, resulting in less number of PMs used by the data center for the allocation of VMs using the branch-and-bound based exact algorithm.

Table 6.5 shows the number of switches (core, aggregation, edge) used for the allocation of VMs in the cloud data center network. The number of switches used at the data center in the case of proposed exact algorithm is less when compared to the Best-Fit algorithm for both data center environments. The proposed network-aware lower bounds are used to select the optimal number of switches from the data center in the case of exact algorithm. Thus, more number of VM allocated in the same rack will result in switching-off more number of network switches.

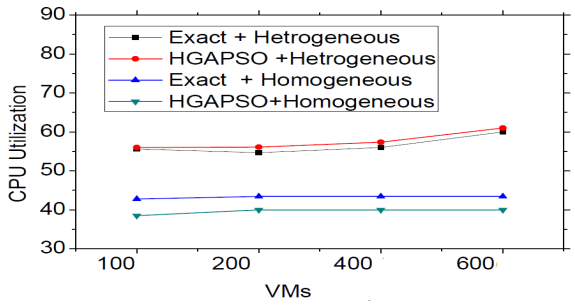

Figure 6.4: % of CPU Utilization

Figure 6.4 shows the % of CPU utilization, Figure6.5 shows the % of RAM utilization, and Figure 6.6 shows the % of Storage utilization at the data center. The CPU, RAM, and Storage utilization are calculated for both homogeneous and heterogeneous data center environments by applying different number of VMs while keeping number of PMs constant at the data center. The % of (CPU, RAM, and Storage) utilization of the proposed branch-and-bound based exact algorithm is high when compared to that of the Best-Fit algorithm in all the cases. And this is due to less number of PMs used for the VM allocation in the case of exact algorithm. Since, large number of PMs are switched-off, hence this will result in less resource wastage in the data center. The gap of resources utilization between exact algorithm and Best-Fit algorithm is increasing when taking more number of VMs at the cloud data center. This is because of the increased gap between number of PMs used for the allocation of VMs to PMs as shown in Table 6.4. Further, the resources utilization graph is almost a straight line in the case of Best-Fit algorithm. This is due to increasing number of used PMs for VM allocation on the similar lines of increasing the number of VMs at the data center. Hence, it is observed that there is a very less variation in the % of resources utilization.

Further, higher resource utilization will lead to excess heat generation resulting in hardware failure or need of more cooling devices at the cloud data center. Hence, to avoid these issues of the data center, we need to maintain an unused buffer area for each type of resource in the PM. Therefore, in our experiment, we

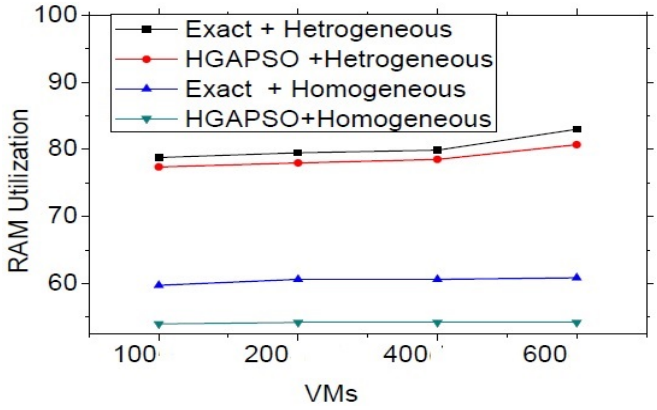

Figure 6.5: % of RAM Utilization

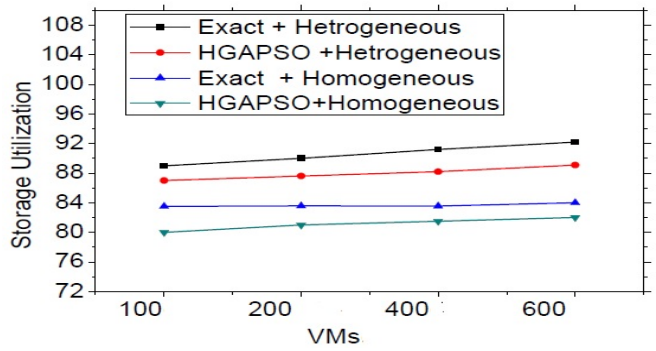

Figure 6.6: % of Storage Utilization

set the maximum resource utilization limit of CPU, RAM, and Storage to 70%, 90%, and 95% respectively. Hence, the maximum value of CPU, RAM, and, Storage utilization are not going beyond the maximum set limit value as shown in Figures 6.4, 6.5, and 6.6 respectively.

Figures 6.7 and 6.8 show the power consumption of PM and switches for different number of VMs in heterogeneous and homogeneous data center environments, respectively. The power consumption of the data center in the case of proposed exact algorithm is low when compared to the Best-Fit algorithm for both heterogeneous and homogeneous environments. The power consumption of the data center in the proposed exact algorithm is low (since less number of type1, type2, and type3 PMs are used for VM allocation) when compared to that of the Best-Fit algorithm. Hence, the power consumption of the data center is low in the case of exact algorithm for different number of VMs. Further, the amount of power

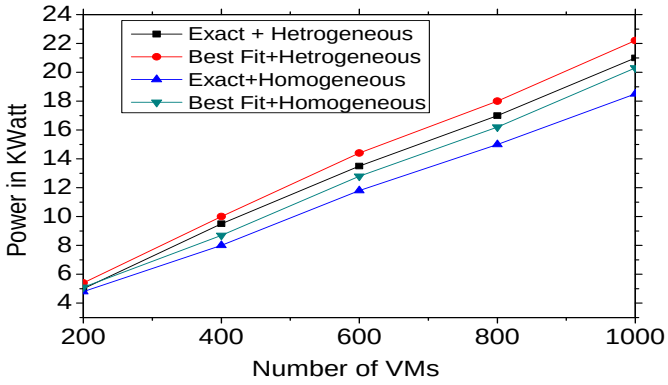

Figure 6.7: Server Power Consumption

consumed by the switches at the cloud data center is also less in the case of exact algorithm as compared to that of the Best-Fit algorithm. This is due to less number of switches used by the data center for the allocation of the VMs to PMs. The optimal placement of VMs to PMs will reduce the wastage of bandwidth and also switching-off more number of idle ports, and non-active switches (access & edge) will result in lower power consumption at the cloud data center in the case of exact algorithm.

The total power consumption of the cloud data center is dependent on the power consumption of the PM and network switches. Hence, reducing the number of switched-on idle PMs and switched-on idle switches will thus reduce the total power consumption of the cloud data center. Figure 6.9 shows the total power consumption of both homogeneous and heterogeneous data center environments. The power consumption of the data center is less in the case of exact algorithm when compared to that of the Best-Fit algorithm. The exact algorithm gives an optimal solution in terms of VM allocation by maintaining the switch bandwidth and PM resource constraints. Hence, an optimal solution will be obtained by switching-off more number of idle PMs and thereby achieving lower power consumption at the cloud data center.

Further, to evaluate the performance of proposed VM migration technique, we created different VMs instances as requested by the users in the discrete time interval at the data center. In our experiment, we considered 1024 PMs at the data

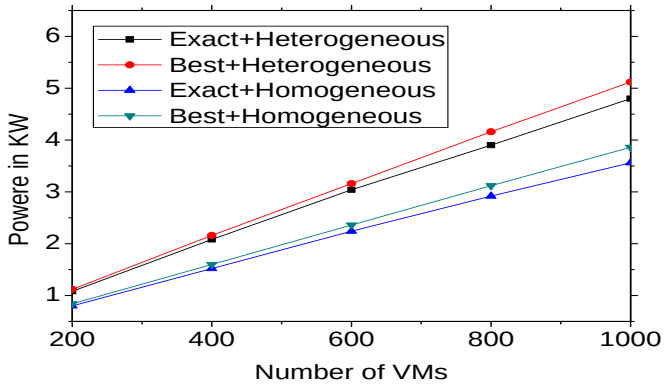

Figure 6.8: Switch Power Consumption

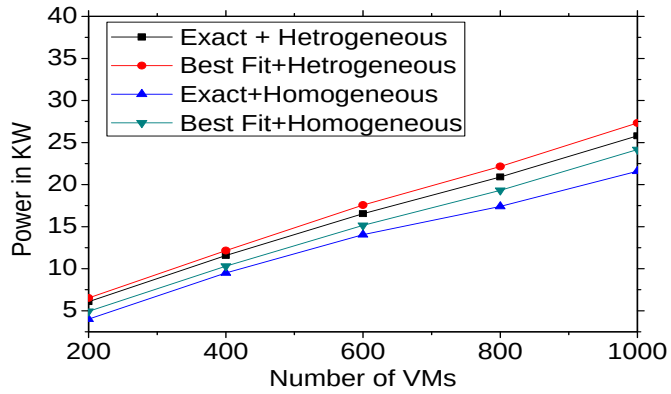

Figure 6.9: Total Power Consumption

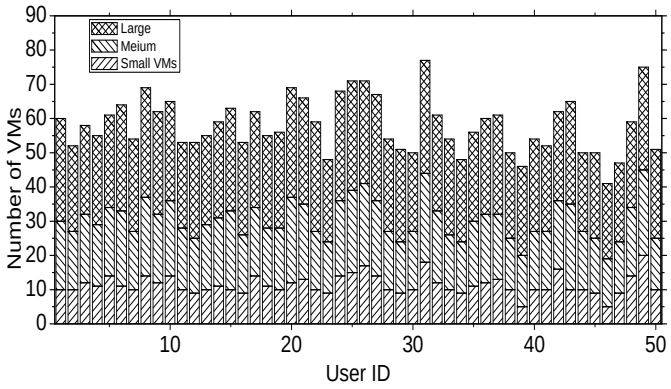

Figure 6.10: Number of VMs Requested by Users.

center (341 type1, 341 type2, and 342 type3). Further, to derive the data center system model, all the users' requests will be sent to the cloud service provider for the creation of VMs at the data center. Hence, by this way, the cloud service provider will collect the user's requests in the form of different VMs instances (small, medium, large). In our experiment, we defined the user's requested VMs in the range of [1, 100].

The lifetime of all the VMs is set uniformly in between the set range of [30 minutes and 200 minutes]. We used constant migration overhead cost (c_o^{mig}) in terms of power consumption during VM migration, such as 10 watt (small), 20 watt (medium), 30 watt (large), and 40 watt (x.large). The VM allocation algorithm is applied to the data center when a new user's requests arrives at the data center. By applying the proposed migration technique, we destroy the VMs from the PMs at the data center after completion of time instance of VMs as requested by the user. The proposed migration technique will migrate the VMs from the underutilized PM to the energy efficient PM by setting the lower threshold utilization of each PM as (u_l=30%). Thus, if the current CPU utilization of a PM is less than the set value of lower utilization threshold then PM will be in underutilized condition. Since, our main focus is to reduce the power consumption of the PM by the optimal allocation of VMs to PMs at the cloud data center, hence we did not check the performance of our proposed VM migration techniques with other existing VM migration techniques at the cloud data center.

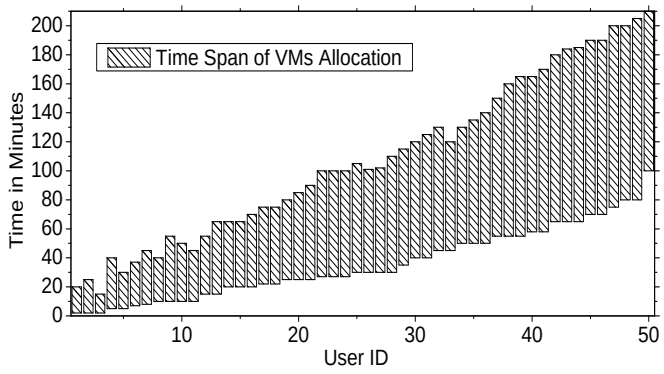

Figure 6.11: Time Span of VMs allocated to Data Center.

Figure 6.10 shows the total number of VMs as requested by 50 users for a predefined period of time at the cloud data center. Figure 6.11 shows the duration of VM as requested by different users during different time intervals. The total time duration of a user requested VM allocated to the cloud data center is calculated by the subtraction of allocation time from the destruction time i.e. (destruction time-allocation time) using Figure 6.11.

Figure 6.12 shows the total number of PMs that are switched-on by different techniques in heterogeneous and homogeneous data center environments during different time instances. The proposed exact algorithm in both cloud data center environments takes less number of PMs when compared to that of the Best-Fit algorithm. The exact algorithm gives an optimal solution in terms of VM allocation to PM. Hence, this will result in less number of PMs used for the VM allocation by our proposed exact algorithm. Initially, in Figure 6.12, the graph is going upward due to continuous arrival of user's requests at the data center. After 50 time instances, the graph is going down due to the destruction of those VMs which have completed their specified time slots. Further, our proposed VM migration technique will not only switch-off idle PMs but also migrate VMs from the underutilized PMs to the energy efficient PM. The maximum number of PMs used by the data center for the VM allocation during peak load is more than 900 as shown in Figure 6.12.

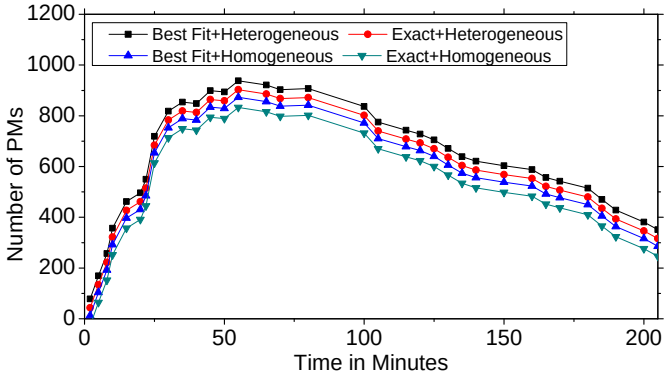

Figure 6.12: Number of PMs used for VM allocation.

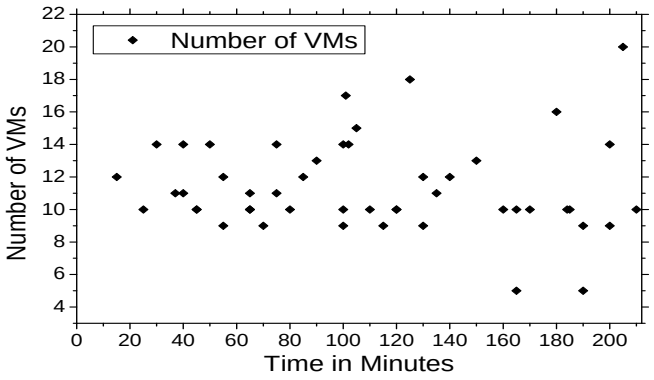

Figure 6.13: Number of VMs Migrated on $u_l = 30\%$.

Figure 6.13 shows the number of VMs migrated during different time intervals on the lower utilization threshold value ($u_l = 30\%$) at the data center. The proposed First-Fit based VM migration policy will not only migrate the VMs from the underutilized PM to the energy efficient PM at the data center, but also switches-off underutilized (idle) PMs at the data center. The energy consumption and CPU utilization of the data center during different periods of time instances are shown in Figure 6.14 and Figure 6.15, respectively. The energy consumption of the data center is low in the case of the proposed exact algorithm approach when compared to that of the other VM allocation techniques for both data center environments. This is due to less number of energy efficient PMs used by exact algorithm and further migration of VMs from the underutilized PM to the energy efficient PM. Further, switching-off the underutilized PM will result in more energy saving at the cloud data center.

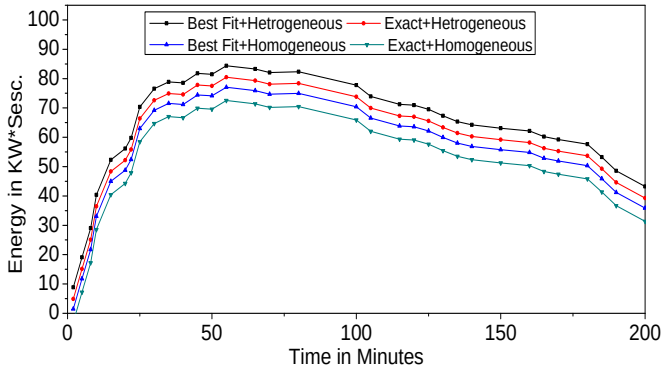

Figure 6.14: Energy Consumption at the Data Center.

Initially, the energy consumption graph, as shown in Figure 6.14, is going upward due to the continuous increment of number of PMs used by the data center for the allocation of user's requested VMs. After 50 seconds, this graph is going down since there are no further user's requests arriving at the cloud data center and those VMs completed their time slots will be destroyed from the data center resulting in switching-off more number of idle PMs. Further, the migration of VMs from underutilized PM to the energy efficient PM will result in continuous drop in energy consumption of the data center.

Figure 6.15 shows the % of CPU utilization of the cloud data center for different time instances. The CPU utilization of the proposed exact algorithm is more than that of the Best-Fit VM allocation technique. This is due to the near optimal placement of VMs to the PMs at the cloud data center which will result in less number of PMs used for the VM allocation by the proposed exact algorithm. Hence, the resource wastage in exact algorithm is less than that of the Best-Fit VM allocation technique.

Figure 6.16 shows the total energy consumption of the data center for a specified period of time. The energy consumption of the PM at the data center by the proposed exact algorithm is (15%, and 12%) less for heterogeneous and homogeneous data center environments, respectively when compared to that of the Best-Fit algorithm. Further, the energy consumption of the switches using the exact algorithm for heterogeneous and homogeneous data center environments is

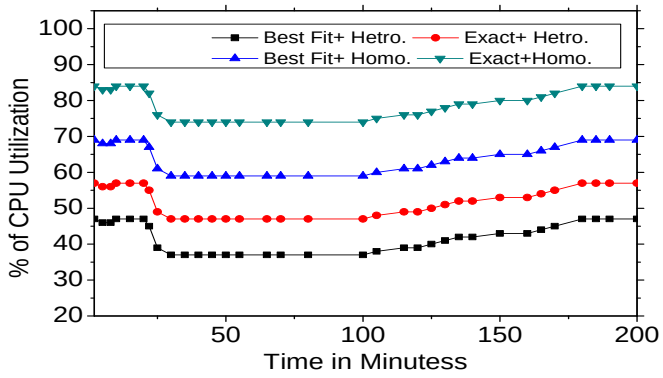

Figure 6.15: % of CPU Utilization.

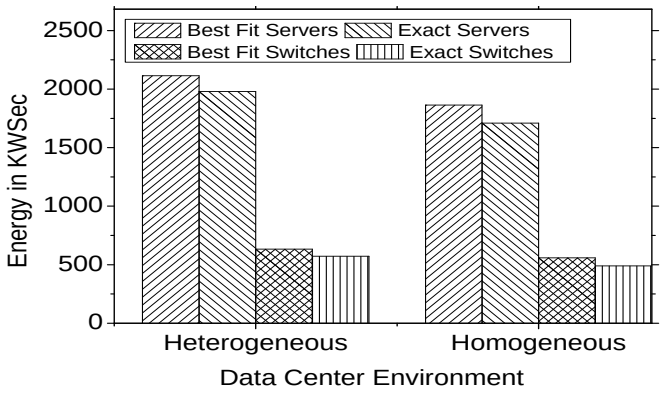

Figure 6.16: Total Energy Consumption.

(10%, and 8%) less than that of the Best-Fit Algorithm. Figure 6.17 shows the average % of CPU utilization. The CPU utilization is (15%, and 10%) more in the case of exact algorithm when compared to Best-Fit algorithm for heterogeneous and homogeneous data center environments, respectively.

To calculate the optimal value of lower utilization threshold (u_l) for VM migration, we set this threshold value in the range of [10%, 90%]. Figure 6.18 shows the energy consumption of the data center for different lower utilization threshold values. The energy consumption of data center is high and SLA violation is low while decreasing the lower utilization threshold value due to less number of VMs migrated from the underutilized PM to the energy efficient PM. On the other hand, the energy consumption of data center is low and SLA violation is high while increasing the lower utilization threshold value because of more number of VMs migrated from the underutilized PM to the energy efficient PM. Hence, in

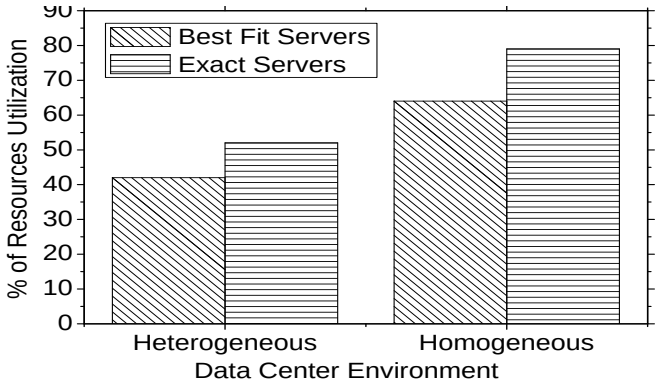

Figure 6.17: Average Resource Utilization.

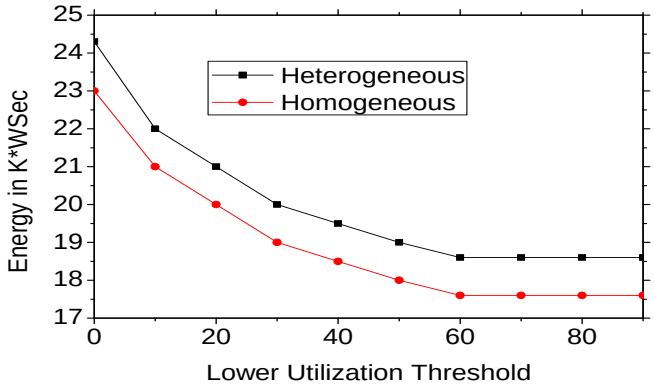

Figure 6.18: Lower Utilization Threshold.

our experiment, we set 30% as the optimal lower utilization threshold value.

For checking the scalability and performance of our proposed exact algorithm in terms of computation time, we execute the proposed algorithm for VM allocation and migration with different lower utilization threshold values. The Java program is conducted on HP Compaq LE1902X machine (i3 processor with 2 GB RAM). Figure 6.19 shows the running time of our proposed VM allocation and migration by taking VMs in between [100 and 1000]. The computation time for VM allocation and migration algorithms is approximately 4 minutes in the case of 800 PMs. The computation time of proposed VM allocation algorithm is low due to the modified heuristic approach used for selecting the optimal PMs for generating the child nodes in the reduced search tree. Hence the proposed exact algorithm based on branch-and-bound technique is suitable for large data centers (homogeneous and heterogeneous).

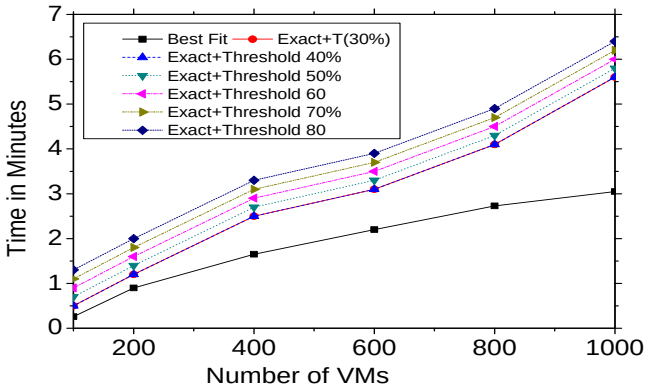

Figure 6.19: Execution Time on Different Threshold Values.

6.3 Summary

This Chapter discussed the energy saving of PMs and switches at the network-aware cloud data center environments (homogeneous and heterogeneous). The proposed exact algorithm for VM allocation not only saved the energy of PMs and network switches, but also reduced the resources wastage at the data center for both cloud environments. Further, energy efficient VM migration using optimal lower utilization threshold value will switch-off underutilized PMs resulting in more energy saving with less SLA violation at the data center for both cloud environments. Further, the performance evaluation of proposed algorithms in the terms of energy efficiency, resources wastage, and, execution time for the VM allocation problem in both cloud data center environments (homogeneous and heterogeneous) are described in Chapter 7.

Chapter 7

Performance Evaluation of HGACSO, HGAPSO, HGAPSOSA and Exact Algorithms

In the previous Chapters of this thesis, we proposed two types of VM allocation algorithms such as non-network aware (HGACSO, HGAPSO, HGAPSOSA), and network-aware (branch-and-bound based Exact algorithm). In Chapter 3, we discussed the HGACSO based VM allocation algorithm which is the hybrid combination of GA and CSO. In Chapter 4, we discussed the proposed HGAPSO algorithm for VM allocation which is the hybrid combination of GA, and PSO. In Chapter 5, we discussed the HGAPSOSA algorithm which is the hybrid combination of GA, PSO, and SA. Further, In Chapter 6, we discussed the network-aware branch-and-bound based Exact algorithm. Hence, we need to check the performance of all the proposed VM allocation algorithms in terms of energy consumption, resources utilization, and execution time. Thus, the key contributions of this Chapter are:

- To evaluate the performance of proposed HGACSO, HGAPSO, and HGAPSOSA in non-network aware data center environment, and branch-and-bound based exact algorithm in network-aware data center environment.
- To evaluate the execution time of proposed HGACSO, HGAPSO, HGAPSOSA, and Exact algorithms.

7.1 Performance Evaluation

To check the performance of proposed VM allocation algorithms we need to setup two different data center environments such as (non-network aware, and network aware). Hence, experimental setup of data center environment is described as follows.

A. Experimental Setup

To check the performance of proposed algorithms, we consider three different types of PMs, and four different types of VMs at the cloud data center. Further, to create the network aware data center environment, we consider 16 port networking switch. The detailed description of PMs, and VMs are described in Section 4.2, and the detailed description of networking switch is described in Section 6.2. Further, all the proposed VM allocation algorithms such as HGACSO, HGAPSO, HGAPSOSA, and branch-and-bound based Exact algorithm are implemented in Java 1.7 version.

B. Non-Network Aware Data Center

To check the superiority of the proposed algorithms, we compared their performance in terms of energy consumption, resources utilization, and execution time in both homogeneous and heterogeneous data center environments. Further, the experiment is conducted in two different cases, in Case-1 we consider different combinations of VMs(PMs) at the cloud data center. In Case-2 we consider variable number of VMs and constant number of PMs at the cloud data center. For each VMs(PMs) combination in Case-1, we set the same number of VMs and PMs of all types. For example, 400 VM consist of (100 small, 100 medium, 100 large, 100 x.large type of VM) and 90 PM consist of (30 Type1, 30 Type2, and 30 Type3).

Case-1

The number of PMs of different types used for the allocation of VMs in two different data center environments (homogeneous and heterogeneous) is shown in Table 7.1. In the case of HGAPSOSA, we need less number of PMs when compared to both HGACSO, and HGAPSO for the allocation of VMs for each of the VMs(PMs) combinations. The reason behind taking less number of PMs by the HGAPSOSA is due to an optimal selection of PMs for the allocation of VMs by improving the performance of mutation operation using SA. Hence, more number of VMs are allocated to less number of PMs in the case of HGAPSOSA as compared to that of both HGACSO, and HGAPSO. Thus, resulting in less number of PMs used by the data center for the allocation of VMs using the HGAPSOSA.

Table 7.1: Number of PMs Used at the Cloud Data Center

VM(PM)	Homgeneous			Heterogeneous		
	HGACSO	**HGAPSO**	**HGAPSOSA**	**HGACSO**	**HGAPSO**	**HGAPSOSA**
100(60)	37	38	33	(10,12,6)	(10,13,6)	(8,10,5)
200(120)	74	75	65	(20,23,12)	(20,25,12)	(16,20,10)
400(240)	148	150	130	(40,48,23)	(40,50,24)	(33,40,20)
600(360)	222	224	196	(60,71,36)	(60,74,36)	(49,60,30)
800(480)	296	298	260	(80,96,47)	(80,98,48)	(65,80,40)
1000(600)	370	372	325	(100,120,59)	(100,122,60)	(80,100,50)

Further, the number of PM used by both HGAPSO and HGACSO is almost same since both seeking mode operation of HGACSO and the mutation operation of GA achieve similar performance. Hence, both operations such as seeking mode and mutation operations generate the same type of chromosomes, resulting in same number of PMs for the VM allocation in the case of HGACSO and HGAPSO.

Figure 7.1 shows the % of CPU utilization, Figure7.2 shows the % of RAM utilization, and Figure 7.3 shows the % of Storage utilization at both heterogeneous and homogeneous data center environments. The % of CPU, RAM, and Storage utilization are calculated for both homogeneous and heterogeneous data center environments by applying different VMs(PMs) combinations. Further, solid and dotted lines in Figures 7.1, 7.2, and 7.3 show the % of resources utilization in heterogeneous, and homogeneous data center environments respectively.

The % of (CPU, RAM, and Storage) utilization in the case of HGAPSOSA is high when compared to that of both HGACSO, and HGAPSO for all VMs(PMs) combinations. And this is due to less number of PMs used for VM allocation in the case of HGAPSOSA, when compared to that of HGACSO, and HGAPSO. Since, large number of PMs are switched-off in the case of HGAPSOSA, hence this will result in less resource wastage at the cloud data center. Further, in the case of homogeneous cloud data center environment, the resources utilization graph is almost straight line for all VMs(PMs) combinations because of the number of PMs are switched-on at the cloud data center in the same proportion as number of VM are increasing.

Further, in heterogeneous data center environment, the resources utilization is improving when allocating more number of VMs among less number of switched-on PMs at the cloud data center. Further, in heterogeneous cloud data center environment, the gap between the resources utilization of HGAPSOSA and that of other proposed algorithms (HGACSO, HGAPSO) is increasing when more number of VMs(PMs) combinations are taken into account. This is because of the increased gap between number of PMs used for the allocation of VMs to PMs as shown in Table 7.1. Further, the resources utilization in the case of HGACSO, is nearly equal to HGAPSO, since the same number of PM are used by both HGACSO, and HGAPSO algorithms.

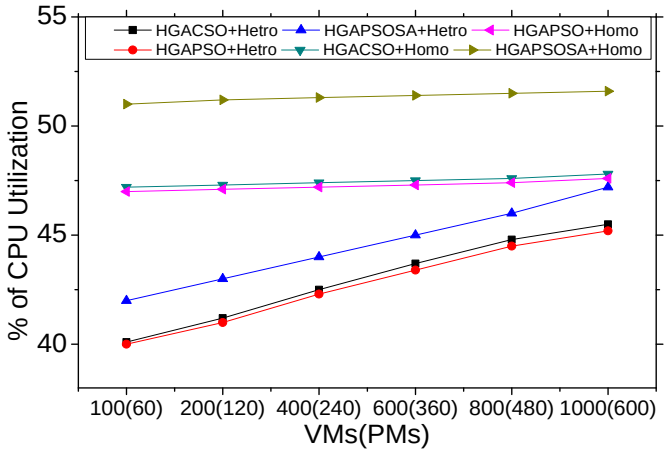

Figure 7.1: % of CPU Utilization at the Data Center

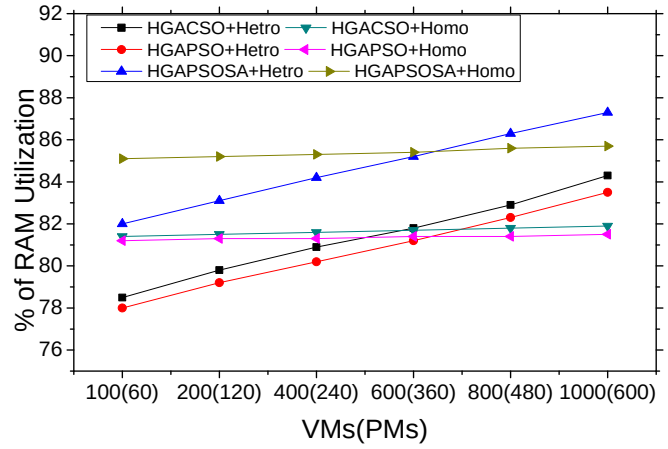

Figure 7.2: % of RAM Utilization at the Data Center

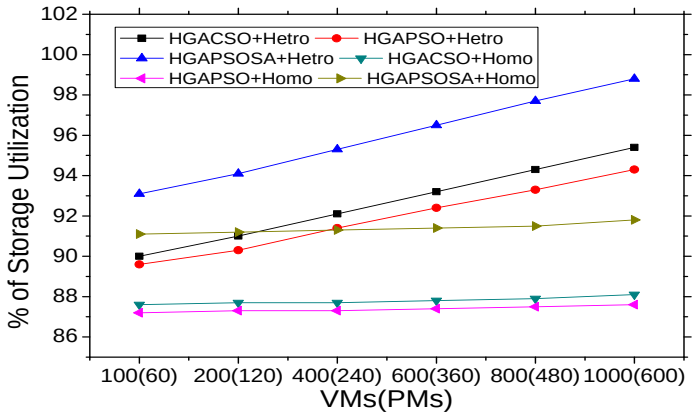

Figure 7.3: % of Storage Utilization at the Data Center

Table 7.2: Number of PMs Used at the Constant Data Center

VM(PM)	Homgeneous			Heterogeneous		
	HGACSO	**HGAPSO**	**HGAPSOSA**	**HGACSO**	**HGAPSO**	**HGAPSOSA**
100(60)	38	40	34	(10,13,6)	(11,13,6)	(9,10,5)
200(120)	77	78	69	(22,26,12)	(23,27,12)	(18,21,10)
400(240)	149	150	132	(41,48,24)	(42,49,24)	(35,41,20)
600(360)	223	224	197	(61,72,36)	(62,73,37)	(50,61,30)
800(480)	299	300	264	(84,97,49)	(85,99,50)	(68,81,41)
1000(600)	371	373	325	(101,120,60)	(102,121,61)	(80,100,50)

Case-2

Table 7.2 shows the number of PMs used for VMs allocation in the case of taking varible number of VMs and keeping constant number of PMs at the cloud data center. Figures 7.4, 7.5, and 7.6 show the % of CPU, RAM, and Storage utilization respectively in both cloud data center environments (homogeneous, heterogeneous). Further, performance of CPU, RAM, and Storage utilization is varying while taking different number of VMs and keeping number of PMs constant in both homogeneous and heterogeneous cloud data center environments. Since, size of the chromosome is large in proportion of number of VMs requested by the user, so there is a high probability that all hybrid bio-inspired algorithms such as HGAPSO, HGACSO, and HGAPSOSA select idle PM in the chromosome for the allocation of VM.

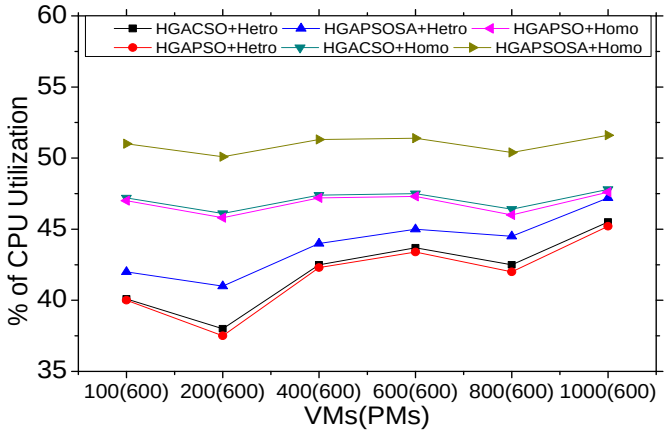

Figure 7.4: % of CPU Utilization at Constant Data Center.

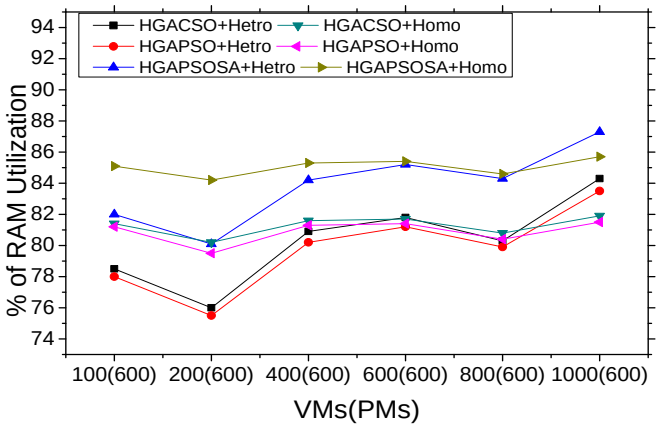

Figure 7.5: % of RAM Utilization at Constant Data Center.

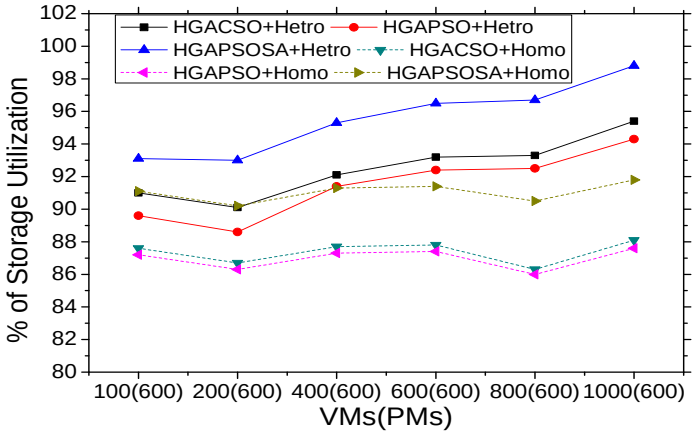

Figure 7.6: % of Storage Utilization at Constant Data Center.

Power Consumption Analysis (Case-1 and Case-2)

Figure 7.7 shows the power consumption of data center in both homogeneous and heterogeneous environments for different VMs(PMs) combinations. The power consumption of the data center in the case of HGAPSOSA is low when compared to the HGACSO, and HGAPSO algorithms for both heterogeneous and homogeneous environments. The power consumption of the data center in the case of HGAPSOSA is low (since less number of type1, type2, and type3 PMs are used for VM allocation) when compared to that of the HGAPSO, and HGACSO algorithms. Hence, more number of PMs are switched-off at the data center, resulting in low power consumption at the data center.

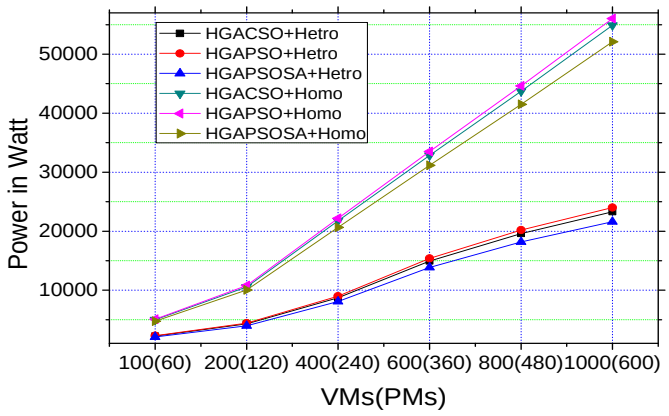

Figure 7.7: Power Consumption at the Data Center

Figure 7.8 shows the power consumption of the cloud data center while taking different number of VMs and keeping number of PMs constant at the cloud data center. The power consumption in the case of HGAPSOSA is low when compared to that of other two hybrid approaches (HGAPSO, HGACSO). Further, in case of constant number of PMs the power consumption of the data center is varying in different proportion over different number of VMs, due to the number of PMs used for the allocation of different number of VMs at the cloud data center.

Further, to check the performance of proposed algorithms in terms of energy consumption for non-network aware data center environment, we created different VM instances as requested by the users in the discrete time interval at the data center. In our experiment, we considered 1024 PMs at the data center (341 type1,

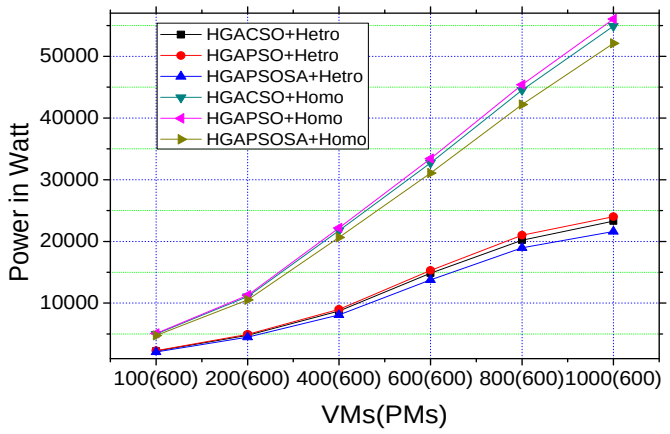

Figure 7.8: Power Consumption at the Constant Data Center

341 type2, and 342 type3). Further, to derive the data center system model, all the users' requests will be sent to the cloud service provider for the creation of VMs at the data center. Hence, by this way, the cloud service provider will collect the user's requests in the form of different VMs instances (small, medium, large). In our experiment, we defined the user's requested VMs in the range of [1, 100].

The lifetime of all the VMs is set uniformly in between the set range of [30 minutes and 200 minutes]. We used constant migration overhead cost (c_o^{mig}) in terms of power consumption during VM migration, such as 10 watt (small), 20 watt (medium), 30 watt (large), and 40 watt (x.large) as defined in Chapter 3. The VM allocation algorithm is applied to the data center when a new user's requests arrive at the data center. By applying the proposed migration technique, we destroy the VMs from the PMs at the data center after completion of time instance of VMs as requested by the user. The proposed migration technique will migrate the VMs from the underutilized PM to the energy efficient PM by setting the lower threshold utilization of each PM as (u_l=30%). Thus, if current CPU utilization of a PM is less than the set value of lower utilization threshold then PM will be in underutilized condition. Hence, we use the same VM migration policy as discussed in Section 3.3. Further, to evaluate the performance of proposed algorithms we used the same number of users requested VMs and the time duration of VMs as discussed in Section 3.3.

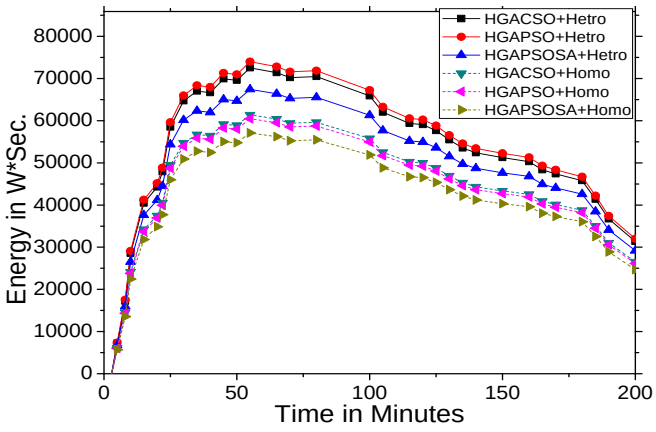

Figure 7.9: Energy Consumption at the Data Center.

Energy Consumption, Average Resource Utilization and Execution Time Analysis of Proposed Algorithms

The energy consumption of the data center in both heterogeneous and homogeneous environments during different periods of time instances is shown in Figure 7.9. Further, solid and dotted lines in Figure 7.9 show the energy consumption in heterogeneous and homogeneous data center environments, respectively. The energy consumption of the data center is low in case of HGAPSOSA when compared to that of HGAPSO, and HGACSO for both cloud data center environments. This is due to less number of energy efficient PMs used by HGAPSOSA and further migration of VMs from the underutilized PM to the energy efficient PM. Further, switching-off the underutilized PMs will result in more energy saving at the cloud data center. Further, the energy consumption of both HGAPSO, and HGACSO is almost similar because of same number of PMs used by the cloud data center for the allocation of VMs.

Initially, the energy consumption graph, as shown in Figure 7.9 is going upward due to the continuous rise in the number of PMs used by the cloud data center for the allocation of user's requested VMs. After 50 seconds, this graph is going downward since, there is no further arrival of user's requests at the cloud data center, and those VMs completed their time slots will be destroyed resulting in switching-off more number of idle PMs at the cloud data center. Further, migration of VMs from underutilized PM to the energy efficient PM will lead to

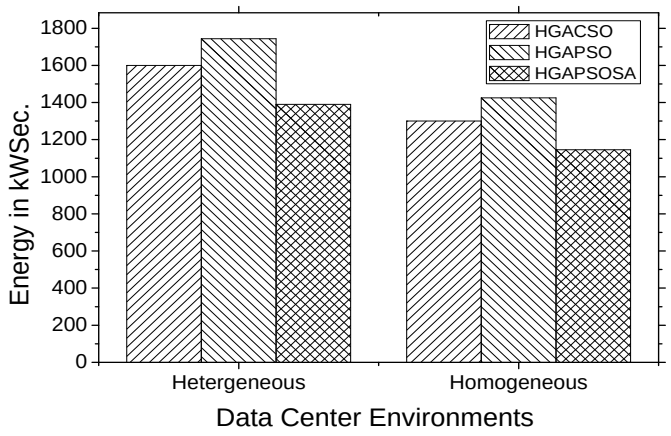

Figure 7.10: Total Energy Consumption.

continuous drop in the energy consumption of both (heterogeneous and homogeneous) cloud data center environments.

The total energy consumption, and average resources utilization at the cloud data center is shown in Figure 7.10 and Figure 7.11 respectively. The total energy consumption of the heterogeneous data center by using the proposed HGAPSOSA algorithm is (8.9%, and 7.0%) less when compared to HGAPSO, and HGACSO respectively. Further, the energy consumption of the homogeneous data center is (7.5%, and 7.0%) less when compared to HGAPSO, and HGACSO respectively. The % of average resources utilization of the data center in the case of HGAPSOSA is (3.9%, and 3%) more when compared to HGAPSO, and HGACSO respectively. Further, the % of average resources utilization of the data center in the case of HGAPSOSA is (4.3%, and 3.7) more when compared to HGAPSO, and HGACSO respectively. Moreover the variation in energy consumption and resources utilization between HGACSO, and HGAPSO is very less.

To check the execution time of the proposed algorithms in non-network aware data center environment, we conducted our experiment on the same machine as discussed in Section 4.2. Figure 7.12 shows the execution time of the proposed HGACSO, HGAPSO, and HGAPSOSA algorithms by taking VMs in between [1000 and 10000]. The computation time for VM allocation in the case of HGAPSOSA algorithm is high as compared to the HGAPSO and this is due to modifi-

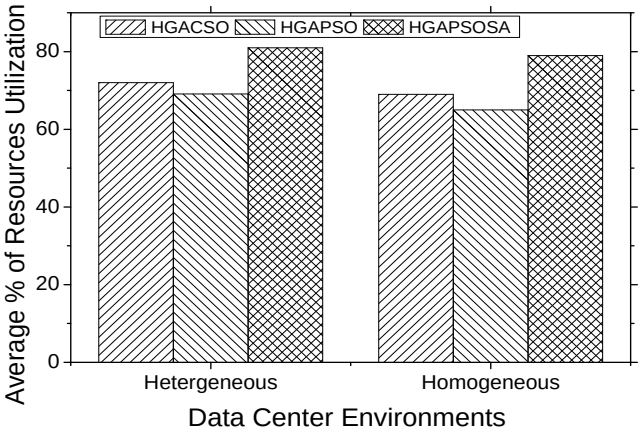

Figure 7.11: Average Resource Utilization.

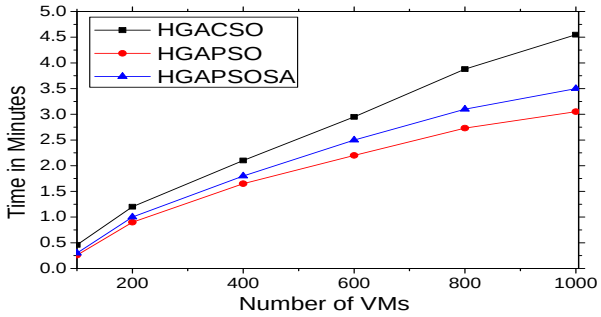

Figure 7.12: Execution time of VM Allocation Algorithms.

cation of mutation operation of HGAPSOSA by using SA. And the computation time of HGAPSOSA is low as compared to HGACSO since, PSO overcomes the limitations of seeking mode operation of CSO, and resulting in less convergence time. Further, computation time of HGACSO is high when compared to that of HGAPSO. This is due to requirement of large amount of time for the convergence in the case of seeking mode of HGACSO. Further, seeking mode operation works in the same manner as mutation operation of the HGAPSO. Hence, HGACSO generates similar types of solution as generated by HGAPSO.

C. Network Aware Data Center

To check the performance of proposed branch-and-bound based Exact algorithm in network aware data center environment, we compared our proposed algorithm with Best-Fit algorithm. The energy consumption, and resources utilization of the data center in network aware (homogeneous and heterogeneous) data cen-

ter environments are discussed in Section 6.2. Further, experimental results show that our proposed branch-and-bound based Exact algorithm outperforms Best-Fit algorithm in terms of energy consumption, and resources utilization. Thus, energy consumption of the PMs at the data center by the proposed exact algorithm is (15%, and 12%) less for heterogeneous and homogeneous data center environments, respectively when compared to that of the Best-Fit algorithm.

Further, the energy consumption of the network switches using the exact algorithm for heterogeneous and homogeneous data center environments is (10%, and 8%) less than that of the Best-Fit Algorithm. The average resources utilization is (15%, and 10%) more in the case of exact algorithm when compared to Best-Fit algorithm for heterogeneous and homogeneous data center environments, respectively. Further, the execution time of branch-and-bound based Exact algorithm is 4.5 minutes by taking 1000 VMs at the cloud data center.

7.2 Summary

This chapter discussed the performance evaluation of the proposed VM allocation algorithms in two different (non-network, and network) data center environments. Further, experimental results are carried out in homogeneous and heterogeneous data center environments. The performance of HGAPSOSA in terms of energy consumption, resources utilization is superior when compared to that of HGACSO, and HGAPSO. The energy consumption and resources utilization is slightly better in the case of HGACSO, when compared to HGAPSO. Further, the execution time of HGAPSOSA is slightly high when compared to HGAPSO, and it is low as compared to HGACSO. Hence, HGAPSOSA, and HGAPSO are the optimal choices for the allocation of VMs to PMs at the non-network aware large scale cloud data center.

In the case of network aware data center environment, the performance of branch-and-bound based Exact algorithm in terms of energy consumption, and resources utilization is better when compared to the Best-Fit algorithm. But, the execution time of Exact algorithm is high when we consider more number of VMs for the allocation at the network-aware cloud data center. Hence, Exact algorithm

is the optimal choice for the allocation of VMs to PMs at the network-aware cloud data centers (small, and medium scale). Further, the conclusion and future work directions of this thesis are described in Chapter 8.

Chapter 8

Conclusion and Future Directions

The energy efficient resources allocation in the form of VMs at the cloud data center is an important and challenging problem in the cloud computing. Further, allocation of VMs to PMs is referred to as a multi-dimensional bin packing problem. The time complexity of the problem is NP-hard/NP-complete in nature. Hence, multi-objective (minimizing energy consumption, resources wastage, and thermal temperature) and multi-constraints VM allocation algorithm is required to solve the VM allocation problem in polynomial time at the cloud data center. Further, allocation and destruction of VMs from the PMs at the cloud data center is the continuous process of a real time cloud data center environment.

Hence, energy efficient SLA aware VM migration policy is required at the cloud data center to migrate the VMs from underutilized PM to energy efficient PM. Further, there is a need of energy efficient task scheduling algorithm which not only reduces the energy consumption, but also avoids the SLA violation at the cloud data center. The energy efficient VM allocation, and migration in the network-aware data center environment is also a challenging and important problem at the data center. Hence, there is a need of energy efficient network-aware VM allocation, and migration policy at the cloud data center.

Hence, the research work in this thesis is directed towards the design and development of multi-objective, multi-constraints VM allocation, and migration policies in both non-network, and network aware cloud data center. Further, this thesis is directed towards the design and development of energy efficient SLA aware VM migration, and task scheduling policies in both non-network, and network aware cloud data center environments.

The first set of contributions of this thesis attempt to address the multi-objective (minimizing energy consumption, and resources wastage) VM allocation

problem at the non-network aware cloud data center environment. The multi-objective VM allocation to PM at the cloud data center not only saves the energy consumption, but also reduces the resources wastage at the data center. The multi-objective VM allocation using a hybrid combination of GA and CSO known as HGACSO is proposed to reduce both the energy consumption and the resources wastage at the cloud data center.

The proposed HGACSO VM allocation algorithm is effective in terms of reducing energy consumption and resources wastage by giving a near optimal solution of VM allocation problem at the data center. Further, proposed First-Fit approximation based VM migration policy migrates the VMs from underutilized PM to energy efficient PM and thus not only saves the energy consumption, but also reduces SLA violation at the cloud data center. The experimental results show that the energy consumption of the data center by the proposed HGACSO algorithm is (20%, 16%, 14%, and 8%) less than First Fit, FFD, GA, and CSO respectively. Further, the CPU utilization is (37%, 30%, 27%, and 15%) more in the case of HGACSO as compared to the First-Fit, FFD, GA, and CSO respectively. But the limitation of HGACSO is that it takes more execution time for the convergence.

The second set of contributions of this thesis attempt to address the multi-objective (minimizing energy consumption, and resources wastage) VM allocation problem at the non-network aware cloud data center environment. The proposed VM allocation using a hybrid combination of GA and PSO known as HGAPSO is proposed to reduce both the energy consumption and the resources wastage at the cloud data center. Further, proposed task scheduling, and VM migration policies based on First-Fit approximation not only save the energy consumption, but also minimize the SLA violation at the cloud data center.

The proposed task scheduling constraints not only ensure the resource requirement of the task but also reduce the CPU wastage of the PM at the cloud data center. Hence, resulting in less energy consumption and less SLA violation at the cloud data center. Further, proposed VM migration policy migrates the VMs from underutilized PM to energy efficient PM at the cloud data center. The ex-

perimental results show that the proposed HGAPSO saves 15% and 10% energy at the cloud data center when compared to GA and PSO respectively. Further, HGAPSO takes less convergence time when compared to HGACSO. But the limitation of this work is that it did not consider the thermal temperature of the PM at the cloud data center.

The third set of contributions of this thesis is to allocate multi-objective (minimizing energy consumption, resources wastage, and thermal tempature) VM allocation problem at non-network aware cloud data center environment. The proposed VM allocation algorithm is the hybrid combination of GA, PSO, and SA, designed for not only reducing the energy consumption, but also for reducing the resources wastage, and thermal temperature at the cloud data center.

Further, we reduced the limitation of the mutation operation by using SA, and thus, we improved the solution quality of HGAPSO by using HGAPSOSA. The experimental results demonstrated that the proposed HGAPSOSA algorithm consumed (19%, 13%, and 5%) less energy over GA, PSO, and HGAPSO respectively. The resources utilization by proposed HGAPSOSA was (33%, 21%, and 6%) more when compared to that of GA, PSO, and HGAPSO respectively at the cloud data center. Further, the average temperature of the data center in the case of HGAPSOSA is reduced by (10 degree, 6 degree, 3 degree) when compared to that of GA, PSO, and HGAPSOSA respectively.

The fourth set of contributions of this thesis is to allocate the mutli-dimensional VMs to PMs in the network-aware cloud data center environment. We considered the fat tree architecture for the cloud based data center. Further, we proposed branch-and-bound based exact algorithm for the allocation of VMs to PMs at the cloud data center. Before applying our proposed branch-and-bound based exact algorithm, we proposed ILP based mathematical model for the allocation of VMs to PMs at the cloud data center.

Further, for the selection of optimal number of PMs and switches for the VM allocation at the cloud data center, we proposed lower bounds. The proposed lower bounds are used to select the optimal PMs and switches in the search tree of

branch-and-bound based exact algorithm. The proposed branch-and-bound based exact algorithm will switch-off both idle PMs and idle switches and thus, resulting in less energy consumption at the cloud data center. Further, to reduce the size of the search tree space of branch-and-bound based exact algorithm we proposed some dominance rules. Thus, resulting in less computation time required for the deployment of our proposed branch-and-bound based exact algorithm.

Further, we propose an energy efficient VM migration using First-Fit approximation technique in network aware cloud data center environment. The proposed VM migration technique migrates the VMs from underutilized PM to energy efficient PM at the network aware cloud data center environment. Further, the experimental results show that the energy consumption of the PMs at the cloud data center by the proposed exact algorithm is (15%, and 12%) less for heterogeneous and homogeneous data center environments, respectively when compared to that of the Best-Fit algorithm.

Further, the energy consumption of the switches using the exact algorithm for heterogeneous and homogeneous cloud data center environments is (10%, and 8%) less than that of the Best-Fit Algorithm. While the CPU utilization is (15%, and 10%) more in the case of exact algorithm when compared to Best-Fit algorithm for heterogeneous and homogeneous cloud data center environments, respectively.

Finally, this thesis highlighted the performance evaluation of proposed VM allocation algorithms in both non-network, and network aware cloud data center environments. The experimental results show that the total energy consumption of the data center by the proposed HGAPSOSA algorithm is (8.9%, and 7.0%) less when compared to HGAPSO, and HGACSO in heterogeneous non-network aware cloud data center environment respectively.

Further, the energy consumption of the cloud data center by the proposed HGAPSOSA algorithm is (7.5%, and 7.0%) less when compared to HGAPSO, and HGACSO in homogeneous non-network aware cloud data center environments, respectively. The % of average resources utilization of the data center in the case of HGAPSOSA is (3.9%, and 3%) more when compared to HGAPSO,

and HGACSO, respectively. Further, the % of average resources utilization of the data center in the case of HGAPSOSA is (4.3%, and 3.7%) more when compared to HGAPSO, and HGACSO respectively. The execution time of proposed HGAPSOSA is slightly more as compared to HGAPSO, and it is less when compare to to that of HGACSO.

In summary, we designed and developed ILP based mathematical model of multi-objective multi-constraints VM allocation problem in non-network, and network aware cloud data center environments. Further, we designed and developed bioinspired algorithms using the hybrid combinations of GA, CSO, PSO, SA known as (HGACSO, HGAPSO, HGAPSOSA) for non-network aware cloud data center environment, and branch-and-bound based exact algorithm in network-aware cloud data center environment. Further, we designed and developed First-Fit approximation based energy efficient, SLA aware VM migration, and task scheduling policies in both non-network, and network aware cloud data center environments.

The future directions of this research are towards the design and development of VM allocation, and migration algorithms for geographically distributed federated cloud data center. Further, there is a need to design and develop an energy efficient SLA aware task scheduling policy by taking work flow based dependent task at the cloud data center.

References

Abdelaal, M. A., Ebrahim, G. A., & Anis, W. R. (2016). Network-aware resource management strategy in cloud computing environments. In *Proceeding of the 11th IEEE International Conference on Computer Engineering Systems (ICCES-2016)*, 26–31.

Ajiro, Y. & Tanaka, A. (2007). Improving packing algorithms for server consolidation. In *Proceeding of the International Conference on Computer Management Group (CMG-2007)*, 253–272.

Alsubaihi, S. & Gaudiot, J. L. (2016). Pets: Performance, energy and thermal aware scheduler for job mapping with resource allocation in heterogeneous systems. In *Proceeding of the 35th IEEE International Performance Computing and Communications Conference (IPCCC-2016)*, 1–2.

Amazon-Website (2014). Specification of ec2 virtual machines instances. Available at https://aws.amazon.com/ec2/instance-types/.

Andersen, D. G., Franklin, J., Kaminsky, M., Phanishayee, A., Tan, L., & Vasudevan, V. (2009). Fawn: A fast array of wimpy nodes. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, SOSP, 1–14.

Aransay, I., Zapater, M., Arroba, P., & Moya, J. M. (2015). A trust and reputation system for energy optimization in cloud data centers. In *Proceeding of the 8th IEEE International Conference on Cloud Computing*, 138–145.

Asemi, R., Doostsadigh, E., Ahmadi, M., & Malazi, H. T. (2015). Energy efficieny in virtual machines allocation for cloud data centers using the imperialist competitive algorithm. In *Proceedings of the 5th IEEE International Conference on Big Data and Cloud Computing 2015*, 62–67.

Beaumont, O., Eyraud-Dubois, L., Caro, C. T., & Rejeb, H. (2013). Heterogeneous resource allocation under degree constraints. *IEEE Transactions on Parallel and Distributed Systems*, *24*(5), 926–937.

Beloglazov, A. & Buyya, R. (2013). Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. *IEEE Transactions on Parallel and Distributed Systems*, *24*(7), 1366–1379.

Benini, L., Bogliolo, A., & Micheli, G. D. (2000). A survey of design techniques for system-level dynamic power management. *IEEE Transactions on VLSI System*, *8*(3), 299–316.

Bermejo, B., Guerrero, C., Lera, I., & Juiz, C. (2016). Cloud resource management to improve energy efficiency based on local nodes optimizations. *Procedia Computer Science*, *83*, 878 – 885.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Softw. Pract. Exper.*, *41*(1), 23–50.

Cardosa, M., Korupolu, M. R., & Singh, A. (2009). Shares and utilities based power consolidation in virtualized server environments. In *Proceeding of the FIP/IEEE International Symposium on Integrated Network Management (INM-2009)*, 327–334.

Caulfield, A. M., Grupp, L. M., & Swanson, S. (2009). Gordon: Using flash memory to build fast, power-efficient clusters for data-intensive applications. *SIGARCH Comput. Archit. News*, *37*(1), 217–228.

Chaudhry, M. T., Ling, T., & Manzoor, A. (2012). Considering thermal-aware proactive and reactive scheduling and cooling for green data-centers. In *Proceeding of the International Conference on Advanced Computer Science Applications and Technologies (ACSAT-2012)*, 87–91.

Chen, J., Tan, R., Wang, Y., Xing, G., Wang, X., Wang, X., Punch, B., & Colbry, D. (2014). A sensor system for high-fidelity temperature distribution forecasting in data centers. *ACM Trans. Sen. Netw.*, *11*(2), 30:1–30:25.

Chen, X., Wang, L., Zomaya, A. Y., Liu, L., & Hu, S. (2015). Cloud computing for vlsi floorplanning considering peak temperature reduction. *IEEE Transactions on Emerging Topics in Computing*, *3*(4), 534–543.

Cheng, D., Rao, J., Jiang, C., & Zhou, X. (2016). Elastic power-aware resource provisioning of heterogeneous workloads in self-sustainable datacenters. *IEEE Transactions on Computers*, *65*(2), 508–521.

Chowdhury, M. R., Mahmud, M. R., & Rahman, R. M. (2015). Implementation and performance analysis of various vm placement strategies in cloudsim. *Journal of Cloud Computing*, *4*(1), 20.

CISCO (2014). The green data center. Available at `http://www.cisco.com/c/en/us/products/collateral/switches/catalyst-2960-serise-switches/product_data_sheet0900aecd80522c0c.html`.

Coffman, Jr., E. G., Garey, M. R., & Johnson, D. S. (1997). : chapter A Survey: Approximation Algorithms for Bin Packing, 46–93. Boston, MA, USA: Publishing Co. PWC.

Dabbagh, M., Hamdaoui, B., Guizani, M., & Rayes, A. (2015). Energy-efficient resource allocation and provisioning framework for cloud data centers. *IEEE Transactions on Network and Service Management*, *12*(3), 377–391.

Dai, X., Wang, J. M., & Bensaou, B. (2016). Energy-efficient virtual machines scheduling in multi-tenant data centers. *IEEE Transactions on Cloud Computing*, *4*(2), 210–221.

Dalvandi, A., Gurusamy, M., & Chua, K. C. (2017). Application scheduling, placement, and routing for power efficiency in cloud data centers. *IEEE Transactions on Parallel and Distributed Systems*, *28*(4), 947–960.

Dell-Power-Model (2014). Power consumption specification of servers. Available at http://www.dell.com/.

Duggan, M., Duggan, J., Howley, E., & Barrett, E. (2016). An autonomous network aware vm migration strategy in cloud data centres. In *Proceeding of the International Conference on Cloud and Autonomic Computing (ICCAC-2016)*, 24–32.

Fang, W., Liang, X., Li, S., Chiaraviglio, L., & Xiong, N. (2013). Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Elsevier Journal on Computer Networks*, *57*(1), 179–196.

Fioccola, G. B., Donadio, P., Canonico, R., & Ventre, G. (2016). Dynamic routing and virtual machine consolidation in green clouds. In *Proceeding of the IEEE International Conference on Cloud Computing Technology and Science (CloudCom-2016)*, 590–595.

Gao, Y., Guan, H., Qi, Z., Hou, Y., & Liu, L. (2013). A multi-objective ant colony system algorithm for virtual machine placement in cloud computing. *Journal of Computer and System Sciences*, *79*(8), 1230–1242.

Gartner (2013). Gartner estimates ict industry accounts for 2 percent of global co2 emissions. Available at `https://www.gartner.com/newsroom/id/503867`.

Gattulli, M., Tornatore, M., Fiandra, R., & Pattavina, A. (2014). Low-emissions routing for cloud computing in ip-over-wdm networks with data centers. *IEEE Journal on Selected Areas in Communications*, *32*(1), 28–38.

Ge, R., Feng, X., & Cameron, K. W. (2005). Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, 34–34.

Gmach, D., Rolia, J., Cherkasova, L., & Kemper, A. (2009). Resource pool management: Reactive versus proactive or let's be friends. *ACM Journal on Computer Networks*, *53*(17), 2905–2922.

Gulati, A., Holler, A., Ji, M., Shanmuganathan, G., Waldspurger, C., & Zhu, X. (2012). Vmware distributed resource management: Design, implementation, and lessons learned. *VMware Technical Journal*, *1*(1), 45–64.

Gunaratne, C., Christensen, K., Nordman, B., & Suen, S. (2008). Reducing the energy consumption of ethernet with adaptive link rate (alr). *IEEE Transactions on Computers*, *57*(4), 448–461.

Guyon, D., Orgerie, A. C., Morin, C., & Agarwal, D. (2017). How much energy can green hpc cloud users save? In *2017 25th Euromicro International Conference on Parallel, Distributed and Network-based Processing (PDP)*, 416–420.

Haouari, M. & Serairi, M. (2011). Relaxations and exact solution of the variable sized bin packing problem. *Springier Journal on Computational Optimization and Applications*, *48*(2), 345–368.

He, H., Xu, G., Pang, S., & Zhao, Z. (2016). Amts: Adaptive multi-objective task scheduling strategy in cloud computing. *IEEE Journal on China Communications*, *13*(4), 162–171.

He, Y., Chen, G., Wei, W., Liu, Q., Zhang, J., Zhou, T., Zhu, P., Zhu, Y., Liu, C., & Ahuja, N. (2015). Hta corrosion resistant technology for free cooling. In *Proceeding of the 31st Thermal Measurement, Modeling Management Symposium (SEMI-THERM)*, 120–126.

Heller, B., Seetharaman, S., Mahadevan, P., Yiakoumis, Y., Sharma, P., Banerjee, S., & McKeown, N. (2010). Elastictree: Saving energy in data center networks. In *Proceedings of the 7th USENIX Conference on Networked Systems Design and Implementation (NSDI'10)*, 17–17., Berkeley, CA, USA.

Hsu, C. h. & Feng, W. c. (2005). A power-aware run-time system for high-performance computing. In *Proceedings of the ACM/IEEE Conference on Supercomputing*, IEEE Computer Society, 1–1.

IBM (2014). A report on green data center. Available at `https://www-935.ibm.com/services/us/cio/pdf/the_green_data_center_oiw03010usen.pdf`.

IBM-Power-Model (2014).

IBM-Switch-Model (2014).

Jung, G., Joshi, K. R., Hiltunen, M. A., Schlichting, R. D., & Pu, C. (2009). A cost-sensitive adaptation engine for server consolidation of multitier applications. In *Proceeding of ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, Springer, 163–183.

Karn, R. R. & Elfadel, I. A. M. (2016). Autoscaling of cores in multicore processors using power and thermal workload signatures. In *Proceeding of the 59th IEEE International Midwest Symposium on Circuits and Systems (MWSCAS)*, 1–4.

Kim, N., Cho, J., & Seo, E. (2014). Energy-credit scheduler: an energy-aware virtual machine scheduler for cloud systems. *Future Generation Computer Systems*, *32*, 128–137.

Kinger, S. & Goyal, K. (2013). Energy-efficient cpu utilization based virtual machine scheduling in green clouds. In *Proceeding of the 5th International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2013)*, 28–34.

Kreith, F. (1999). *The CRC handbook of thermal engineering*. CRC Press, Boca Raton, FL (US).

Kumar, S., Talwar, V., Kumar, V., Ranganathan, P., & Schwan, K. (2009). vmanage: Loosely coupled platform and virtualization management in data centers. In *Proceedings of the 6th International Conference on Autonomic Computing (ICAC 2009)*, ACM, 127–136., New York, NY, USA.

Kuo, C.-F. & Lu, Y.-F. (2014). Energy-efficient assignment for tasks on non-dvs heterogeneous multiprocessor system. In *Proceedings of the International Conference on Research in Adaptive and Convergent Systems (RACS 2014)*, 314–319., New York, NY, USA.

Kusic, D., Kephart, J. O., Hanson, J. E., Kandasamy, N., & Jiang, G. (2008). Power and performance management of virtualized computing environments via lookahead control. In *Proceeding of the International Conference on Autonomic Computing (ICAC 2008)*, Chicago, IL, USA, 3–12.

Kveton, B., Gandhi, P., Theocharous, G., Mannor, S., Rosario, B., & Shah, N. (2007). Adaptive timeout policies for fast fine-grained power management. In *Proceedings of the National Conference on Artificial Intelligence*, Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press, volume 22, 1795.

Lawey, A. Q., El-Gorashi, T. E. H., & Elmirghani, J. M. H. (2014). Distributed energy efficient clouds over core networks. *IEEE Journal on Lightwave Technology*, *32*(7), 1261–1281.

Lee, E. K., Viswanathan, H., & Pompili, D. (2012). Vmap: Proactive thermal-aware virtual machine allocation in hpc cloud datacenters. In *Proceeding of the 19th IEEE International Conference on High Performance Computing (HIPC-2012)*, 1–10.

Lefevre, L. & Orgerie, A.-C. (2010). Designing and evaluating an energy efficient cloud. *The Journal of Supercomputing*, *51*(3), 352–373.

Li, D., Yu, Y., He, W., Zheng, K., & He, B. (2015). Willow: Saving data center network energy for network-limited flows. *IEEE Transactions on Parallel Distributed Systems*, *26*(9), 2610–2620.

Li, H., Li, J., Yao, W., Nazarian, S., Lin, X., & Wang, Y. (2017). Fast and energy-aware resource provisioning and task scheduling for cloud systems. In *Proceeding of the 18th IEEE Iternational Symposium on Quality Electronic Design (ISQED-2017)*, 174–179.

Li, H., Zhu, H., Ren, G., Wang, H., Zhang, H., & Chen, L. (2016). Energy-aware scheduling of workflow in cloud center with deadline constraint. In *Proceeding of the 12th IEEE International Conference on Computational Intelligence and Security (CIS-2016)*, 415–418.

Li, X., Garraghan, P., JIANG, X., Wu, Z., & Xu, J. (2017). Holistic virtual machine scheduling in cloud datacenters towards minimizing total energy. *IEEE Transactions on Parallel and Distributed Systems*, *PP*(99), 1–1.

Li, X., Jiang, X., & He, Y. (2014). Virtual machine scheduling considering both computing and cooling energy. In *Proceeding of the 6th IEEE International Symposium on Cyberspace Safety and Security (CSS-2014)*, 244–247.

Liu, J., Zhang, N., Kang, C., Kirschen, D. S., & Xia, Q. (2017). Decision-making models for the participants in cloud energy storage. *IEEE Transactions on Smart Grid*, *PP*(99), 1–1.

Liu, N., Dong, Z., & Rojas-Cessa, R. (2013). Task scheduling and server provisioning for energy-efficient cloud-computing data centers. In *Proceeding of the 33rd IEEE International Conference on Distributed Computing Systems Workshops (ICDCSW-2013)*, 226–231.

Mahadevan, P., Sharma, P., Banerjee, S., & Ranganathan, P. (2009). Energy aware network operations. In *IEEE INFOCOM Workshops 2009*, 1–6.

Martello, S. & Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. New York, NY, USA: John Wiley & Sons, Inc.

Minas, L. & Ellison, B. (2009). *Energy efficiency for information technology: How to reduce power consumption in servers and data centers*. Intel Press.

Nathuji, R., Isci, C., & Gorbatov, E. (2007). Exploiting platform heterogeneity for power efficient data centers. In *Proceeding of the 4th IEEE International Conference on Autonomic Computing*, 5–5., Los Alamitos, CA, USA.

Nathuji, R. & Schwan, K. (2007). Virtualpower: coordinated power management in virtualized enterprise systems. In *SIGOPS Operating Systems Review*, volume 41, 265–278.

Nedevschi, S., Popa, L., Iannaccone, G., Ratnasamy, S., & Wetherall, D. (2008). Reducing network energy consumption via sleeping and rate-adaptation. In

Proceedings of the 5th USENIX Symposium on Networked Systems Design and Implementation, USENIX Association, NSDI'08, 323–336., Berkeley, CA, USA.

Nguyen, H., Shen, Z., Gu, X., Subbiah, S., & Wilkes, J. (2013). Agile: Elastic distributed resource scaling for infrastructure-as-a-service. In *Proceeding of the USENIX International Conference on Automated Computing (ICAC13). San Jose, CA*.

Oxley, M. A., Jonardi, E., Pasricha, S., Maciejewski, A. A., Koenig, G. A., & Siegel, H. J. (2014). Thermal, power, and co-location aware resource allocation in heterogeneous high performance computing systems. In *Proceeding of the IEEE International Green Computing Conference (IGCC-2014)*, IEEE Computer Society, 1–10., Los Alamitos, CA, USA.

Pantazoglou, M., Tzortzakis, G., & Delis, A. (2016). Decentralized and energy-efficient workload management in enterprise clouds. *IEEE Transactions on Cloud Computing*, *4*(2), 196–209.

Pisinger, D. & Sigurd, M. (2005). The two-dimensional bin packing problem with variable bin sizes and costs. *Discrete Optimization*, *2*(2), 154–167.

Polverini, M., Cianfrani, A., Ren, S., & Vasilakos, A. V. (2014). Thermal-aware scheduling of batch jobs in geographically distributed data centers. *IEEE Transactions on Cloud Computing*, *2*(1), 71–84.

Prevost, J. J., Nagothu, K., Kelley, B., & Jamshidi, M. (2011). Prediction of cloud data center networks loads using stochastic and neural models. In *Proceeding of 6th IEEE International Conference on System of Systems Engineering (SSE-2011)*, 276–281.

Raghavendra, R., Ranganathan, P., Talwar, V., Wang, Z., & Zhu, X. (2008). No power struggles: Coordinated multi-level power management for the data center. *SIGARCH Comput. Archit. News*, *36*(1), 48–59.

Shu, W., Wang, W., & Wang, Y. (2014). A novel energy-efficient resource allocation algorithm based on immune clonal optimization for green cloud computing. *EURASIP Journal on Wireless Communications and Networking*, *64*(1), 64.

Snowdon, D. C., Ruocco, S., & Heiser, G. (2005). Power management and dynamic voltage scaling: Myths and facts. In *Proceedings of the workshop on power aware real-time computing*, volume 12.

Son, J., Dastjerdi, A. V., Calheiros, R., & Buyya, R. (2017). Sla-aware and energy-efficient dynamic overbooking in sdn-based cloud data centers. *IEEE Transactions on Sustainable Computing*, *PP*(99), 1–1.

Song, C., Wang, C., Ahuja, N., Zhou, X., & Daniel, A. (2015). Optimize data center management with multi-tier thermal-intelligent workload placement. In *Proceeding of the 31st Thermal Measurement, Modeling Management Symposium (SEMI-THERM-2015)*, 25–30.

Song, Y., Wang, H., Li, Y., Feng, B., & Sun, Y. (2009). Multi-tiered on-demand resource scheduling for vm-based data center. In *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, IEEE Computer Society, 148–155.

Stillwell, M., Schanzenbach, D., Vivien, F., & Casanova, H. (2009). Resource allocation using virtual clusters. In *Proceedings of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid-2009)*, 260–267.

Su, C. L., Tsui, C. Y., & Despain, A. M. (1994). Saving power in the control path of embedded processors. *IEEE Design Test of Computers*, *11*(4), 24–31.

Tighe, M. & Bauer, M. (2014). Integrating cloud application autoscaling with dynamic vm allocation. In *Proceeding of the IEEE International Conference on Network Operations and Management Symposium (NOMS-2014)*, 1–9.

Tiwari, V., Ashar, P., & Malik, S. (1993). Technology mapping for lower power. In *Proceedings of the 30th ACM International Design Automation Conference*, DAC '93, 74–79., New York, NY, USA.

Valentini, G., Lassonde, W., Khan, S. U., Min-Allah, N., Madani, S. A., Li, J., Zhang, L., Wang, L., Ghani, N., Kolodziej, J., Li, H., Zomaya, A. Y., Xu, C.-Z., Balaji, P., Vishnu, A., Pinel, F., Pecero, J. E., Kliazovich, D., & Bouvry, P. (2013). An overview of energy efficiency techniques in cluster computing systems. *Cluster Computing*, *16*(1), 3–15.

Vasudevan, V., Andersen, D., Kaminsky, M., Tan, L., Franklin, J., & Moraru, I. (2010). Energy-efficient cluster computing with fawn: Workloads and implications. In *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*, ACM, 195–204., New York, NY, USA.

Verma, A., Ahuja, P., & Neogi, A. (2008). pmapper: power and migration cost aware application placement in virtualized systems. In *ACM/IFIP/USENIX International Conference on Distributed Systems Platforms and Open Distributed Processing*, Springer, 243–264.

Wang, J. V., Cheng, C.-T., & Chi, K. T. (2015). A power and thermal-aware virtual machine allocation mechanism for cloud data centers. In *Communication Workshop (ICCW), 2015 IEEE International Conference on*, IEEE, 2850–2855.

Wang, S., Liu, Z., Zheng, Z., Sun, Q., & Yang, F. (2013). Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers. In *Parallel and Distributed Systems (ICPADS), 2013 International Conference on*, IEEE, 102–109.

Wang, S., Zhou, A., Hsu, C. H., Xiao, X., & Yang, F. (2016). Provision of data-intensive services through energy- and qos-aware virtual machine placement in national cloud data centers. *IEEE Transactions on Emerging Topics in Computing*, *4*(2), 290–300.

Wang, T., Xia, Y., Muppala, J., & Hamdi, M. (2017). Achieving energy efficiency in data centers using an artificial intelligence abstraction model. *IEEE Transactions on Cloud Computing*, *PP*(99), 1–1.

Wang, Z. & Zhang, Y.-Q. (2011). Energy-efficient task scheduling algorithms with human intelligence based task shuffling and task relocation. In *Proceedings of the IEEE/ACM International Conference on Green Computing and Communications*, IEEE Computer Society, 38–43.

Whitley, D., Gordon, V. S., & Mathias, K. (1994). Lamarckian evolution, the baldwin effect and function optimization. *3*, 5–15.

Wu, C.-M., Chang, R.-S., & Chan, H.-Y. (2014). A green energy-efficient scheduling algorithm using the dvfs technique for cloud datacenters. *Future Generation Computer Systems*, *37*, 141–147.

Wu, Z., Li, X., Garraghan, P., Jiang, X., Ye, K., & Zomaya, A. Y. (2016). Virtual machine level temperature profiling and prediction in cloud datacenters. In *Proceeding of the 36th International Conference on Distributed Computing Systems (ICDCS-2016)*, 735–736.

Xie, R., Jia, X., Yang, K., & Zhang, B. (2013). Energy saving virtual machine allocation in cloud computing. In *Proceeding of the 33rd IEEE International Conference on Distributed Computing Systems Workshops*, 132–137.

Xiong, A.-p. & Xu, C.-x. (2014). Energy efficient multiresource allocation of virtual machine based on pso in cloud data center. *Mathematical Problems in Engineering*, *2014*.

Xu, J. & Fortes, J. A. (2010). Multi-objective virtual machine placement in virtualized data center environments. In *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*, IEEE, 179–188.

Xu, M., Shang, Y., Li, D., & Wang, X. (2013). Greening data center networks with throughput-guaranteed power-aware routing. *Computer Network*, *57*(15), 2880–2899.

Yan, F., Lee, T. T., Hu, W., undefined, undefined, undefined, & undefined (2017). Congestion-aware embedding of heterogeneous bandwidth virtual data centers

with hose model abstraction. *IEEE/ACM Transactions on Networking*, *25*(2), 806–819.

Yao, J., Guan, H., Luo, J., Rao, L., & Liu, X. (2015). Adaptive power management through thermal aware workload balancing in internet data centers. *IEEE Transactions on Parallel and Distributed Systems*, *26*(9), 2400–2409.

List of Publications

International Journals

1. N. K. Sharma, and G. R.M. Reddy, "Multi Objective Energy Efficient Virtual Machine Allocation at the Cloud Data Center "in IEEE Transactions on Services Computing, vol. PP, no. 99, pp. 1-1., July 2016.

2. N. K. Sharma and G. R.M. Reddy, "A Novel Energy Efficient Network-Aware Virtual Machines Allocation and Migration at Cloud Data Center", in IEEE Transactions on Services Computing (Under Review).

3. N. K. Sharma and G. Ram Mohana Reddy, A Hybrid Bio-inspired Energy Efficient Thermal-Aware Virtual Machines Allocation and Migration at the Cloud Data Center, Journal of Grid Computing (Under Review).

International Conferences

1. N. K. Sharma, and G. R.M. Reddy, A Novel Approach for Multi-Dimensional Variable Sized Virtual Machine Allocation and Migration at Cloud Data Center, 9th IEEE International Conferences on Communication Systems, and Networks (COMSNETS-2017) 8-9 January 2017, IISC Bangalore, India.

2. N. K. Sharma, and G. R.M. Reddy, On Demand Virtual Machine Allocation and Migration at Cloud Data Center using Hybrid of Cat Swarm Optimization and Genetic Algorithm, 5th IEEE International Conference on Eco-Friendly Computing and Communication Systems (ICECCS-2016), 8-9 December 2016, Maulana Azad National Institute of Technology MANIT, Bhopal, India.

3. N. K. Sharma, and G. R.M. Reddy, Multi-Objective Resources Allocation Using Improved Genetic Algorithm at Cloud Data Center, 5th IEEE International Conference of Cloud Computing for Emerging Markets (CCEM-2016), 19-21 October 2016 Conduced by IBM Bangalore, India.

4. N. K. Sharma, and G. R.M. Reddy, A Novel Energy Efficient Resources Allocation using Hybrid Approach of Genetic DVFS With Bin Packing, 6th IEEE International Conference on Communication System and Network Technology (CSNT-2015), 4-6 April, 2015, MIR Lab Gwalior, India.

5. N. K. Sharma, and G. R.M. Reddy, Novel Energy Efficient Virtual Machine Allocation at Data Center Using Genetic Algorithm, 3rd IEEE International Conference on Signal Processing, Communication and Networking (ICSCN-2015), 26-28 March 2015, Anna University, Chennai, India.

6. N. K. Sharma and G. R. M. Reddy, Energy Efficient Virtual Machine Allocation and Migration at Cloud Data Center, 13th IEEE International Conference on Services Computing (IEEE SCC 2016), 25-30 June 2016, San Francisco, USA (Accepted).

7. N. K. Sharma and G. R. M. Reddy, "Energy Efficient Quality of Service Aware Virtual Machine Migration in Cloud Computing", 4th IEEE International Conference on Recent Advances in Information Technology (RAIT 2018), March 15-17, 2018, IIT(ISM) Dhanbad, India.

Brief Bio-Data

Neeraj Kumar Sharma

Research Scholar

Department of Information and Technology

National Institute of Technology Karnataka, Surathkal

P.O. Srinivasnagar

Manglore, 575025

Phone: 07353092246

Email: neeraj16ks@gmail.com

Permanent Address

Neeraj Kumar Sharma

26, Srvodya Colony, Bhandariya Road

khnadwa, 450001

Madhya Pradesh

Qualification

M.E. Software IET, Devi Ahilya Vishwavidyalaya, Indore, Madhya Pradesh, 2010.

B.E. Computer Science and Engineering, Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal, Madhya Pradesh, 2005.