

PSO BASED SCHEDULING TECHNIQUES TO IMPROVE QoS PARAMETERS IN CLOUD COMPUTING

Ph.D. THESIS

by

MOHIT KUMAR



**DEPARTMENT OF PAPER TECHNOLOGY
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)
MAY, 2018**

PSO BASED SCHEDULING TECHNIQUES TO IMPROVE QoS PARAMETERS IN CLOUD COMPUTING

A THESIS

*Submitted in partial fulfilment of the
requirements for the award of the degree*

of

DOCTOR OF PHILOSOPHY

in

COMPUTER SCIENCE AND ENGINEERING

by

MOHIT KUMAR



**DEPARTMENT OF PAPER TECHNOLOGY
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE – 247667 (INDIA)
MAY, 2018**



**©INDIAN INSTITUTE OF TECHNOLOGY ROORKEE, ROORKEE-2018
ALL RIGHTS RESERVED**



INDIAN INSTITUTE OF TECHNOLOGY ROORKEE ROORKEE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in the thesis entitled “**PSO BASED SCHEDULING TECHNIQUES TO IMPROVE QoS PARAMETERS IN CLOUD COMPUTING**” in partial fulfilment of the requirements for the award of the Degree of Doctor of Philosophy and submitted in the Department of Paper Technology of the Indian Institute of Technology Roorkee, Saharanpur Campus, Saharanpur is an authentic record of my own work carried out during a period from December, 2014 to May, 2018 under the supervision of Dr. S.C. Sharma, Professor, Indian Institute of Technology Roorkee, Saharanpur Campus, Saharanpur.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institution.

(MOHIT KUMAR)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

(S.C. Sharma)
Supervisor

Dated:

ABSTRACT

Cloud computing provides the services either in the form of software application or hardware infrastructure on the basis of pay per use over the internet. There are lots of challenges in the field of cloud computing due to improper management of cloud resources. Scheduling challenges occurs due to dispersion, uncertainty and heterogeneity of resources that are not resolved with traditional resource management mechanisms. Over provisioning and under provisioning types of problem occurs in cloud environment due to which cloud resources are not utilizing properly. One of the challenging features of cloud computing is to provide on demand huge number of resources to the user as per their need (elasticity and scalability) and satisfy the quality of service (QoS) parameters like reliability, elasticity, deadline, priority of task etc. to minimizing the makespan time and task rejection ratio. Therefore we need an efficient scheduling algorithm that utilize the cloud resources properly and improve the QoS parameters.

Following are the major objectives of the thesis:

1. To develop a load balancing algorithm with elasticity concept that balances the workload among the virtual machine and analyzes the QoS parameters (execution time, makespan time, task rejection ratio) considering deadline as constraint.
2. To propose a dynamic transfer function based modified binary PSO to solve the real world discrete problems and analyze the QoS parameters in cloud computing
3. To Develop a PSO based multi-objective scheduling algorithm to analyze the effect of execution time, execution cost and energy consumption in the field of cloud computing.

Objective 1: Development a load balancing algorithm with elasticity

Most of the published scheduling algorithm deals with only one parameter either scheduling the upcoming tasks to optimize the required parameter or scalability within user defined deadline constraint. It is difficult to predict and calculate all possible task-resource mapping in cloud environment. One of the important aspects of scheduling is to balance the workload among the cloud resources (virtual machines) and monitors the load at each virtual machine continuously.

Contribution:

To solve the issue of load balancing with elasticity in cloud computing, we have developed a

dynamic scheduling algorithm that balances the workload among all the virtual machines with elastic resource provisioning and deprovisioning based on the last optimal k-interval considering deadline as constraint. Developed algorithm distributes the tasks and adds the cloud resource if task rejection ratio is more than the service level agreement (SLA) defined threshold value. The computational results proved that the developed algorithm decreases the makespan time (up to 6% in comparison with min-min, up to 15% from shortest job first and up to 20 % from first come first serve algorithm) and task meet with deadline ratio of developed algorithm is up to 93% compare to other algorithm (min-min up to 80.8 %, shortest-job-first up to 75% and first come first serve 73%) in all conditions.

Objective 2: Development of a dynamic transfer function based modified BPSO

Binary particle swarm optimization (BPSO) is used to solve discrete optimization problems but it does not maintain the good balance between exploration and exploitation for transfer function.

Contribution:

To overcome this problem, we have developed a dynamic transfer function (TF_p-BPSO) for BPSO which provides better exploration at the early stage by high flipping of bit of particle position for any velocity and it has the ability to move from exploration to exploitation in the intermediate stage of execution. It provides the stronger exploitation (less probability of flipping of bits) in the last stage of execution. Results proved that developed TF_p-BPSO algorithm reduces the execution time (up to 10% compare with BPSO and up to 20% compare with FCFS) makespan time (up to 15% comparison with BPSO and up to 32% comparison with FCFS). Throughput has been increased (up to 20% compare with BPSO and up to 40% compare with FCFS) in better way than existing algorithm like first come first serve and BPSO.

Objective 3: PSO based cost and energy efficient scheduling algorithm with deadline constraint

The main purpose of cloud service provider is to maximize the profit and minimize the energy consumption from cloud infrastructure while cloud users want to execute their applications in minimum time and execution cost. There is always a conflict between execution cost and time parameters because low cost resources are less computation oriented than expensive. So a trade-off solution is required to optimize both the parameters at the same time. The rapid growths in the demand of computational power tends to massive growth in cloud data centers

and require large amount of energy consumption in cloud data centers which has become a serious threat to the environment. To reduce the energy consumption in cloud computing is a challenging problem due to incompatibility between workstation (physical machine) and unpredictable users demand.

Contribution:

We have proposed a resource allocation model for processing the applications efficiently and particle swarm optimization based scheduling algorithm that not only optimize execution cost and time but also reduce the energy consumption of cloud data centers considering deadline as constraint. The developed algorithm has been simulated at cloudsim and it is observed that it reduces the execution time (up to 8% from existing PSO, 15% honey bee, 20 % min-min algorithms), makespan time (up to 10% from existing PSO, 20% honey bee, 18.8 % min-min algorithms) execution cost (up to 8% from existing PSO, 12% honey bee, 15 % min-min algorithms), task rejection ratio (up to 8% from existing PSO, 23% honey bee, 19.6 % min-min algorithms), energy consumption (up to 7% from existing PSO, 11% honey bee, 18 % min-min algorithms) and increase the throughput (up to 6.5% from existing PSO, 9.8% honey bee, 12.6 % min-min algorithms) in comparison to PSO, honey bee and min-min algorithm.

The thesis is organized into six chapters including the introduction.

The thesis is organized into six chapters.

CHAPTER 1: INTRODUCTION

In the introduction part, details with general concepts of cloud services, motivation, research challenges, objectives and contributions of this thesis.

CHAPTER 2: FUNDAMENTALS AND SCHEDULING TECHNIQUES

In this chapter, we discuss the existing resource provisioning techniques, advantages of resource provisioning in the field of cloud computing, static and dynamic scheduling algorithm, classification of scheduling algorithms in terms of heuristic, meta-heuristic and hybrid algorithm. Further resource allocation model and simulation tool are discussed in brief that are used to measure the performance of the scheduling algorithm.

CHAPTER 3: LOAD BALANCING WITH ELASTICITY USING HEURISTIC TECHNIQUE

The details about the development of a cloud resource broker architecture and dynamic scheduling algorithm that is able to automatically manage and monitor the virtual machines to minimize the QoS parameters based on the last optimal k-interval of considering deadline as constraint and simultaneously fulfill the objective of elasticity in cloud environment have been discussed.

CHAPTER 4: DYNAMIC TRANSFER FUNCTION BASED MODIFIED BINARY PSO FOR SCHEDULING THE TASKS

Development of dynamic transfer function (TF_p-BPSO) based BPSO algorithm that provides better exploration at the early stage, its move from exploration to exploitation in the intermediate stage of execution and provides the stronger exploitation in the last stage of execution to improve QoS parameters have been discussed.

CHAPTER 5: MULTI-OBJECTIVE SCHEDULING ALGORITHM USING PSO

Particle swarm optimization (PSO) based scheduling algorithm and resource allocation model for improvement of QoS parameters have been discussed.

CHAPTER 6- CONCLUSIONS AND FUTURE WORK

This chapter concludes the work reported in the thesis and discusses about future research directions

ACKNOWLEDGEMENTS

It is with great pleasure and felicity that I would like to express thanks to all people who have made this thesis possible.

First of all, I would like to thank my supervisor, **Dr. S. C. Sharma**, Professor, Computer Science and Engineering Discipline, Indian Institute of Technology Roorkee. I feel privileged to express my deep sense of gratitude to my supervisor for his valuable guidance, endless support and constant encouragement throughout the course of my research work. His truly scientific intuition and broad vision inspired and enriched my growth as a student and researcher. The critical comments, rendered by him during the discussions are deeply acknowledged. I humbly acknowledge a lifetime's gratitude to him.

I express my regards to Prof. Y.S. Negi, Head of the Department for his support and encouragement. Special thanks to my SRC members: Dr. S. P. Singh (Professor), Dr. Dharam Dutt (Professor) and Dr. Millie Pant (Associate Professor) for spending their valuable time during the discussions over the seminars. I am indebted to the Department of Paper Technology, IIT Roorkee, to all its faculty and staff for their academic support and encouragement. I am very thankful to University Grants Commission (UGC), India, for providing me financial assistantship during my research work at IIT Roorkee.

I would like to express my gratitude to all member of wireless and cloud computing lab. Thanks to Mr. Akhtar Hussain, Mr. Santar Pal Singh, Mr. Tushar Bhardwaj, Mr. Ram Kumar, Mr. Kalka Dubey, Mr. Kuldeep Tripathi, Mr. Trilok Saini, Mr. Ashish Mohan Yadav, Mr. Sanjeev Kumar and Mr. Saureng Kumar. Apart from these, I would like to thanks Mr. Suryakant, Ms. Neetu Kushwaha, Mr. Anubhav Goel and Mr. Manoj Bhatt for their valuable suggestions and support that helped me a lot to accomplish my research work.

A special thanks to my family. Words cannot express my gratitude towards my parents for all of the sacrifices that they have made. I want to thank my wife, Rubi Saini for supporting me in everything throughout all my doctoral work and my naughty son Manan Saini for all the happiness and love that he brings to my life.

I would like to express my sincere thanks to all anonymous reviewers of the various conferences and journals. Their constructive comments have helped me to shape the course of my research work.

Finally, I would like to thank all the readers of this work, since any piece of academia is useful if it is read and understood by others so that it can become a bridge for further research.

With profound gratitude, love and devotion, I dedicate this thesis to my family.

(Mohit Kumar)



TABLE OF CONTENTS

	CANDIDATE'S DECLARATION	
	ABSTRACT	I-IV
	ACKNOWLEDGEMENT	V-VI
	TABLE OF CONTENTS	VII-X
	LIST OF FIGURES	XI-XIII
	LIST OF TABLES	XV-XVI
	SYMBOLS	XVII
	LIST OF ACRONYMS	XIX-XX
	GLOSSARY	XXI-XXII
	LIST OF PUBLICATIONS	XXIII- XXIV
1	INTRODUCTION	1
1.1	Overview	1
1.2	Motivation	6
1.3	Objectives	8
1.4	Organization and Contribution	9
1.5	References	11
2	FUNDAMENTALS AND SCHEDULING TECHNIQUES	13
2.1	Resource Management in cloud computing	13
2.2	Cloud Resource provisioning	14
2.2.1	Need of resource provisioning	14
2.2.2	Advanced reservation	14
2.2.3	On demand resources allocation	15

2.2.4	Spot Instances	15
2.2.5	Advantages of cloud resource provisioning	15
2.3	Scheduling	16
2.3.1	Static Scheduling Algorithm	18
2.3.2	Dynamic Task Scheduling Algorithm	18
2.3.3	Online and offline (Batch) mode scheduling	19
2.3.4	Pre-emptive and Non-pre-emptive scheduling	20
2.4	Classification of scheduling scheme in cloud computing	20
2.4.1	Heuristic scheduling algorithm	20
2.4.2	Meta-heuristic scheduling algorithm	21
2.4.2.1	Particle Swarm Optimization	22
2.4.3	Hybrid scheduling algorithm	25
2.4.4	Benefit of Resource Scheduling	25
2.5	Resource Allocation	26
2.5.1	Classification of resource allocation	28
2.6	Simulation Tool used in cloud computing	30
2.7	Summary	32
2.8	References	32
3	LOAD BALANCING WITH ELASTICITY USING HEURISTIC TECHNIQUE	37
3.1	Concept of Load Balancing and Elasticity	37
3.2	Contribution	38
3.3	Related Work and Research Gap	38
3.4	Problem Formulation	42
3.5	Proposed Architecture	45
3.6	Dynamic Load balancing algorithm with Elasticity	49

3.7	Analysis and Comparison of Results	54
3.7.1	Makespan Time Calculations	54
3.7.2	Number of Task Meets to Deadline	56
3.7.3	Provisioning and Deprovisioning (elasticity) of Resources	58
3.7.4	Scalability	60
3.8	Summary	61
3.9	References	62
4	DYNAMIC TRANSFER BASED MODIFIED BINARY PSO FOR SCHEDULING THE TASKS	65
4.1	Concept of Task Scheduling and Binary PSO	65
4.2	Contribution	67
4.3	Related Work and Research Gap	67
4.3.1	Sigmoid transfer function	71
4.3.2	Linear normalized transfer function	72
4.3.3	V-Shape transfer function	72
4.4	Problem Formulation	73
4.5	Proposed Cloud Architecture	76
4.6	BPSO based modified transfer function (TF _p -BPSO)	78
4.7	Modified BPSO (Dynamic transfer based (TF _p -BPSO)) based scheduling algorithm	80
4.8	Analysis and comparison of simulation results	83
4.8.1	Execution time of tasks	83
4.8.2	Makespan Time of Tasks	87
4.8.3	Convergence rate	89
4.8.4	Throughput	90
4.9	Summary	91
4.10	References	92
5	MULTI-OBJECTIVE SCHEDULING ALGORITHM USING PSO	97
5.1	Energy Consumption and Execution Cost	97

5.2	Contribution	98
5.3	Related Work and Research Gap	98
5.4	Proposed Resource allocation model for Cloud environment	103
5.5	Problem Statement and Formulation	105
5.6	Particle Swarm Optimization (PSO)	112
5.6.1	PSO working methodology	113
5.7	Modified proposed PSO	114
5.7.1	Exploration and Exploitation	114
5.7.2	Flow chart of modified PSO algorithm	115
5.7.3	Proposed PSO algorithm for multi-objective scheduling	117
5.8	Analysis, Comparison and Simulation Results	120
5.8.1	Execution Time Calculation	121
5.8.2	Makespan Time calculation	122
5.8.3	Task rejection ratio	124
5.8.4	Execution cost calculations	125
5.8.5	Throughput	128
5.8.6	Energy consumption	129
5.9	Summary	130
5.10	References	131
6	CONCLUSIONS AND FUTURE WORK	135
6.1	Conclusions	135
6.2	Future Work	136

LIST OF FIGURES

Figure 1.1.	Cloud computing service layers with example	2
Figure 1.2.	Cloud deployment models	3
Figure 1.3.	Resource Provisioning in non-cloud computing	6
Figure 1.4.	Resource Provisioning in cloud computing environment	7
Figure 1.5.	Brief layout of research plan	10
Figure 2.1.	Resource management in cloud computing	13
Figure 2.2.	Resource provisioning plans	14
Figure 2.3.	Flowchart of resource provisioning and scheduling in cloud computing	17
Figure 2.4	Classification of scheduling scheme in cloud computing	21
Figure 2.5.	Flying of a particle in the search space	23
Figure 2.6.	Flow chart of PSO algorithm	24
Figure 2.7.	Percentage of meta-heuristic algorithm used in scheduling problem	25
Figure 2.8.	Process of efficient resource allocation in cloud computing	27
Figure 2.9.	Taxonomy of resource allocation in cloud computing	29
Figure 2.10.	Tools used in cloud computing for scheduling	31
Figure 3.1.	Proposed cloud resource broker architecture for load balancing with elasticity	46
Figure 3.2.	Cloud task scheduling model	49
Figure 3.3.	Flow chart to determine the overloaded and underloaded virtual machines	53
Figure 3.4.	Basic architecture of cloudsim	54
Figure 3.5.	Makespan time comparisons between proposed algorithms with FCFS, SJF, dynamic min-min	55

Figure 3.6.	Task acceptance ratio comparison between proposed algorithm with FCFS, SJF and dynamic min-min	57
Figure 3.7.	Task acceptance ratio comparisons between proposed algorithm with FCFS, SJF, dynamic min-min	58
Figure 3.8.	Provisioning and deprovisioning of cloud resource based on upcoming tasks	59
Figure 3.9.	Provisioning and deprovisioning of cloud resource based on upcoming tasks	59
Figure 3.10.	Scale-out of cloud resource based on upcoming task	61
Figure 4.1.	Sigmoid transfer functions	71
Figure 4.2.	Proposed Cloud Task scheduling architecture	77
Figure 4.3.	Comparison of proposed transfer function with sigmoid transfer function	79
Figure 4.4.	Proposed dynamic transfer function based algorithm for task scheduling	82
Figure 4.5.	Execution Time comparison proposed BPSO with BPSO and FCFS at fixed Tasks	86
Figure 4.6.	Execution Time comparison proposed BPSO with BPSO and FCFS	86
Figure 4.7.	Makespan time comparisons between FCFS, BPSO and proposed BPSO	87
Figure 4.8.	Makespan time comparisons between FCFS, BPSO and proposed BPSO	88
Figure 4.9.	Makespan time comparisons between FCFS, BPSO and proposed BPSO	89
Figure 4.10.	Convergence rate comparisons between BPSO and proposed BPSO	90

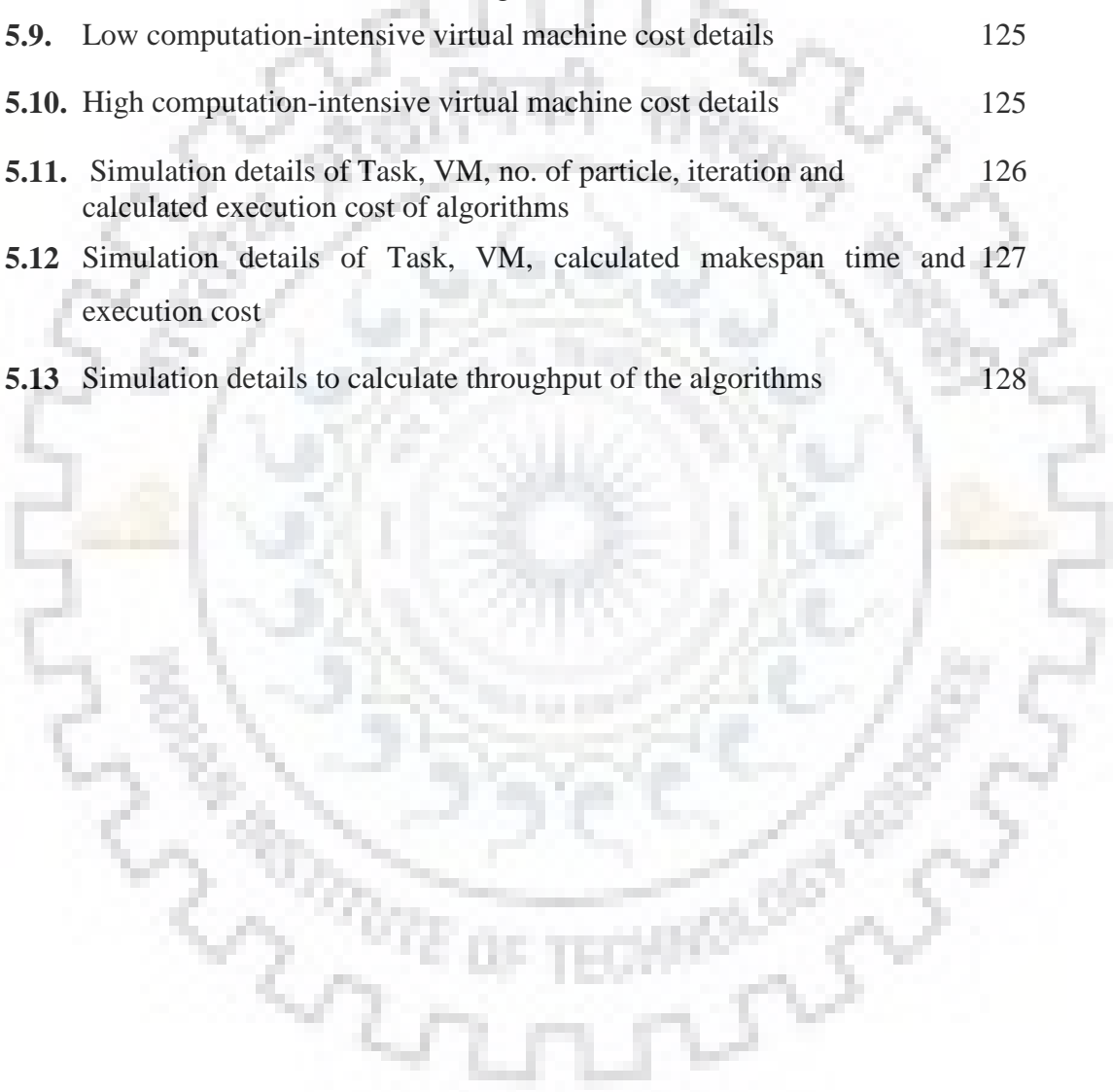
Figure 4.11.	Throughput comparisons between FCFS, BPSO and proposed BPSO	91
Figure 5.1.	Resource allocation model for cloud environment	104
Figure 5.2.	PSO working methodology	113
Figure 5.3.	Flowchart of the modified PSO (proposed) algorithm	116
Figure 5.4.	Steps of proposed PSO based algorithm for scheduling algorithm	119
Figure 5.5.	Execution time comparisons between Proposed PSO vs. PSO, Honey Bee and Min-Min	122
Figure 5.6.	Makespan time comparisons between proposed PSO vs. PSO, Honey Bee and Min-Min algorithm	123
Figure 5.7.	Task rejection ratio comparisons between proposed modified PSO vs. PSO, Honey Bee and Min-Min algorithm	124
Figure 5.8.	Execution cost comparison between proposed PSO vs. PSO, Honey Bee and Min-Min algorithm	126
Figure 5.9.	Results of makespan time and execution cost at 1000 tasks using proposed PSO	127
Figure 5.10.	Throughput comparisons between proposed PSO vs. PSO, Honey Bee and Min-Min algorithm	128
Figure 5.11.	Energy Consumption comparisons between proposed PSO vs. PSO, Honey Bee and Min-Min algorithm	129



LIST OF TABLES

Table 2.1.	Differences between static and dynamic algorithms	19
Table 3.1.	Literature review on dynamic scheduling, load balancing & elasticity based algorithm	40
Table 3.2.	Notations and their description	43
Table 3.3.	VM Properties	54
Table 3.4.	Task Properties	55
Table 3.5.	Task with deadline	57
Table 3.6.	Task with deadline	57
Table 4.1.	Literature review on meta-heuristic based scheduling algorithm	69
Table 4.2.	Notation and description	74
Table 4.3.	Position matrix	81
Table 4.4.	VM properties	84
Table 4.5.	Tasks properties	84
Table 4.6.	Execution time comparison between BPSO and proposed BPSO	84
Table 4.7.	Execution time comparison between FCFS, BPSO and developed BPSO	85
Table 4.8.	Throughput comparison between FCFS, BPSO and developed BPSO	91
Table 5.1.	Literature Review of Traditional and Meta-heuristic algorithm with their limitations	100
Table 5.2.	Notation and their description are used in problem formulation and fitness function	107
Table 5.3.	Position of the particles	118
Table 5.4.	Mapping task with cloud resources	118
Table 5.5.	VM properties	120

Table 5.6.	Detail of task, VM, PSO parameters and calculated execution time of algorithms	121
Table 5.7.	Detail of task, VMs, PSO parameters and calculated makespan time of algorithms	123
Table 5.8.	Detail of task, VM, PSO parameters and calculated task rejection ratio of algorithms	124
Table 5.9.	Low computation-intensive virtual machine cost details	125
Table 5.10.	High computation-intensive virtual machine cost details	125
Table 5.11.	Simulation details of Task, VM, no. of particle, iteration and calculated execution cost of algorithms	126
Table 5.12	Simulation details of Task, VM, calculated makespan time and execution cost	127
Table 5.13	Simulation details to calculate throughput of the algorithms	128

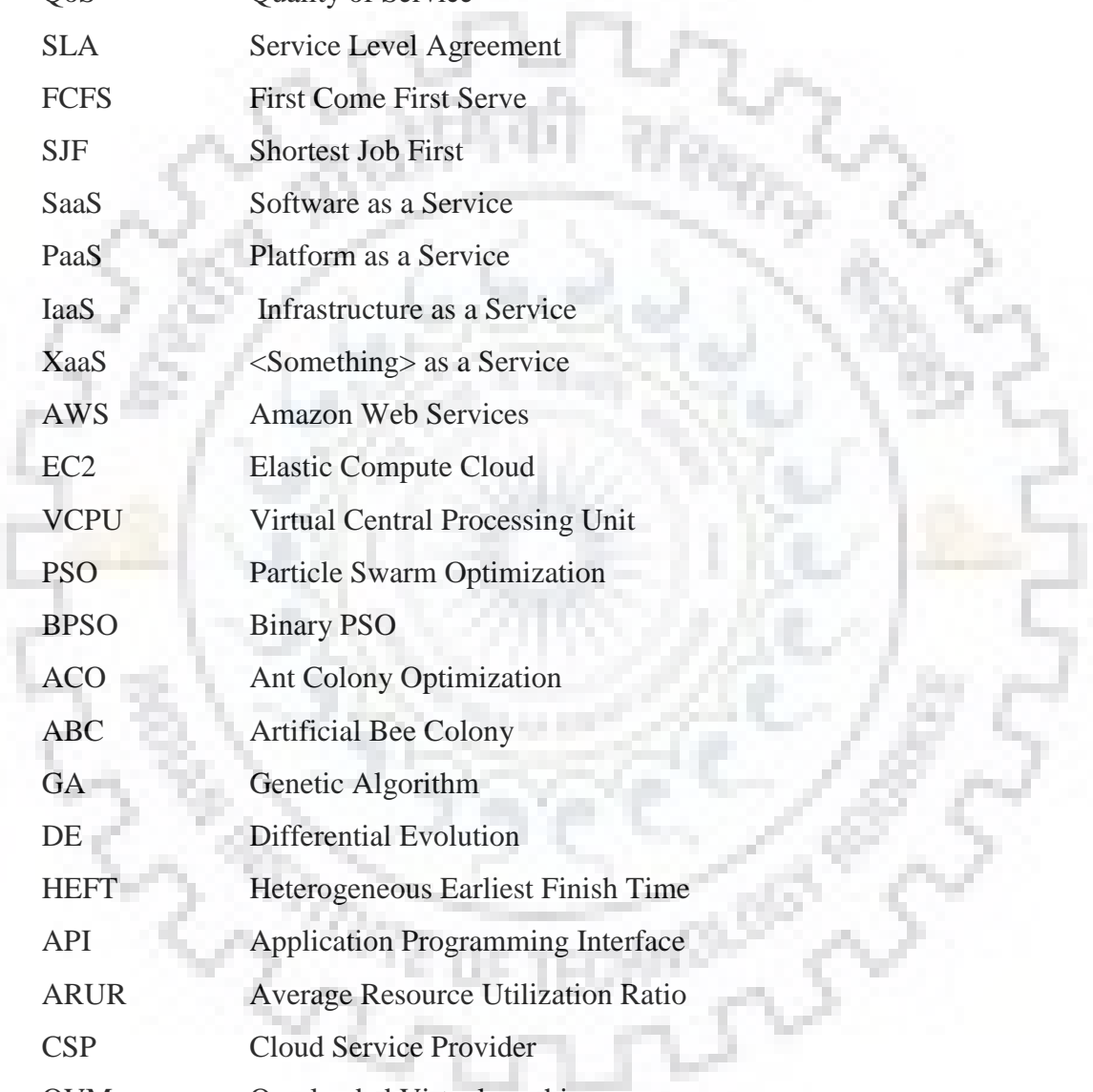


SYMBOLS

N	Number of Tasks
M	Number of Virtual machines
S_p	Represent the p^{th} schedule of workload
$\Phi_{T_i S_p}$	Represents the number of matched resources for task T_i for schedule S_p
$ET_{T_i R_j \in \Phi_{T_i S_p}}$	Execution time of task T_i on matched resource R_j
$EET_{T_i R_j \in \Phi_{T_i S_p}}$	Excepted Execution Time of resource R_j to execute the task T_i
$TPT_{T_i}^{R_j}$	Total processing time of tasks T_i at cloud resources R_j
TL_{T_i}	Length of task T_i in Millions of instructions
B_{R_j}	Bandwidth of resource R_j
$\partial(T_i)$	Deadline of task T_i
FT_{T_i}	Finishing time of task T_i
WT_{T_i}	Waiting time of task T_i
W_{R_j}	Workload available on resource R_j
p_s	Processing speed of resource (R_j)
$\psi_{T_i R_j}$	Binary decision variable such that $\psi_{T_i R_j}=1$ if T_i is allocated to resource R_j
X_{ijk}	If task i is allocated to resource R_k from R_j then value is 1 otherwise 0
$EC_{T_i R_j}$	Execution cost of task T_i at resource R_j
$\mathcal{EC}_{T_i R_j \in \theta_{T_i S_p}}$	Energy consumption by resources R_j for tasks T_i in schedule S_p
\mathcal{EC}_{Max}	Energy consumption when resource is completely
\mathcal{EC}_{Min}	Energy consumed by resources when they are ideal or low utilization(0 to5 %)
t_{max}	Maximum number of iteration

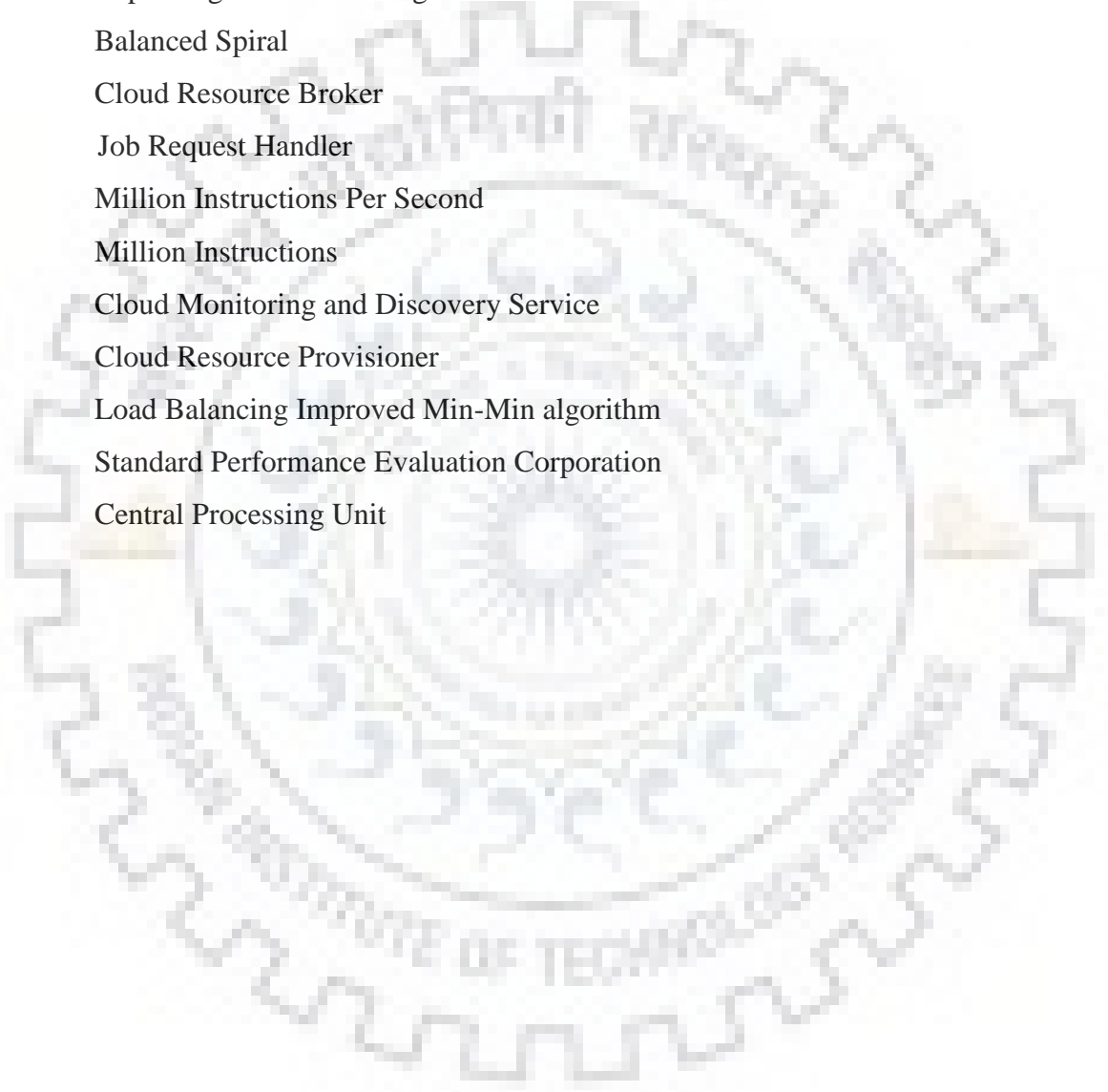


LIST OF ACRONYMS



VM	Virtual Machine
VMM	VM Monitor
QoS	Quality of Service
SLA	Service Level Agreement
FCFS	First Come First Serve
SJF	Shortest Job First
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
XaaS	<Something> as a Service
AWS	Amazon Web Services
EC2	Elastic Compute Cloud
VCPU	Virtual Central Processing Unit
PSO	Particle Swarm Optimization
BPSO	Binary PSO
ACO	Ant Colony Optimization
ABC	Artificial Bee Colony
GA	Genetic Algorithm
DE	Differential Evolution
HEFT	Heterogeneous Earliest Finish Time
API	Application Programming Interface
ARUR	Average Resource Utilization Ratio
CSP	Cloud Service Provider
OVM	Overloaded Virtual machine
UVM	Underloaded Virtual Machine
BVM	Balance Virtual Machine
EC	Execution Cost
MST	Makespan Time
IoT	Internet of Things

ET	Execution Time
LP	Linear Programming
TTT	Task Transfer Time
HPC	High Performance Computing
NP	Nondeterministic Polynomial Time
IBA	Improving the Backfill Algorithm
BS	Balanced Spiral
CRB	Cloud Resource Broker
JRH	Job Request Handler
MIPS	Million Instructions Per Second
MI	Million Instructions
CMDS	Cloud Monitoring and Discovery Service
CRP	Cloud Resource Provisioner
LBIMM	Load Balancing Improved Min-Min algorithm
SPEC	Standard Performance Evaluation Corporation
CPU	Central Processing Unit



GLOSSARY

Virtual Machine: Virtual machine is a processing entity in cloud environment that is controlled by hypervisor such as KVM, XEN etc.

Task/cloudlet: In cloud computing, cloudlet is a mini cloud set to serve a specific purpose in a given environment on the demand of the cloud users. However, in the simulation tools, it is known as a task to perform certain operation.

Execution Time: It is the time to execute an application at cloud resources (virtual machine). It should be minimum for better cloud services and users satisfaction.

Makespan Time: It is the total time of applications that elapses from starting to end. The aim of scheduling is to execute the user's application in minimum time.

Waiting Time: Time spends by task or application in ready queue before assigned to the cpu is called waiting time of task. Waiting time should be minimum for better performance.

Turnaround Time: Turnaround time is the combination of both execution time and waiting time.

Response Time: It is the time to produce first response after submitting the task, when task start their execution on virtual machine. It sends a response to the user that is called the response time.

Execution Cost: Total cost spend to execute the upcoming task in a schedule is called execution cost. It is measured in dollar (\$) per hour basis for each schedule.

Throughput: It is the measure of the rate at which consumer requests are being processed.

Energy consumption: It represents the efficiency and effectiveness in using electrical energy for different datacenter operations e.g., powering of servers and cooling system.

Scalability: It is the ability of a system to fit in a problem such that if scope of the problem increases (number of request increase).

Resource utilization: It measures the degree of resource utilization of computing resources in the cloud datacenter.

SLA: It represents the reduction in the number of SLA violations. SLA violations should be minimized to provide consumer satisfaction.

Bandwidth: Potential capacity of a link is called bandwidth.

Memory: is a process in which the cloudlet or tasks are encoded, retrieved or stored as the requirement of the cloud users in cloud computing.

Performance: is an amount of cloudlet or task accomplished on the demand of the cloud users.

Priority: is a cloudlet or task that has more importance than other or has right to execute or proceed before others.

Reliability: is the ability of cloudlet or task to execute its required function within specific time successfully. It provides the assurance of completion and avoid or reduce the failure rate in cloud computing.

Workload: is the amount of processing to be done or handled within given time period.

Particle Swarm Optimization (PSO): It is a meta-heuristic technique that optimizes the hard computational problem by iteratively to improve QoS parameters (candidate solutions) with regard to a given measure of quality.

LIST OF PUBLICATIONS

Refereed Journals

1. **Mohit Kumar** and S.C.Sharma, “Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment” in journal of Computers & Electrical Engineering (CAEE), Nov-2017 (Elsevier, IF=1.57) [SCI-E Indexed] Status: **Published.** <https://doi.org/10.1016/j.compeleceng.2017.11.018>
2. **Mohit Kumar** and S.C.Sharma, “Dynamic load balancing algorithm to minimize the makespan time and utilize the resources effectively in cloud environment” in international journal of Computers and applications, pp. 1-10, Nov. 2017 (Taylor & Francis, Scopus Indexed) Status: **Published.**
<https://doi.org/10.1080/1206212X.2017.1404823>
3. **Mohit Kumar** and S.C.Sharma, “Load balancing algorithm to minimize the makespan time in cloud environment,” World journal of modelling and simulation (Scopus Indexed) Status: **Accepted.**
4. **Mohit Kumar** and S.C.Sharma, “PSO-BOOST: Multi-Objective scheduling algorithm to optimize the execution cost and time in cloud computing,” Swarm and Evolutionary Computation (Elsevier IF=3.893) [SCI-E Indexed] Status: **Under-Review. 2018**
5. **Mohit Kumar** and S.C.Sharma, “Task scheduling in cloud environment using dynamic transfer function based modified Binary Particle Swarm Optimization algorithm,” Cluster Computing (Springer IF=2.04) [SCI-E Indexed] Status: **Under-Review. 2017**
6. **Mohit Kumar** and S.C.Sharma, “Modified Binary PSO based Multi-objective scheduling in cloud computing,” Concurrency and Computation: Practice and Experience (Wiley IF=1.133) Status [SCI-E Indexed] Status: **Under-Review. 2018**

7. **Mohit Kumar** and S.C.Sharma, "PSO-COAGENT: Cost and Energy Efficient scheduling in Cloud environment with deadline constraint," *Sustainable Computing (Elsevier IF=1.80) [SCI-E Indexed]* Status: **Under-Review. 2017**

International Conferences

1. **Mohit Kumar** and S.C. Sharma," Priority Aware Longest Job First (PA-LJF) Algorithm for Utilization of the Resource in Cloud Environment," in *3rd International Conference on Computing for Sustainable Global Development (IndiaCom)*, pp. 415-420, New Delhi, India, March 2016 (**IEEE**) [**Scopus Indexed**] Status: **Published.**
2. **Mohit Kumar**, K. Dubey and S.C.Sharma, "Job Scheduling algorithm in cloud environment considering the priority and cost of job," in *6th International Conference on soft computing for problem solving (SocProS)*, Thapar, Patiyala, India 2016 (**Springer**) [**Scopus Indexed**] Status: **Published.**
3. **Mohit Kumar** and S.C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," in *Procedia Computer Science*, vol. 115, pp. 322-329, 2017 (**Elsevier Scopus**) [**Scopus Indexed**] Status: **Published.**
4. **Mohit Kumar**, K. Dubey and S.C.Sharma, "Elastic and flexible deadline constraint load Balancing algorithm for Cloud Computing" in *Procedia Computer Science*, vol. 125, pp. 717-724, 2018 (**Elsevier Scopus**) [**Scopus Indexed**] Status: **Published.**

CHAPTER-1

INTRODUCTION

Cloud computing is emerging computing technology that is rapidly gaining popularity and admired in IT industry as well as academic [1]. It is also known as utility based system, since cloud computing still is in its infancy, there are many open research challenges exist in the field of cloud computing like security, resource provisioning and scheduling. In this thesis, we address the problem of resource provisioning and scheduling in cloud computing to optimize QoS parameters like execution time, execution cost, task rejection ratio, throughput and energy consumption considering deadline as constraint. In this chapter, we provide an overview of the cloud computing that contains the cloud services model, deployment models, resource provisioning and scheduling. Further the key motivation that guided us to research on this topic and to formulate the objectives of the research work and original contribution of research work is given in the last.

1.1 Overview

Cloud computing is collection of heterogeneous resources that provides the services either in the form of software application or hardware infrastructure on the basis of pay per use over the internet. Computing resources of cloud environment contains on demand self-service, scalability (scale-out and scale-up), resource pooling, broad network access, rapid elasticity and higher availability types of characteristic. Three types of basic service provided by cloud computing namely as: Infrastructure as a service (IaaS), Platform as a service (PaaS) and Software as a service (SaaS). These types of services are useful in scientific, business and industrial applications. Based on the delivery model, cloud computing can broadly be divided into three basic service models [2] as shown in Fig. 1.1.

1.1.1 Cloud services models

- (i) Software as a service: Cloud consumers use the provider application from anywhere using the web browser or a program interface in SaaS. There is no need to manage and control the infrastructure, operating system as well as applications by cloud consumers. Examples of SaaS are: Email, virtual desktop, facebook, YouTube etc.

- (ii) Platform as a service: PaaS vendors offer a development environment for application developers (consumers) where developers is free to build their own applications as per requirement using programming languages, libraries and tools. There is no need to manage and control the underlying infrastructure (virtual machines, servers, operating system etc) by consumers but has to managed and control own deployed applications. Examples of PaaS are: Window azure, Google App Engine, Force.com etc.
- (iii)Infrastructure as a service: Consumers can get on demand infrastructure (virtual machines, servers, storage etc) from the cloud in IaaS and they are able to deploy different types of operating systems and applications as per requirements. Cloud consumer is responsible to manage and control the infrastructure as well as operating system and deployed applications. Examples of IaaS are: Amazon elastic compute cloud (EC2), virtual machines, servers, storages etc.

These types of services are useful in scientific, business and industrial applications. User can send the request at anytime and anywhere for the cloud resources using the graphical user interface or web browser.

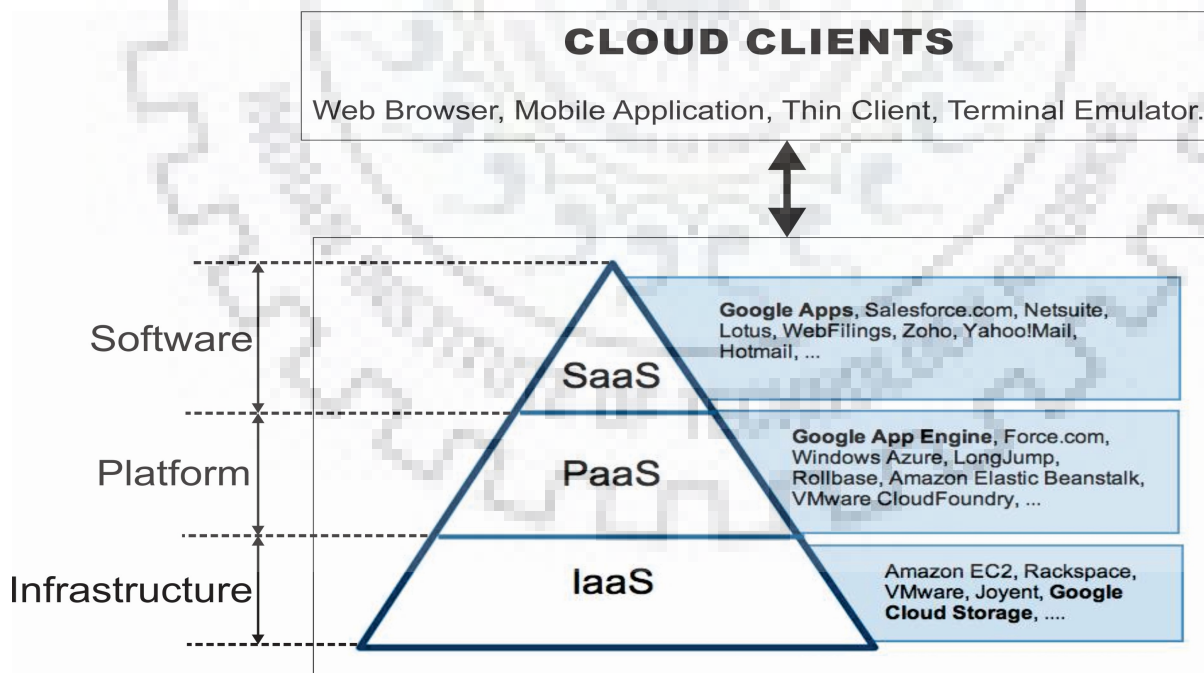


Figure 1.1 Cloud computing service layers with example

According to Mell et al. [3], from National Institute of Standards and Technology (NIST), cloud computing is defined as:

“a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction”.

1.1.2 Cloud deployment models

There are four deployment models of cloud are public, private, hybrid and community as shown in Fig. 1.2. Public cloud services are available for general public over the internet. Amazon elastic compute cloud, Google appEngine, Window azure service platform etc are examples of public cloud those offered the services either pay per use basis or free [4]. Lightweight framework is proposed by A. Abraham et al., to monitor the public cloud [5]. Private cloud is used for personal use or provides the service to single organization. Eucalyptus, OpenNebula, Openstack etc are example of private cloud those offered the similar advantages to public cloud. A hybrid cloud is combination of two or more than two public and private cloud which are bounded by service level agreement (SLA).

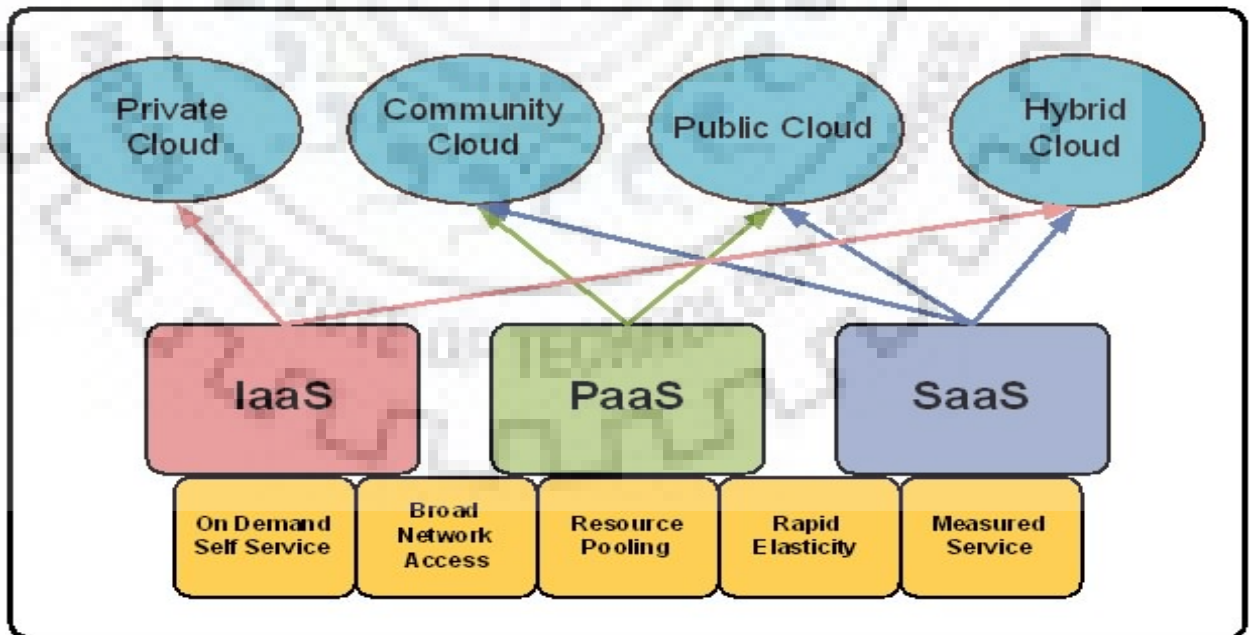


Figure 1.2 Cloud deployment models

Community cloud is multi-tenant platform that allows many companies to work at the same platform and it is managed by one or more than one companies in the community.

The number of users and applications are increasing gradually in cloud environment and in turn there is increase unknown mixed workloads (HPC applications, web) and traffic at the web applications which are deployed in the virtual machine (cloud resource). Therefore resource provisioning and scheduling of application becomes a critical problem for services provider due to over-provisioning and under-provisioning types of problem [6]. Scheduling of application is a NP Complete optimization problem in cloud environment due to heterogeneous nature of cloud resources and dynamic nature of upcoming user application request. It is difficult to find out the optimal solution for NP-complete problem but one can find the suboptimal solution for short period of time using the metaheuristic (optimization) algorithm. The main objective of scheduling algorithm is to assign the application (tasks) in such a manner that all tasks should be executed in minimum time and cost considering deadline as constraint.

1.1.3 Applications of cloud computing

Some applications of cloud computing [7] as follows:

- It is easy to use the cloud services, applications and its associated data directly without getting them installed on their machine.
- There is no need to worry about data loss or virus types of problem because cloud service provider provides dependable and secure data storage center to cloud users.
- Data sharing is possible between different equipments by cloud computing.
- Cloud computing provides number of applications as well as services in low cost rather than buying the infrastructure.
- Cloud computing is also useful in agriculture field and health sector. There is no need to buy the servers, software and no need to maintain the infrastructure by the farmers.

1.1.4 Research Challenges

Research challenges of cloud computing can summarize as follows [8-17]:

1.1.4.1 Security of data

Movement of data and application over the networks is a critical issue in the field of cloud computing because there is possibility of loss of control on data. When data is move from one

organization to other organization for either processing or storage then there are maximum chances of inherent risk and possibility of various attacks [8-13].

1.1.4.2 Reliability and Availability

Strength of technology is measured with the help of reliability and availability parameters. Reliability denotes that cloud resources are totally free of technical errors without disruption (loss of data, code reset during execution). Downtimes and slowdowns creates serious problem for the reliability of cloud computing. Reliability in cloud computing can be achieved using the redundant resources. Availability in cloud environment can be understood as the probability of acquiring the virtual machines whenever they are needed [14-15].

1.1.4.3 Scalability and Elasticity

Most challenging features of cloud computing is to provide on demand huge number of resources to the user as per their need (elasticity and scalability) [16]. Scalability is the ability of a system to fit in a problem such that if scope of the problem increases such as number of request increase, length of request vary randomly etc. The ability of auto scaling on upcoming demands in cloud computing is biggest advantages for services provider as well as user. Elasticity is the ability of provisioning and deprovisioning of resources as per user demand.

1.1.4.4 Resource Management and Scheduling

To utilize the cloud resources efficiently is also a challenging issue in the field of cloud computing. Scheduling challenges occurs due to dispersion, uncertainty and heterogeneity of resources that are not resolved with traditional resource management mechanisms. Still over provisioning and under provisioning types of problem occurs in cloud environment due to which cloud resources are not utilizing properly [17]. Resource management is needed at various levels like hardware, software, virtualization etc. Job scheduling is a type of resource provisioning where jobs are executed in particular order to optimize the parameters like execution time, execution cost, energy consumption, throughput etc.

1.1.4.5 Energy Consumption

Cloud data centers are increasing day by day due to huge computational demands of cloud user which is serious threat for the environment. In 2006, United States data centers consumed more than 1.5% of the total energy produced in that year, and this percentage is expected to increase 18% annually [18].

1.2 Motivation

Cloud computing is growing in popularity among computing paradigms to deliver computing resources as a services over the internet on the basis of pay per use. There was some shortcoming of non-cloud computing environment as shown in Fig. 1.3. It is difficult to predict the workload and allocate on demand resources in non-cloud environment. Therefore over provisioning and under provisioning types of problems occurs regularly due to which most of time virtual machines either in ideals conditions (waste of capacity) or overloaded conditions. Cloud computing environment partially solve the problem of over provisioning and under provisioning as shown in Fig. 1.4. IaaS cloud services have provided a paradigm shift in the way resources are provisioned and utilized. One more advantage with cloud computing is that there is no need to invest in a huge computer system by the users to do their business; instead, they can purchase the on demand services from the cloud anywhere and anytime. Still some research challenges exist in cloud computing: Most of the published scheduling algorithm deals with only one parameter either scheduling the upcoming tasks to optimize the required parameter or scalability within user defined deadline constraint.

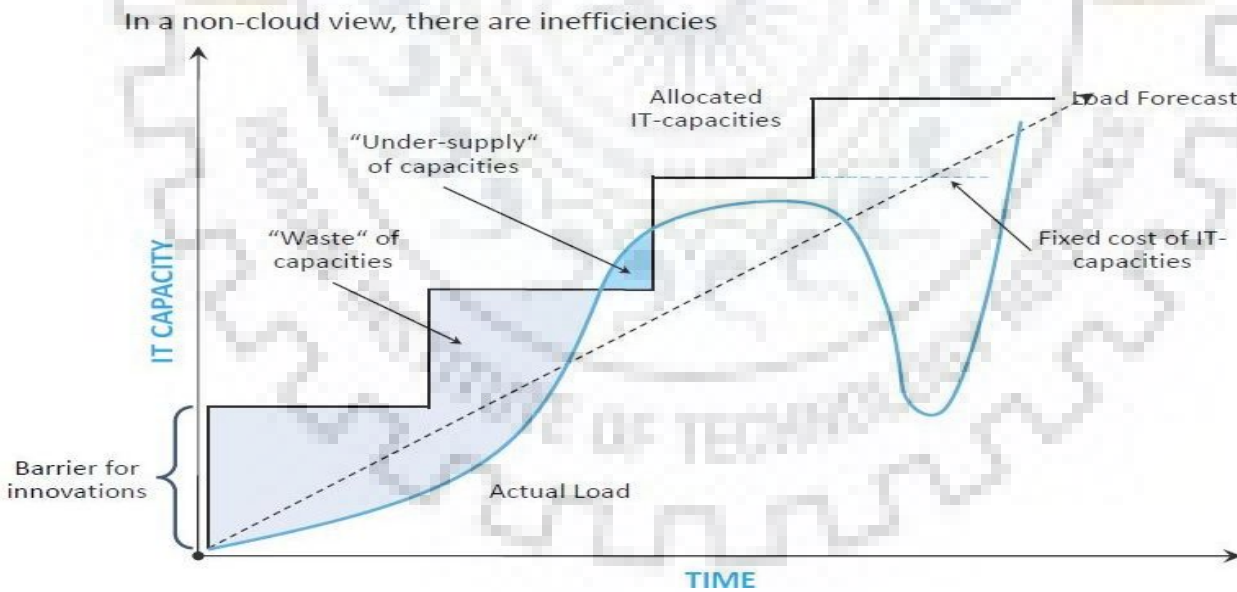


Figure 1.3 Resource Provisioning in non-cloud computing

The most challenging problem for a cloud service provider is maintaining the quality of service parameters like reliability, elasticity, keeping the deadline and minimizing the makespan time and also the task rejection ratio. Therefore, the cloud service provider needs a dynamic task

scheduling algorithm that reduces the makespan time while increasing the utilization ratio of cloud resources and meeting the user defined QoS parameters [19].

Many real-world problems belong to the family of discrete optimization problems. Most of these problems are NP-complete and difficult to solve efficiently using classical linear and convex optimization method [20]. There is no algorithm exists to solve the NP complete problem in polynomial time i.e., required time to solve the problem is increase exponentially when size of problem is increased linearly. Calculating the all possible task-resource mapping (scheduling) and selecting the optimal mapping is not feasible in cloud environment. Therefore researchers are using the metaheuristic techniques to solve the NP Complete problems. Simple binary particle swarm optimization (BPSO) does not provide satisfactory solution due to inappropriate behavior (unable to maintain the good balance between exploration and exploitation) of transfer function. To overcome this problem, we have modify the transfer function that provides the exploration and exploitation capability in better way to solve the problem of scheduling. The main goal of cloud service provider is to maximize the profit from cloud infrastructure while cloud users want to execute their applications in minimum time and cost. There is always a conflict between execution time and cost due to heterogeneous nature of cloud resources as well as upcoming user's application.

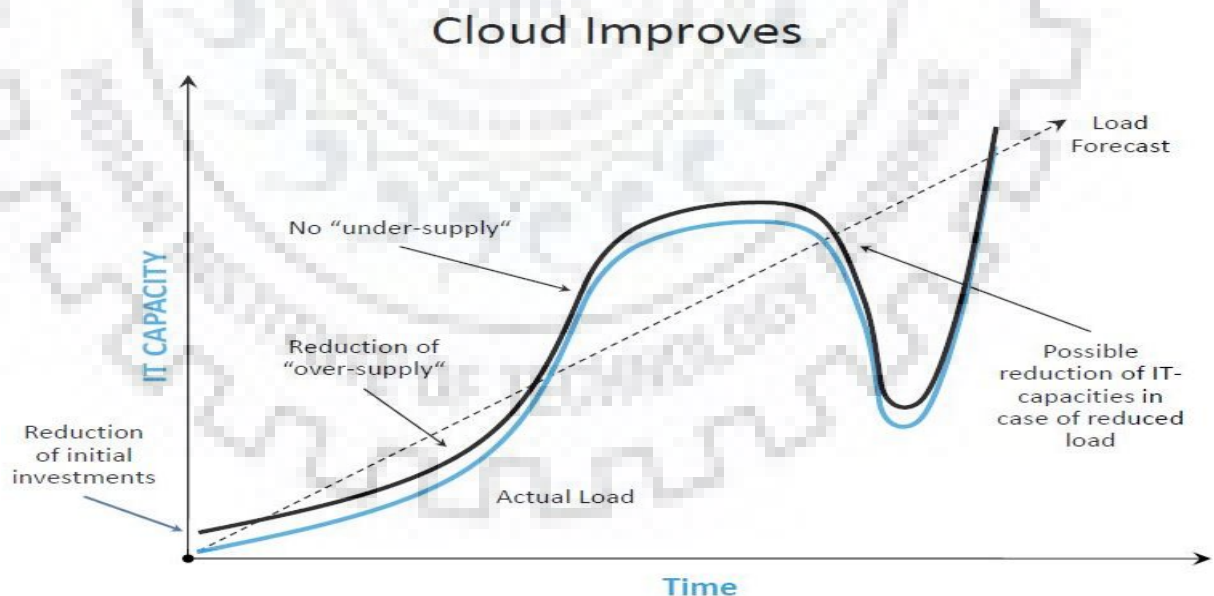


Figure 1.4 Resource Provisioning in cloud computing environment

To make a balance between execution time and cost, a trade-off solution is required. The rapid growths in demand of computational power tends to massive growth in cloud data centers and

require large amount of energy consumption in cloud data centers which becomes a serious threat to the environment. To reduce the energy consumption and gain the maximum profit in cloud computing is a challenging problem due to incompatibility between workstation (physical machine) and unpredictable users demand.

1.3 Objectives

Keeping in view the resource provisioning and scheduling problem in cloud computing, the aim of present work is to propose a technique that can execute the tasks in minimum time, execution cost and energy consumption while considering deadline as a constraint. The major objectives of this thesis are summarized below:

- 1) Dynamic load balancing with elasticity play an important role in the field of cloud computing. Thus, the first objective of the thesis is to developed a dynamic scheduling algorithm that balances the work load among all the virtual machines with elastic resource provisioning and deprovisioning based on the last optimal k-interval considering deadline as constraint.
- 2) Simple BPSO unable to maintain the good balance between exploration and exploitation. The aim of proposed dynamic transfer function based binary particle swarm optimization (TF_p-BPSO) is to provide the exploration at the early stage by high flipping of bits of particle position for any velocity and It has the ability to decrease the probability of flipping of bits in intermediate stage so that it can move from exploration to exploitation. It has been observed that in the last stage of run should provide stronger exploitation (less probability of flipping of bits).
- 3) Most of the scheduling algorithm deals with only one parameter either optimize the execution cost of running cloud resources or execution time within user defined deadline constraint. There is always a conflict between execution time and cost. Therefore a trade-off solution is required to solve this issue. We have formulated our multi-objective scheduling problem in the form of mathematical model and proposed a particle swarm optimization (PSO) based algorithm that not only minimize the execution time but also optimize the execution cost of cloud resources.
- 4) To reduce the energy consumption and gain the maximum profit in cloud computing is a challenging problem due to incompatibility between workstation (physical machine) and unpredictable users demand. We proposed resource allocation model for processing

the applications efficiently and particle swarm optimization based scheduling algorithm that not only optimize execution cost but also reduce the energy consumption of cloud data centers considering deadline as constraint.

1.4 Organization and Contribution

The thesis is organized into six chapters including the introduction.

CHAPTER 1: INTRODUCTION

In the introduction part, details with general concepts of cloud services, motivation, research challenges, objectives and contributions of this thesis.

CHAPTER 2: FUNDAMENTALS AND SCHEDULING TECHNIQUES

In this chapter, we discuss the existing resource provisioning techniques, advantages of resource provisioning in the field of cloud computing, static and dynamic scheduling algorithm, classification of scheduling algorithms in terms of heuristic, meta-heuristic and hybrid algorithm. Further resource allocation model and simulation tool are discussed in brief that are used to measure the performance of the scheduling algorithm.

CHAPTER 3: LOAD BALANCING WITH ELASTICITY USING HEURISTIC TECHNIQUE

The details about the development of a cloud resource broker architecture and dynamic scheduling algorithm that is able to automatically manage and monitor the virtual machines to minimize the QoS parameters based on the last optimal k-interval of considering deadline as constraint and simultaneously fulfill the objective of elasticity in cloud environment have been discussed.

CHAPTER 4: DYNAMIC TRANSFER FUNCTION BASED MODIFIED BINARY PSO FOR SCHEDULING THE TASKS

Development of dynamic transfer function (TF_p-BPSO) based BPSO algorithm that provides better exploration at the early stage, its move from exploration to exploitation in the intermediate stage of execution and provides the stronger exploitation in the last stage of execution to improve QoS parameters have been discussed.

CHAPTER 5: MULTI-OBJECTIVE SCHEDULING ALGORITHM USING PSO

Particle swarm optimization (PSO) based scheduling algorithm and resource allocation model for improvement of QoS parameters have been discussed.

CHAPTER 6- CONCLUSIONS AND FUTURE WORK

This chapter concludes the work reported in the thesis and discusses about future research directions. Thesis methodology is divided into different step in order to simplify the work. The steps are shown in terms of flowchart as shown in Fig. 1.5

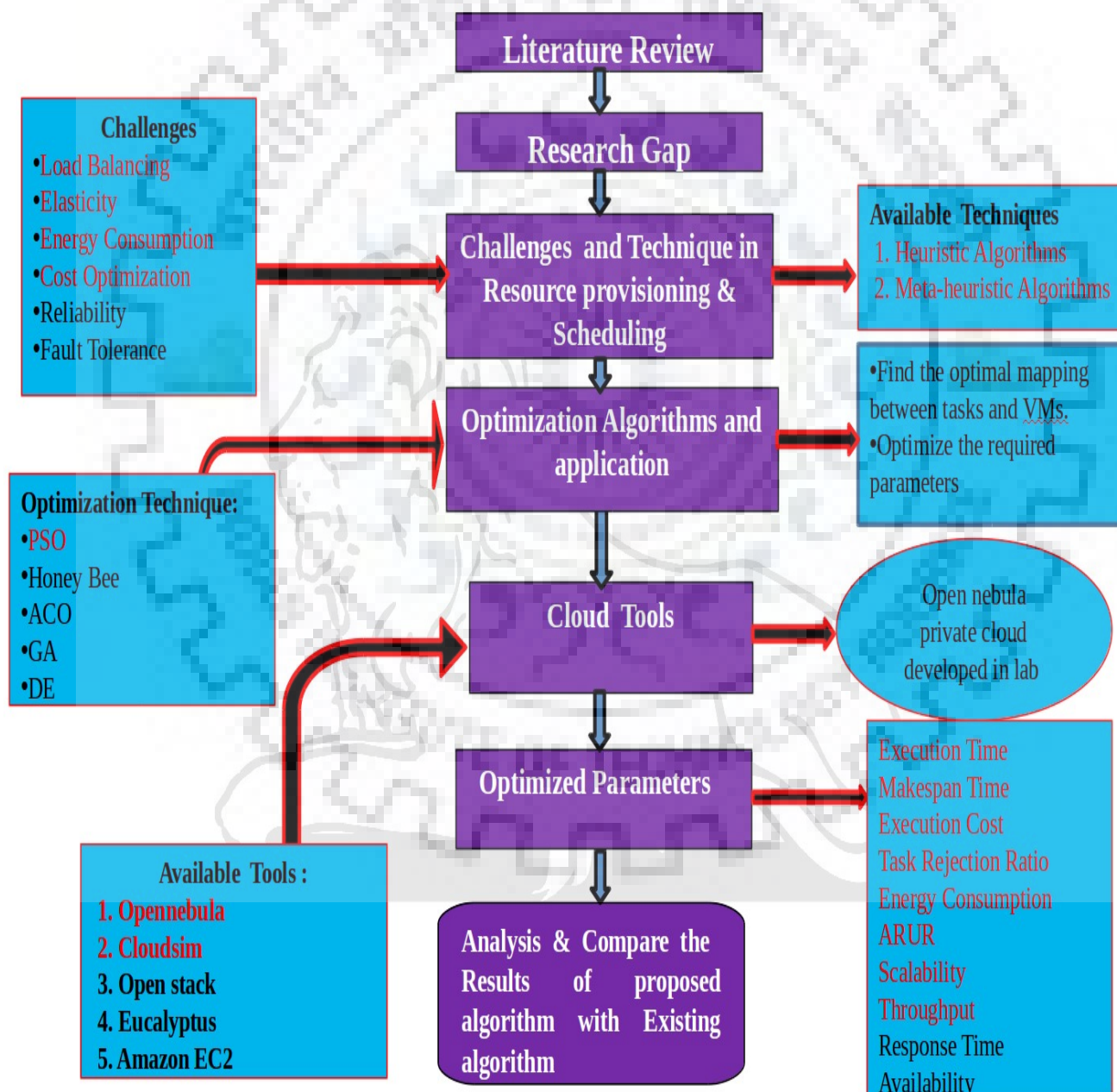


Figure 1.5 Brief layout of research plan

1.5 References:

- [1] B. Pring, R. H. Brown, A. Frank, S. Hayward and L. Leong, "Forecast: Sizing the Cloud; understanding the opportunities in cloud services," *Gartner, Inc., Research Report G 166525*, 2009.
- [2] O. Serfaoui, M. Aissaoui and M. Eleuldj , "OpenStack: Toward an Open-Source Solution for Cloud Computing," *International Journal of Computer Applications*, vol. 55, no. 03, Oct. 2012.
- [3] P. Mell and T. Grance, "The NIST definition of Cloud computing: Recommendations of the national institute of standards and technology," Special Publication 800-145, NIST, Sep 2011.
- [4] M. Kumar, K. Dubey and S. C. Sharma, "Elastic and flexible deadline constraint load balancing algorithm for cloud computing," in *Procedia Computer Science*, vol. 125, pp. 717-724, India, 2018.
- [5] K. Ma, R. Sun, and A. Abraham, "Toward a lightweight framework for monitoring public clouds," in 4th *International Conference on Computational Aspects of Social Networks (CASoN)*, pp. 361-365, 2012.
- [6] S. Khatua, P. K Sur, R. K Das and N Mukherjee, "Heuristic-based resource reservation strategies for public cloud," *IEEE Transactions on Cloud Computing*, vol. 4, no. 4, pp. 392-401, Oct. 2016.
- [7] S. Zhang, S. F. Zhang, X. B. Chen, and X. Z. Huo, "Cloud Computing Research and Development Trend," in *Second International Conference on Future Networks (ICFN '10)*, IEEE Computer Society, pp. 93-97, Washington, USA, Mar. 2010.
- [8] S. Kumar, S. K. Singh, A. K. Singh, S. Tiwari and R. S. Singh, "Privacy preserving security using biometrics in cloud computing," *Multimedia Tools and Applications*, PP. 1-23, 2017.
- [9] A. H. Gamlo, Ning Zhang, Omaimah Bamasag , "Mobile Cloud Computing : Security Analysis," 5th *International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud)*, 2017.
- [10] G. Sahoo and S. Mehfuz, "Securing Software as a Service Model of Cloud Computing : Issues and Solutions," vol. 3, no. 4, pp. 1–11, 2013.
- [11] R. Rai, G. Sahoo and S. Mehfuz, "Exploring the factors influencing the cloud computing adoption: a systematic study on cloud migration," SpringerPlus, vol. 4, no. 1, pp. 197-208. 2015.

- [12] A. Naureen and N. Zhang, "A Comparative Study of Data Aggregation Approaches for Wireless Sensor Networks," *Proceedings of the 12th ACM Symposium on QoS and Security for Wireless and Mobile Networks*, Malta, Malta, 13-17 Nov., 2016.
- [13] A. Al-Riyami, N. Zhang, and J. Keane, "An Adaptive Early Node Compromise Detection Scheme for Hierarchical WSNs," *IEEE Access*, vol.4, pp.4183 - 4206, 2016.
- [14] Y. Ghanam, J. Ferreira and F. Maurer, "Emerging issues & challenges in Cloud- A hybrid approach," *Journal of software engineering and applications*, vol. 5, no. 11, pp. 923-937, Nov. 2012.
- [15] M. A. Vouk, "Cloud computing Issues, research and implementations," *Journal of computing and information technology*, vol. 16, no. 4, pp. 235-246, June 2008.
- [16] T. Dillon, C. Wu and E. Chang, "Cloud computing: Issues and challenges," in 24th *International Conference on Advanced Information Networking and Applications*, pp. 27- 33, June 2010.
- [17] European CIO Cloud Survey, Addressing security, risk and transition, May -2011.
- [18] R. Brown et al., "Report to congress on server and data center energy efficiency: Public law 109-431," *Lawrence Berkeley National Laboratory*, 2008.
- [19] M. Kumar and S. C. Sharma, "Deadline constrained based dynamic load balancing algorithm with elasticity in cloud environment" *Computers and Electrical Engineering journal*, pp. 1-17, 2017.
- [20] Md. J. Islam, X. Li, and Y. Mei, "A Time-Varying Transfer Function for Balancing the Exploration and Exploitation ability of a Binary PSO," *Applied Soft Computing*, vol. 59, pp. 182-196, 2017.

CHAPTER-2

FUNDAMENTALS AND SCHEDULING TECHNIQUES

In this chapter, we initially present the techniques of resource management that are used in cloud computing. Resource management includes resource provisioning, resource mapping, resource brokering, scheduling, resource allocation and load balancing related problems in the field of cloud computing [1]. At the end we have discussed the existing resource provisioning, scheduling, resource allocation and load balancing algorithm used by different researchers to optimize the QoS parameters and addressed the shortcoming of the algorithms.

2.1 Resource Management in cloud computing

Cloud computing offers resource provisioning and scheduling to the users as per their demand and provides the guarantee of reliable services on the basis of pay per use. To overcome the challenging issues the various resource provisioning and scheduling techniques are reported in literature [2]. Resource management is divided into two basic stages in cloud computing: resource provisioning and resource scheduling as shown in Figure 2.1. Resource scheduling will be more beneficial for users as well as service providers, if perform after the efficient resource provisioning.

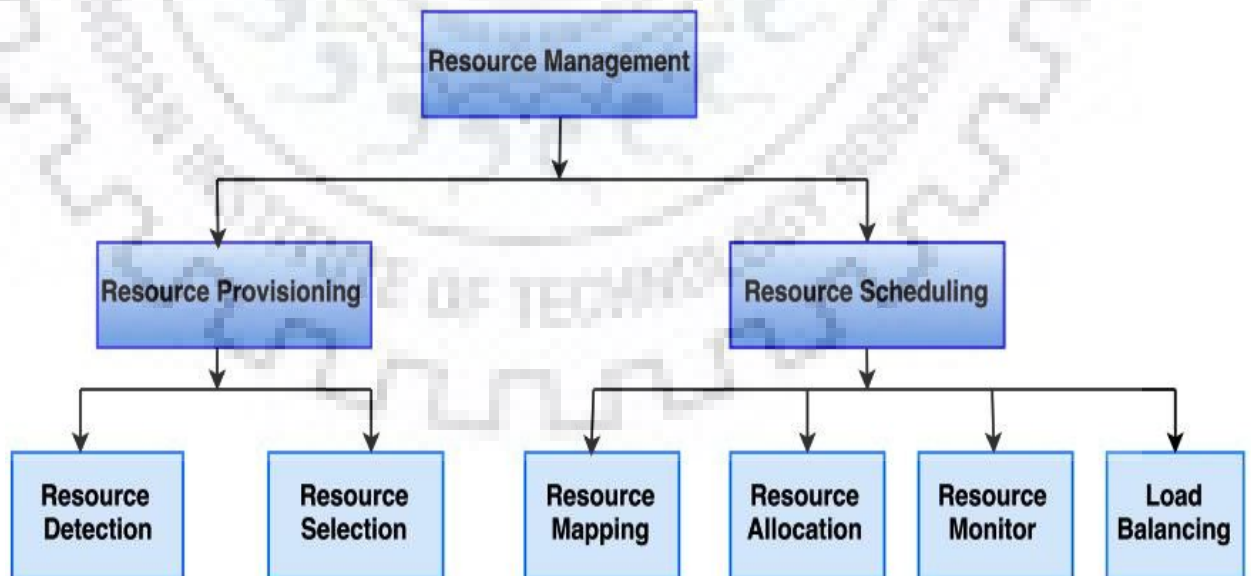


Fig. 2.1 Resource management in cloud computing

2.2 Cloud Resource provisioning

Cloud resource provisioning is the technique to enable the virtualized resources for allocation the users. When cloud service provider accepts the request for resources from the users, it creates appropriate number of virtual machines and allocate to users as per their demand. Resource provisioning is also responsible to fulfill the user's need based upon the quality of service parameters (QoS), SLA negotiations and match the resources to the upcoming workloads.

2.2.1 Need of resource provisioning

The aim of resource provisioning is to detect and select the appropriate resources for upcoming request, so that request (applications) can get optimal resource i.e. number of resources should be minimum for the applications to maintain a desirable level of service quality (minimum execution time and maximum throughput). The objective of resource provisioning is to map the upcoming request with the running virtual machines so that user get the services in minimum cost and time while service provider get the maximum profit [3].

There are three resources provisioning techniques available for the cloud users as shown in Fig. 2.2:

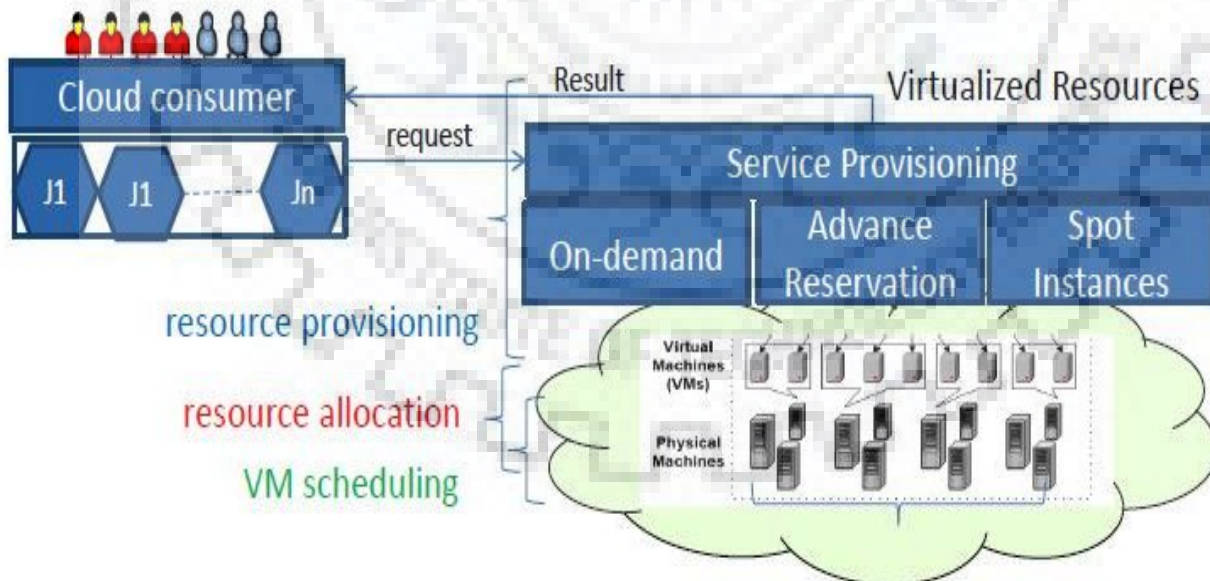


Figure 2.2 Resource provisioning plans

2.2.2 Advanced reservation

This is a long term plan that allows the users to reserve the resources in advances for a specific time period. This technique is very useful in federated cloud as well as elastic compute cloud (EC2). There are some drawbacks of this technique like it is difficult to predict the future demand of users and prices of cloud resources. Over-provisioning and under-provisioning types of problem also occur in this technique [4].

2.2.3 On demand resources allocation

It is an intermediate level plan that allows users to pay per hour basis based upon the resources has been used. If the demand for the cloud resources at a given time t exceeds the reserved value, then additional resources are required for on-demand resource provisioning. The under-provisioning problem can be solved by provisioning more resources at higher cost with on-demand plan. Generally on demand additional resources are allocated to the users at higher cost than advanced reservation resources.

2.2.4 Spot Instances

It is a short-term plan that allows customers to bid on unused resources. Spot instances are Amazon's third plan that offer unused resources at a much lower cost than both on-demand and advanced reservation. Major cloud service providers (AWS, Google, and Azure) offer the option to use Spot Instances. Spot Instances are a cost-effective technique if you can be flexible about when your applications run and if your applications can be interrupted. The problem with spot instances is that their price changes periodically based on supply and demand of spot instances.

2.2.5 Advantages of cloud resource provisioning

Advantages of cloud resource provisioning are given below [5]

- Execution time as well as makespan time of upcoming workload is reduced by efficient resource provisioning techniques.
- Better resource utilization can reduce the problem of over-provisioning and under-provisioning.
- If virtual machine startup delays is less that provides better resource provisioning in cloud environment.
- Effective cloud resource provisioning algorithm increases the robustness as well as fault tolerance capability.

- Resource provisioning algorithm reduces the power consumption without violation of SLA.
- Efficient load balancing algorithm distributes the workload at the virtual machines in such a manner that no virtual machine is in overloaded or underloaded condition.
- Improve user deadline violation rate by effective resources provisioning before start the scheduling.
- Resource provisioning also reduces waiting time in workload queue.

2.3 Scheduling

Scheduling is the way to determine, which activity should be performed based upon the required quality of service (QoS) parameter. Scheduling is responsible to select optimal virtual machines for execute the tasks using either heuristic or meta-heuristic algorithm and also responsible for QoS constraints are met.

Resource scheduling can be done in two ways; first one is on demand scheduling in which cloud service provider provides the resources quickly to random workload. This approach has a problem of unequal distribution of workload i.e. there is possibility of executing more tasks at a single virtual machine (VM), therefore performance start to degrade and over provisioning types of problem can occurs. Second is long term reservation in which many number of virtual machine are in ideal condition due to which under provisioning type of problem occurs. Over provisioning and under provisioning types of problem increase the cost of services due to unnecessary wastage of resources and time. To handle with these types of problems, we need an efficient resource provisioning algorithm that analyze and schedule the upcoming workload in efficient way. Modified flowchart of resource provisioning with scheduling is shown in Fig. 2.3 [6].

The objective of resource provisioning with scheduling (RPS) is provision the resources to users without violation of SLA and fulfill the users' demand [6]. Understand the expectation and requirement of the cloud users at the starting on the basis of upcoming workload (applications). Service level agreement (SLA) commitment is defined between users and service provider after analyzed the upcoming workload properly. Fitness function (FF_{QoS}) is calculated based upon the required QoS parameters for every workload and compare it with the value calculated without considering QoS parameters ($FF_{non-QoS}$). We check the condition if value of FF_{QoS} is less than the value of $FF_{non-QoS}$ then it will provision; otherwise it analyses the

workload again after resubmission of SLA by the cloud consumer through re-negotiation. If resource provisioning is completed successfully then choose the scheduling algorithm to execute the tasks in specified budget and deadline with the help of scheduler.

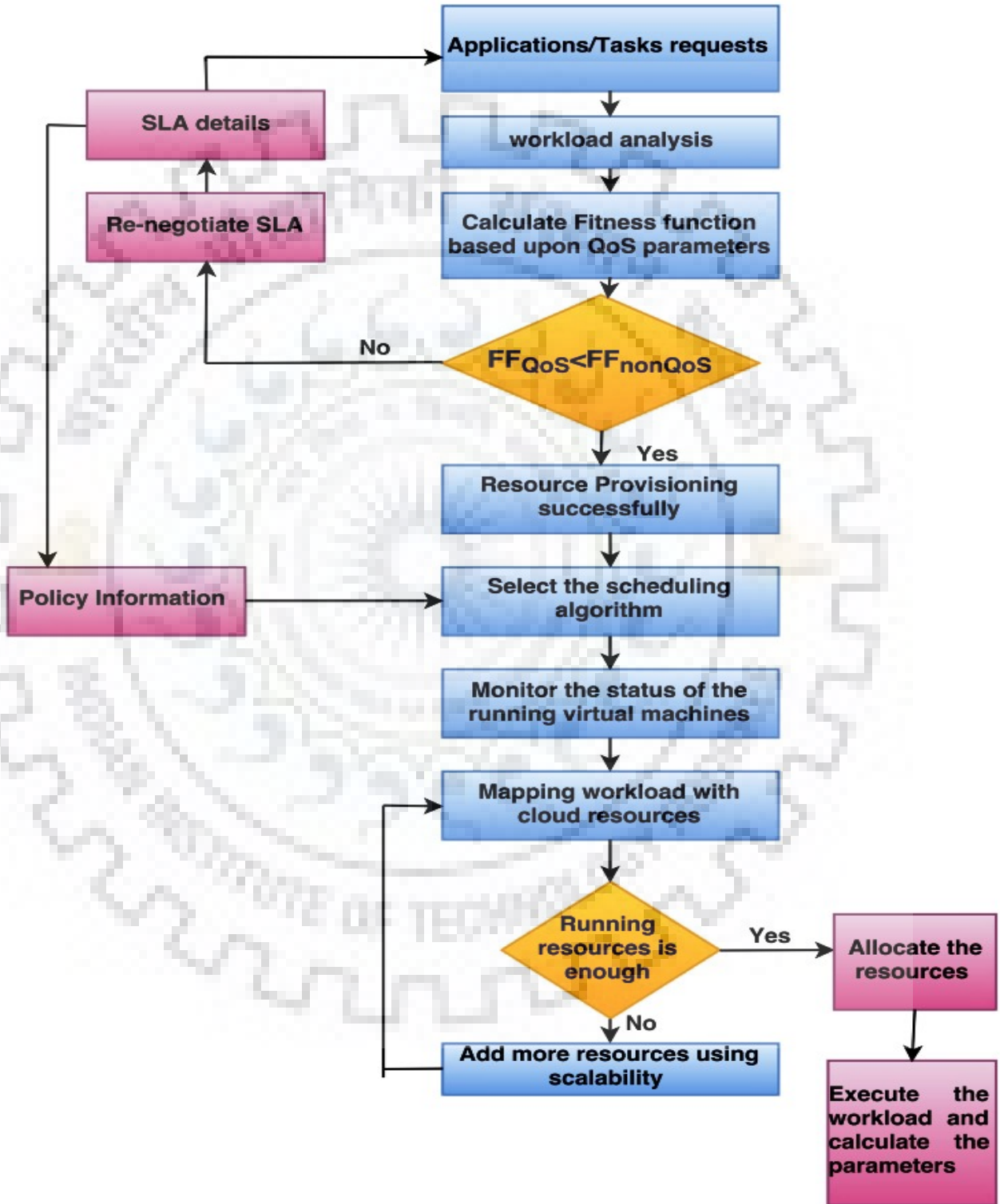


Figure 2.3 Flowchart of resource provisioning and scheduling in cloud computing

Before allocation the workload or tasks at the virtual machines (resources), cloud running resources is monitored and calculate the load at each resources. If any virtual machine is in overloaded condition then task is not allocated to such types resources. Further upcoming workload is map with the available resources and check the condition that running virtual machine is enough or not to execute the workload. If running resources are not enough then increase the resources using the horizontal scalability concept otherwise allocate the resources to the workload and calculate the required QoS parameters. There are various types of scheduling algorithm in cloud computing based upon: static and dynamic, online v/s batch mode, preemptive and non-preemptive scheduling algorithm etc.

2.3.1 Static Scheduling Algorithm

Static scheduling algorithms need the information about the task (length of task, number of tasks, deadline of tasks) and resource (node processing capacity, processing power, memory etc) in advance. Static algorithm work well when node has low variation in workload. These algorithms are not suitable for cloud environment where load vary instantaneously time to time. It is very easy to implement static algorithm but these algorithm don't optimize the quality of service parameters and not provides the good performance in real environment. Therefore we need dynamic task scheduling algorithm for cloud environment. Example of static algorithm are first in first out (FIFO), round robin (RR), shortest job first (SJF), longest job first (LJF) etc.

2.3.2 Dynamic Task Scheduling Algorithm

There is no need of advance information about the task and node in dynamic algorithm but need to monitor the node continuously. These algorithms are more efficient and accurate for cloud environment because if any node is in overloaded condition then transfer the task from overloaded node to under loaded node i.e., algorithm condition change frequently when load change (increase or decrease) at a node. Example of dynamic algorithm are dynamic round robin, heterogeneous earliest finish time (HEFT), clustering based heterogeneous with duplication (CBHD), weighted least connection (WLC), particle swarm optimization (PSO), ant colony optimization (ACO) etc. Both the algorithms (static and dynamic) have their advantages and disadvantages as shown in Table 2.1.

Table 2.1 Differences between static and dynamic algorithms

Static algorithm	Dynamic algorithm
Need the advanced information about the upcoming jobs/requests	There is no need of advance information about the jobs and resources
Scheduling decision is taken at compile time	Scheduling decision is taken at run time
Easy to implement i.e. complexity is low	It is not easy to implement, complexity is high
Static algorithms don't gives optimal results for large computational problem.	Dynamic algorithm is useful for large computational problem.
Difficult to find optimal solution of NP Complete problem	Sub optimal solution of NP complete problem can be find by dynamic algorithm
Only traditional algorithm comes under static algorithm	Meta-heuristic algorithms comes under dynamic algorithm
Static algorithms take more time to solve computational problem.	Dynamic algorithm solves the computational problem in less time.
It is difficult to find optimal result of multi-objective problem by static algorithms.	We can find the optimal results of multi-objective problem using dynamic algorithms.
Static algorithm work well when workload does not change frequently.	Dynamic algorithms work well when workload vary frequently
These algorithms do not monitor the node continuously	Dynamic algorithm monitor the node continuously either event basis or time interval
Static algorithms do not balance the workload properly at the running virtual machines (node).	Dynamic algorithms balance the workload in efficient way at the nodes.

2.3.3 Online and offline (Batch) mode scheduling

In on-line mode, a customer request is mapped with the running virtual machines when scheduler gets the request from customer side and each task is scheduled only once, the scheduling result remains unchanged. Some example of online modes scheduling algorithm are

opportunistic load balancing (OLB), minimum execution time (MET), minimum completion time (MCT) etc.

Offline scheduling is called batch mode scheduling in which upcoming application request is allocated to resources only at some predefined moments. It is used to calculate the processing time of larger number of tasks. Some example of batch modes scheduling algorithm are max-min, min-min etc.

2.3.4 Preemptive and Non-preemptive scheduling

In preemptive scheduling algorithm tasks can be interrupted to the current execution and task can be migrated to another resources.

In non-preemptive scheduling algorithm, when a task is allocated to cloud resource, it will not be free until task cannot be finished i.e. task is execute completely at the resource without interrupted. One task is executed at one resource in cloud environment i.e. interrupted is not allow in cloud environment during the execution of tasks.

2.4 Classification of scheduling scheme in cloud computing

The scheduling scheme is classified into three categories: Heuristic, meta-heuristic and hybrid scheme. The detailed classifications are presented as shown in Fig. 2.4. The objective of this study is to build the base of the scheduling algorithm used in cloud computing to carry out research in this area [7].

2.4.1 Heuristic scheduling algorithm

Heuristic algorithms are problem dependent and give good performance for a specific domain of problems but low performance for other domains. Normally, Heuristic algorithms give exact solution for specific domain of problem in finite amount of time but cannot solve hard optimization problems. There are lots of heuristic algorithms used in cloud environment like HEFT [8], min-min [9], max-min [10], round robin [11], dynamic round robin [12], first come first serve [13], shortest job first [14], bin-packing [15], deadline based scheduling algorithm [16], agent based scheduling algorithm [17], best fit[18] etc. These algorithms schedule the tasks at the virtual machine using different scheduling approach and optimize the parameters.

2.4.2 Meta-heuristic scheduling algorithm

Metaheuristic algorithms have gained huge popularity in the last twenty years due to its efficiency and effectiveness to solve large and complex problems. There are some properties of meta-heuristics algorithms like

- (i) These algorithms are not problem-specific.
- (ii) Meta-heuristic algorithm efficiently explores the search space to find (near) optimal solutions or sub-optimal solution of NP Complete problems.
- (iii) Meta-heuristic algorithms are approximate and usually non-deterministic.

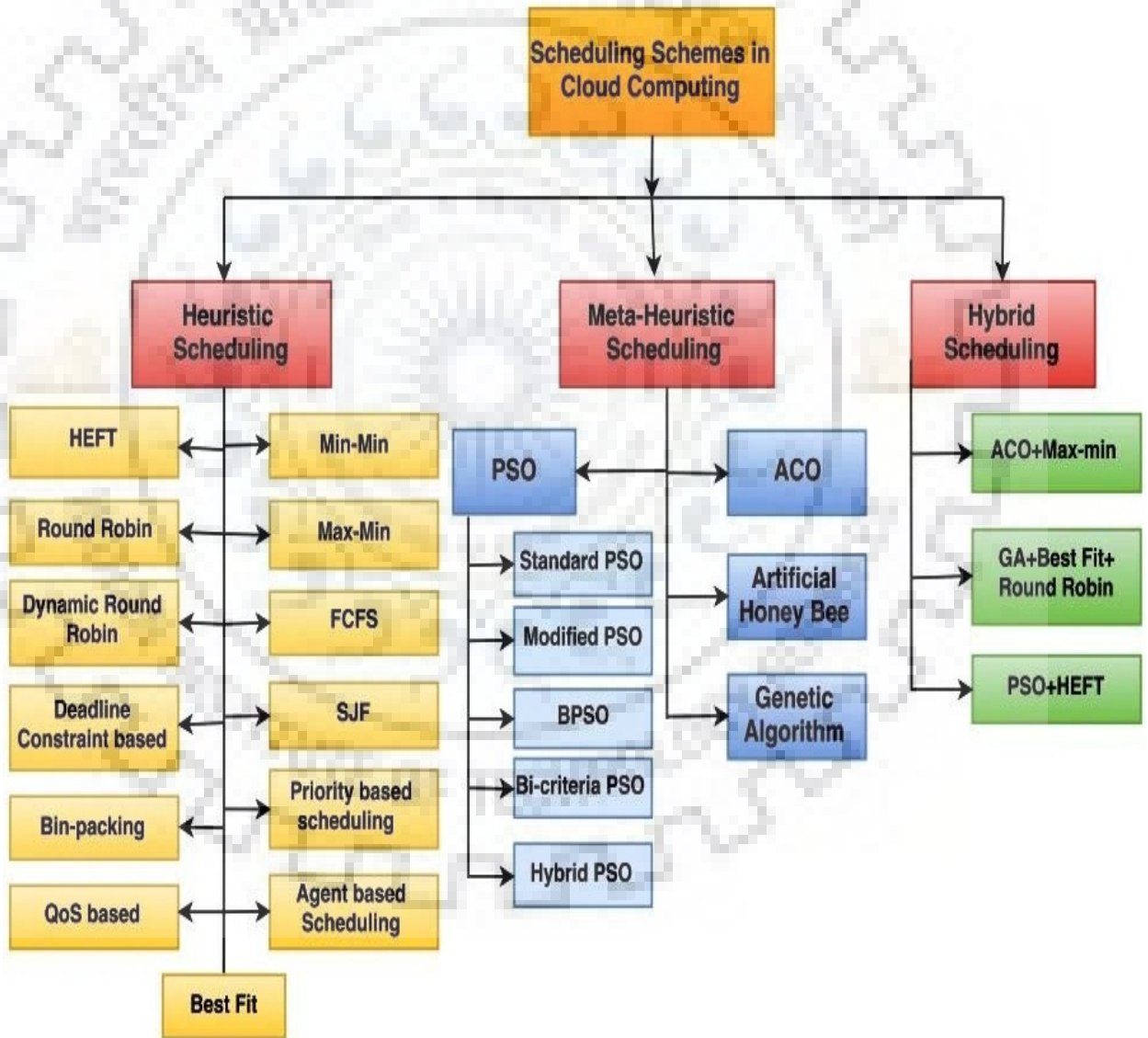


Figure 2.4 Classification of scheduling scheme in cloud computing

Metaheuristic algorithms are problem independent and applicable to solve various domains of problems with acceptable performance. Meta-heuristic methods are one of the common strategies for solving NP-hard optimization problems.

Meta-heuristic = Heuristic + Randomization

There are various meta-heuristic algorithm exist in cloud environment to find the approximate (suboptimal) solution of NP-Complete problem in short period of time. Scheduling of task is a NP-Complete problem due to large solution space and takes the long time to find the optimal solution.

The varied choice of meta-heuristic algorithms like particle swarm optimization (PSO)[19-23] , ant colony optimization (ACO) [24-25], artificial honey bee (ABC) [26] and genetic algorithm (GA)[27] etc. are shown in Fig. 2.4. The correct choice of optimization algorithm may be significantly useful in determining the precise solutions for a particular problem. We used particle swarm optimization in our work because convergence rate and complexity of PSO algorithm is better than the others meta-heuristic algorithms.

2.4.2.1 Particle Swarm Optimization

PSO is population based stochastic optimization algorithm which was proposed by Eberhart and Kennedy [19] in 1995 from swarm intelligence. PSO has been applied in many research and scientific application (model classification, function optimization, machine study, neural network training etc.) due to distinguishing characteristics [20-21] like as (a) Consist of limited number of parameters, there is no need to calculate the overlapping and mutation (b) simple and easy enumeration, (c) it is attractive because there are few parameters to adjust, (d) being free from derivation, (e) sensitivity move towards the fitness function and parameters, (f) less dependency at initial parameter, (g) relatively faster convergence and cheaper way rather than other meta-heuristic algorithm like GA, ABC, ACO etc. (h) high precision solutions. Particle swarm optimization (PSO) is a nature inspired optimization technique, modeled after the societal behavior of flocks of birds (or school of fish) i.e. how they discover and use the multi-dimensional search space in search of food as well as shelter [22].

PSO comprised of certain quantity of particles say N_p , entitled as swarm. Every particle gives a prospective solution. A Particle P_i , $1 \leq i \leq N_p$ has positions $X_{i,d}$ with velocity $V_{i,d}$, $1 \leq d \leq D$ in the d^{th} dimension of the space search. The value of D is identical for the entire particles. The fitness function is applied to estimate every particle for validating the worth of solutions. The

objective of PSO is to determine the position of particle those results best estimation of the fitness function. During the initialization procedure of PSO, every particle is allotted a random position as well as velocity to travel in the search space. In each iteration, every particle discovers its own best, i.e., personal best ($Pbest_i$) and the global best ($Gbest$). $Pbest_i$ represent the personal best position of particle has visited and $Gbest$ represent the global best position of the particle and its neighbors have visited since the starting of the iteration. To attain the best position globally, it makes use of its personal as well as global best for updating of the velocity $V_{i,d}$ and position $X_{i,d}$ with help of the following equations:

$$V_{i,d}(t+1) = \omega \times V_{i,d}(t) + c_1 \times r_1 \times (X_{Pbest_{i,d}} - X_{i,d}) + c_2 \times r_2 \times (X_{Gbest} - X_{i,d}) \quad (1)$$

$$X_{i,d}(t+1) = X_{i,d}(t) + V_{i,d}(t+1) \quad (2)$$

Where value of w is between 0 to 1 denotes inertia weight, $c_1, c_2, 0 \leq c_1, c_2 \leq 2.05$ denotes acceleration coefficients, r_1, r_2 are random number between 0 to 1. The updating procedure is reiterated until and unless it's reached to an adequate value of $Gbest$. After obtaining the newly updated position, the particle estimates the fitness function as well as updates $Pbest_i$ and $Gbest$ for the minimization problem as follows:

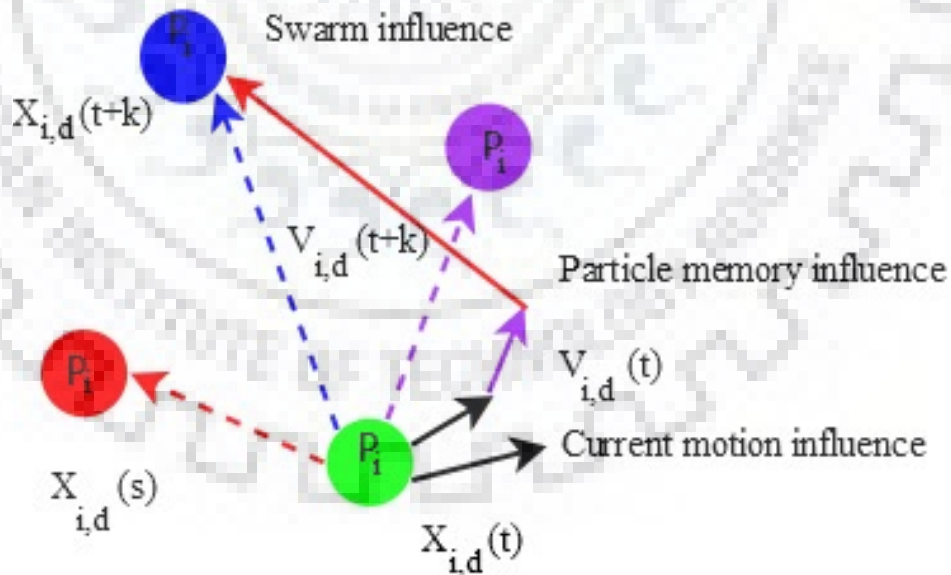


Figure 2.5 Flying of a particle in the search space

$$Pbest_i = \begin{cases} P_i, & \text{if } (fitness(P_i) < fitness(Pbest_i)) \\ Pbest_i, & \text{Otherwise} \end{cases} \quad (3)$$

$$Gbest = \begin{cases} P_i, & \text{if } (fitness(P_i) < fitness(Gbest)) \\ Gbest_i, & \text{Otherwise} \end{cases} \quad (4)$$

Fig.2.5 clarifies how a particle travels around the search space to attain a best solution globally. In the beginning a particle P_i , takes up the position $X_{i,d}(t)$ with velocity $V_{i,d}(t)$ at a point of time and traveling in some direction. Afterward the particle adjusts the direction with the swarm's influence and takes up a new position $X_{i,d}(t+k)$ with velocity $V_{i,d}(t+k)$ and ultimately attains the global best position $X_{i,d}(s)$, where $s > t+k$ and the variable s , t , and k are characterized on a specific time. The flow chart of a PSO algorithm is shown in Fig.2.6.

PSO-based scheduling algorithm is further classifies as standard PSO [28], modified PSO [29], binary PSO [30] etc. which have been applied to schedule the upcoming tasks or workflow at virtual machines (cloud resources).

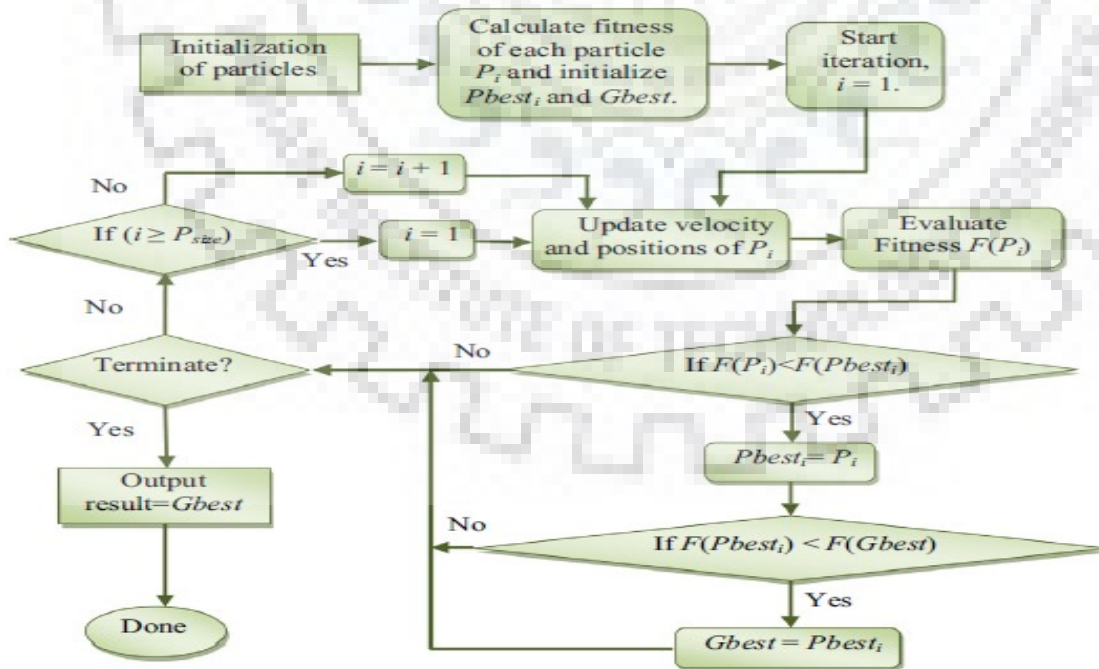


Figure 2.6 Flow chart of PSO algorithm [23]

2.4.3 Hybrid scheduling algorithm

Hybrid scheduling algorithms are combination of meta-heuristic algorithm with heuristic algorithm to solve the problem of task scheduling in cloud environment. Yassa et al. combine Ant colony meta-heuristic algorithm with max-min heuristic algorithm and optimize the total processing time and cost parameters. It helps to improve the energy consumption using hybrid algorithm (PSO with HEFT) in this paper [31]. Delavar and Aryan proposed an algorithm that optimizes the QoS parameters like makespan time, load balancing at virtual machines as well as host and speed-up ratio. It combines the Genetic Algorithm with the Best Fit and Round Robin algorithms [32]. Fig. 2.7 represent the percentage of meta-heuristic algorithm used to solve the problem of scheduling in the field of cloud computing.

2.4.4 Benefit of Resource Scheduling

Advantages of resource scheduling in the field of cloud computing are given below:

- Effective resource scheduling reduces (optimizes) the execution time as well as makespan time of tasks simultaneously.

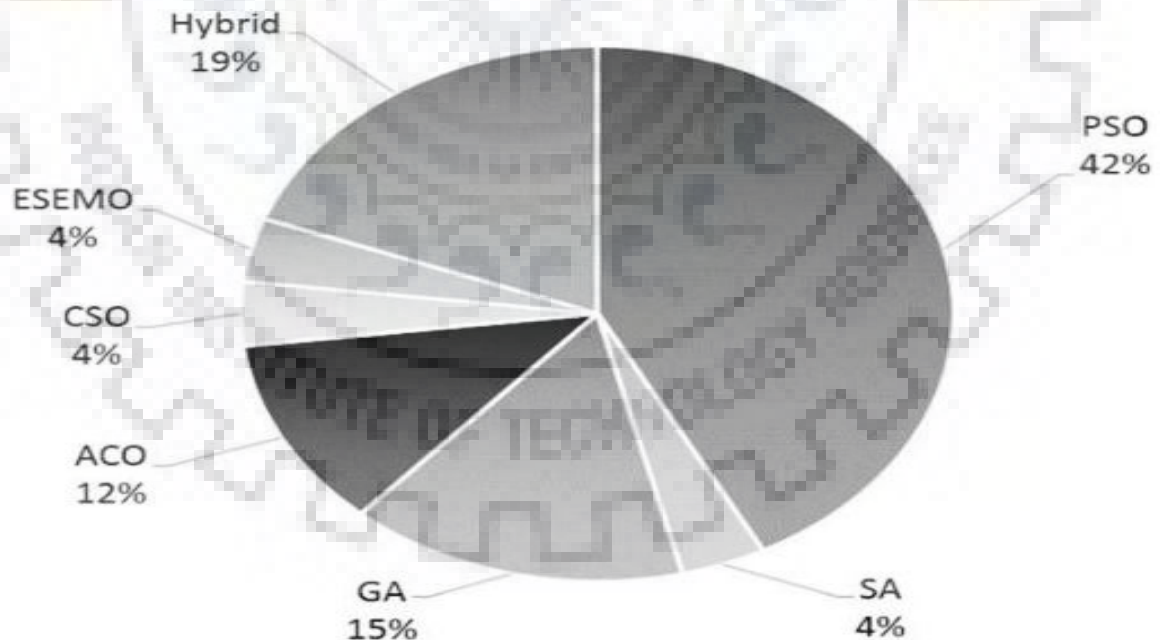


Figure 2.7 Percentage of meta-heuristic algorithm used in scheduling problem [7]

- There is always conflict between execution time and execution cost but effective resource scheduling algorithm can optimize both the parameters simultaneously.
- Efficient resource utilization under different requirements of priority and avoid the over provisioning and under provisioning types of problem in cloud computing.
- Increases the robustness and decrease the failure rate of cloud resources by efficient scheduling techniques.
- Power consumption is a serious issue in the field of cloud computing that is reduces with the help of efficient resource scheduling without the violation of SLA.
- Deadline violation rate is improved by better resource scheduling techniques after provisioning of resources in the field of cloud computing.
- Efficient Resource scheduling distributes the workload among the virtual machines in such a manner so that no virtual machine is in overloaded or under loaded condition i.e. reduces the chances of overloaded and underloaded problem.

2.5 Resource Allocation

Resource allocation is a challenging problem for the service provider due to resource heterogeneity, heterogeneous application (CPU intensive, memory intensive) locality limitations and on demand requests in cloud environment. Cloud users submit the request for service (resources) from anywhere and anytime with the help of graphical user interface or web interface. There are lots of data center available to process the request in cloud environment but request is directed to nearest data center due to low latency.

If request is not directed to nearest data centers then there is possibility of high latency due to which some of the quality of service (QoS) parameter affected, like deadline, response time, elasticity etc. SLA violation is increased after affecting the QoS parameters. User service request is received by job request handler or gatekeeper at the data center. Job request handler performs the Turing test to verify that it is coming from legitimate user or attacker. If request is coming from attacker then block the user based upon the source IP address, port address etc. If request is coming from legitimate user then it is passed to controller node that contains the information about all the resources. Cloud service provider contains all the resources in resource pool and performs the resource provisioning.

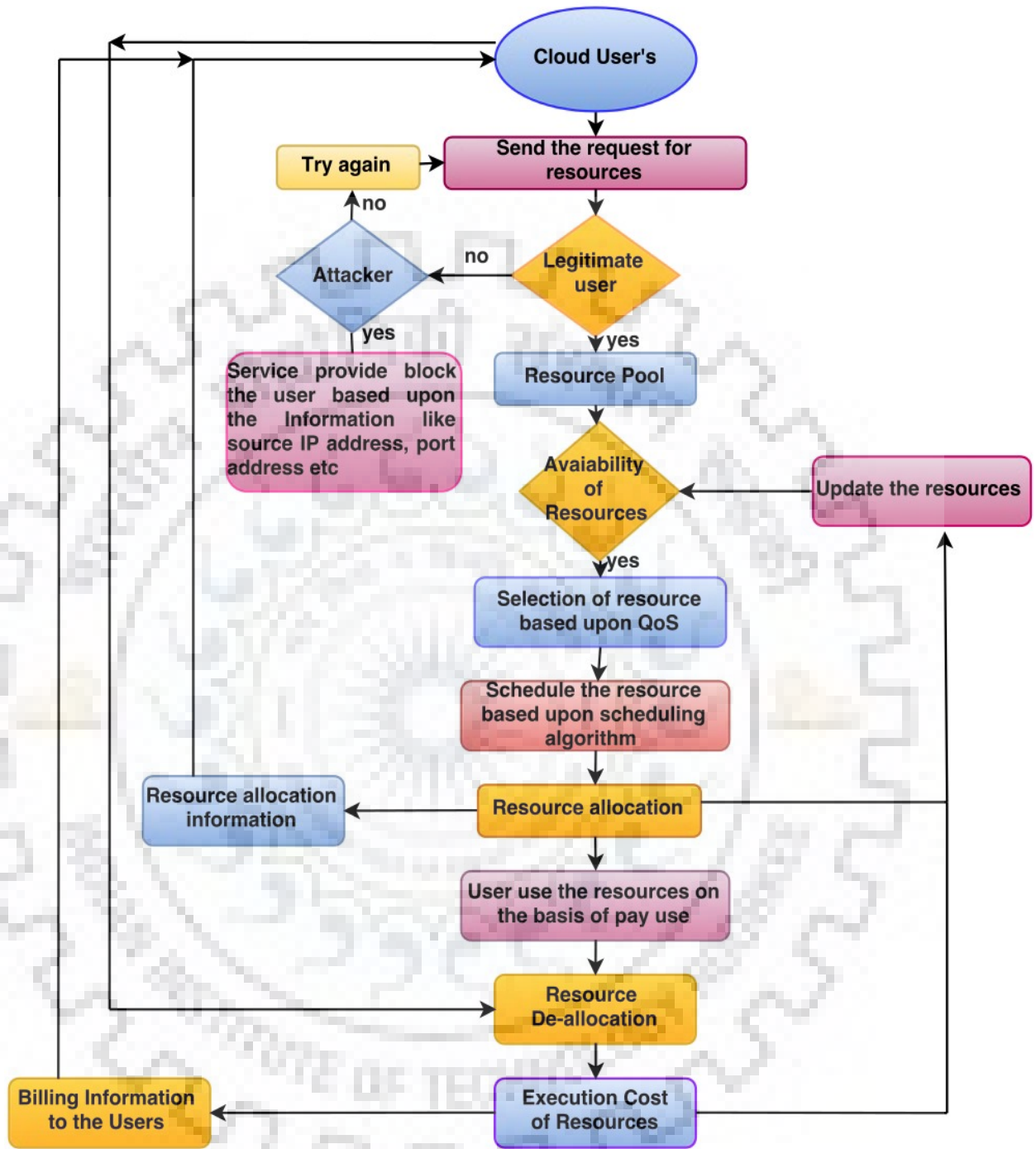


Figure 2.8 Process of efficient resource allocation in cloud computing

Controller node check the availability of cloud resources, if resources are available then select the resources based upon the QoS parameters, like deadline, response time etc otherwise wait for availability of resources. The aim of cloud users is to execute their application within budget and minimum time while cloud service provider tries to utilize the resources efficiently.

There is always a trade-off between execution time and cost due to heterogeneous nature of cloud resources as well as upcoming user's application. To make a balance between execution time and cost, a trade-off solution is required. Fig. 2.8 represents the process of efficient resource allocation in cloud environment to minimize the execution cost as well as time.

2.5.1 Classification of resource allocation

Resource allocation technique is divided into two group's namely strategic based and parametric based resources allocation. These groups are further classified into different subgroup in Fig. 2.9. The aim of this classification is to provide the basic concept and techniques of resource allocation that is helpful for future research in the field of cloud computing. Strategic based resource allocation is divided into three subcategory namely prediction based resource allocation, artificial intelligence based resource allocation and dynamic resource allocation in cloud environment. Prediction based resource allocation approach is necessary to predicts the future demands of upcoming request on the basis of past history in cloud computing. Prediction based algorithm is used to assigned the virtual machines for the future before they are needed [33]. This is an effective and necessary approach for efficient resource allocation in infrastructure as a service cloud computing [34].

Artificial intelligence is used in cloud environment to reduce the failure rate as well as chances of error occurrence and focus the creation of intelligent methodology that work for resource allocation like human. It also provides better precision and greater accuracy for resource allocation in IaaS [35-38]. To handle the fluctuating demand (request) of cloud user, we need dynamic resource allocation technique in cloud computing. This technique is used to allocate the resource efficiently and fulfill the unstable demands of users' [39]. Dynamic allocation technique also provides guaranteed quality of services for avoiding the service level agreement (SLA) violence [40].

Parametric based allocation technique is divided into six subgroup namely utilization aware resource allocation, execution cost based resource allocation, efficiency based resource allocation, load balancing based resource allocation, power consumption based resource allocation and QoS parameters based resource allocation in cloud computing [1]. The objective of service provider is to allocate the resource in such a manner so that utilization of cloud resource should be maximum and execution time should be minimum [41-42]. Cloud service providers provides the services to fulfill the user's demand, In return, they want the maximum

profit and revenue with extreme resource utilization, while cloud users ‘want to pay minimum amount for high quality services [43].

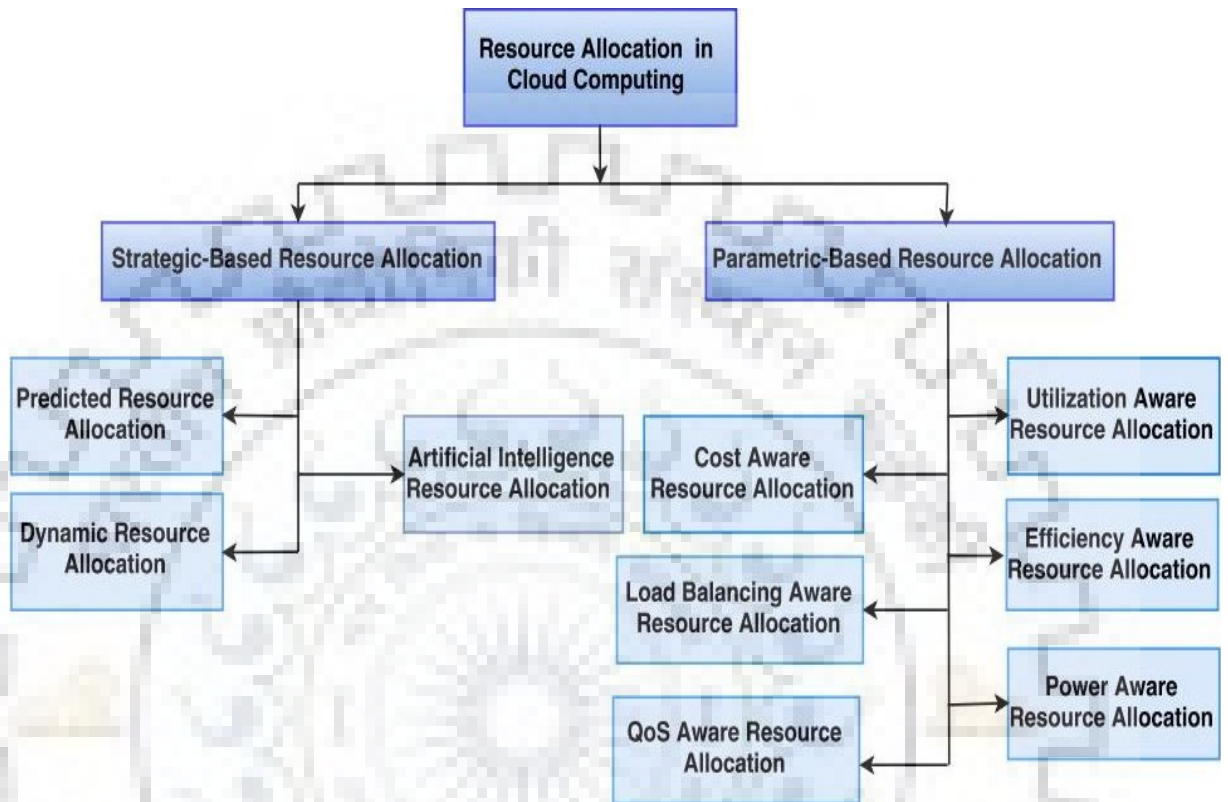


Figure 2.9 Taxonomy of resource allocation in cloud computing

Therefore, efficient resource allocation algorithms play an important role in cloud environment for cloud service provider as well as users. Workload is important parameter in load balancing because status of the resources (overloaded or underloaded) is monitor on the basis of available workload at particular resource. If any resource is in overloaded state then transfer the workload from overloaded resource to underloaded resource [44].

Efficiency aware resource allocation algorithms directly affects the performance, it optimize the different parameters like bandwidth, response time, execution time etc. for allocation the resources efficient way [45]. The rapid growths in demand of computational power tends to massive growth in cloud data centers and require large amount of energy consumption in cloud data centers which becomes a serious threat to the environment. To reduce the energy consumption in cloud computing is a challenging problem due to incompatibility between workstation (physical machine) and unpredictable users demand. Cloud data centers should

manage in such a manner so that they generate less heat, due to which energy consumption and cost can reduce [46]. QoS based scheduling is a key issue in the field of cloud computing in which resources is distributed as per user's demand to optimize the QoS parameters like availability, fault tolerance, recovery time, reliability, throughput and SLA for the both cloud providers and users [47].

2.6 Simulation Tool used in cloud computing

Experimenting new technique in real cloud environment is not possible practically because some experiments compromise the end user quality of service. There are some prominent simulations tools are available in cloud computing to test and analyze the new proposed scheduling techniques in different context. The most popular simulation tool for resource provisioning and scheduling is cloudsim toolkit to calculate the QoS parameters like execution time, makespan time, execution cost, throughput, energy consumption etc. by extending existing classes according to the requirements of algorithm.

2.6.1 Cloudsim

Cloudsim is an extensible simulation toolkit that is used for simulation and experimentation of infrastructures and provides the application environments of cloud computing. Cloudsim contain the service broker that is used for provisioning and scheduling of cloud resources. It supports the simulation of the network connections between the simulated hosts and virtualization engine aids in the management of multiple, independent, virtualized services on a data center host. Cloudsim contains large number of datacenters (infinite cloud resources) and many number of host can be created in one datacenter, depends upon the configuration and processing capacity of datacenter. Virtual machine is a processing entity in cloud environment that is controlled by hypervisor such as KVM, XEN etc. each host contains the number of virtual machines depending upon the configuration of host (processing speed, number of cpu, memory etc.). Therefore most of the researches are using cloudsim simulator (Fig. 2.10) to implement the new algorithm for resource provisioning and scheduling.

2.6.2 DCSim

Data Centre Simulator (DCSim) is used to provides the service into multiple tenants using data center based resource provisioning and scheduling algorithms in virtualization environment.

2.6.3 Cloud Analyst

Cloud analyst extends the functionalities of cloudsims to test and analyze the behavior of large scaled Internet application and repeat the simulation for performing the variations in the required parameters.

2.6.4 EMUSIM

To test the performance and service behavior of new proposed algorithm, this tool (Automated Emulation Framework based) is used emulation in cloud computing.

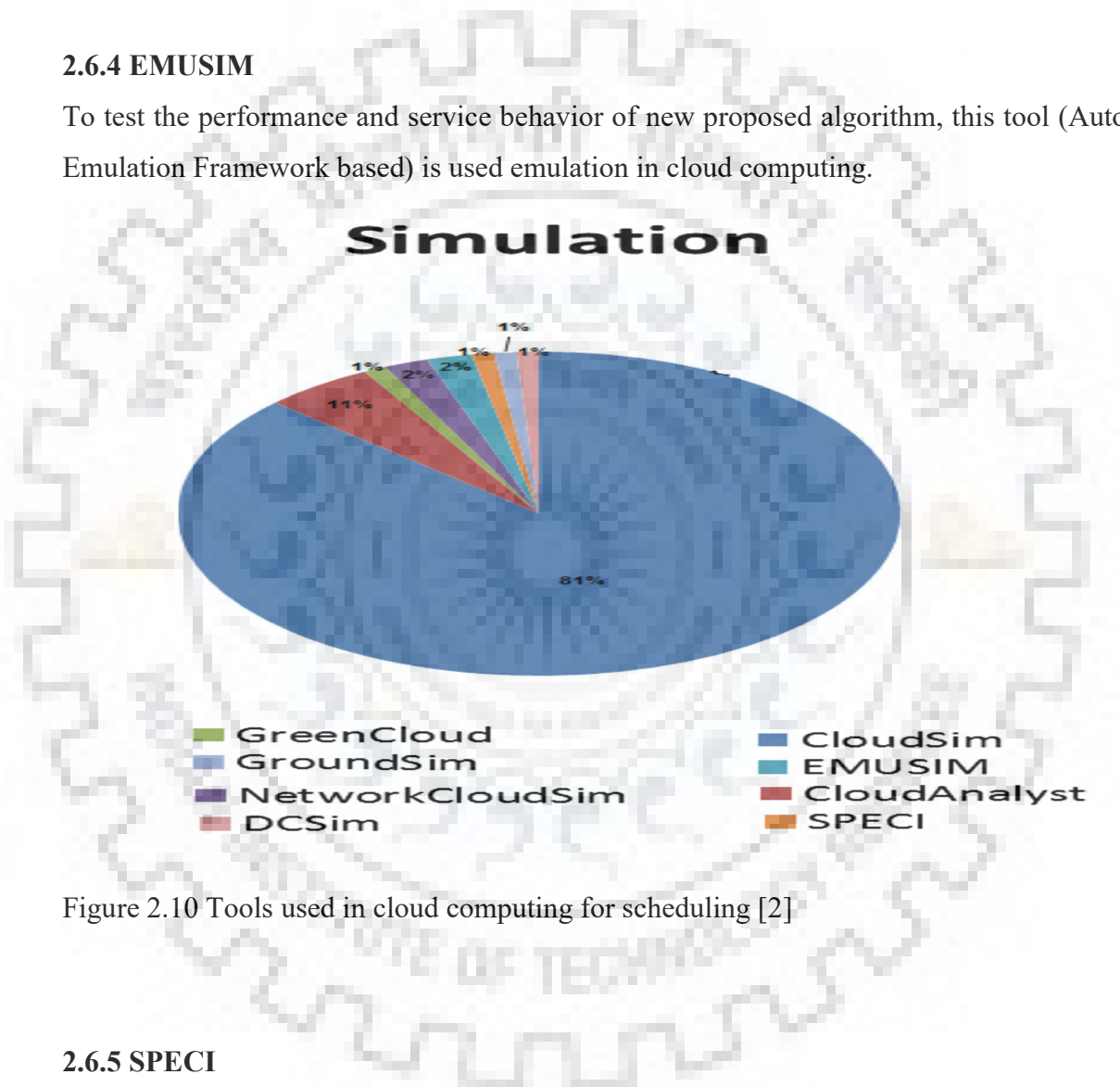


Figure 2.10 Tools used in cloud computing for scheduling [2]

2.6.5 SPECI

Simulation Program for Elastic Cloud Infrastructures (SPECI) is collection of two packages: experiment execution of component and data center layout and topology is used to evaluate the performance of large datacenters under design and size policy.

2.6.6 GroundSim

Infrastructure as a service based GroundSim is used to detect the events for scientific applications based upon simulation thread. Real environment can be realized by the integration of GroudSim into the ASKALON.

2.6.7 GreenCloud

GreenCloud is an extension of cloudsim simulator and used to evaluate the performance of energy efficient scheduling algorithms by calculating the parameters like energy consumption of communication links, computing servers and network switches.

2.6.8 NetworkCloudSim

It is an extension of cloudsim toolkit to evaluate the performance of high performance computing applications and complex workflows in real cloud environment.

2.7 Summary

In this chapter, fundamentals concepts and advantages of existing resource provisioning techniques is discussed briefly in the field of cloud computing. The categorization of scheduling algorithm is also discussed in terms of static and dynamic, offline and online mode, preemptive and non-preemptive scheduling algorithm. Classification of scheduling algorithm (heuristic, meta-heuristic and hybrid) has been described. Characteristics and basic concept of particle swarm optimization algorithm is discussed. Further taxonomy of resource allocation techniques based upon different parameters is also described in this chapter. Simulation tools that are used for implementing new algorithm in the field cloud computing is also discussed briefly.

2.8 References:

- [1] S.H.H. Madni, M.S.A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review," *Journal of Cluster Computing*, vol. 20, no. 3, pp. 2489-2533, Dec. 2016.
- [2] S. Singh and I. Chana, "A survey on resource scheduling in cloud computing issues and challenges," *Journal of Grid Computing*, vol. 14, no. 2, pp. 217-264, Feb. 2016.
- [3] B. Javadi, J. Abawajy and R. O. Sinnott, "Hybrid cloud resource provisioning policy in the presence of resource failures," in *4th International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 10-17, 2012.

- [4] PK Sur, RK Das and N Mukherjee, "Heuristic-based resource reservation strategies for public cloud," *IEEE Transactions on Cloud Computing*, vol. 4, no. 4, pp. 392-401, Nov. 2014.
- [5] S. Singh, I. Chana, "Cloud resource provisioning: survey, status and future research directions," *Knowledge and Information System*, vol. 49, no. 3, pp. 1005-1069, Feb. 2016. <http://dx.doi.org/10.1007/s10115-016-0922-3>.
- [6] S. Singh, I. Chana, "Resource provisioning and scheduling in clouds: QoS perspective," *Journal of Supercomputing*, vol. 72, no. 3, pp. 926-960, Jan. 2016. <http://dx.doi.org/10.1007/s11227-016-1626-x>.
- [7] M. Masdari, S. ValiKardan, Z. Shahi, S. I. Azar, "Towards workflow scheduling in cloud computing: a comprehensive analysis," *Journal of Network and Computer Application*, vol. 66, pp. 64-82, May 2016.
- [8] N. Chopra, S. Singh, "HEFT based workflow scheduling algorithm for cost optimization within deadline in hybrid clouds," in 4th international conference on computing, communications and networking technologies (ICCCNT), India, Jan. 2014.
- [9] H. Chen, F. Wang, N. Helian and G. Akanmu, "User Priority Guided Min-Min scheduling algorithm for cloud computing," in national conference on Parallel Computing Technologies (PARCOMPTECH), Bangalore, India, Oct. 2013.
- [10] Y. Mao, X. Chen, and X. Li, "Max-min task scheduling algorithm for load balance in cloud computing," in *Proceedings of International Conference on Computer Science and Information Technology*, vol. 255, pp. 457-465, Springer, New Delhi, India, 2014.
- [11] S. Subramanian, G. N. Krishna, M. K. Kumar, P. Sreesh, and G. Karpagam, "An adaptive algorithm for dynamic priority based virtual machine scheduling in cloud," *International Journal of Computer Science Issues*, vol. 9, no. 6, pp. 397-402, 2012.
- [12] C.C. Lin, P. Liu, and J. J. Wu, "Energy-aware virtual machine dynamic provision and scheduling for cloud computing," in *International Conference on Cloud Computing*, pp. 736-737, DC, USA, July 2011.
- [13] W. Li, H. Shi, "Dynamic load balancing algorithm based on FCFS," in 4th international conference on Innovative computing, information and control (ICICIC), Kaohsiung, Taiwan, Dec. 2009.
- [14] R. K. Mondal, E. Nandi and D. Sarddar, "Load balancing scheduling with shortest load first," *International Journal of Grid Distribution Computing*, vol. 8, no. 4, pp. 171-178, Oct. 2015.

- [15] M. Sheikhalishahi, R. M. Wallace, L. Grandinetti, J. L. V. Poletti and F. Guerriero, "A multi-dimensional job scheduling," *Future Generation Computer System*, vol. 54, pp. 123–131, 2016.
- [16] R. Singh, S. Singh, "Score based deadline constrained workflow scheduling algorithm for Cloud systems," *International Journal on Cloud Computing: Services and Architecture*, vol. 3, no. 6, pp. 31-41, Dec. 2013.
- [17] J. O. G. Garcia, and A. R. Nafarrate, "Collaborative agents for distributed load management in cloud data centers using live migration of virtual machines," *IEEE transactions on services computing*, vol. 8, no. 6, pp. 916–929, Dec. 2015.
- [18] K. C. Gouda, T. V. Radhika and M. Akshatha, "Priority based resource allocation model for cloud computing", *International Journal of Science, Engineering and Technology Research (IJSETR)*, vol. 2, no. 1, pp. 215-219, Jan. 2013.
- [19] J. Kennedy and R.C. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, pp. 1942–1948, Perth Australia, Dec. 1995.
- [20] Q. Bai, "Analysis of particle swarm optimization algorithm," *Computer and Information Science*, vol. 3, no. 1, pp. 180–184, Feb. 2010.
- [21] N. Singh, R. Arya and R.K. Agarwal, "A novel approach to combine features for salient object detection using constrained particle swarm optimization," *Pattern Recognition*, vol. 47, no. 4, pp. 1731–1739, 2014 .
- [22] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in sensor network: A brief survey," *IEEE Transaction on systems, man, and cybernetics—part c: applications and reviews*, vol. 41, no. 2, pp.262-267, Mar. 2011.
- [23] P. Kulia and P. K. Jana, "Energy efficient clustering and routing Algorithms for wireless sensor network: particle swarm optimization approach," *Engineering Applications of Artificial Intelligence*, vol. 33, pp. 127-140, 2014.
- [24] E.Pacini, C.Mateos and C.G.Garino, "Balancing throughput and response time in online scientific Clouds via Ant Colony Optimization (SP2013/2013/00006)", *Advances in Engineering Software*, vol. 84, pp. 31-47, June 2015.
- [25] X LU, Z. Gu, "A load-adaptive cloud resource scheduling model based on ant colony algorithm," in *International conference on Cloud Computing and Intelligence Systems (CCIS)*, pp. 296–300, China, Sep. 2011. <http://dx.doi.org/10.1109/CCIS.2011.6045078>.
- [26] D. Babu and P. Venkata, "Honey bee behavior inspired load balancing of tasks in cloud

- computing environments,” *Applied Soft Computing*, vol.13, no. 5, pp. 2292–2303, May 2013.
- [27] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mondal and S. Dam, "A genetic algorithm (GA) based load balancing strategy for cloud computing," *Procedia Technology*, vol. 10, pp. 340-347, 2013.
- [28] L. Guo, S. Zhao, S. Shen and C. Jiang, “Task scheduling optimization in cloud computing based on heuristic algorithm,” *Journal of Network*, vol. 7, no. 3, pp. 547–553, March 2012.
- [29] Z. Tarek, M. Zakria, F.A. Omara, “Pso optimization algorithm for task scheduling on the cloud computing environment,” *International Journal of Computers and Technology*, vol.13, 2014.
- [30] J. Kennedy, R. C. Eberhart, A discrete binary version of the particle swarm algorithm, in: *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 5, pp. 4104-4108, 1997.
- [31] S. Yassa, R. Chelouah, H. Kadima and B. Granado, “Multi-objective approach for energy-aware workflow scheduling in cloud computing environments, *The Scientific World Journal*, Sep. 2013. <http://dx.doi.org/10.1155/2013/350934>.
- [32] A. G. Delavar and Y. Aryan, “HSGA: a hybrid heuristic algorithm for workflow scheduling in cloud systems,” *Cluster Computing*, vol. 17, no. 1, pp.129–37, March 2014.
- [33] S. Islam, J. Keung, K. Lee and Liu, “Empirical prediction models for adaptive resource provisioning in the cloud,” *Future Generation Computer System*, vol. 28, no. 1, pp. 155–162, 2012.
- [34] R. Patel, D. Dahiya, “Aggregation of cloud providers: a review of opportunities and challenges,” in *International Conference on Computing, Communication & Automation (ICCCA)*, pp. 620–626, Noida, India, 2015.
- [35] S. K. Panda and P.K. Jana, “An efficient resource allocation algorithm for IaaS cloud,” in *Distributed Computing and Internet Technology*, vol. 8956, pp. 351–355, New York, 2015.
- [36] J. Kumar and A. K. Singh, “Workload prediction in cloud using artificial neural network and adaptive differential evolution,” *Future Generation Computer system*, vol. 81, pp. 41-52, April 2018.
- [37] J. Kumar, R. Goomer and A.K. Singh, “Long Short Term Memory Recurrent Neural Network (LSTM-RNN) Based Workload Forecasting Model For Cloud Datacenters,” in *Procedia Computer Science*, vol. 125, pp. 676-682, 2018.

- [38] S. Mehfuz, S. Urooj, S. Sinha, “Wireless Body Area Networks: A Review with Intelligent Sensor Network-Based Emerging Technology,” in *Advances in Intelligent Systems and Computing*, vol. 339, pp. 813-821, New Delhi, 2015.
- [39] R. Shelke and R. Rajani, “Dynamic resource allocation in cloud computing,” *International Journal of Engineering Research & Technology*, vol. 2, no. 10, 2013.
- [40] S. Jayanthi, “Literature review: dynamic resource allocation mechanism in cloud computing environment” in *International Conference on Electronics, Communication and Computational Engineering (ICECCE)*, pp. 279–281, India, 2014.
- [41] H. Wang, F. Wang, J. Liu, D. Wang and J. Groen, “Enabling customer-provided resources for cloud computing: potentials, challenges, and implementation,” *IEEE Transaction Parallel and Distributed System*, vol. 26, no. 7, pp. 1874–1886 , 2015.
- [42] M.Kumar and S. C. Sharma, "Priority Aware Longest Job First (PA-LJF) algorithm for utilization of the resource in cloud environment," in *International conference on Computing for Sustainable Global Development (INDIACom)*, pp. 415-420, New Delhi, India, 2016.
- [43] Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” *Journal of Internet Services and Applications*, vol. 1, no. 1, pp. 7–18, 2010.
- [44] M. Kumar and S. C. Sharma, “Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing,” in *7th International Conference on Advances in Computing & Communications (ICACC)*, vol. 115, pp.322-329, Cochin, India, 2017.
- [45] A. J. Younge, G. V. Laszewski, L. Wang, S. L. Alarcon and W. Carithers, “Efficient resource management for cloud computing environments,” in *international conference on green computing*, pp. 357–364. USA, 2010.
- [46] R. Buyya, A. Beloglazov and J. Abawajy, “Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges,” in *International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)*, Las Vegas, USA, 2010.
- [47] A. Abdelmaboud, D.N.A Jawawi, I. Ghani, A. Elsafi and B. Kitchenham, “Quality of service approaches in cloud computing: asystematic mapping study,” *Journal of System and Software*, vol. 101, pp. 159–179, 2015.

CHAPTER-3

LOAD BALANCING WITH ELASTICITY USING HEURISTIC TECHNIQUE

3.1 Concept of Load Balancing and Elasticity

Cloud computing provides the services either in the form of software application or hardware infrastructure on the basis of pay per use over the internet. User can send the request at any time for the resource to cloud service provider and cloud resource broker (cloud service provider) selects the best resource within user-defined deadline and budget. Cloud resource broker provides the on-demand service to the user. The number of users and applications are increasing gradually in cloud environment and in turn there is increase in the workload and traffic at the web applications which are deployed in the virtual machine (cloud resource). Therefore cloud resource broker needs an efficient algorithm that distributes the task fairly in all the running virtual machines and reduces the task rejection ratio so that the entire user task can be executed.

The main objective of the load balancing is to utilize the cloud resource in a manner that improves the average resource utilization ratio, response time, task acceptance ratio and scalability of the web application. Efficient load balancing gives the minimum makespan time of tasks and balance the workload among the virtual machines. It also prevents bottleneck of the system which may occur due to load imbalance. Load balancing is one of the challenging research areas in the field of cloud computing. It is difficult to predict and calculate all possible task-resource mapping in cloud environment. So we need an efficient scheduling algorithm which can distribute the task in effective manner, so that less number of virtual machines faces overload or under-load condition. Further need to monitor the virtual machine continuously and perform the load balancing operation. Cloud resource broker (CRB) monitor the virtual machine continuously in cloud environment. If any virtual machine is in overload or under-load condition after the scheduling the tasks then cloud resource broker start the load balancing operation at virtual machines and migrate the task from overloaded to under loaded virtual machines.

The ability of auto scaling on upcoming demands in cloud computing is biggest advantage for services provider as well as for user [1]. Auto scaling can reduce the risk, which is associated with request/load overflow causing server failure. Two types of auto scaling approaches are

available in cloud environment i.e. reactive and proactive. In this chapter we are using prediction based proactive approach that predicts the future demands on the basis of past history. Scalability can be classified into two type's horizontal scalability (scale out) and vertical scalability (scale up). Vertical scalability can be achieved by making changes in the existing resources such as memory, hard drives, CPU's, etc. Vertical scalability is not generally used in cloud environment, because common operating systems don't support these changes without rebooting on existing resources like memory, CPU's. In adding or releasing of one or more machine instance or computing node of same type is used horizontal scaling. Adding the IT resource in horizontally is called scale-out and releasing the IT resource horizontally scale-in. horizontal scaling is better than vertical scaling in cloud environment, because it is less expensive and not limited by hardware capacity.

3.2 Contribution

Elastic resource provisioning with quality of service (QoS) parameter (deadline, high availability, priority etc.) is one of the most challenging problem in the field of cloud computing. Therefore cloud service provider needs an efficient load balancing algorithm that reduces the makespan time as well as task rejection ratio within user defined deadline.

Specific contribution of this chapter includes:

- The developed algorithm distributes the task and adds the cloud resource if task rejection ratio is more than the SLA defined threshold value.
- The developed algorithm monitors the load at each virtual machine and data centre continuously. If any virtual machine is overloaded condition then transfer the task from over loaded to under loaded virtual machine using the task migration policy.
- We have developed a scheduling algorithm based on the last optimal k-interval that balances the workload among all the virtual machine with elastic resource provisioning and deprovisioning which overcome the drawback of algorithm proposed by Somasundaram et al [14].

3.3 Related Work and Research Gap

Several static [2, 5-6] and dynamic algorithms [7, 12-19] have been reported for load balancing in last decade. Existing job scheduling algorithms like conservative backfill, EASY etc. are

unable to fill the resource gap efficiently. The work done by [2] focus on improving the backfill algorithm (IBA), it not only improves the processing time of jobs but also provide the guarantee of quality of services in cloud environment. Authors improve the IBA using balanced spiral (BS) method but this algorithm does not provide better processing time when job requests enter randomly in cloud environment. Dubey et al. [3] proposed an algorithm for metascheduler to solve the job scheduling problem in cloud computing and removed the limitation of IBA algorithm. Authors improved the processing time of upcoming jobs and resource utilization ratio of cloud resources considering priority of job as quality of service parameter. Sahoo et al. [4] proposed an algorithm based upon the greedy technique that reduces the makespan time and execution time of the tasks without using task migration or virtual machine migration approach for load balancing; it does not provide better results in real environment. First come first serve (FCFS) and shortest job first (SJF) [5-6] are static algorithms that are suitable for batch system.

Literature review on dynamic based algorithm is reported in Table 3.1. Huankai Chen et al. [7] proposed a user guided min-min load balancing algorithm that not only minimize the execution time of the tasks but also remove the drawback of min-min algorithm (load is not properly balanced at each node). Proposed algorithm is simulated at matlab toolbox to reduce the average task completion time and increase the average resource utilization ratio. There are some dynamic algorithms which are using soft computing approach like Honey bee behavior [8], particle swarm optimization (PSO) [9], ant colony optimization (ACO) [10], differential evaluation algorithm [11] etc. to solve the problem of load balancing.

S. Chhabra, and A. K. Singh proposed Optimal Physical Host with effective Load Balancing based algorithm [25] that find the optimal host using the probabilistic model and optimize the parameters makespan time, energy consumption and throughput in cloud environment. Bharti and K. K. Pattanaik proposed task requirement preprocessing and scheduling based mechanism that optimizes the energy consumption a well as network output load parameters [26]. S. Javanmardi et al., proposed hybrid job scheduling algorithm (genetic algorithm and fuzzy logic based) for cloud environment that not only reduce the execution but also reduce the execution cost [27]. Table 3.1 summarizes the research papers related to the present work in terms of type of dynamic scheduling & load balancing algorithm, tool used to implement the algorithm, performance metrics migration technique and limitation of the proposed algorithm.

Table 3.1 Literature review on dynamic scheduling, load balancing & elasticity based algorithm

S. No.	Year	Parameters	Technique	Tool	Limitations
1	2011 [17]	computation time and cost with deadline	Proposed a new QoS-based workflow scheduling algorithm based on Partial Critical Paths concept.	Java based simulator	Algorithm is implemented on Java based simulator and does not consider elasticity and deadline QoS parameters.
2	2013 [7]	Makespan time, Resource utilization ratio	Apply improved LBIMM algorithm and reassigned the tasks based upon load at node.	Matlab	Rescheduling of task will increase complexity and time.
3	2013 [14]	Response time, throughput	Proposed resource broker that provide adaptive load balancing and elastic resource provisioning and deprovisioning.	Eucalyptus cloud	Load balancer unable to maintain the session in multi-tenant environment when same user request the multiple VM instances.
4	2014 [13]	Response time, Processing time	Proposed a new cloud-brokering architecture to improve brokering performance.	Cloud Analyst	Experiment is conducted on cloudsim; author is not ensuring that algorithm has not been tested in real test bed environment.
5	2014 [15]	Energy consumption, reduce VM migration time	Proposed a VM migration policy on a host that has the minimum correlation coefficient.	Cloudsim	VM migration is costly and time consuming rather than task migration. Proposed algorithm does not consider execution time and elasticity concept.
6	2015 [3]	Resource utilization, processing time	Apply IBA algorithm to balance the load considering priority as QoS parameter of jobs.	Cloudsim	Proposed algorithm does not give the guarantee of load balancing and does not consider elasticity concept.
7	2015 [19]	Ratio cost to budget, ratio makespan time to deadline	Proposed an algorithm for efficient management of budget with deadline constraint.	Cloudsim	SPSS algorithm take lots of time for planning (10 min or more for 100 work flow) to distribute the work flow to virtual machine. While proposed dynamic algorithm have high failure rate.

8	2016 [12]	Execution time, makespan time	Dynamic and automatically provides the resources by improving knowledge model	Openstack and cloudsim	Some important constraint is not considered like elasticity, scalability, execution cost, heterogeneous virtual machine
9	2016 [16]	Profit of cloud service provider, resource utilization ratio.	Proposed scheduling algorithm based on auction mechanism	Cloudsim	When number of client is increase , large auction deadline interval will have a negative impact on the profit of the cloud service provider to some extent
10	2016 [18]	Resource utilization ratio, no. of leased scheduled, no of leased rejected	Proposed a new algorithm to improve the performance of backfilling algorithm by Analytic Hierarchy Process.	Open nebula private cloud	Main limitation of backfilling algorithm is estimation of program execution must be known and it works as a static algorithm.
11	2016 [24]	Resource utilization and total execution cost	Autonomic resource provisioning algorithm is proposed based upon the concept of MAPE.	Cloudsim	It does not consider QoS parameters like deadline, priority, elasticity etc.
12	2017 [23]	Elasticity	Proposed algorithm is based upon Live Thresholding (LT) technique for controlling the elasticity.	OpenNebula	It focuses only static threshold based elasticity and does not considered any load balancing techniques.
13	2018 [25]	Makespan time, throughput, energy consumption	Probabilistic based model is used to find the optimal host.	Cloudsim	Proposed algorithm neither monitor the virtual machines continuously for balancing the workload or does not consider other QoS parameters like elasticity, deadline.
14	Our Algo .	Makespan time, task meet to deadline, Task rejection ratio, elasticity, scalability	Developed dynamic scheduling algorithm that balances the workload with elastic resource provisioning and deprovisioning based on the last optimal k-interval.	Cloudsim	If avgofCount become large at the first iteration then it creates more virtual machine than required.

3.4 Problem Formulation

For efficient load balancing schedule all the jobs (tasks) to cloud resource (virtual machine) in such a ways that cloud user can execute their task in minimum time (within the deadline) with maximum resource utilization i.e., cloud user is expected to minimum makespan time and cost while cloud service provider expectation is to utilize the resource maximally. Cloud resource broker received N number of task request $T_1, T_2, T_3 \dots \dots T_N$, that are independent in nature and non priority basis. Every task has task length TL_{Ti} which is expressed in MI (million instructions) and deadline of each task is D_{Ti} . Every task requires p processing speed, q number of cpu, r amount of main memory given in Table 3.2 notations and its descriptions which are used in equations, bandwidth B in MBPS. Cloud resource broker have M number of resources (virtual machine) $R_1, R_2, R_3 \dots \dots R_M$, which are heterogeneous in nature in terms of processing speed, memory, bandwidth etc. Matchmaker try to match each task T_i to virtual machine R_j (value of j is 0 to M-1), if a resource R_j is match with task T_i then value of decision variable V_{TiRj} is 1 otherwise its value is 0. Cloud resource broker contain the matched resource list (ϕ) of tasks available in a schedule (S_i) that may contain all the cloud resources M or less than M.

The aim of objective function is to minimize the makespan time of scheduling algorithm considering deadline as constraint. Our problem is based on single objective function therefore Objective function minimize the total execution time of tasks $T_1, T_2, T_3 \dots \dots T_N$ submitted in a particular schedule S_p as shown in Table 3.2

Objective function:

$$\text{Min } TET_{Ti} = EET_{TiRj \in \Phi_{Ti} S_p} \quad (1)$$

$$ET_{TiRj \in \Phi_{Ti} S_p} = TL_{Ti} / p * q \quad (2)$$

$$TT_{TiRj \in \Phi_{Ti} S_p} = TL_{Ti} / B_{Rj} \quad (3)$$

$$EET_{TiRj \in \Phi_{Ti} S_p} = ET_{TiRj} + TT_{TiRj} \quad (4)$$

Where $|\Phi_{Ti} S_p| \leq M$

Excepted execution time is the sum of execution time and task transfer time. Number of match resource for task T_i may be less than or equal to total number of available resource in cloud environment. When user submits the task with deadline, its task execution time is depend that how much workload is available on that resource.

Table 3.2 Notations and their description

Notations	Description
$T_1, T_2, T_3, \dots, T_n$	Task request 1 to N submitted for schedule.
$R_1, R_2, R_3, \dots, R_n$	Cloud computing resource 1 to M are available for execution the task
Sp	Represent the p th schedule of workload (value of p is 1 to k)
Φ_{TiSp}	Represents the number of matched resources for task T_i for schedule Sp
$TT_{TiRj \in \Phi_{TiSp}}$	Transfer time for task T_i on resource R_j in the matched cloud resource Φ_{TiSp}
$ET_{TiRj \in \Phi_{TiSp}}$	Execution time of task T_i on matched resource R_j
$EET_{TiRj \in \Phi_{TiSp}}$	Excepted execution time of resource R_j to execute the task T_i
TET(Sp)	Total execution time of task in schedule Sp
TL_{Ti}	Length of task T_i in Millions of instructions
p	Processing speed of resource R_j in MIPS
B_{Rj}	Bandwidth of resource R_j
V_{TiRj}	Decision variable is used to represent whether resource R_j is in matched resource list for the task T_i
D_{Ti}	Deadline of task T_i
FT_{Ti}	Finishing time of task T_i
WT_{Ti}	Waiting time of task T_i
TM_{Sp}	Number of task match with resource in schedule Sp
q	Number of cpu's
MIPS	Millions instruction per second
W_{Rj}	Workload available on resource R_j
MET_{Rj}	Maximum execution time of resource R_j
MST	makespan time

The time required to complete the task on available cloud resource is expressed in equation 5 and task will be assigned to that resource that satisfied the condition shown in equation 6.

$$RT_{TiRj} = D_{Ti} - W_{Rj} \quad (5)$$

$$ET_{TiRj} \leq RT_{TiRj} \quad (6)$$

Here W_{Rj} represent the available workload on resource before assigned the Task T_i . Workload on virtual machine (cloud resource) at a particular time can be calculated by equation 7 & 8 [8].

$$L_{V,Mi,T} = K * T L_{Ti(t)} / S_{(V,Mi,t)} \quad (7)$$

Where $K = \{1, 2, 3, \dots, N\}$

$S_{(V,Mi,t)}$ is defined the service rate of virtual machine at time t, that can be expressed in the form of processing power (p) and number of cpu (q) at a time t.

$$\text{Equation can be defined as } S_{(V,Mi,t)} = p * x_{(t)} \quad (8)$$

Where $x = \{1, 2, 3, \dots, q\}$

Load on a virtual machine at a time t can be calculated as the number of task on particular virtual machine is divided by service rate of that virtual machine VM. Total load on all available virtual machine can be calculated as

$$L = \sum_{j=1}^M L_{VMj} \quad (9)$$

After assigning the tasks to virtual machine workload on virtual machine will be increase. We can calculate the total workload by equation 10

$$W_{Rj} = W_{Rj} + ET_{TiRj} \quad (10)$$

Cloud data centre contain M number of resource and each resource contain one or more than one task therefore we have to find the task completion time of all resource after that find the makespan time of resource. Makespan time is the largest time of virtual machine that is required to execute all the tasks/job.

$$MET_{Rj} = \sum EET_{TiRj} \quad (11)$$

$$T_i \in \varphi_{Rj}$$

$$\text{Makespan time (MST)} = \max(MET_{Rj}) \quad (12)$$

Subject to:

$$FT_{Ti} > a_i + ET_{TiRj} \quad \forall i \in N \quad (13)$$

FT_{Ti} is the finishing time of task T_i at virtual machine VM_j .

a_i is task arrival time, if it is known and certain then problem is static otherwise problem is dynamic. ET_{TiRj} is the execution time of task T_i at virtual machine VM_j . Equation 13 indicates that a task can't be started before its time.

$$FT_{i,j} \geq FT_{i-1,j} + ET_{TiRj} \quad (14)$$

Equation 14 represent that a task start to execute at a virtual machine only when previous task has been completed its execution at that particular virtual machine.

Find out the capacity of virtual machine (resource) to know that how many number of virtual machine are overloaded condition and under loaded condition. If any virtual machine is overloaded then transfer the task to under loaded virtual machine, if large number of virtual machine are in under loaded condition then check the condition and reduce the virtual machine.

$$\text{Capacity of virtual machine } C_{Rj} = p * q \quad (15)$$

After that find out the average number of task unable to meet deadline in all interval because on the basis of those task number of virtual machine can be increase or decrease if more number of task are unable to meet deadline then create new virtual machine. Firstly we calculate number of task unable to meet deadline in each interval with the help of count variable and add the value of count after every interval (iteration) with the help of arraylist. Initially declare the variable avgofCount and sumofCount and finally find the average number of task unable to meet deadline after each interval rather than completing the entire interval.

```

    listAvg.add(count);
    declare avgofCount = 1, sumofCount= 0;
    for (Integer index : listAvg)
    {sumofCount = sumofCount + index; }
    avgofCount = sumofCount/listAvg.size();

```

(16)

3.5. Proposed Architecture

The proposed architecture of cloud resource broker for dynamic task scheduling and load balancing with elasticity is represented in Fig. 3.1 which is a modified architecture proposed by Somasundaram et.al. [14]. the brief description of the proposed architecture is given below.

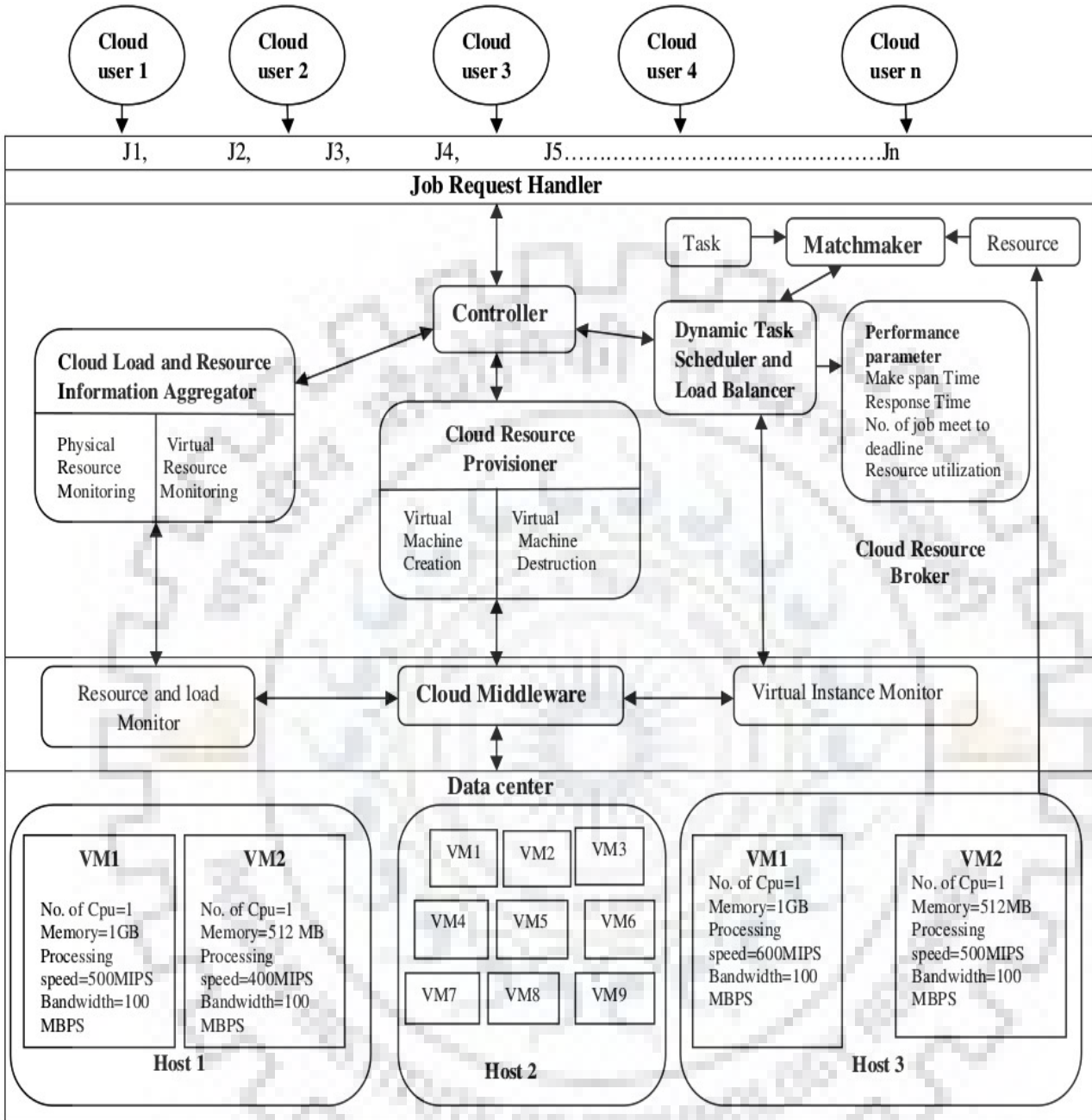


Figure 3.1 Proposed cloud resource broker architecture for load balancing with elasticity

3.5.1 Job Request Handler

Cloud user can access the cloud services (resources) from anywhere and anytime. Cloud contains huge number of datacenters that are distributed in geographical area. When a user submits the request for service, request goes to nearest datacenter. User submit the job J_1, J_2, \dots, J_n through the graphical user interface or web interface with service requirement in terms of quality of service (QoS), hardware, software etc. cloud user requirement vary

dynamically in terms of QoS (throughput, efficiency, user satisfaction, deadline, response time etc.), hardware (number of processor required, memory required, bandwidth required etc.) and software (Mpich-1.2.7, FFTW-3.X etc.). When user submit the job to the cloud, it is accepted by job request handler (gatekeeper), job request handler check the request at verify node to know who is sending the request, a legitimate user or an attacker, if request is coming from legitimate user then it is sent to the controller otherwise it discards the request. Cloud resource cannot be access directly in real environment but can be accessed through RESTFUL web API /SOAP, the main challenge is to allocate the resource to end user because user request is unpredictable and change at run time based on their application.

3.5.2 Controller Node

This is the main component of cloud resource broker. job request handler forward the authentic request to controller node for further processing, controller node send the all application (job) request to matchmaker and also handles the incoming and outgoing request from other the components like job request handler (JRH), cloud resource provisioner (CRP), dynamic task scheduler and load balancer (DTSLB), cloud load and resource information (CLRI) aggregator.

3.5.3 Matchmaker

It contains the information about all the virtual machines (which virtual machine is in idle condition or which one is busy) and user job request. When matchmaker receives the request from the controller it checks the following quality of service parameter:

- Upcoming request is priority based or non priority based
- Upcoming request has a deadline or not

User required parameter (response time, makespan time, resource utilization etc.)

Matchmaker map the user request to available running virtual machine as per application requirement and forward the mapping list to task scheduler and load balancer component.

3.5.4 Dynamic Task Scheduler and Load Balancer

Dynamic task scheduler receives the mapping list from the matchmaker and allocates the task (job) to the virtual machine as per scheduling algorithm. There are lot of task scheduling and load balancing algorithms present in cloud computing that works on parameters like makespan time, response time, throughput, resource utilization, cost etc. Task scheduler schedules the task in such a ways that all tasks are completed in minimum time. Haizea works as resource scheduler in private cloud OpenNebula. Cloud load balancer monitors all the virtual machine and compute the load on all the virtual machine. If any virtual machine is in overloaded

condition then the task of that virtual machine is migrated to other under loaded virtual machine. If overloaded virtual machine does not exist then task is migrated to the balance virtual machine which can execute the task in minimum time. If load at the running virtual machine is more than the threshold value then it invoke the CRP and create new virtual machine. If load at the running virtual machine is below the threshold then it invoke the CRP for destroying the virtual machine.

3.5.5 Cloud Load and Resource Information Aggregator (CLRI)

CLRI is mainly used for aggregating the resource information like memory, processor, bandwidth, load etc from CSP's. It continuously monitors the resource and collects the information about the resource and interacts with Cloud Monitoring and Discovery Service (CMDS) to retrieve the information about cloud resource. CMDS is used to monitor the virtual machine (idle or busy state) and discover the resource information. CLRI trigger a query (request) about the virtual machine, CMDS collect the information from available virtual machine and reply to CLRI.

3.5.6 Resource and Load Monitor

It is mainly used to monitor the private cloud (such as open Nebula, eucalyptus etc.) resource. Ganglia work as external information provider about the host and virtual machine in open Nebula. It collects the information about the resources and passes to the resource and load monitor component.

3.5.7 Cloud Resource Provisioner

The main objective of CRP is creation and deletion of virtual machine as per application (job) requirement. It interacts with cloud middleware (OpenNebula use extended version of D-Grid Resource centre Ruhr as middleware) for provisioning/deprovisioning of the virtual machine.

3.5.8 Virtual Instance Monitor

Monitors the load of each virtual machine continuously and forward it to load balancer that further passes it to controller node. There are two method of monitoring the resources

- *Event based:* When task is removed from a virtual machine or assigned to the virtual machine, after that monitor the status of all virtual machines. We are using event base approach.
- *Time based:* continuously monitor the resource after a particular time interval.

3.6 Dynamic Load balancing algorithm with Elasticity

Dynamic task scheduling and load balancing algorithm with elasticity is developed and discussed in this chapter. The developed algorithm not only minimizes the makespan time but also increase the ratio of task meet to the deadline using elasticity concept. Cloud task scheduling model is shown in Fig. 3.2. We model our scheduling algorithm and analyze the performance of makespan time and number of task meet to deadline as per the parameters given in Table 3.3 to 3.6. To develop this algorithm, we have created N number of task and length of task is generated randomly (range of task between 20000MI to 400000 MI) and created M number of heterogeneous virtual machine, each virtual machines have different processing power in terms of processor speed in MIPS, RAM etc.

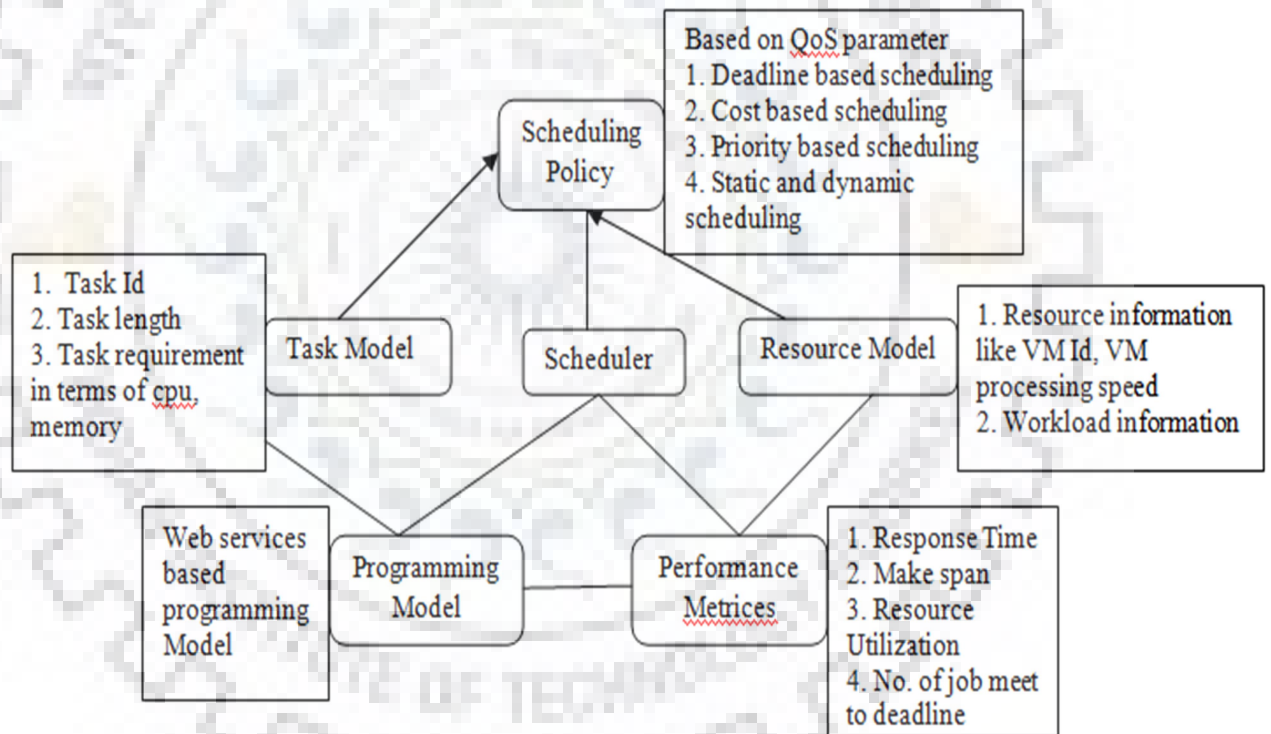


Figure 3.2 Cloud task scheduling model

Algorithm 3.1 sorts the task based on deadline. To test the algorithm we have taken two array, one have task length other have deadline of task; use the sorting algorithm to sort the task. The task which has minimum deadline value that task will be executed first because our aim is to execute more number of tasks before deadline expires.

Algorithm 3.1 for task sorting based on deadline

1. deadline[i] array represent the deadline of i^{th} task
2. Task [i] array represent the i^{th} task
3. Sort the task based on deadline
4. For $i=1$ to deadline_array length
5. For $j=i+1$ to deadline_array length
6. if (deadline[j]<deadline[i])
7. if true then swap the content of deadline[j] with deadline[i]
8. Swap the content (Task length) of Task[j] with Task[i]

Algorithm 3.2 for task scheduling based on deadline

- # EPT[VmSize] and MET[VmSize], min and a=0 are variable
1. Generate M number of virtual machine.
 2. We have to schedule N number of Task based on deadline.
 3. $\forall T_i \in \{T_1, T_2, T_3 \dots \dots T_N\}$
 4. At the starting number of task assigned to resource is null
 5. $\phi_{R_j} \leftarrow \{\text{null}\}$
 6. $\forall R_j \in \{R_1, R_2, R_3 \dots \dots R_M\}$
 7. Find the execution time of task T_i at R_j
 8. $ET_{T_i R_j \in \phi_{T_i S_p}} = EPT[j] = TL_{T_i} / p * q$
 9. End for loop of resource R_j
 10. Assigned the task T_i to resource R_j for minimum execution time
 11. $\forall R_j$ from 0 to M-1
 12. $EPT[j] = EPT[j] + MET[j];$
 13. End of resource R_j loop
 14. $\text{min} = EPT[0];$
 15. $\forall R_j$ from 0 to M-1
 16. if($\text{min} > EPT[j]$)
 17. $a = j;$
 18. $\text{min} = EPT[j]$
 19. if($\text{deadline}[T_i] > \text{min}$)
 20. assigned the task to virtual machine with index a
 21. $\text{vm} = \text{getVmsCreatedList}().\text{get}(a)$
 22. $MET[a] = EPT[a];$
 23. End of for loop of resource R_j ;
 24. End of for loop of tasks T_i ,
 25. Load increase at resource R_j
 26. $W_{R_j} = W_{R_j} + ET_{T_i R_j}$
 28. $\phi_{R_j} = \phi_{R_j} \cup \{T_i\}$

We proposed an efficient task scheduling algorithm that not only allocates the task to virtual machine (cloud resource) but also decrease the makespan time of task as shown in algorithm 3.2. Matchmaker and dynamic task scheduler component participate to allocate the task in effective way to virtual machine (shown in Fig. 3.1). Matchmaker match each task to available running virtual machine and pass the list of match resource to task scheduler that will allocate the task based upon scheduling algorithm. At the starting list of task assigned to resource is null shown in step 5 of algorithm 3.2 i.e. no task is assigned to virtual machine. To assign the task to virtual machine, we calculate the execution time of task at all the running virtual machine and current load at all virtual machine, compare it with deadline. If deadline of task is more than the execution time of task to virtual machine then assign the task to that virtual machine who can execute this task in minimum time. Load at the virtual machine is increased after assigning the task. It can be calculated by sum of previous available load and load of current assigned task.

This process is continued until all tasks has not finished. When dynamic task scheduler try to allocate the task to virtual machine, there are some tasks that do not fulfill the condition of deadline i.e., some tasks have execution time more than their deadline value therefore these type of tasks are unable to meet their deadline. These types of tasks are discarded in cloud environment. If large number of task is discarded then cloud service performance will degrade and user satisfaction level will also decrease. Therefore we proposed an algorithm that balances the load at all virtual machine and increase the ratio of task to meet with deadline using the elasticity concept (provisioning and deprovisioning of resource at run time) as shown in algorithm 3.3. Flow chart of algorithm 3.3 (shown in Fig. 3.3) represents the threshold condition and virtual machine overloaded or underloaded condition briefly in well specified way. We calculated the number of task unable to meet deadline in each interval after that find the average of number of task that are unable to meet deadline in last k interval using the equation 16 and applied the condition that if more than 25% average number of task are rejected then increase the new virtual machine by 20%. If average rejected task is more than 10% then increase the virtual machine 10%. If average rejected task is less than or equal to 10% then there is no need to increase the virtual machine at current time because sometimes instantaneous small peak load (less than 10%) can come at the virtual machine. We start to increase the virtual machine (up to 5 % or 10 %) which leads to unnecessary overhead because

after sometime either I have to reduce the virtual machine or some virtual machine will be in under loaded condition.

Algorithm 3.3 for Load Balancing decision with scalability

UVM =under loaded VM, BVM =balanced VM, OVM =overloaded VM
count represent number of task unable to meet deadline
avgofCount represent avg. number of task unable to meet deadline, NumberofTask=N

1. $\forall T_i \in \{T_1, T_2, T_3 \dots \dots T_N\}$
2. $\forall R_j \in \{R_1, R_2, R_3 \dots \dots R_M\}$
// Allocate the task T_i to that resource R_j who can execute in minimum time and find out the min value as shown in step 7 in algorithm 2
3. if (deadline[T_i] > min) then
4. Assigned the task to virtual machine with index a
5. else
6. count++;
7. find avgofCount after each interval
8. if (avgofCount > N * .25) then
9. increase R_j by 20%.
10. else if (avgofCount > N * .1) then
11. increase R_j by 10%.
12. else (avgofCount \leq N * .1)
13. no need to increase the virtual machine at current time
14. end of for loop;
15. end of for loop
16. $\forall R_j \forall T_i$, Calculate the load at each R_j
17. Calculate the capacity (C_{R_j}) of each R_j
18. Find the number of OVM, UVM, and BVM
19. $OVM \geq .9 * C_{R_j}$, $UVM < .2 * C_{R_j}$
20. Sort the virtual machine in OVM decreasing order
21. Sort the virtual machine in UVM in ascending order
22. Assigned the task of OVM to UVM and calculate the task transfer time.
23. Calculate the average number of UVM
24. if (avgUVM > .25 * R_j) then
25. reduce R_j by 20%.
26. else if (avgUVM > .1 * R_j) then
27. reduce R_j by 10%
28. else (avgUVM \leq .1 * R_j)
29. no need to decrease the virtual machine at current time
30. end of for loop;
31. end of loop

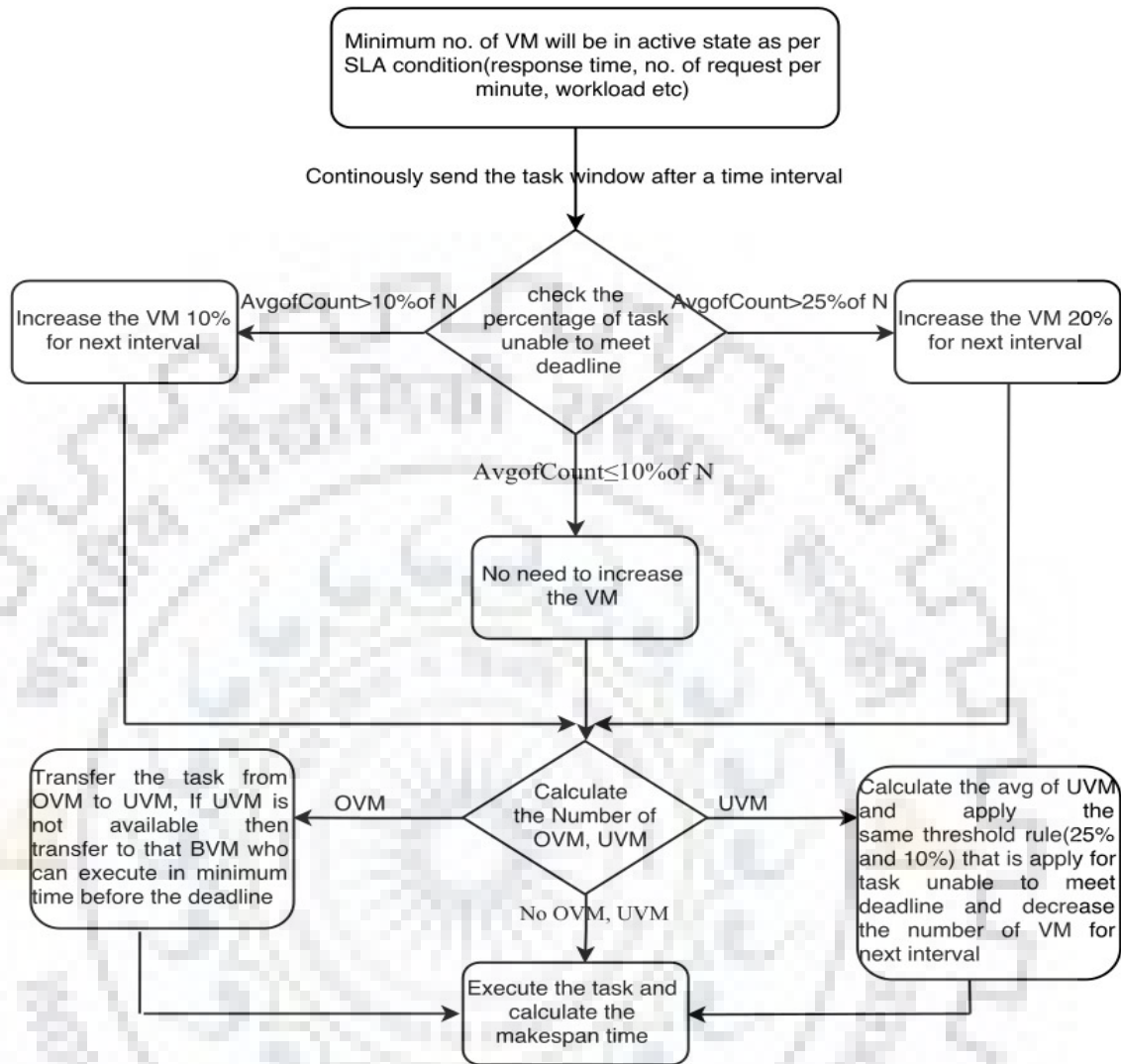


Figure 3.3 Flow chart to determine the overloaded and underloaded virtual machines

We calculate the parameter avgofCount that is based upon last k interval and increase the virtual machine based upon the defined threshold value (25 % or 10%) for next upcoming interval. Threshold value is defined at the time of service level agreement (SLA) based upon the parameter like upcoming number of request per minutes, workload, number of task unable to meet deadline etc. SLA is a contract between the service provider and users in which the Quality of Service (QoS) is defined. We have studied threshold based papers [14, 20-24] related to our work for cloud environment and set the value of threshold based on the historical (Past) data. For this we have carried out various experiments with different value of threshold to find out the optimum results. Finally we have selected the threshold value that gives the optimum result. We find the overloaded, under loaded and balanced virtual machine after

assigning the task to virtual machine. If any virtual machine is overloaded or under loaded condition then Sort the OVM and UVM in decreasing and increasing order and assign the task. The algorithm checks the condition of under loaded virtual machine if average of under loaded virtual machine is greater than the 25% of all available virtual machine then decrease the virtual machine by 20% for next interval. If it is more than by 10% then decrease the virtual machine 10 % for next interval.

3.7 Analysis and Comparison of Results

The developed load balancing algorithm minimizes the makespan time and increase the ratio of task to meet deadline using the cloudsim platform. To test the performance of the algorithm, we choose cloudsim simulator [28] for experimental purpose (architecture is shown in Fig. 3.4). Each virtual machine (VM) has their parameter like Id, MIPS, number of CPU etc as shown in Table 3.3. Further we have generated cloudlet (Task) with their parameter like TaskId, length, filesize etc as shown in Table 3.4. Cloud resource broker submit bounded task to specific virtual machine (depends upon the policy) with the help of broker. `bindCloudletToVm(cloudlet.getId(),vm.getId())` method. In this chapter, we have calculated and analyzed makespan time and task unable to meet deadline with the help of cloudsim simulator.

3.7.1 Makespan Time Calculations

Let’s consider the example; consider 10 virtual machine of different processing power and bandwidth of each virtual machine is 1000 MBPS, number of cpu for each virtual machine is one as shown in Table 3.3.

Figure 3.4 Basic architecture of cloudsim

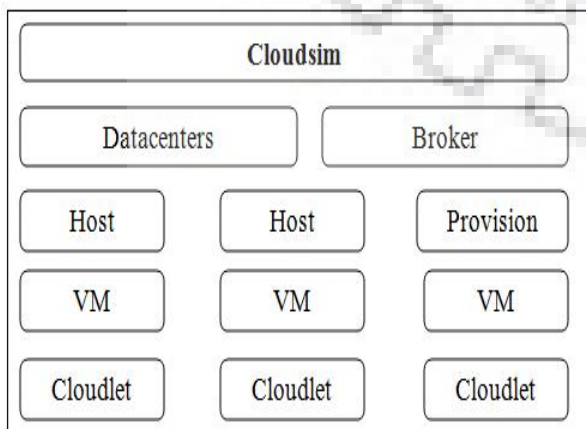


Table 3.3 VM Properties

VM Id	VM MIPS	VM Image size	Memory	No. Of cpu	Hypervisor
0	500	1000	256	1	Xen
1	520	1000	512	1	Xen
2	540	1000	512	1	Xen
3	560	1000	256	1	Xen
4	580	1000	256	1	Xen
5	600	1000	512	1	Xen
6	620	1000	256	1	Xen
7	640	1000	512	1	Xen
8	660	1000	215	1	Xen
9	680	1000	512	1	Xen

Table 3.4 Task Properties

Task Id	Length	File Size	Output Size	No. of cpu required
0	130795	300	300	1
1	224339	300	300	1
2	48212	300	300	1
3	330838	300	300	1
4	269322	300	300	1
5	65245	300	300	1
6	383678	300	300	1
7	263607	300	300	1
8	137286	300	300	1
9	328394	300	300	1

The range of task is 10 to 100 and length of task is varying from 20000MI to 400000 MI as shown in Table 3.4. The numerical calculation to determine the performance of the algorithm based upon the selecting the application (tasks) and resources instances and if application is memory-intensive that needed high-memory VM instances for database operation tasks.

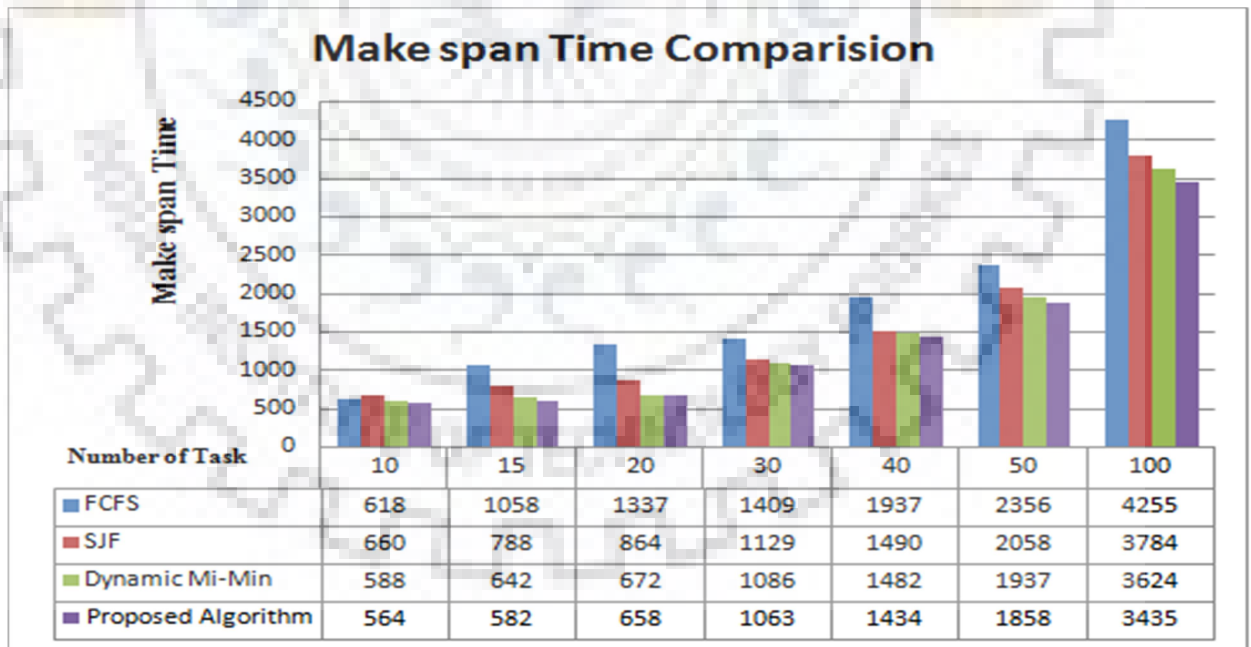


Figure 3.5 Makespan time comparisons between proposed algorithms with FCFS, SJF, dynamic min-min

Therefore the proposed algorithm selects the length of task in a range (20000MI to 400000MI) and creates the virtual machine instance such that they can process the task. If the range of task

is increased or decreased then results (makespan time, number of task unable to meet deadline and elasticity) are affected i.e. because same virtual machine instance either process the task early (underloaded condition) or delay (overloaded condition). Simulation have been run for more than 200 times on different number of task with random length and results are found using the space shared policy [28] in cloudsim. Results of Table 3.3 and Table 3.4 are graphically represented in Fig. 3.5, where x-axis represent the number of task and y-axis represent the makespan time of task in second. The developed scheduling algorithm allocates the task to all the virtual machines. The comparisons of makespan time with other popular scheduling algorithms like FCFS, SJF and dynamic min-min algorithm is shown in Fig. 3.5. Computational results shows that proposed algorithm reduces the makespan time of task compared to existing algorithm (FCFS, SJF and dynamic min-min algorithm) as shown in Fig. 3.5

3.7.2 Number of Task Meets to Deadline

To calculate the number of task meet to deadline we have created a window of 15 tasks with random length which is sent continuously to cloud resource broker after every 5 second interval. Different configuration of 10 virtual is created to process the upcoming task and deadline of task is created randomly (example shown in Table 3.5). We computed the performance metric for analyzing the number of task completed on or before the deadline specified by the user. Our proposed task scheduling algorithm considers the deadline as an important factor and find out the best resource for the task so that task can be executed before the deadline expired. Simulation has been run at different task with corresponding deadline and calculated results shows that proposed algorithm completes more tasks before deadline compare to FCFS, SJF and dynamic min-min as shown in Fig. 3.6. The proposed algorithm shows approximate 90% of the task meeting with deadline compare to 78% of FCFS, 81% of SJF and 76% of dynamic min-min algorithm. Further we have tested our algorithm increase the number of task (15 to 20) with random length and deadline of task as shown in Table 3.6. Consider all the virtual machine has different processing capacity. The simulation has been run when cloud resource broker send 20 tasks at every 5 second interval. The calculated results are shown in Fig. 3.7 which indicates that the proposed algorithm shows better results compare to FCFS, SJF and dynamic min-min algorithm in terms of number of tasks meet to deadline.

Table 3.5 Task with deadline

Task Id	Task length	Deadline of task	VM MIPS
0	319775	500	500
1	43753	720	520
2	50077	780	540
3	276496	685	560
4	133858	745	580
5	45215	620	600
6	132290	490	620
7	367951	450	640
8	291873	700	660
9	385050	660	680
10	93020	870	
11	42947	850	
12	30362	590	
13	348858	400	
14	69942	560	

Table 3.6 Task with deadline

Task Id	Task length	Deadline of task	VM MIPS
0	381771	785	500
1	392397	745	520
2	339760	920	540
3	323461	690	560
4	272332	750	580
5	325970	700	600
6	333911	660	620
7	182561	2000	640
8	396156	1200	660
9	215744	1300	680
10	253197	1250	
11	334013	450	
12	344630	400	
13	222784	550	
14	253745	1000	
15	153285	800	
16	205633	600	
17	98049	1050	
18	107218	300	
19	147281	1150	

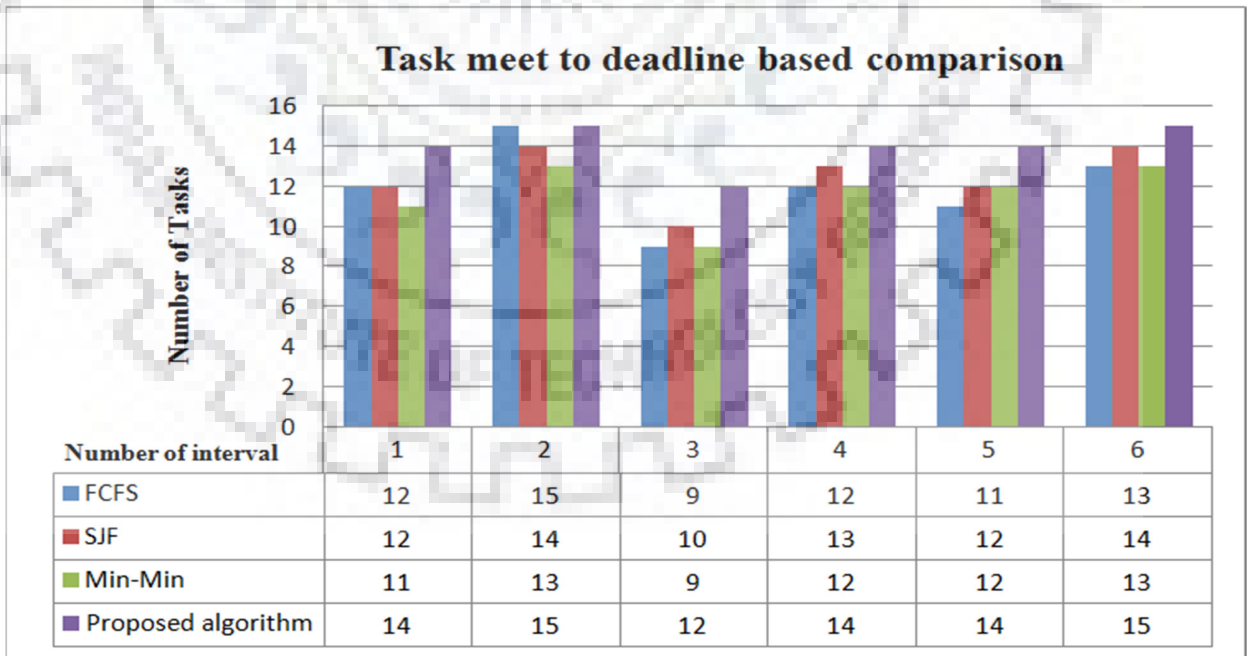


Figure 3.6 Task acceptance ratio comparison between proposed algorithm with FCFS, SJF and dynamic min-min

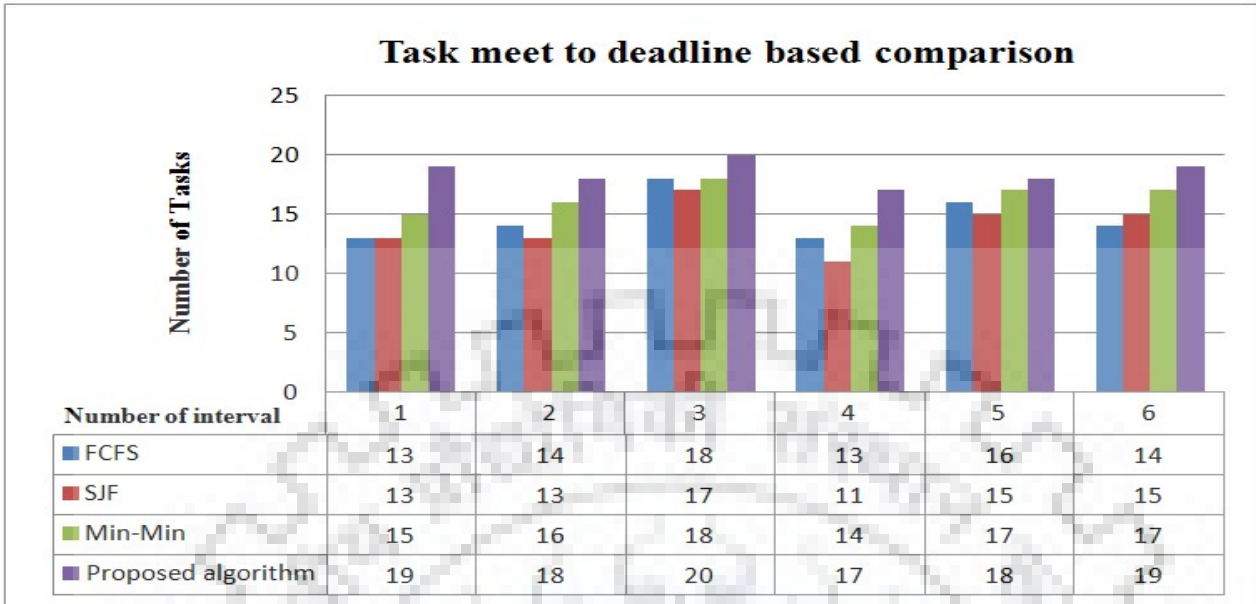


Figure 3.7 Task acceptance ratio comparisons proposed algorithm with FCFS, SJF, min-min

3.7.3 Provisioning and Deprovisioning (elasticity) of Resources

Elasticity is one of the important factors in cloud environment where user use the resources on the basis of pay per use and don't want to pay for resource which is not used by user. Elasticity is the ability to fit the resource needed to cope with dynamically upcoming load at the virtual machine. If load is increased at the resources then controller pass the instruction to cloud resource provisioner to increase resource in scale out fashion. When demand wanes, resource provisioners start to shrink back and remove unneeded resources. Let's consider the example in that we send continuously 15 task of random length after 5 second interval (time interval will be large in real environment) and 10 virtual machine are ready to process the task at the initial phase. Deadline of task is given randomly. The proposed algorithm is tested with 15 number of task and the results show that 14 number of task meet deadline in first iteration, only one task is rejected (average is 1) i.e. less than 10 % of the total task. It shows that there is no need to increase the virtual machine.

In next iteration three tasks have been rejected because of length of task is randomly selected (average become two) i.e. more than 10% task is rejected so cloud resource provisioner add 10% more virtual machine to process the upcoming tasks as per reported algorithm 3.3. In the next iteration two tasks are rejected and calculated average task rejection is more than 10 % so one more virtual machine is added. By randomly generated task, virtual machine is increases

from 10 to 14 based upon the average number of task rejection but after some interval if workload starts to decrease and some of the virtual machine goes to underloaded condition. The number of virtual machine is started to decrease as per underloaded threshold condition given in algorithm. After 10 intervals virtual machine is reduces from 14 to 12 for executing the 15 random length tasks as shown in Fig. 3.8.

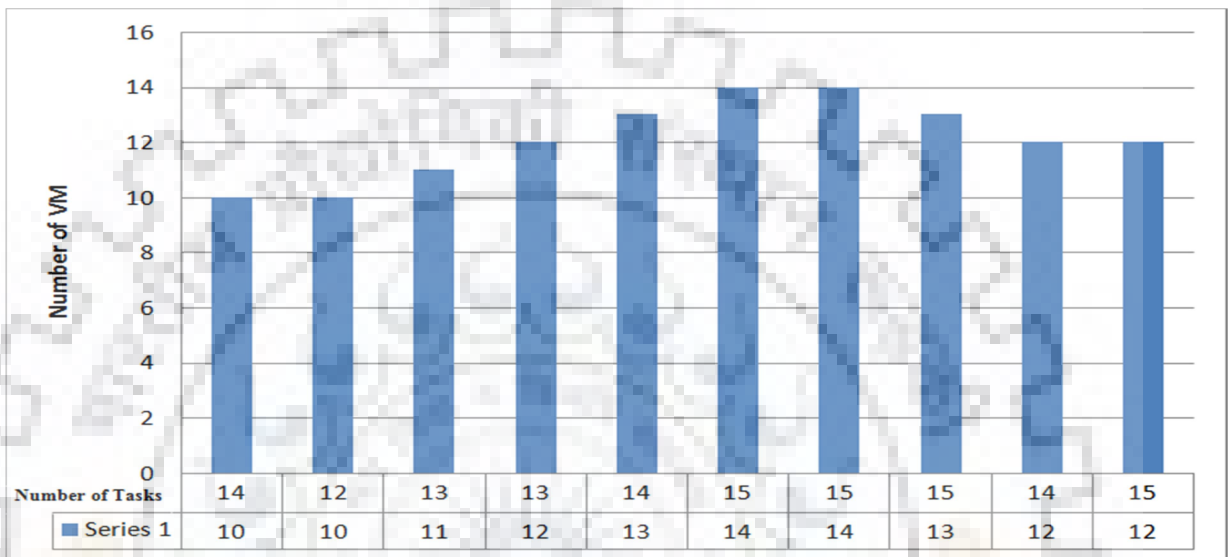


Figure 3.8 Provisioning and deprovisioning of cloud resource based on upcoming tasks

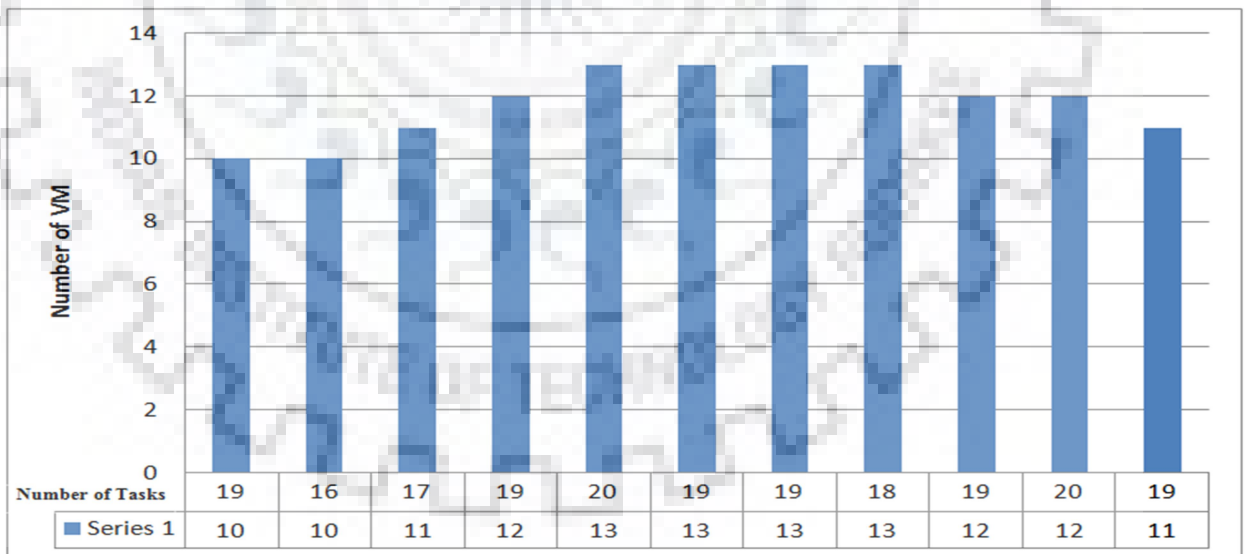


Figure 3.9 Provisioning and deprovisioning of cloud resource based on upcoming tasks

The algorithm is further tested with sending randomly window of 20 tasks. At the starting 10 virtual machine are ready to process the tasks. Simulation has been run for 11 intervals and it is observed that when the workload at the virtual machine is increases then the cloud resource

provisioner create more virtual machines depending upon the condition given in algorithm 3.3. If workload is decreased and virtual machine comes in under loaded condition then cloud resource provisioner decreases the virtual machine instances. Fig. 3.9 shows that virtual machine is increased from 10 to 13 at the time of high workload and decreased from 13 to 11 at the time of low work load after the end of 11th iteration. Most of the existing approaches reported in the literature [20, 23-24] take only single threshold limit for elasticity but we have considered 3 threshold limits ($>25\%$, >10 , $\leq 10\%$) that gives better results than previous approaches. We have considered the average task unable to meet deadline in last k interval, while previous approach decide the elasticity only the basis of last interval. In the present chapter, we have set the values for the threshold statically, however some authors set the threshold value dynamically [20, 23-24].

3.7.4 Scalability

Scalability is the ability of the system to accommodate more loads by adding resource either horizontally (scale-out) or vertically (scale-up). Vertical scalability means that we are adding the additional hardware. This type of approach is apply in web server, database servers etc but limitation of vertical scaling is that hardware should be specific and how much memory, processor and disk a single server support. Therefore horizontal scaling is better than vertical scaling that add the number of node in horizontal scaling. We have tested our algorithm for horizontal scalability with number of tasks (10 to 30) with fixed length (200000 MI) and 10 virtual machines is process to execute the task at the starting of the simulation. On the basis of task rejection cloud resource provisioner increase the virtual machine instance and cloud environment provides the better scalability.

The simulation has been run approximately 10 times for 10 tasks at the starting and it are observed that all the tasks are easily executed by 10 virtual machines. If all the tasks executed by 10 virtual machine in first iteration then results are same for next all the iteration, because all tasks have the same length. Now simulation have been run for 15 tasks and it is observed that only 11 tasks meet to deadline i.e. approximate 26% tasks has been rejected. So, cloud resource provisioner added the 20 % virtual machine and it become 10 to 12 virtual machine. For the next iteration 20% average task has been rejected and virtual machine increase 12 to 13. This process is continuing until all the tasks have not been accepted as shown in Fig. 3.10. Further the algorithm has been tested by increasing the tasks from 15 to 20, 25, 30 and same procedure is repeated. It is observed that approximate 16 virtual machine is needed to execute

the 30 tasks of same length. Fig. 3.10 represents that number of tasks we have submitted and number of task meet with deadline.

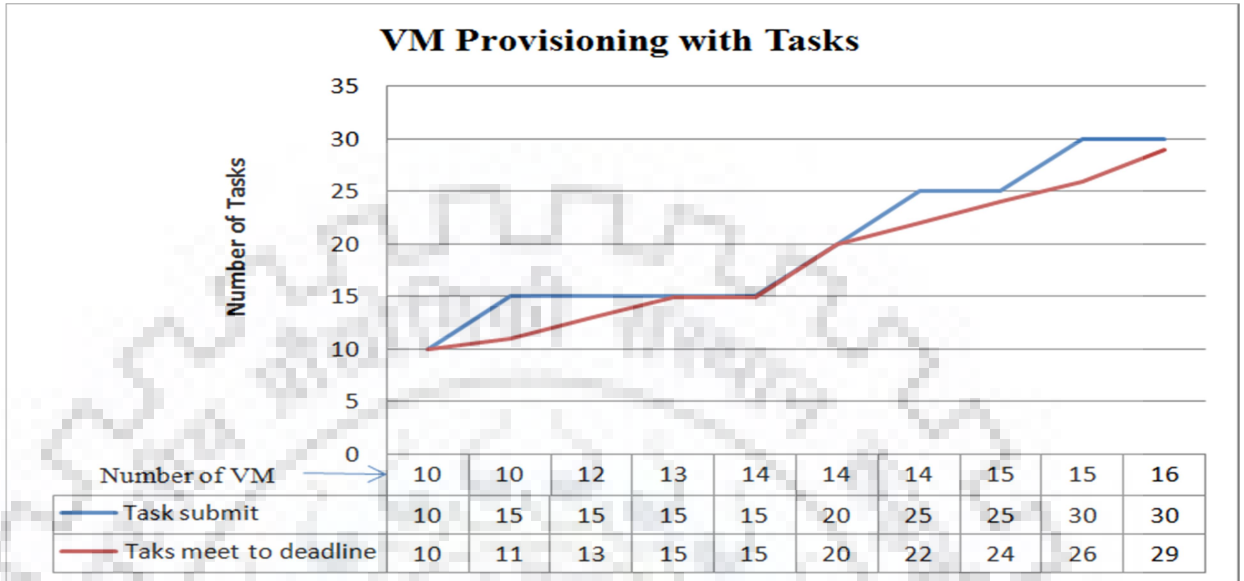


Figure 3.10 Scale-out of cloud resource based on upcoming task

Y axis represent the number of task, x axis represent the number of virtual machine required to execute those task by using the proposed algorithm 3.3. The overhead of the proposed algorithm is calculated based on the value of k which should not be more than last15 intervals. We have analyzed the performance of the algorithm till last 30 intervals, as after 15 intervals its performance start to degrade. We have compared overhead of developed algorithm with the other algorithms available in literature like min-min, SJF and FCFS. It is observed that proposed algorithm perform better than the existing algorithm because at the time of scheduling these algorithm unable to distribute the task efficiently to existing virtual machine due to which more tasks are rejected (unable to meet deadline). It is because more number of virtual machine is in overloaded and underloaded conditions hence SLA violation is increased.

3.8. Summary

In this chapter, we have modified the architecture of cloud resource broker and developed an efficient dynamic algorithm for task scheduling, which is based on the last optimal k-interval that not only minimizes the makespan time of tasks but also increase the ratio of tasks to meet the deadline and fulfill the objective of elasticity in cloud environment. The algorithm has been tested at variable number of task to achieve better scalability. Further, the algorithm has also been tested with modified architecture of cloud resource broker and a test scenario has been

created in cloudsim. It has been observed that the developed algorithm is helpful for making intelligent scheduling decision for increasing (scale-out) or decreasing (scale in) the virtual machine instance based on the upcoming workload request/application. The main idea of our threshold-based dynamic resource allocation scheme is to monitor and predict the resources based on the needs of the cloud applications. The performance of the reported algorithm starts to degrade when the value of last interval 'k' is more than 15. Experimental results show that under all possible conditions, the algorithm improves the makespan time and also number of tasks to meets the deadline. The results have proved that the developed algorithm provide better elasticity and reduce the rejection ratio of task in comparison to the existing conventional algorithms like FCFS, SJF and min-min as shown in Figs. 3.5 to 3.10. This proposed model can be extended to improve other QoS parameters like execution cost, energy consumption and reliability for ensuring the high-priority requests.

3.9 References

- [1] K. Hwang, Y. Shi and X. Bai, "Scale-Out vs. Scale-Up Techniques for Cloud Performance and Productivity," in *6th International Conference on Cloud Computing Technology and Science (CloudCom)*, pp. 763-768, Singapore, Dec. 2014.
- [2] A. Suresh and P. Vijayakarthick, "Improving scheduling of backfill algorithms using balanced spiral method for cloud metascheduler," in *International Conference on Recent Trends in Information Technology*, pp. 624- 627, Chennai, India, 2011,
- [3] K. Dubey, M. Kumar and M. Chandra, "A Priority Based Job Scheduling Algorithm Using IBA and EASY Algorithm for Cloud Metascheduler," in *International Conference on Advances in Computer Engineering and Applications*, pp. 66-70, Ghaziabad, India, 2015.
- [4] B. Shaoo, D. Kumar and S. K. Jena, "Analyzing the Impact of Heterogeneity with Greedy Resource Allocation Algorithms for Dynamic Load Balancing in Heterogeneous Distributed Computing System," *International Journal of Computer Applications*, vol. 62, no. 19, pp. 25-34, Jan. 2013.
- [5] W. Li, Wenzheng, and H. Shi, "Dynamic load balancing algorithm based on FCFS," in *Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, pp. 1528-1531, Taiwan, Dec. 2009.

- [6] R.K. Mondal, E. Nandi, and D. Sarddar, "Load Balancing Scheduling with Shortest Load First" *International Journal of Computer Science and Information Technology Research*, Vol. 3, no. 4, pp. 162-166 , Dec. 2015.
- [7] H. Chen, F. Wang, N. Helian and G. Akanmu ,“User Priority Guided Min-Min Scheduling Algorithm For Cloud Computing,” in *national conference on Parallel Computing Technologies*, pp. 1-8, Bangalore, India, Oct. 2013.
- [8] D. Babu and P. Venkata, “Honey bee behavior inspired load balancing of tasks in cloud computing environments,” *Applied Soft Computing*, vol.13, no. 5, pp. 2292–2303, May 2013.
- [9] F. Ramezani and F. K. hussain, “Task-based System Load Balancing in cloud computing using Particle Swarm Optimization,” *International Journal of Parallel Programming*, vol. 42, no. 5, pp. 739-754, Oct. 2013.
- [10] E.Pacini, C. Mateos and C. G. Garino, “Balancing throughput and response time in online scientific Clouds via Ant Colony Optimization (SP2013/2013/00006)”, *Advances in Engineering Software*, vol. 84, pp. 31-47, June 2015.
- [11] J. T. Tsai, J. C. Fang and J. H. Chou, “Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm,” *Computer Operation Research*, vol. 40, no. 12, pp.3045-3055, Dec. 2013.
- [12] E.D. Coninck, T. Verbelen, B. Vankeirsbilck, S. Bohez and P. Simoens, "Dynamic auto-scaling and scheduling of deadline constrained service workloads on IaaS clouds," *Journal of Systems and Software*, vol. 118, pp. 101-114, 2016.
- [13] R.K. Naha and M. Othman, "Brokering and load-balancing mechanism in the cloud Revisited," *IETE Technical Review*, vol. 31, no. 4, pp. 271-276, 2014.
- [14] T. Somasundaram, K. Govindarajan, M. Rajagopalan and S.M. Rao, “A broker based architecture for adaptive load balancing and elastic resource provisioning and deprovisioning in multi-tenant based cloud environments,” in *International conference at Advances in Computing*, pp. 561–573, New Delhi, India, 2013.
- [15] X. Fu and Z. Chen, "Virtual machine selection and placement for dynamic consolidation in Cloud computing environment," *Frontiers of Computer Science*, vol. 9, no. 2, pp. 322-330, 2015.
- [16] W. Kong, Y. Lei and J. Ma, "Virtual machine resource scheduling algorithm for cloud computing based on auction mechanism," *Optik-International Journal for Light and Electron Optics*, vol. 127, no. 12, pp. 5099-5104, 2016.

- [17] S. Abrishami and M. Naghibzadeh, "Deadline-constrained workflow scheduling in software as a service cloud," *Scientia Iranica*, vol. 19, no. 3, pp. 680-689, 2012.
- [18] S.C. Nayak and C. Tripathy, "Deadline Sensitive Lease Scheduling in Cloud Computing Environment Using AHP," *Journal of King Saud University-Computer and Information Sciences*, vol. 30, no. 2, pp. 152-163, April 2018.
- [19] M. Malawski, G. Juve, E. Deelman and J. Nabrzyski, "Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," in *International Conference on High Performance Computing, Networking, Storage and Analysis*, USA, 2012.
- [20] Lorido-Botran T, Miguel-Alonso J, Lozano J.A, "Comparison of Auto-scaling Techniques for Cloud Environments," 2013.
- [21] X. Li and Z. Cai, "Elastic resource provisioning for cloud workflow applications," in *IEEE Transactions on Automation Science and Engineering*, vol. 14, no. 2, pp. 1195-1210, 2017.
- [22] R.N. Calheiros, R. Ranjan and R. Buyya, "Virtual machine provisioning based on analytical performance and QoS in cloud computing environments," in *International Conference on Parallel processing (ICPP)*, pp. 295-304, Taiwan, Sep. 2011.
- [23] R.D.R. Righi, V.F. Rodrigues, G.Rostirolla, C.A.D. Costa, E. Roloff and P.O.A. Navaux, "A lightweight plug-and-play elasticity service for self-organizing resource provisioning on parallel applications," *Future Generation Computer Systems*, vol. 78, pp. 176-190, 2017.
- [24] M. G. Arani, S. Jabbehdari and M. A. Pourmina, "An autonomic approach for resource provisioning of cloud services," *Cluster Computing*, vol. 19, no. 3, pp. 1017-1036, 2017.
- [25] S. Chhabra, and A. K. Singh, "A Probabilistic Model for Finding an Optimal Host Framework and Load Distribution in Cloud Environment," in *Procedia Computer Science*, vol. 125, pp. 683-690, 2018.
- [26] S. Bharti and K. K. Pattanaik, "Task requirement aware pre-processing and Scheduling for IoT sensory environments," *Ad Hoc Networks*, vol. 50, pp. 102-114, 2016.
- [27] S. Javanmardi, M. Shojafar, D. Amendola, N. Cordeschi, H. Liu and A. Abraham, "Hybrid Job Scheduling Algorithm for Cloud Computing Environment," in *Advances in Intelligent Systems and Computing*, vol 303, pp. 43-52, 2014.
- [28] H.S. Sindhu, "Comparative analysis of scheduling algorithms of Cloudsim in cloud computing," *International Journal of Computer Applications*, vol. 97, no. 16, July 2014.

CHAPTER-4

DYNAMIC TRANSFER BASED MODIFIED BINARY PSO FOR SCHEDULING THE TASKS

4.1. Concept of Task Scheduling and Binary PSO

Cloud computing provides on demand resources for computation and data intensive types of applications [1]. Number of users and heterogeneity nature of resources (different core, memory etc) as well as application (computation intensive, data intensive or normal application) brings challenges for scheduling of applications in cloud environment. Task scheduling is a nondeterministic polynomial time complete (NPC) problem in the field of computer science. There is no algorithm exists to solve the NP Complete problem in polynomial time. It is preferable to find suboptimal solution, but in short period of time. Calculating the all possible task-resource mapping (scheduling) and selecting the optimal mapping is not feasible in cloud environment [2]. Therefore researcher are using the soft computing techniques to solve the NP Complete problems like metaheuristic algorithms (artificial honey bee [3-6], particle swarm optimization (PSO) [7], ant colony optimization (ACO)[8], Evolutionary algorithms (differential evolution [9-10], genetic algorithm (GA) [11], Gravitational Search Algorithm (GSA) [12]). The merits of PSO over GA include easier implementation and less number of variable parameters has been involved in the optimization process. PSO achieves a faster convergence rate and global optimum solution within minimal time as compared to ACO and GA. Therefore researchers choose the PSO algorithm for job scheduling in cloud and grid environment.

PSO is a population-based search algorithm that was developed for continuous optimization problems in 1995 by Kennedy and Eberhart [13]. Each particle contains the position and velocity in PSO algorithm. Velocity of each particle is updated in each time step to find out the two best positions Pbest and Gbest. Velocity is updated by equation 1

$$V_i^{k+1}(j) = w * V_i^k(j) + c_1 r_1 (Pbest_i^k(j) - Z_i^k(j)) + c_2 r_2 (Gbest_i^k(j) - Z_i^k(j)) \quad (1)$$

Where $k+1$ represent the current instruction, $Z_i^k(j)$ represent the j^{th} element of i^{th} particle in k^{th} iteration of the PSO algorithm and $V_i^k(j)$ represent the j^{th} element of velocity matrix of i^{th} particle in k^{th} iteration. c_1 and c_2 are cognitive learning factor and social interaction coefficient.

Cognitive factor produce the self confidence in the particle and control the influence of Pbest (local search) on the search process. Social interaction factor c_2 is used for Gbest (global search). Initially c_1 factor should be large so that particle can move with own confidence and c_2 should be low, as search progress c_1 factor should be decrease and c_2 should be increase.

$$c_1(k)=2.5-2*(k/MaxIteration) \quad (2)$$

$$c_2(k)=0.5+2*(k/MaxIteration) \quad (3)$$

r_1 and r_2 are random number in the range $[0,1]$. Inertia weight factor w is used to control the momentum of velocity and acceleration i.e. maintains the balance between exploration and exploitation. We start the value to w with large value (.9) which decrease when iteration increase over the time to small value (.4) approximately. Value of w is decrease linearly by given equation 4.

$$w = w_{max} - \{(w_{max} - w_{min}) * (Itr_k / Itr_{max})\} \quad (4)$$

Velocity $V_i^{k+1}(j)$ is bounded by threshold limit shown in equation 5.

$$V_i^{k+1}(j) = \begin{cases} V_{max}, & \text{if } (V_i^{k+1}(j)) > V_{max} \\ -V_{max}, & \text{if } (V_i^{k+1}(j)) < -V_{max} \end{cases} \quad (5)$$

After that particle position is updated using velocity equation.

$$Z_i^{k+1}(j) = Z_i^k(j) + V_i^{k+1}(j) \quad (6)$$

Many real-world optimization problems are typically discrete problems like task scheduling, 0-1 knapsack problem, traveling salesman problem, airline scheduling problem etc. that can be solved by Binary PSO. Kennedy and Eberhart [14] proposed the binary version of the algorithm in 1997 for discrete optimization problems. Each particle contain a matrix of size $m*n$ where m represent the number of virtual machine and n represents the number of upcoming tasks for service. Elements of each particle matrix will be either 0 or 1 where 1 means task is selected or 0 means task is not selected. Elements of particle can be change from 0 to 1 and vice versa. Range of velocity matrix for each particle is $[-V_{max}, V_{max}]$. Velocities are defined in terms of probabilities that a bit will be in state or the other state. Position matrix and velocity matrix of each particle is generated randomly at the starting of the algorithm. Then algorithm try to find out some optimal or suboptimal solution based upon the fitness function after some iteration because fitness value of every particle is calculated using the fitness function (objective function) so that fitness function to be improved. Firstly we transform the

value of velocity from continuous space to binary space using the sigmoid transfer function shown in equation 7 that perform the calculation based upon the value of current velocity (equation 1 & 5) and produce the value always less than 1. Finally we find out the modify position of the particle using the equation 7, where $Rand(i, j)$ is function that produces the value between 0 to 1.

$$Z_i^{k+1}(j) = \begin{cases} 1 & \text{if Sigmoid}(V_i^{k+1}(j)) > Rand(i, j) \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

$$\text{Where Sigmoid}(V_i^{k+1}(j)) = \frac{1}{1 + e^{-V_i^{k+1}(j)}} \quad (8)$$

The major difference between BPSO and typical PSO is that the relevant variables (velocities and positions of the particle) are defined in terms of the change in probabilities and the particles are formed by integer in $\{0, 1\}$.

4.2 Contribution

Simple binary particle swarm optimization (BPSO) does not provide satisfactory solution due to inappropriate behavior of transfer function. To overcome this problem, we have modified transfer function that provides the exploration and exploitation capability in better way to solve the problem of scheduling in the field of cloud computing.

Specific contribution of this chapter:

- We have proposed a dynamic transfer function (TF_P-BPSO) for BPSO that provides exploration (high probability of flipping the bits) at the starting phase of the simulation.
- At the middle phase it move from exploration to exploitation due to less probability of flipping of bits. Proposed transfer function provides the exploitation in the last phase.
- Results shows that proposed dynamic transfer function maintains the good balance between exploration and exploitation and improve the QoS parameters (execution time, makespan time, convergence rate, throughput) compare to existing algorithm.

4.3 Related Work and Research Gap

There are lots of traditional algorithm [15-20] have been proposed in last decade for task scheduling in cloud environment. Suresh and Vijayakarhick [15] propose a technique of balanced spiral (BS) method to improve the processing time of jobs. To achieve better quality of service with high resource utilization an algorithm IBA with EASY [16] has been proposed

for scheduling. Chen et al.[17] proposes an improved min-min load balancing algorithm (LBIMM) to reduce the makespan time of tasks and increase the utilization ratio of cloud resources while considering priority as quality of service (QoS) parameter. Mao et al.[18] proposed Max-Min scheduling algorithm to reduce the response time and improve the resource utilization ratio of the cloud resources. Mohit Kumar and S.C.Sharma proposed an algorithm to reduce the makespan time of tasks and improved the utilization ratio of cloud resource considering the priority of task as quality of service parameter [19]. P.Samal and P.Mishra proposed round robin technique considering the parameter response time and resource utilization to solve the problem of load balancing in cloud environment [20]. The selection process of non-PSO based resource identifier stops after a pre-defined number of iterations. But in PSO, set a fixed number of iterations and particle rejects the new solution if it is poorer than the current solution.

Modified PSO has been reported to optimize the parameters like makespan time, execution cost, energy consumption etc while considering deadline and budget as constraint [21-24, 29] but these algorithms is used to solve the continuous optimization problem. Binary PSO and its variant has been used to solve the discrete optimization problems but unable to provide satisfactory results due to inappropriate transfer function. Therefore Islam et al.[25] proposed a new time varying transfer function for balancing the better exploration and exploitation and provide the satisfactory solution. The main aim of proposed time varying transfer function is to generate the value of velocity nearby 0.5 at the early stage of the simulation run so that BPSO can provide stronger exploration due to high probability of flipping the bits. BPSO should start to shift exploration to exploitation in the intermediate stage and in the last stage transfer function provides the stronger exploitation due to low probability of flipping all the bits i.e. it generate the value of velocity either nearby 1 or nearby 0. K. Suresh and N. Kumarappan proposed a hybrid improve BPSO algorithm to reduce the loss of load probability and minimize the annual supply reserve ratio deviation for power system [26]. Chen et al.[27] proposed the improve PSO algorithm to solve the resource-constrained scheduling problem to minimize the makespan time using the two rules delay local search and bidirectional scheduling rule. Cho et al.[28] proposed a hybrid metaheuristic algorithm for virtual machine scheduling that reduce the makespan time, execution time and average number of request is rejected or accepted.

Table 4.1 Literature review on meta-heuristic based scheduling algorithm

S. No.	Year	Technique	Parameters	Tool	Limitations
1	1997 [33]	Meta-heuristic (BPSO based)	Robustness of optimization functions	Personal computer	Sigmoid transfer function does not make a good balance between exploration and exploitation.
2	2009 [37]	Meta-heuristic (BPSO based)	Makespan time and flow time	Personal computer Pentium IV, 3.2 GHz	Algorithm does not consider any QoS constraint like deadline, priority, scalability. Exploration and exploitation of proposed algorithm transfer function is poor.
3	2012 [26]	Meta-heuristic (Hybrid improved BPSO based)	Load probability, annual supply reserve ratio	Matlab	Hybrid algorithm has slow convergence rate.
4	2012 [30]	Meta-heuristic (BPSO based)	Fitness value of function	Personal computer	Sigmoid-kind function does not make a good balance between exploration and exploitation.
5	2012 [34]	Meta-heuristic (modified BPSO based)	0-1 knapsack problem	Personal computer	Transfer function provides better exploration but does not provide better exploitation at the last phase of iteration.
6	2013 [5]	Meta-heuristic (Modify ABC)	Reliability, efficiency and accuracy	Personal computer	Proposed algorithm is used for continuous optimization problems.
7	2014 [21]	Meta-heuristic (Self adaptive learning PSO based)	Profit and execution cost	Matlab	Does not provide better exploration and exploitations. Algorithm does not considered important QoS parameters like makespan time, throughput etc.
8	2014 [28]	Meta-heuristic (Hybrid PSO based)	Execution time, makespan time, task rejection ratio	Personal system with core i7 and 3.4 GHz	Pre-reject operator degrades the performance when more tasks are rejected, it is better to use scalability concept when size of requests is larger than the available

					resources like cpu, memory.
9	2016 [23]	Meta-heuristic (Improved PSO based)	makespan time, energy consumption	Personal computer that have 1GM RAM	Real relative data of production scheduling is limited and more evaluation of the energy saving model by specifying the given parameters in factory applications needs to be performed.
10	2016 [31]	Meta-heuristic (BPSO based)	High-utility item set mining	PC Core2 i3- 4160 CPU and 4GB of RAM	Sigmoid function does not provide the exploitation at the last phase of execution for better results of parameters.
11	2016 [35]	Meta-heuristic (s shaped versus v shaped BPSO based)	Global minima of benchmark function	Personal computer	V shaped transfer function perform better than s shaped transfer function. Proposed v shaped transfer function does not provide the exploitation in efficient way.
12	2017 [22]	Meta-heuristic (Hybrid PSO based)	Makespan time, Execution cost	Cloudsim	Does not provide better exploitation at the last phase of execution and algorithm perform well for workflow scheduling and there is no guarantee of good performance at independent tasks.
13	2017 [25]	Meta- heuristic(Dynamic transfer function based BPSO)	Exploration and exploitation	NA	Time varying transfer function is work well for at 0-1 knapsack problem but does not improve the parameters of scheduling algorithm at large scale.
14	2018 [12]	Meta-heuristic (Gravitational search algorithm)	Convergence rate, reliability, accuracy	NA	Proposed algorithm is used for continuous optimization problems.
15	Our algo.	Meta-heuristic (dynamic transfer function based BPSO)	Execution time, makespan time, Throughput	Cloudsim	Algorithm does not consider other QoS parameters like deadline, elasticity etc.

Binary PSO is used to solve various types of discrete problems [30-32] but it has been observed that BSPO unable to maintain the good balance between exploration and exploitation. The binary PSO has good capability of convergence but it suffers the demerit of premature convergence due to the loss of diversity. Improving the exploration and exploitation ability of PSO is an active research topic. To overcome the problem of exploration and exploitation of binary PSO, sigmoid transfer function, a linear transfer function and two different V-Shaped transfer function were proposed in literature [33-36].

4.3.1 Sigmoid transfer function

The main aim of sigmoid transfer function (shown in Fig. 4.1) is to map a calculated or given velocity V_i^{k+1} to a probability value which have the range [0,1] for changing the binary particle position using the equations 7 & 8. If value of velocity is higher either positive or negative (4 or -4) then probability of flipping of bit is lower [33]. Transfer function provides the higher probability of flipping the bit when velocity value is low i.e. nearby zero. As per the transfer function output if velocity value is close to zero then maximum chance for better exploration. If value of velocity is high then chance for exploitation. Sigmoid transfer function is not able to maintain the sequence of velocity from near to zero to higher velocity range hence transfer function is facing the problem to maintain the good balance between exploration and exploitation.

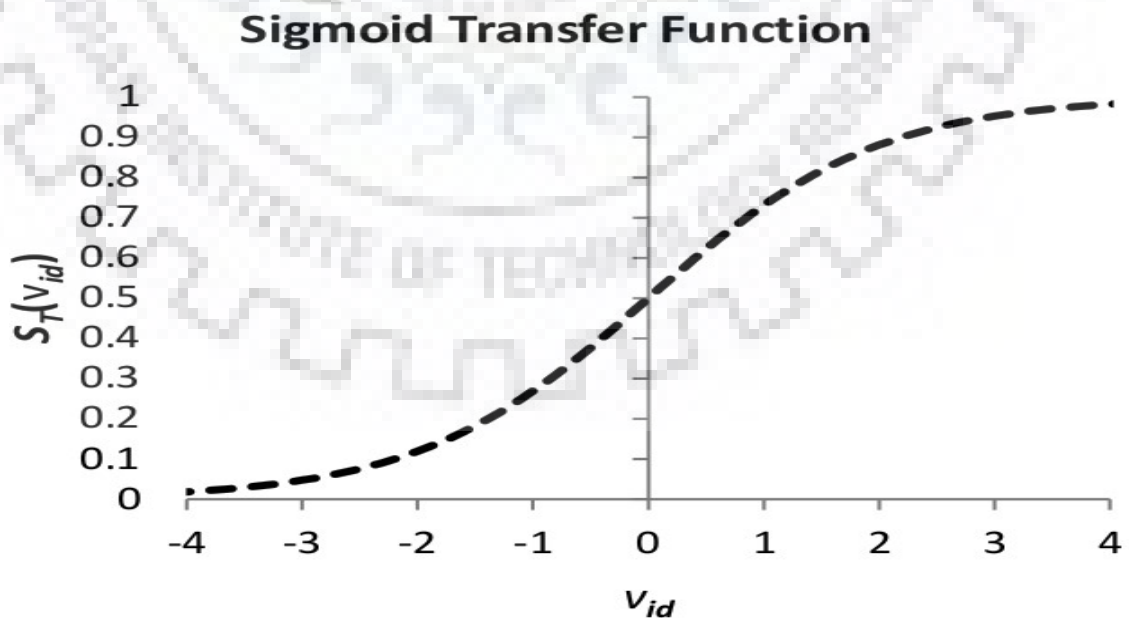


Figure 4.1 Sigmoid transfer functions

4.3.2 Linear normalized transfer function

It is used to improve the exploration ability of sigmoid function [34]. Linear normalized transfer function is work based on defined mathematical equations 9 & 10.

$$L_T(Z_i^k, V_i^{k+1}) = (Z_i^k + V_i^{k+1} + V_{\max}) / 1 + 2V_{\max} \quad (9)$$

$$Z_i^{k+1} = \begin{cases} 1 & \text{if } L_T(Z_i^k, V_i^{k+1}) > \text{Rand}() \\ 0 & \text{otherwise} \end{cases} \quad (10)$$

It is observed that L_T -BPSO face the same challenge as sigmoid function but it provides better exploration than sigmoid function.

4.3.3 V-Shape transfer function

Two different V-shaped transfer function V_{T1} and V_{T2} are reported in literature [35-36].

Transfer function V_{T1} transforms the particle velocity to binary position using the equation 11 & 12.

$$V_{T1}(V_i^{k+1}) = \begin{cases} 1 - \frac{2}{1 + e^{-(V_i^{k+1})}} & \text{if } V_i^{k+1} \leq 0 \\ \frac{2}{1 + e^{-(V_i^{k+1})}} - 1 & \text{otherwise} \end{cases} \quad (11)$$

$$Z_i^{k+1} = \begin{cases} 0 & \text{if } \text{rand}() \leq V_{T1}(V_i^{k+1}) \text{ and } V_i^{k+1} \leq 0 \\ 1 & \text{if } \text{rand}() \leq V_{T1}(V_i^{k+1}) \text{ and } V_i^{k+1} > 0 \\ Z_i^k & \text{if } \text{rand}() > V_{T1}(V_i^{k+1}) \end{cases} \quad (12)$$

Like sigmoid and linear transfer function V_{T1} – BPSO is unable to maintain good balance between exploration and exploitation.

V_{T2} is a tangent hyperbolic function which is used to transform real value of velocity into probability value.

$$V_{T2}(V_i^{k+1}) = \left| \frac{2}{\pi} * \arctan\left(\frac{\pi}{2} * V_i^{k+1}\right) \right| \quad (13)$$

$$Z_i^{k+1} = \begin{cases} (Z_i^k)^{-1} & \text{if } \text{Rand}() < V_{T2}(V_i^{k+1}) \\ Z_i^k & \text{otherwise} \end{cases} \quad (14)$$

It is seen that V_{T2} improve the exploration and exploitation ability of the PSO particle and provide the better results than V_{T1} , sigmoid function and linear function but it cannot provide better exploration at the early stage of run and exploitation in the last stage of the run. To overcome this problem, we proposed dynamic transfer function based binary particle swarm

optimization algorithm (TF_P-BPSO) that maintains the good balance between exploration and exploitation.

4.4 Problem Formulation

Cloud resource broker (CRB) is responsible for scheduling the tasks (job/applications) in such a way that all tasks complete their execution in minimum time i.e., CRB choose the best virtual machine among all the available virtual machine for every upcoming tasks. Users expected to complete their tasks in minimum time therefore cloud service provider need high end workstation. Suppose after a time interval a task window is submitted to CRB that contain the n tasks $T_1, T_2, T_3 \dots T_n$. Each task is independent in nature and pre-emption is not allow at the time of execution i.e., if a task is executing then other task has to until first one complete his execution. Every task has the task length TL_i that is expressed in million instructions (MI) as per the Standard Performance Evaluation Corporation (SPEC) benchmark. Every task required p number of processor, s is the speed of processor, required amount of RAM is M, S is for secondary storage required and B is for bandwidth of nodes. Cloud service provider contains m number of heterogeneous and dynamic resources $R_1, R_2, R_3 \dots R_m$. Cloud resources are heterogeneous in terms of RAM memory, processor speed, number of processor, secondary memory and bandwidth etc. The processing speed of the resources is measured in MIPS as per standard SPEC benchmark. If any resource R_j is matched with the upcoming task T_i then value of decision variable $\Phi_{T_i R_j}$ is 1 otherwise its value is 0.

The main objective of the proposed algorithm is to minimize the total execution time using the dynamic transfer function based binary particle swarm optimization algorithm (TF_P-BPSO). The notations and their description are shown in Table 4.2 which we are used for formulation in objective function. Suppose n tasks $T_1, T_2, T_3 \dots T_n$ are sending by user in a particular task window w_p for services and its total execution time can be find out using the equation 15 to 19.

We define the objective function that aim is to minimize the total execution time

$$\text{Objective: Min TET}_{T_i} = E E T_{T_i R_j} \in \mu_{T_i w_p} \quad (15)$$

$$E T_{T_i R_j} \in \mu_{T_i w_p} = TL_i / PC_{R_j} \quad (16)$$

$$PC_{R_j} = p * s \quad (17)$$

$$T T_{T_i R_j} \in \mu_{T_i w_p} = TL_i / B_{R_j} \quad (18)$$

$$EET_{T_i R_j \in \mu_{T_i w_p}} = ET_{T_i R_j} + TT_{T_i R_j} \quad (19)$$

$$\text{Where } |\mu_{T_i w_p}| \leq m$$

Table 4.2 Notation and description

Notations	Description
$w_1, w_2, w_3, \dots, w_k$	Represent the task window from 1 to k
w_p	Represent the p^{th} task window
$T_1, T_2, T_3 \dots T_n$	Task request 1 to n
TL_i	Length of the tasks T_i in MI
$R_1, R_2, R_3 \dots R_m$	Available cloud resources for executing the tasks
PC_{R_j}	Processing capacity of resource R_j in MIPS
p	Number of processor
s	Speed of processor
B_{R_j}	Bandwidth of resource R_j
$\mu_{T_i w_p}$	Represent the matched resource for task T_i in task window w_p
L_{R_j}	Load at the resource R_j
D_{R_j}	Delay of resource R_j
WT_{T_i}	Waiting time of task T_i
$TT_{T_i R_j \in \mu_{T_i w_p}}$	Task transfer time on resource R_j to other resource in the match list $\mu_{T_i w_p}$
$ET_{T_i R_j \in \mu_{T_i w_p}}$	Execution time of task T_i on resource R_j in the match cloud resource $\mu_{T_i w_p}$
$EET_{T_i R_j \in \mu_{T_i w_p}}$	Expected execution time of resources R_j to process the tasks T_i
$TET(w_p)$	Total execution time for the tasks available in task window w_p
n_k	Represent the number of task submit by user k
WT_{T_i}	Waiting time of task T_i
FT_{T_i}	Finishing time of task T_i
$\Phi_{T_i R_j}$	Decision variable contain the value either 1 or 0 depends upon the resource R_j meet to task T_i

Expected execution time is the sum of execution time and task transfer time. We are not considering booting time of virtual machine in total execution time of tasks. Fitness function to minimize the execution time is defined in equation 20

$$\text{Min } f(R_j) = \Phi_{T_i R_j} * EET_{T_i R_j} \quad (20)$$

Constraints:

$$\begin{aligned} |\mu_{T_i w_p}| &\leq m \\ R_j \in \mu_{T_i w_p}, \quad \Phi_{T_i R_j} &= 1 \end{aligned} \quad (21)$$

$$R_j \notin \mu_{T_i w_p}, \quad \Phi_{T_i R_j} = 0 \quad (22)$$

Once task is allocated to cloud resources (virtual machine) then load at the resource can be calculated using the equation 23.

$$L_{R_j} = L_{R_j} + ET_{T_i R_j} \quad (23)$$

$$FT_{T_i} > a_i + ET_{T_i R_j} \quad \forall i \in n \quad (24)$$

FT_{T_i} is the finishing time of task T_i at virtual machine VM_j . a_i is task arrival time, if it is known and certain then problem is static otherwise problem is dynamic. $ET_{T_i R_j}$ is the execution time of task T_i at virtual machine VM_j . Equation 24 indicates that a task can't be started before its time.

$$FT_{i,j} \geq FT_{i-1,j} + ET_{T_i R_j} \quad (25)$$

Equation 25 represent that a task start to execute at a virtual machine only when previous task has been completed its execution at that particular virtual machine.

Makespan time or total time taken by cloud resource to execute all the tasks can be calculated using the equations 26 and 27.

$$\text{Makespan time (MST)} = \max \{FT_{R_j}\} \quad (26)$$

$$FT_{R_j} = \sum_{j=1}^m EET_{T_i R_j} \quad (27)$$

Throughput of the system is calculated using the equation 28

$$\text{Throughput } (\Gamma) = \frac{\text{number of tasks completed successfully}}{\text{Total processing time}} \quad (28)$$

Main objective of proposed dynamic transfer function based binary particle swarm optimization algorithm (TF_P-BPSO) algorithm is to execute the task in minimum time with maximum throughput i.e. execution time as well as makespan time should be minimum.

4.5 Proposed Cloud Architecture

The proposed architecture shown in Fig. 4.2 contains three major parts: User level phase, Cloud resource broker phase or scheduling phase and Cloud level phase or infrastructure level phase.

4.5.1 User level phase

First phase is situated at the top of cloud architecture as shown in Fig. 4.2 cloud users submit job/task/applications request $T_1, T_2, T_3 \dots T_n$ through the user interface either graphical user interface or command line and specifying the requirement of service in terms of software, hardware and quality of services (QoS). Hardware requirement in terms of number of cpu required, amount of main memory required, amount of secondary memory required, bandwidth etc. and software requirement Mpich-1.2.7, Charm++ etc and QoS (deadline, throughput, priority, execution time, response time etc) to process the user request. The entire upcoming requests authentication is checked by gatekeeper or job request handler to identify that request is coming from legitimate user or an attacker using the turing test types of approach. If request is coming from legitimate user, it is send to the next phase for further processing otherwise request is rejected.

4.5.2 Cloud resource broker (CRB) phase

The request accepted by user level phase is send to cloud resource broker phase. CRB contain many component like controller node, matchmaker or task scheduling, workload monitor etc. each component working is useful for task scheduling in cloud environment. Controller node accepts the entire authentic tasks request which is coming from job request handler. Controller node contains all the information about the virtual machine (busy, ideal, underloaded or overloaded) and sends all these information (upcoming tasks and current virtual machine status) to matchmaker node. Matchmaker node map all the tasks request with available cloud resources. Matchmaking strategy is mainly depends upon the user estimated task execution time of a task. It firstly check the quality of service parameter i.e. request contain any priority or deadline, if yes, and then allocate the virtual machine based upon the QoS parameter using the scheduling algorithm otherwise allocate the task based upon the scheduling algorithm. There are lots of heuristic and meta-heuristic scheduling algorithm has been proposed for cloud environment like Min-Min, Max-Min, artificial bee colony, particle swarm optimization, ant colony optimization, shortest job first, first come first serve etc.

In this chapter, we are using dynamic transfer function based binary particle swarm optimization (TF_p-BPSO) metaheuristic algorithm for scheduling the tasks. The main objective of scheduling algorithm is to optimize (minimize or maximize) the parameter (makespan time, execution time, response time, cost etc.). The entire process of task-resource mapping is control by cloud task scheduler phase. CRB component scheduling parameter shows that we optimize the parameter makespan time, total execution time and utilization of resource using the modify BPSO algorithm. The main aim of workload component is to monitor all the virtual machine continuously and pass the status of virtual machine to schedulers that pass it further to controller node.

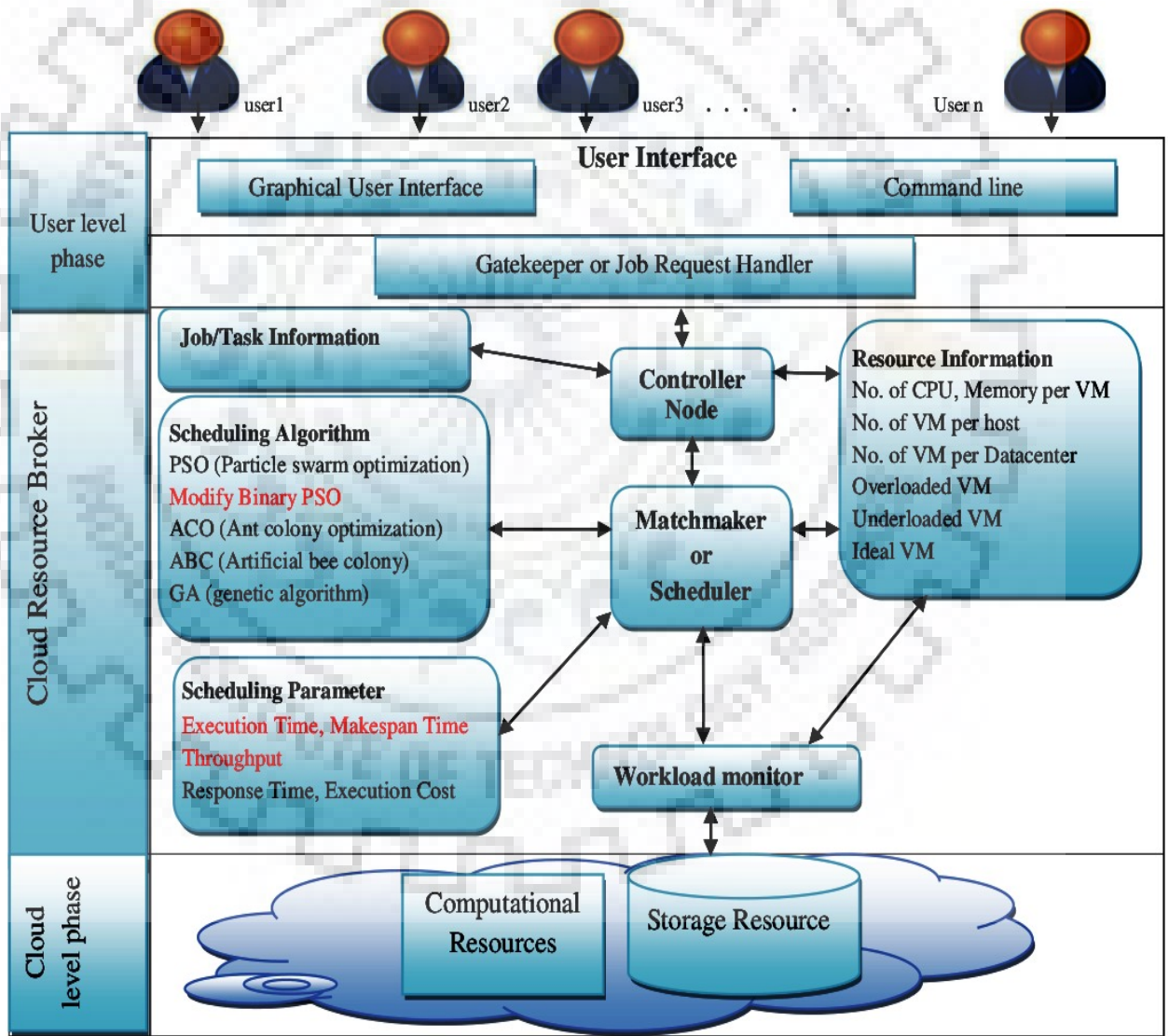


Figure 4.2 Proposed Cloud Task scheduling architecture

4.5.3 Cloud level phase

It is also called the infrastructure level phase because all the cloud infrastructure (datacenter, hosts, workstation, nodes etc) are exist in this phase. Cloud is collection of heterogeneous resources in terms of computational resources, storages resources etc. each node contain the number of virtual machine for processing the task depends upon the configuration of node. Number of virtual machine can be increase or decrease (elasticity) at the run time depends upon the upcoming request. Generally a task is allocated to one virtual machine if task is highly computation oriented then CRB allocate the high end virtual machine for the task so that it can execute easily otherwise task takes more time to execute due to which response time is increase and user satisfaction is decrease. Therefore all the virtual machine is in heterogeneous nature in cloud environment.

4.6 BPSO based modified transfer function (TF_P-BPSO)

Binary particle swarm optimization algorithm is focus at exploration at the early stage of the run to avoid the condition of trapped in local optima, but when iteration increase, algorithm start to move from exploration to exploitation and more emphasizing on exploitation at the last stage of run. We proposed a dynamic transfer function (TF_P-BPSO) with some consideration

- Modify transfer function TF_P-BPSO should provide the exploration at the early stage by high flipping of bits of particle position Z_i^k for any velocity V_i^k .
- TF_P-BPSO should have the ability to decrease the probability of flipping of bits in intermediate stage for particle position Z_i^k at any velocity V_i^k so that it can move from exploration to exploitation.
- Last stage of run should provide stronger exploitation i.e., there should be very less probability of flipping of bits for position Z_i^k at any velocity V_i^k .

We proposed a new dynamic transfer function representing in equation 29 that consider the above concept

$$TF_P(V_i^{k+1}, \lambda) = \frac{1}{1 + e^{-(V_i^{k+1})/\lambda}} \quad (29)$$

λ is control parameter in equation 29 that start with the high value and randomly decrease within a interval after each iteration when run is in progresses. If we provide a fix value to control parameter λ it will not shift from exploration to exploitation as run progress therefore value of λ is decided based upon the given equation 30

$$\lambda = \lambda_{max} - \text{Itr}_{k+1} \left(\frac{\lambda_{max}}{\text{Itr}_{max}} \right) * T^q / (T^q + a^q) \quad (30)$$

Where $T \in (3 \text{ to } 11)$ and value of a is always 1. Value of q is 3 to 11 but it gives good results at value 7. λ_{max} is maximum value of $\lambda = 4$ (equal to maximum value of velocity $V=4$) is used to bound the control parameter. Itr_{k+1} represent the current iteration and Itr_{max} represent the maximum iteration. Proposed transfer function $\text{TF}_P(V_i^k, \lambda_{min})$ represent the final shape of the curve with value $\lambda = .05$. The proposed transfer function change the position of the i^{th} particle instead of sigmoid function as shown in equation 31.

$$z_i^{k+1} = \begin{cases} 1 & \text{if Rand() < TF}_P(V_i^{k+1}, \lambda) \\ 0 & \text{otherwise} \end{cases} \quad (31)$$

Proposed transfer function (TF_P) shape changes over the time depending upon the value of λ . Value of λ will be high at the starting phase that divides the value of velocity. Suppose initial random velocity value is $\{3.8, -3, 1, 2.5\}$ and calculated probability of flipping of bits using the

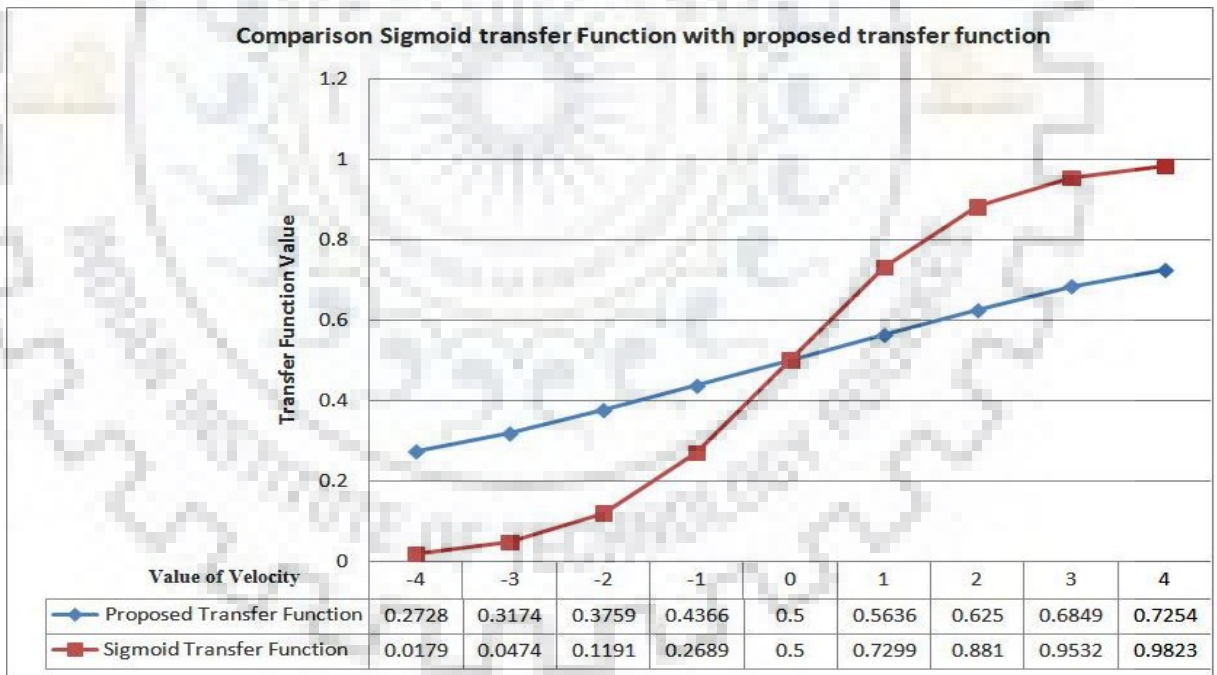


Figure 4.3 Comparison of proposed transfer function with sigmoid transfer function

proposed transfer function is $\{.71, .32, .574, .64\}$ that is much better than sigmoid function $\{.99, .0476, .735, .9259\}$. Let's consider one more example of random velocity set with values $\{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$ in the range -4 to 4. Calculated probability of flipping of bits using

the proposed transfer function is $\{.2728, .3174, .3759, .4366, .5, .5636, .625, .6849, .7254\}$ that is better than the sigmoid function $\{.0179, .0474, .1194, .2689, .5, .7299, .881, .9532, .9823\}$. Results shown in Fig. 4.3 prove that proposed transfer function provides the better exploration (maximum probability of flipping the bits) in comparison of sigmoid transfer function because its values is nearby .5.

We can further change the proposed transfer function $(TF_P(V_i^k, \lambda_{max} - .5), TF_P(V_i^k, \lambda_{max} - 1))$ to $TF_P(V_i^k, \lambda_{max} \pm 3)$ for better exploration and exploitation results in binary PSO. Curve of modified transfer function $TF_P(V_i^k, \lambda_{max})$ is closest to probability value 0.5 i.e. it gives the highest possibility of flipping the bit than any other curve. $TF_P(V_i^k, \lambda_{min})$ provide the lowest probability of flipping the bit therefore it provides the stronger exploitation at the final stage of run.

4.7. Modified BPSO (Dynamic transfer based (TF_P-BPSO)) based scheduling algorithm

We have modified BPSO algorithm for scheduling in cloud computing using the dynamic transfer function based binary particle swarm optimization algorithm (TF_P-BPSO) algorithm. Each particle contains a position matrix of size $m \times n$, where m represent the number of virtual machine and n represent the number of tasks. Each particle's position matrix has two properties:

(i) All the elements of the matrix are either 0 or 1. If Z_k is position matrix of k^{th} particle then

$$Z_k(i, j) \in \{0, 1\} \quad (\forall i, j) \text{ where } i \in \{1, 2, \dots, m\} \text{ and } j \in \{1, 2, \dots, n\} \quad (32)$$

(ii) Each column contains only one element value is 1 other element should be 0 because a task is allocated to only one virtual machine. We can represent this condition by equation 33

$$Z_k(i, j) = 1 \quad \text{if } T_j \rightarrow VM_i \quad (33)$$

$$\text{Otherwise } Z_k(i, L) = 0 \quad L \neq j \quad \forall L \in \{1 \text{ to } n\} \text{ and } \forall i \in \{1 \text{ to } m\}$$

If $Z_k(i, j) = 1$ then j^{th} task is executed by i^{th} virtual machine for any k^{th} particle. In the position matrix row represent the task allocated to virtual machine and column represent the task allocation. Tables 4.3 represent the position matrix that contains six tasks and three virtual machines. Task are allocated randomly at the starting, position matrix represent that task T_2, T_4, T_5, T_6 are allocated to resource R_1 , task T_1 is allocated to R_2 task T_3 is allocated to R_4 and no task is allocated to resource R_3 .

Table 4.3 Position matrix

Task/VM	T ₁	T ₂	T ₃	T ₄	T ₅	T ₆
R ₁	0	1	0	1	1	1
R ₂	1	0	0	0	0	0
R ₃	0	0	0	0	0	0
R ₄	0	0	1	0	0	0

Particle velocity: Particle velocity is represented in $m \times n$ matrix form whose range is $[-V_{\max}, V_{\max}]$.

$$V_k(i, j) \in \{-V_{\max}, V_{\max}\} \quad (\forall i, j) \text{ where } i \in \{1, 2, \dots, m\} \text{ and } j \in \{1, 2, \dots, n\} \quad (34)$$

Where V_k represent the velocity matrix of k^{th} particle.

Pbest and Gbest represent the position matrix of $m \times n$ size with their elements either 0 or 1.

Pbest_k is the best position of individual particle (k^{th} particle) has visited from the starting of the algorithm and Gbest_k is the best position of the k^{th} particle and its neighbor has visited from starting of the iteration. Pbest_k and Gbest_k are updated based upon the fitness function (objective function) in each iteration. Fitness value of each particle Z_k is calculated if its current calculated value is smaller (minimizing the execution time based upon the objective function) than the Pbest_k then replace it with Z_k . To update the value of Gbest_k its fitness value is compared with all the neighborhood Pbest_k value. If neighborhood Pbest_k fitness value is smaller than the Gbest_k then replace it with neighborhood Pbest_k.

Particle updating equation: Particle updating velocity matrix is represented in equation 35 and updating position matrix of each particle with the help of modified transfer function in BPSO is represented in equation 36.

$$V_k^{t+1}(i, j) = w * V_k^t(i, j) + c_1 r_1 (Pbest_k^t(i, j) - Z_k^t(i, j)) + c_2 r_2 (Gbest_k^t(i, j) - Z_k^t(i, j)) \quad (35)$$

After calculating the real velocity value by equation 34, transform the value of velocity from continuous space to binary space by the proposed dynamic transfer function using the equation 29 & 30.

$$Z_k^{t+1}(i, j) = Z_k^t(i, j) + V_k^t(i, j) \quad (36)$$

Sometimes calculated value of $V_k^t(i, j)$ and $Z_k^t(i, j)$ comes 1 and total sum of both the value is 2 for $Z_k^{t+1}(i, j)$ but in BPSO value should be either 0 or 1. Therefore we use the mod function to

convert the value only 0 or 1 form. Proposed algorithm for task scheduling dynamic transfer function based modified BPSO algorithm is shown in Fig. 4.4.

```

1. Input: Number of tasks and number of virtual machine
2. Output: Calculate the minimum execution time of tasks at running virtual machine
   by fitness function
3. Start the iteration Itr
4.  $S_S$ =Swarm Size ,  $V_R$ =Random Velocity,  $V_k(i, j)$  =Velocity of Particle,
    $Z_k(i, j)$ =Position of Particle,  $P_R$ =Random position
5. Pbest=Local best, Gbest=Global best
   // Pbest and Gbest are position matrix of size m*n where m is no. of resource
   and n is no. of tasks
6. For i=1 to  $S_S$  do
   // randomly initialize the velocity and position matrix at the starting
7.  $V_k(i, j) \leftarrow V_R() \in \{-V_{max}, V_{max}\}$ 
8.  $Z_k(i, j) \leftarrow P_R \in \{0, 1\}$ 
9. Evaluate the fitness function  $f(Z_i)$ ;
10. If  $f(Z_i) < f(Pbest)$ 
11. Then Pbest  $\leftarrow Z_i$ 
12. End if loop
13. If  $f(Z_i) < f(Gbest)$ 
14. Then Gbest  $\leftarrow Z_i$ ;  $f(Gbest) \leftarrow f(Z_i)$ ;
15. End if
16. End for loop of  $S_S$ 
17. While Itr <  $Itr_{max}$ 
18. For loop of  $S_S$ 
19. Update the value of velocity using the equation 34 & 35
20. Calculate the proposed  $\lambda$  using equation 30
21. Calculate modified transfer function  $TF_P(V_i^{k+1}, \lambda)$  using the equation 29
22. Update the position matrix  $Z_k^{t+1}(i, j)$  of particle using the equation 36
23. If fitness function  $f(Z_i) < f(Pbest)$ 
24. Then Pbest  $\leftarrow Z_i$ 
25. End if loop
26. If  $f(Z_i) < f(Gbest)$ 
27. Then
28. Gbest  $\leftarrow Z_i$ ;
29.  $f(Gbest) \leftarrow f(Z_i)$ ;
30. End if
31. .End for of  $S_S$ 
32. Itr=Itr+1;

```

Figure 4.4 Proposed dynamic transfer function based (TF_P-BPSO) algorithm for task scheduling

4.8 Analysis and comparison of simulation results

Modified transfer function based proposed TF_p -BPSO algorithm minimizes the execution time as well as makespan time of tasks in cloud environment using the cloudsim platform. We choose cloudsim simulator for experimental purpose because to implement the work in real environment is costly (need to established cloud infrastructure). To test the proposed algorithm we have created a datacenter that contains number of host and each host contains the number of heterogeneous virtual machine depends upon the configuration of host (processing speed, number of cpu, memory etc.). Each virtual machine (VM) has the parameter like Id, MIPS, number of cpu etc as shown in Table 4.4. After that we have generated cloudlet (Task) with their parameter like TaskID, Length etc. as shown in Table 4.5. Cloud resource broker allocate the tasks to virtual machine based upon the proposed BPSO algorithm. In this chapter, we have calculated and analyzed the execution time, makespan time and resource utilization ratio with the help of cloudsim simulator.

4.8.1 Execution time of tasks:

Let's consider the example to solve the problem of task scheduling in cloud environment by proposed binary particle swarm optimization algorithm based on dynamic transfer function. Each particle is represented by matrix of size $m \times n$ where m is number of virtual machine and n is number of upcoming tasks at run time (swarm size is 10 in this example i.e. 10 matrix is generated by the entire 10 particle). Particle position matrix elements are in $[0, 1]$ interval and sum of the elements of each column should be 1 i.e. there is only single entry of 1 in each column rest of element should be 0. Consider four heterogeneous virtual machine (different processing speed and memory) and six tasks with different length. Apply the algorithm shown in Fig.4.4. All the tasks are allocated randomly to virtual machine at the first iteration as shown in Table 4.3 by particle position matrix. Velocity matrix is also allocated randomly at the starting phase. After that calculate the Pbest and Gbest of each particle based upon the fitness function (execution time). If value of a particle current position matrix is smaller than the Pbest then assigned the current position matrix to Pbest.

If fitness value of any neighborhood Pbest is smaller than the Gbest then replace Gbest with Pbest. Start for loop of iteration until the end of simulation or end of iteration to find out the results. Calculate the velocity and position matrix based upon the modified transfer function and find the new optimize values of Gbest that represent the value of optimize position matrix

i.e. execution time of tasks. To find out the value of execution time of tasks, range of BPSO parameter is set in first experiment as

Table 4.4 VM properties

VM Id	VM MIPS	Memory	No. Of cpu
0	40	256	1
1	50	512	1
2	100	512	1
3	80	256	1

Table 4.5 Task properties

Task Id	Length	No. of cpu required
0	1000	1
1	1500	1
2	2000	1
3	1200	1
4	2400	1
5	1800	1

Table 4.6 Execution time comparison between BPSO and proposed BPSO

Case study	BPSO	Developed Modified BPSO
1	114	108
2	125	101
3	120	109
4	113	99
5	123	104
6	124	108
7	119	112
8	109	101
9	115	99
10	116	104

First case: $w \in \{.9 \text{ to } .4\}$, c_1 and $c_2 \in \{.5 \text{ to } 2.5\}$, velocity of particle $\in \{-4 \text{ to } 4\}$, number of task and number of VM are shown in Table 4. and Table 4.5. Value of $\lambda \in \{4 \text{ to } .05\}$, number of iteration is 200 and number of particle is 10. We run the simulation 10 times for both the approaches existing BPSO [37] and developed BPSO to calculate the execution time. Calculated execution time of tasks from the proposed MBPSO algorithm and existing BPSO algorithm can be change in each simulation at low number of iteration because randomization is exist in both the algorithm (particle velocity, position is initialized randomly at the starting, r_1 and r_2 also random etc.). If number of iteration and number of particle is increased then proposed MBPSO algorithm results is not be fluctuate because it easily converge to the solution.

Table 4.7 Execution time comparison between FCFS, BPSO and developed BPSO

Case study	No. of Task	No. of VM	No. of Iteration	No. of particle	FCFS [38]	BPSO [37]	Developed Modified BPSO
1	6	4	100	10	186	121	102
2	8	4	100	10	262	146	123
3	10	6	100	10	223	137	112
4	15	6	200	20	394	257	192
5	15	8	200	20	332	223	162
6	15	10	200	20	287	204	138
7	20	10	300	20	367	241	173
8	30	15	300	30	388	247	178
9	40	20	400	40	356	234	157
10	50	20	500	50	436	298	203

Calculated results shows (shown in Table 4.6) that the proposed MBPSO perform better than existing BSPO algorithm i.e. developed MBPSO maintain the better ratio of exploration and exploitation during the execution of iteration.

Second case: Dynamic transfer function based proposed BPSO algorithm is further tested at different number of tasks with random length and number of virtual machine with different processing power. Range of tasks is extended from 6 to 50 and virtual machine range is extended from 4 to 20. To calculate the execution time of tasks, range of the proposed BSPO algorithm parameter (w , $c1$ & $c2$ etc.) is same as in first case. First result is calculated at 4 VM that process 6 tasks in 100 iteration considering 10 as swarm size. We compare proposed algorithm result with other existing algorithm like first come first serve (FCFS)[38] and simple BPSO. Calculated results shows that dynamic transfer function based proposed BPSO algorithm perform better than the existing algorithm in literature. In next iteration, result is calculated at 8 numbers of tasks and 4 number of VM. This process is continue up to the 50 number of tasks, 20 number of virtual machine using 100 to 500 iteration and swarm size (number of particle) is increased from 10 to 50 to calculated the execution time. Calculated results is shown in Table 4.7 prove that dynamic transfer function based proposed BPSO algorithm performs better than the existing algorithm in all the conditions. Table 4.7 results represent that proposed dynamic transfer function provides the good balance between exploration and exploitation therefore algorithm converge easily in less step and gives the minimum execution time.

Third case: Further we tested dynamic transfer function based proposed BPSO algorithm at large number of request because huge number of request comes at cloud after a time interval.

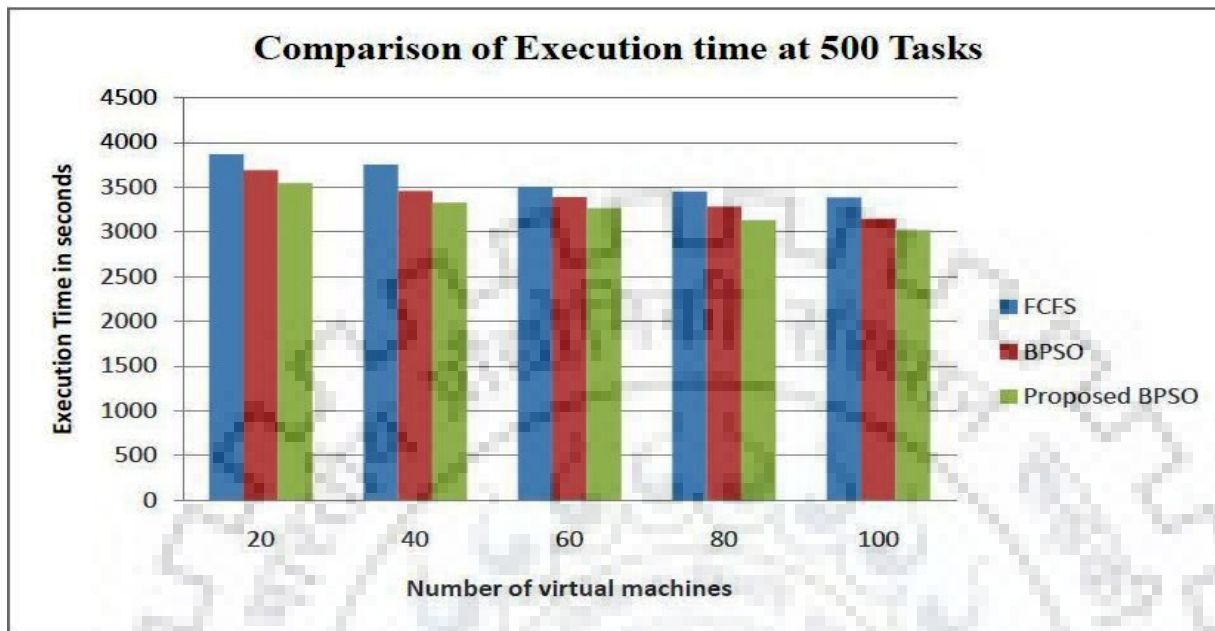


Figure 4.5 Execution Time comparison proposed BPSO with BPSO and FCFS at fixed Tasks

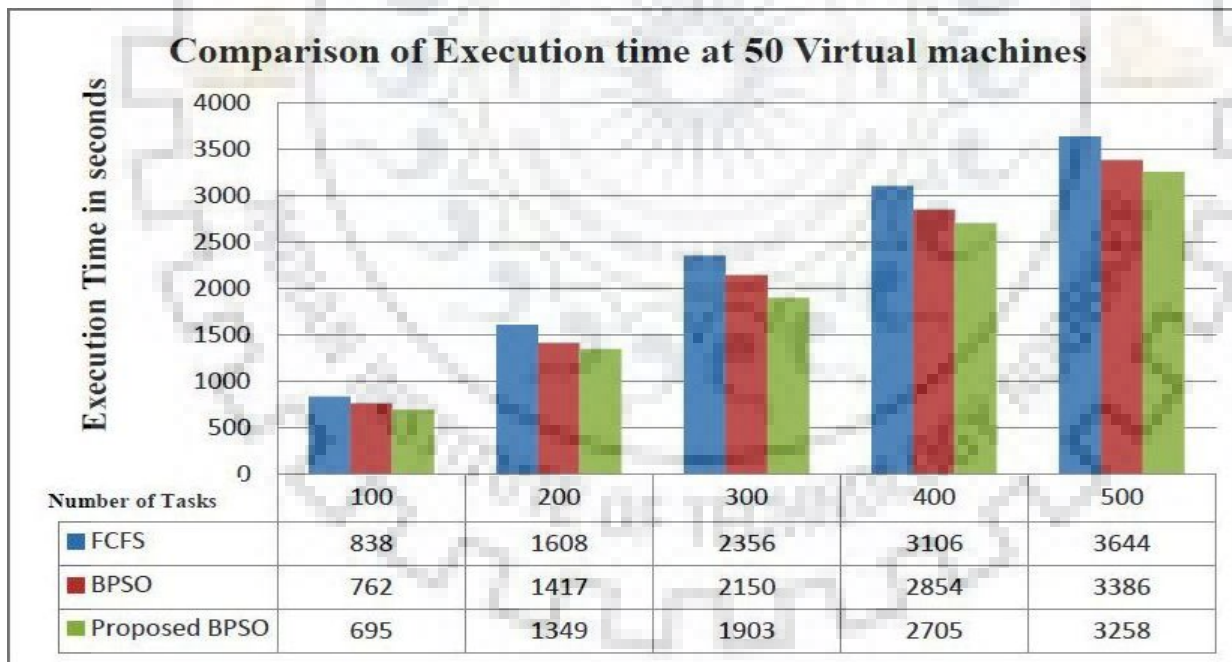


Figure 4.6 Execution Time comparison proposed BPSO with BPSO and FCFS

We have taken fixed 500 numbers of tasks with random length (1000MI to 5000MI) to test the proposed BPSO algorithm but number of virtual machines is vary from 20 to 100 with length 100 to 500 MIPS. Firstly all the tasks are allocated to 20 virtual machines by proposed BPSO

algorithm and all the tasks are executed in 3543 seconds while others algorithm take more time as shown in Fig. 4.5. After that we increase the number of virtual machines from 20, 40 up to 100 and execute all the tasks at virtual machines. Fig. 4.5 results represent that proposed dynamic transfer function based BPSO algorithm perform better than other algorithm.

Fourth case: We also tested our proposed BPSO algorithm at variable number of tasks from 100, 200 up to 500 which are running at 50 fixed number of virtual machines. Computational results shown in Fig. 4.6 proved developed BPSO reduce the execution time up to 10% in comparison with BPSO and up to 20% comparison with FCFS algorithm.

4.8.2 Makespan Time of Tasks

Makespan time is the completion time of tasks at resource (virtual machine) which has the maximum execution time after completion of all the tasks. The main aim of task scheduling is to minimize the makespan time so that user's applications can execute minimum time. Consider 4 virtual machine of different processing power and 6 task of different length at the starting to test the proposed algorithm as shown in Table 4.3 & 4.4. Apply the proposed algorithm and

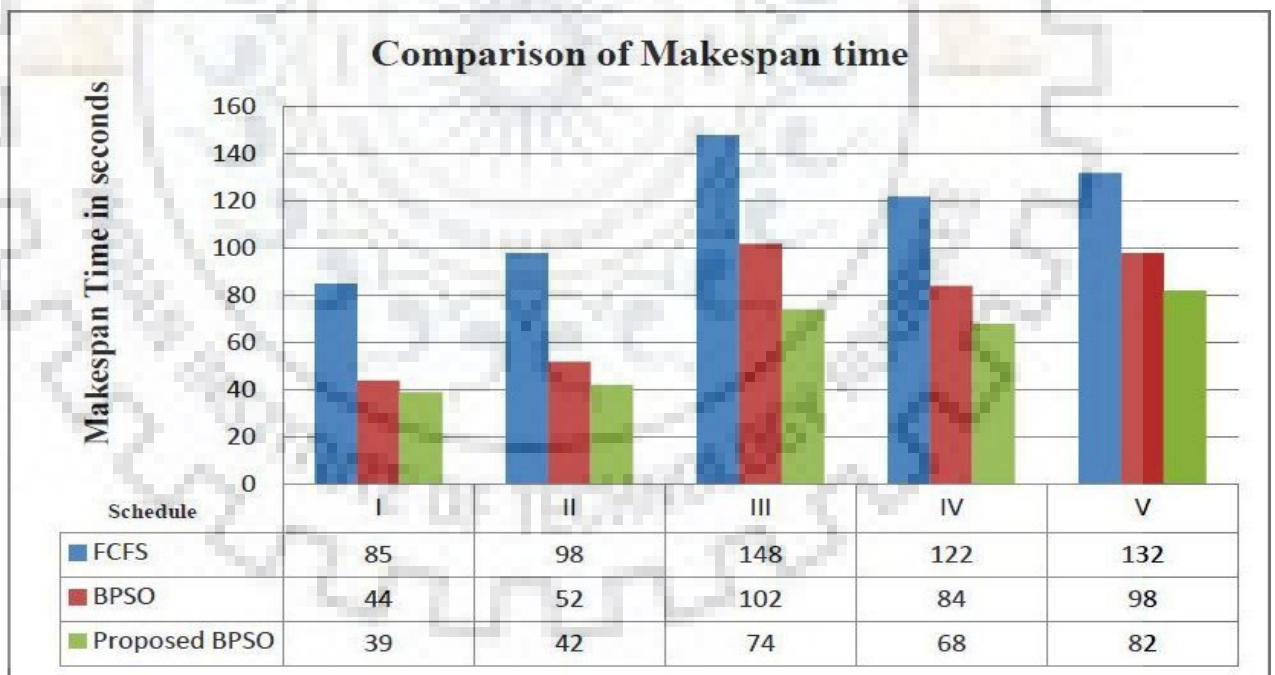


Figure 4.7 Makespan time comparisons between FCFS, BPSO and proposed BPSO

calculate the makespan time as shown in case study I. Calculated more results (case study II to V) in different scenario (different task and VM) to test the correctness of the algorithm. The

range of task is extended from 6 to 50 and length of task is varying from 500MI to 2500 MI. Range of virtual machine is extended from 4 to 20 and MIPS range of VM is 40 to 300. The numerical calculation to determine the performance of the algorithm based upon the selecting the application (tasks) and resources instances and if application is memory-intensive that needed high-memory VM instances for database operation tasks. Therefore the proposed algorithm selects the length of task in a range (500MI to 2500MI) and created the virtual machine instance such that they can process the task.

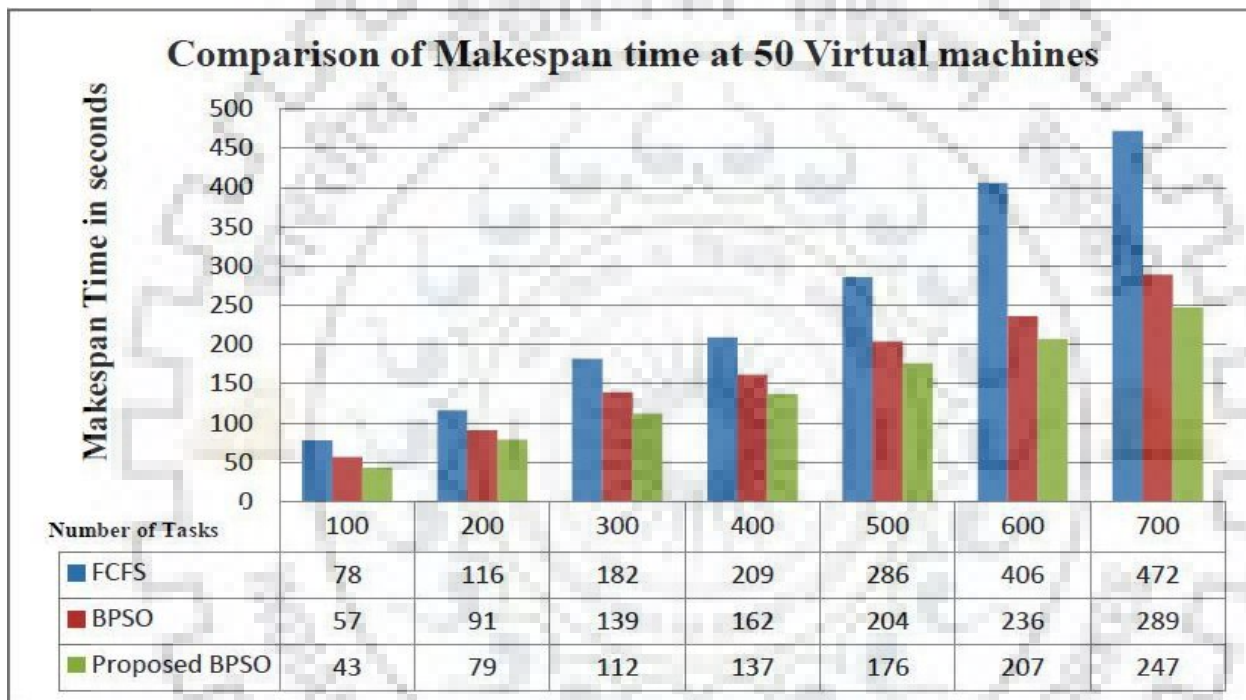


Figure 4.8 Makespan time comparisons between FCFS, BPSO and proposed BPSO

If the range of task is increase or decrease then results (makespan time) is affected. Calculated results of extended task and VM are shown in case study II to V in Figure 4.7. Calculated results prove that dynamic transfer function based proposed BPSO algorithm performs better than the other existing algorithm

We have extended number of tasks to test the performance of the proposed BPSO algorithm at large number of request because huge number of request comes at cloud after a time interval. We increase number of requests/tasks number from 100, 200 up to 500 which are running at 50 fixed number of virtual machines and calculate the makespan time. Computational results shown in Fig. 4.8 proved that execution time of proposed BPSO is better than the existing

BPSO and FCFS algorithm. Further we test the performance of the algorithm at fixed number of tasks while number of virtual machine is variable. We have taken fixed 500 numbers of tasks with random length (1000MI to 5000MI) and allocated to 20 virtual machines by proposed BPSO algorithm and calculate the makespan time which is better than the other algorithms like FCFS and existing BPSO. After that we increase the number of virtual machines from 20, 40 up to 100 and execute all the tasks at running virtual machines as shown in Fig. 4.9. Calculated results proved that developed dynamic transfer function based BPSO algorithm reduce the makespan time up to 15% comparison with BPSO and up to 32% comparison with FCFS algorithm.

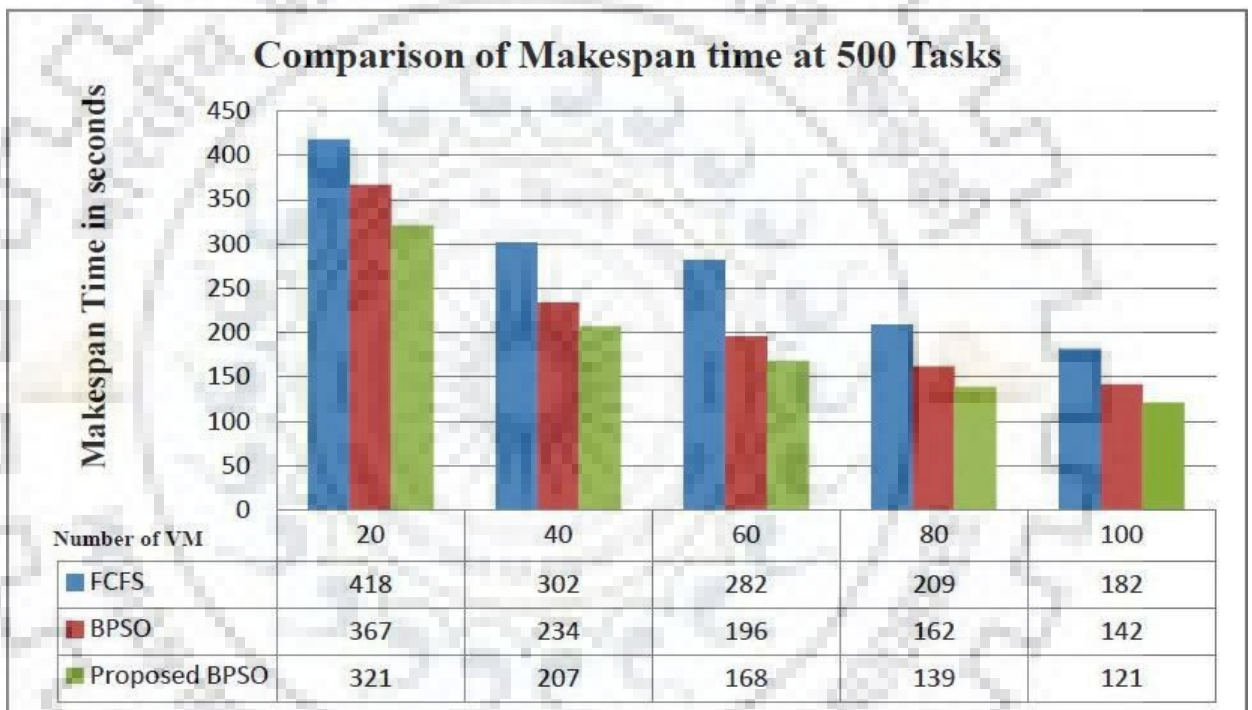


Figure 4.9 Makespan time comparisons between FCFS, BPSO and proposed BPSO

4.8.3 Convergence rate

Convergence rate of algorithms represents that after how much iteration algorithm is converging to final solution (minimum execution time or makespan time) i.e. how much time algorithm is taken to optimize the fitness function. To check the convergence rate of both the algorithm (BPSO and proposed BPSO) simulation environment is created in which 500 iteration is run with swarm size 20. Number of tasks and number of VM is considering 20, 10. Rest of the BPSO parameter value is considering same as taken section 4.9.1 during the calculation of execution time. Run the simulation and calculated results shows (Fig. 4.10) that

proposed BPSO algorithm is converge in less iteration (early) as compare to general BPSO algorithm because proposed BPSO algorithm use dynamic transfer function that provide the good balance between exploration and exploitation.

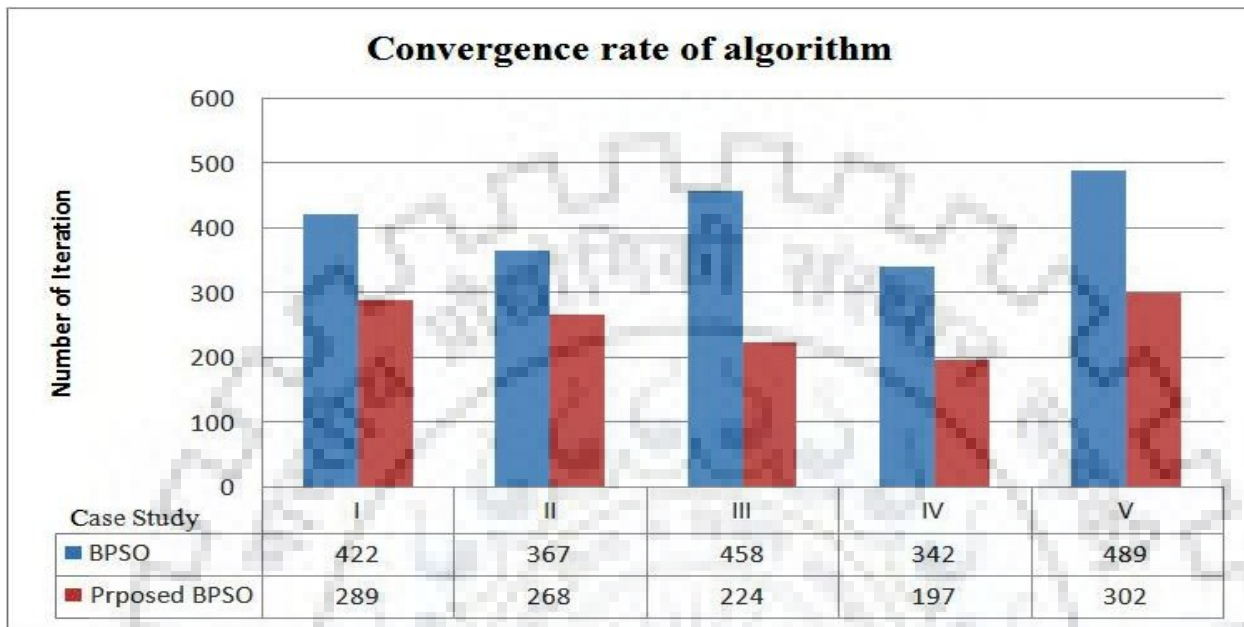


Figure 4.10 Convergence rate comparisons between BPSO and proposed BPSO

4.8.4 Throughput

This experiment analyzes the performance of the algorithms regarding the parameter throughput. It is calculated using the equation 28. Ten different schedules are generated to test the throughput of the developed algorithm. Range of the tasks is extended from six to 50 and range of virtual machine is extended from four to twenty in the generated schedule. Length of the task is generated randomly in each schedule. Calculate the results of the developed algorithm with BPSO parameter like number of iteration and number of particle is shown in Table 4.8. Calculated results represents that when number of task is increase then throughput of the algorithm is decrease due to different type of delay like waiting time of task, buffer delay, task transfer time etc. Developed BPSO based algorithm execute more task in a minute rather than others algorithm because it has the lowest processing time. FCFS [38] algorithm average throughput is 1.19 and BPSO [37] algorithm average throughput is 1.85 per minute which is lower than the developed BPSO algorithm (2.045. Fig. 4.11 represent that developed modified BPSO algorithm has better throughput than other existing algorithm in the entire schedule and all the condition.

Table 4.8 Throughput comparison between FCFS, BPSO and developed BPSO

Schedule	Task	Resource	Iteration	No. of particle	Throughput of FCFS [38]	Throughput of BPSO [37]	Throughput of developed Modified BPSO
S_1	6	4	100	10	1.93548	2.975206	3.5294117
S_2	8	4	100	10	1.374045	2.465753	2.9268292
S_3	10	6	100	10	1.614349	2.627737	3.2142857
S_4	15	6	200	20	.9137055	1.400778	1.875000
S_5	15	8	200	20	1.084337	1.614349	2.222222
S_6	15	10	200	20	1.254355	1.764705	2.608695
S_7	20	10	300	20	.9809264	1.4937759	2.0809248
S_8	30	15	300	30	.92783505	1.4574898	2.0224719
S_9	40	20	400	40	1.01123595	1.5384615	2.2929936
S_{10}	50	20	500	50	.825688073	1.2080536	1.77339901

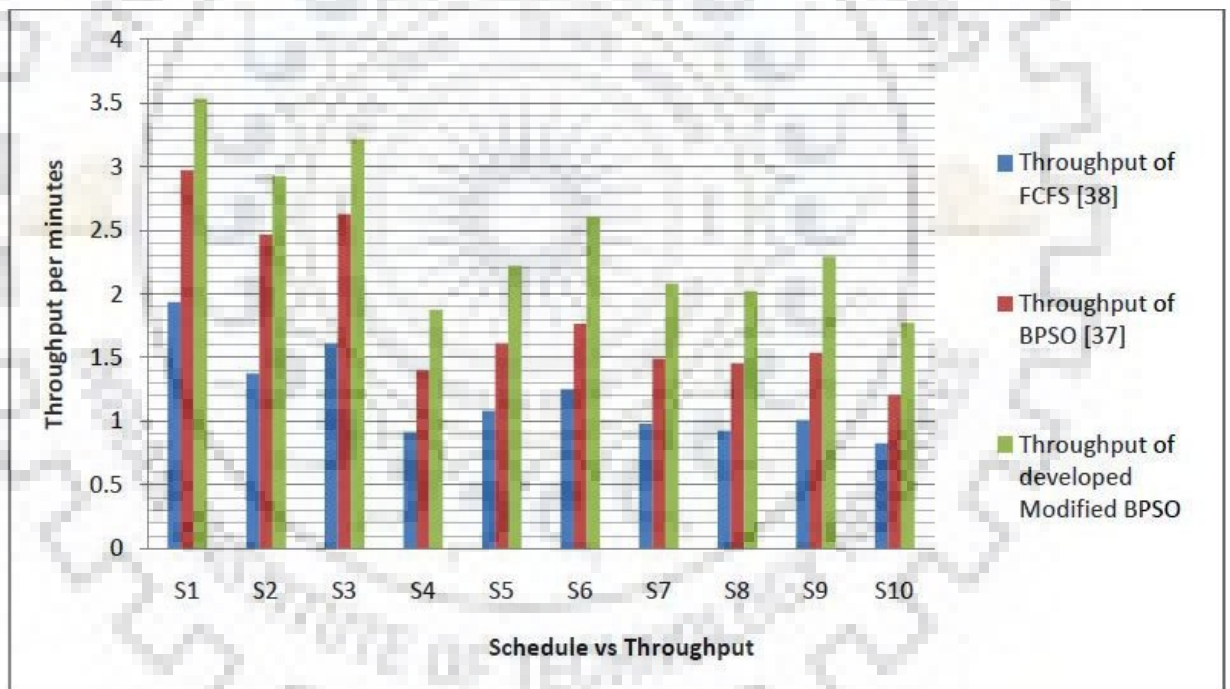


Figure 4.11 Throughput comparisons between FCFS, BPSO and proposed BPSO

4.9. Summary

We have developed a TF_p -BPSO algorithm in which modified transfer function remove the shortcoming of existing BPSO and maintain the good balance between exploration and exploitation in this paper. We have study different transfer function in the literature but all the existing transfer functions have some limitation i.e. some transfer function provides good

exploration but exploitation and some provide better exploitation not exploration. Further we proposed a cloud task scheduling architecture and component of architecture like job request handler, controller node, matchmaker etc. with working principle. The main aim of dynamic transfer function based BPSO algorithm for task scheduling is to complete the tasks in minimum time. Cloudsim simulator has been used for the analysis the performance of the dynamic transfer function based modify BPSO. Simulation results show (Table 4.5 to 4.7, Figs. 4.5 to 4.11) that proposed MBPSO algorithm gives minimum execution time of task comparison to other existing algorithm in all the condition.

4.10 References:

- [1] R. Gupta, "Review on existing load balancing techniques of cloud computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 2, pp. 168-71, Feb. 2014.
- [2] M. E. Fríncu, "Scheduling highly available applications on cloud environments," *Future Generation Computer System*, vol. 32, no. 6, pp. 138–153, May 2012.
- [3] D. Babu and P. Venkata, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol.13, no. 5, pp. 2292–2303, May 2013.
- [4] D. P. Mahato, and R. S. Singh, "Load balanced transaction scheduling using Honey Bee Optimization considering performability in on-demand computing system," *Concurrency and Computation: Practice and Experience*, vol.29, no. 21, 2017.
- [5] J. C. Bansal, H. Sharma, K. V. Arya, and Atulya Nagar, "Memetic search in artificial bee colony algorithm," *Soft Computing*, vol. 17, no. 10, pp. 1911-1928, 2013.
- [6] J.C. Bansal, A. Gopal and A. K. Nagar, "Stability analysis of Artificial Bee Colony optimization algorithm" *Swarm and Evolutionary Computation*.
DOI: 10.1016/j.swevo.2018.01.003.
- [7] F. Ramezani and F.K. hussain, "Task-based System Load Balancing in cloud computing using Particle Swarm Optimization," *International Journal of Parallel Programming*, vol. 42, no. 5, pp. 739-754, Oct. 2013.
- [8] E.Pacini, C. Mateos and C. G. Garino, "Balancing throughput and response time in online scientific Clouds via Ant Colony Optimization (SP2013/2013/00006)", *Advances in Engineering Software*, vol. 84, pp. 31-47, June 2015.

- [9] J. T. Tsai, J. C. Fang and J. H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," *Computer Operation Research*, vol. 40, no. 12, pp.3045-3055, Dec. 2013.
- [10] S. Roy, S. M. Islam, S. Ghosh, S. Das, and A. Abraham, "An Adaptive Differential Evolution Algorithm For Autonomous Deployment and Localization of Sensor Nodes," *Electromagnetic Research*, vol.29, pp.289-309, 2011.
- [11] K. Dasgupta, B. Mandal, P. Dutta, J. K. Mondal and S. Dam, "A genetic algorithm (GA) based load balancing strategy for cloud computing," *Procedia Technology*, vol. 10, pp. 340-347, 2013.
- [12] J.C. Bansal, S. K. Joshi and A. K. Nagar, "Fitness Varying Gravitational Constant in GSA," *Applied Intelligence (APIN)* (2017).
- [13] J. Kennedy, R.C. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, pp. 1942–1948, 1995.
- [14] J. Kennedy, R.C. Eberhart, "A discrete binary version of the particle swarm algorithm," in *International conference on Systems, Man, and Cybernetics*, pp. 4104 – 4108, 1997.
- [15] A. Suresh and P. Vijayakarthick, "Improving scheduling of backfill algorithms using balanced spiral method for cloud metascheduler," in *International Conference on Recent Trends in Information Technology*, pp. 624- 627, Chennai, India, 2011,
- [16] K. Dubey, M. Kumar and M. Chandra, "A Priority Based Job Scheduling Algorithm Using IBA and EASY Algorithm for Cloud Metascheduler," in *International Conference on Advances in Computer Engineering and Applications*, pp. 66-70, Ghaziabad, India, 2015.
- [17] H. Chen, F. Wang, N. Helian and G. Akanmu, "User Priority Guided Min-Min Scheduling Algorithm For Cloud Computing," in *national conference on Parallel Computing Technologies (PARCOMPTECH)*, pp. 1-8, Bangalore, India, Oct. 2013.
- [18] Y. Mao, X. Chen and X. Li, "Max–min task scheduling algorithm for load balance in cloud computing," in *International Conference on Computer Science and Information Technology*, vol. 255, pp. 457-465, New Delhi, India, 2014.
- [19] M. Kumar and S.C.Sharma, "Priority Aware Longest Job First (PA-LJF) algorithm for utilization of the resource in cloud environment," in *3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 415-420, New Delhi, India, 2016.

- [20] P. Samal and P. Mishra "Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing," *International Journal of Computer Science and Information Technologies*, vol. 4, no. 3, pp. 416-419, 2013.
- [21] X. Zuo, G. Zhang and W. Tan, "Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud," *IEEE Transactions on Automation Science and Engineering*, vol. 11, no. 2, pp. 564-573, April 2014.
- [22] A. Verma and S. Kaushal, "A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling," *Parallel Computing*, vol. 62, pp. 1-19, 2017.
- [23] D. Tang, M. Dai, M. A. Salido and A. Giret, "Energy-efficient dynamic scheduling for a flexible flow shop using an improved particle swarm optimization." *Computers in Industry*, vol. 81, pp. 82-95, Sep. 2016.
- [24] Y. Wang, L. Bin, T. Weise, J. Wang, B. Yuan and Q. Tian, "Self-adaptive learning based particle swarm optimization," *Information Sciences*, vol. 181, no. 20, pp. 4515-4538, Oct. 2011.
- [25] Md. J. Islam, X. Li, and Y. Mei, "A Time-Varying Transfer Function for Balancing the Exploration and Exploitation ability of a Binary PSO," *Applied Soft Computing*, vol. 59, pp. 182-196, 2017.
- [26] K. Suresh and N. Kumarappan, "Hybrid improved binary particle swarm optimization approach for generation maintenance scheduling problem," *Swarm and Evolutionary Computation*, vol. 9, pp.69-89, 2013.
- [27] Chen, Ruey-Maw, et al. "Using novel particle swarm optimization scheme to solve resource-constrained scheduling problem in PSPLIB." *Expert systems with applications* 37.3 (2010): 1899-1910.
- [28] K. M. Cho, P. W. Tsai, C. W. Tsai and C. S. Ynag, "A hybrid meta-heuristic algorithm for VM scheduling with load balancing in cloud computing." *Neural Computing and Applications*, vol. 26, no. 6, pp. 1297-1309, 2015.
- [29] T. S. Somasundaram, and K. Govindarajan, "CLOUDRB: A framework for scheduling and managing High-Performance Computing (HPC) applications in science cloud," *Future Generation Computer Systems*, vol. 34, pp. 47-65, 2014.
- [30] M. Naeem, U. Pareek and D. C. Lee, "Swarm intelligence for sensor selection problems," *IEEE Sensors Journal*, vol. 12, pp. 2577-2585, 2012.

- [31] J. C.W. Lin, L. Yang, P. F. Viger, T. P. Hong and M. Voznak, "A binary pso approach to mine high-utility itemsets," *Soft Computing*, vol. 21, no. 17, pp. 1-19, Mar 2016.
- [32] L. Han, C. Huang, S. Zheng, Z. Zhang and L. Xu, "Vanishing point detection and line classification with bps0," *Signal, Image and Video Processing*, vol. 11, pp. 17-24, 2017.
- [33] J. Kennedy, R. C. Eberhart, A discrete binary version of the particle swarm algorithm, in *International Conference on Systems, Man, and Cybernetics*, volume 5, pp. 4104-4108, 1997.
- [34] J. C. Bansal and K. Deep, "A modified binary particle swarm optimization for knapsack problems," *Applied Mathematics and Computation*, vol. 218, no. 22, pp. 11042-11061, July 2012.
- [35] S. Mirjalili and A. Lewis, "S-shaped versus v-shaped transfer functions for binary particle swarm optimization," *Swarm and Evolutionary Computation*, vol. 9, pp. 1-14, 2013.
- [36] L. Jian-Hua, Y. Rong-Hua, S. Shui-Hua, The analysis of binary particle swarm optimization, *Journal of Nanjing University (Natural Sciences)*, vol. 47, pp. 504-514, 2011.
- [37] H. Izakian, B. T. Ladani, K. Zamanifar and A. Abraham, "A novel particle swarm optimization approach for grid job scheduling," in *International Conference on Information Systems, Technology and Management*, pp. 100-109, Berlin, Heidelberg, 2009.
- [38] W. Li, Wenzheng, and H. Shi, "Dynamic load balancing algorithm based on FCFS," In *Fourth International Conference on Innovative Computing, Information and Control (ICICIC)*, Taiwan, 2009.



CHAPTER 5

MULTI-OBJECTIVE SCHEDULING ALGORITHM USING PSO

The main objective of scheduling algorithm is allocation of applications to cloud resource in such a manner that cloud user complete their tasks in minimum time and cost before the deadline expires. Efficient resources allocation becomes a critical problem for cloud service provider which needs to be resolved because it can greatly reduce the energy consumption in cloud data center.

5.1 Energy Consumption and Execution Cost

Cloud service provider provides the elastic and flexible services to the user [1] due to which rapid growth in demand of computational power and number of users is increasing day by day in cloud environment. Therefore large scale data centers (group of computers) demand is increased and more data centers are adding to the cloud server for better services but these servers consumes huge amount of electric energy. As per the survey, IT infrastructures in USA consumes approximately 61 billion kWh whose cost was 4.5 billion \$ in 2006 [2]. This electricity consumption was the double of electricity consumed in 2000 by IT infrastructure and it is also predicted [3] that cloud data centers consumed .5% of world electricity and it could be four times by 2020 if the same trend continues. However, utilization of cloud data center is approximately 20% to 30% [4] only, i.e., a large amount of energy will be wasted.

There are two main reason of day by day increasing the energy consumption in cloud data center: first one is rapid increase in cloud server as well as customers due to which energy consumption has been increased approximately 56% from 2005 to 2010. To minimize the total energy consumption, the number of active nodes should be reduced and the idle nodes (host) should be turned off. Second reason is that resource is not allocated properly in cloud environment due to which energy consumption has been increased.

There are two main entity works in cloud environment one is cloud services provider and other is customer or user [5]. Cloud service provider contains the huge number of computational resources that are provided to the users to maximize the profit by achieving high resource utilization. On the other hand cloud user want to execute their dynamic applications in minimum time and execution cost. From a provider's perspective, the key issue is to maximize

profits by minimizing the energy consumption and execution costs considering deadline as constraint. There is always trade-off between profit and energy consumption, so trade-off solution is required to solve the problem. Therefore we have developed a scheduling algorithm that optimize the parameters (makespan time, execution cost, task rejection ratio, throughput, energy consumption etc.) based upon the defined fitness function considering deadline as constraint. We are discussing PSO based metaheuristic scheduling algorithm in this chapter.

5.2 Contribution

The goal of cloud service provider is to maximize the profit from cloud infrastructure while cloud users want to execute their applications in minimum time and cost. The rapid growths in demand of computational power tends to massive growth in cloud data centers and require large amount of energy consumption in cloud data centers which becomes a serious threat to the environment. To reduce the energy consumption and gain the maximum profit in cloud computing is a challenging problem.

Special contribution of this chapter:

- We have formulated our multi-objective scheduling problem in the form of mathematical model and defined the objective function & fitness function.
- Resource allocation model has been designed for processing the applications/tasks in efficient way.
- We have modified PSO algorithm that optimizes the parameters (execution time, makespan time, execution cost, energy consumption, task rejection ratio and throughput) based upon the defined fitness function using independent random tasks while considering deadline as constraint. Cloudsim demonstrated that whatever simulation condition is near to the real environment developed algorithm performs better than existing PSO, Honey Bee and Min-Min.

5.3 Related Work and Research Gap

Traditional algorithm in cloud computing mainly focuses on the optimization constrained by time or cost without paying attention to energy consumption. Lots of algorithms have been proposed for resource scheduling in cloud environment. M Kumar and SC Sharma [6-7] proposed algorithms that balance the workload to enhance the utilization of cloud resources and

reduce the makespan time of upcoming task. K. Dubey et al. [8] proposed priority based job scheduling algorithm using the IBA & EASY for cloud metascheduler that reduce the time of job and improve the utilization ratio of cloud resources. M. Malawski et al. [9] developed an algorithm for IaaS cloud to optimize the cost and time considering deadline as constraint. Some more traditional algorithms have been proposed [10-13] but none of the algorithm optimizes time, cost and energy simultaneously in cloud environment as shown in Table 5.1.

Traditional algorithms are not good for finding the optimal solution of the complex problem. Therefore, authors are using metaheuristic algorithms to find out the sub optimal solution (approximate solution) of NP Complete problem in short period of time. L.D Babu and P.V Krishna have proposed an algorithm that reduces the response time and makespan time with considering the priority of tasks as quality of service parameter [14]. E.Pacini et al. [15] proposed cloud scheduler based ant colony optimization (ACO) algorithm to optimize the throughput and response time in cloud environment. Tsai has proposed improved differential evolution algorithm to optimize the execution cost and makespan time parameter [16].

PSO algorithm is used to solve single objective, multi-objective and bi-objective problem [17]. R.Fahimeh et al. [18] have developed task based system for load balancing using particle swarm optimization (TBSLBPSO) algorithm to optimize the execution time and task transfer time considering task migration approach. T.S Somasundaram and K Govindarajan have developed PSO based scheduling algorithm that optimize execution cost, job rejection ratio and time considering deadline as quality of service parameter [19]. A.Verma and S. Kaushal have proposed Bi-Criteria Priority based PSO algorithm that optimize the execution time and execution cost parameter consider deadline as constraint [20]. Netjinda et al. [21] have proposed PSO with variable neighborhood search based algorithm that improve only execution. Yassa et al. [22] have proposed Dynamic Voltage and Frequency Scaling (DVFS) and PSO based scheduling policy (PSO-DVFS) to optimize the energy consumption and execution cost. Best of the author knowledge none of the above papers have considered all three parameters like execution time, energy and execution cost simultaneously in existing literature review as per my knowledge. The work reported in this chapter, considered all three parameter simultaneously and used PSO algorithm to optimize the parameters considering deadline as constraint.

Table 5.1 Literature Review of Traditional and Meta-heuristic algorithm with their limitations

S N.	Year	Technique	Parameters	Tool	Limitations
1	2012 [10]	After predicting the load at node, VM migration technique is used	Time of user request, Resource Utilization, Throughput	Eucalyptus cloud	It is very difficult to predict the future load on the basis of history of load. Energy and execution time is not considered.
2	2012 [11]	Continuously monitor the load at each server	Memory, Response Time	Openstack	It did not consider user priority and Load on a specific server. Execution cost and energy is not considered.
3	2013 [12]	Task migration approach is used after scheduling	Time, Ordinary and VIP resource utilization	Matlab	Rescheduling of task will increase complexity and time. Cost and energy is not considered.
4	2015 [13]	Interval number theory is used to describe uncertainty in cloud	Resource utilization, energy consumption	Cloudsim	Execution cost and time is not considered.
5	2013 [14]	ABC algorithm is used for balancing the workload among the virtual machines.	Execution time, Throughput Response Time	Cloudsim	Algorithm does not work for other QoS parameters like Cost and energy.
6	2015 [15]	ACO is used for scheduling and VM migration approach is used for load balancing.	Throughput, Response time	Cloudsim	ACO is used indirect communication mechanism for information exchange between entities (pheromone), energy and cost is not considered.

7	2013 [16]	Improved DEA algorithm for better exploration and exploitation	Makespan time, execution cost	Cloudsim	Energy consumption is not considered and does not provides satisfactory solution for scalable complex problems
8	2013 [18]	PSO based algorithm is used for multi-objective scheduling using task migration approach.	Task execution time, task transfer time	Cloudsim	Execution Cost and energy consumption is not considered.
9	2014 [19]	Task migration technique is used	Execution time and execution Cost, task rejection ratio	Matlab, Eucalyptus	PSO easily drop in local minima or regional optimum and energy consumption is not considered.
10	2014 [20]	Bi-criteria priority based PSO	Execution cost and execution time	cloudsim	Energy consumption parameter is not considered.
11	2014 [21]	PSO with variable neighborhood search technique	Execution cost	cloudsim	Execution Time and energy consumption parameters are not considered.
12	2013 [22]	Hybrid PSO algorithm with DVFS	Energy consumption and execution cost	cloudsim	Execution time and cost parameters are not considered.
13	2016 [23]	Self-Optimization of Cloud Computing Energy-efficient Resources allocation technique is used	Energy Consumption	Cloudsim	Execution time and execution cost is not considered. Proposed algorithm used heuristic technique to optimize the parameters.

14	2017 [24]	Configuring, Healing, Optimizing and Protecting technique is used for Efficient Resource allocation	Execution cost, time and SLA violation	Cloudsim	Some important QoS parameters Energy Consumption, task rejection ratio and deadline of tasks is not considered in proposed algorithm.
15	2017 [25]	Measure the size of the file and decision is taken based upon threshold values.	Energy Consumption and process time	CPN	Proposed algorithm does not consider execution cost, makespan time and task rejection ratio types of QoS parameters.
16	2017 [26]	Two novel adaptive energy-aware algorithms have been proposed.	Energy consumption and SLA violation	Cloudsim	Execution time, execution cost, task rejection ratio and deadline constraint does not consider in proposed algorithm.
17	2016 [27]	MMGreen framework is proposed for exploiting virtualization technologies	Energy consumption and communication cost	Cloudsim	Execution time, throughput, execution cost etc. QoS parameters are not considered by proposed algorithm.
18	2018 [28]	Adaptive task allocation algorithm is proposed	Energy consumption and makespan time	Cloudsim	Execution cost is not considered and traditional technique is used to optimize the energy and time parameters.
19	2018 [29]	Complete mapping (VM to tasks) algorithm is proposed	Energy consumption and resource utilization	Cloudsim	Execution cost, task rejection ratio, throughput etc. QoS parameters are not considered by proposed algorithm.
20	Our Algo	PSO based approach	Time, cost, energy consumption, throughput, and task rejection ratio.	cloudsim	Algorithm is tested for independent task only.

5.4 Proposed Resource allocation model for Cloud environment

Cloud computing resource allocation model is depicted in Fig. 5.1. This model contains the six components but three are main components: controller node, task scheduler and resource monitoring node. Cloud user submits the applications/jobs (tasks) request \mathcal{T} in cloud environment and specify the requirement of software (Mpich-1.2.7 etc), hardware (memory, CPU etc) and quality of service parameter (deadline, priority, elasticity etc). Each task request contain the deadline $\partial(\mathcal{T}_i)$ that are assigned to them. Each task \mathcal{T}_i is executed with in defined deadline. If large number of tasks exceeds the deadline then task rejection ratio is increased due to increased SLA violation. Therefore we need an algorithm that execute maximum tasks before their deadline i.e. task rejection ratio should be low and throughput of the algorithm should be high. Working of each component of the resource allocation model is given below.

5.4.1 Controller Node

Controller node controls (continuously monitor or interact) all the components of the model like scheduler, resource monitor, performance metrics task model, and programming model. It contains the information about the entire upcoming authentic tasks or applications request that needs cloud resources (virtual machine). Controller node forwards the details of tasks model (no. of required CPU, amount of memory required, bandwidth etc.) and virtual machine (idle, overloaded, under load) to scheduler for scheduling the upcoming requests.

5.4.2 Task Scheduler

Scheduler is responsible for mapping the upcoming tasks (request) with suitable cloud resources (virtual machine). When scheduler receives the details of tasks from task model and resources details from controller node, it starts to map the cloud resource with tasks based upon the following quality of service parameters. Scheduler checks how much total number of tasks are coming in a schedule. After that QoS parameter (priority and deadline) of the upcoming task is checked if upcoming tasks in a schedule contain the deadline then main objective of the scheduler is to schedule all the task in such a manner that maximum task completed their execution before the deadline expired i.e. task rejection ratio should be low so that SLA violation is not increased. If tasks are priority based, executes those tasks at the starting that have high priority and lower priority task is executed at the last. Scheduler use the scheduling algorithm to optimize the parameters like makespan time, execution time, cost, task rejection ratio, throughput, energy consumption based upon the fitness function. Scheduler sent the mapping list (match task with cloud resource) to controller node so that it can update its

information. We have proposed scheduling algorithm in this chapter that assigned the task to virtual machine in effective way and optimize the parameters considering deadline as QoS parameter.

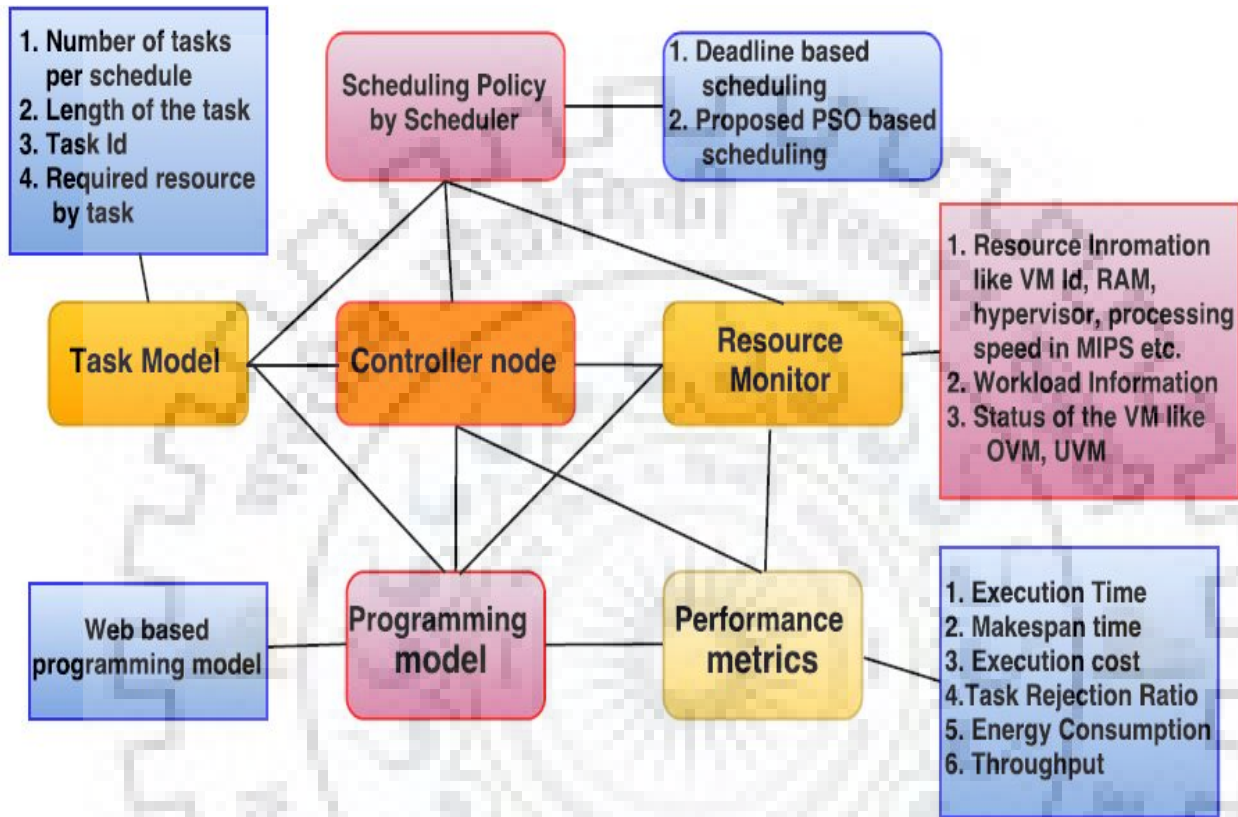


Figure 5.1 Resource allocation model for cloud environment

5.4.3 Resource Monitoring Node

After the allocation of tasks to virtual machine, this node start to monitor the status of each virtual machine either periodically (particular time interval) or event basis and pass to the controller node. If any of the virtual machines hold the load above their threshold limit (threshold limit is defined at the time of SLA) then virtual machine is in overloaded condition. If virtual machine holds the load below the threshold limit then virtual machine is in underloaded condition otherwise virtual machine is in balance condition. Resource monitor node contain all information about the each virtual machine i.e. VM Id, VM processing speed, hypervisor, primary as well as secondary storage memory etc. When task is allocated to a virtual machine, workload at the virtual machine is increased and resource monitoring node update the status of that virtual machine i.e. this virtual machine is able to process more task or

not at that particular moment and forward the complete details to controller node because controller should know the capacity of all virtual machines. On the basis of capacity and current load at the virtual machine it calculates overloaded, underloaded and balanced virtual machine. If large number of virtual machine are in overloaded and underloaded condition or task rejection ratio is high then controller node reschedule the tasks (if enough deadline is available) otherwise calculate the optimize parameter based upon the fitness function.

5.4.4 Performance metrics

It is used to calculate the performance improvements that are gained by proposed resource allocation model in Fig. 5.1. In this chapter, we have evaluated the various performance metrics such as execution time, execution cost, task rejection ratio, energy consumption, throughput consider deadline as constraint.

5.4.5 Programming model

The programming model is helpful for providing the interface to the controller node. Web services based programming model is used for interfacing the controller node with other components of the proposed model.

5.4.6 Task model

It contain the information about the tasks/applications and their requirements. The requirements are in terms of hardware resources and software resources. The upcoming request/ applications can be of many types such as memory intensive, CPU intensive, workflow tasks etc. The Tasks are modeled as

<TaskID, Task length, InputFileID, OutputFileID, required resource by task, QoS ParametersID>

5.5 Problem Statement and Formulation

The aim of PSO based scheduling algorithm is to schedule the upcoming applications/tasks to running resource (virtual machine) in such a manner that cloud user's complete their applications in minimum time (execution as well as makespan time) with minimum energy consumption and reduced execution cost considering deadline as constraint. As per cloud customer's expectation throughput should be high so that maximum task is executed in unit time due to which task rejection ratio will be low i.e. maximum task should execute before the deadline expires. The objective of completing the tasks in minimum time, minimum execution cost and minimum energy consumption becomes a multi-objective and complex problem in

cloud environment. There is always a trade-off between time, cost and energy consumption. To make a balance between execution time, execution cost and energy consumption, a trade-off solution is required. We formulate this multi-objective problem into mathematical form and defined the fitness function as well as objective function for it. Controller node accept n number of tasks request $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3 \dots \mathcal{T}_n$ which are independent in nature. Each task has the task length $\mathcal{L}(\mathcal{T}_i)$ and deadline $\partial(\mathcal{T}_i)$ that are assigned to them. Length of the task is expressed in MI. Every task required ρ_s processing speed, η is the number of required processor, \mathcal{M}_{mem} amount of main memory, s_c storage capacity and \mathcal{B} amount of bandwidth to process the upcoming task. Controller node contain the m number of cloud resource $r_1, r_2, r_3 \dots r_m$ that are heterogeneous in nature in terms of processing speed, number of core, memory, hypervisor etc. If tasks \mathcal{T}_i meet their requirement with resources r_j within defined deadline then value of decision variable $\psi_{\mathcal{T}_i r_j}$ is 1 otherwise value is 0. Symbol notation and its related description are shown in Table 5.2.

5.5.1 Objective function

The main aim of objective function is to maximize or minimize the parameter (Execution cost, energy efficiency, makespan time, task rejection ratio, throughput etc). Cloud service provider wants to minimize time (execution time as well as makespan time) and energy consumption while cloud user needs the services in minimum cost. Therefore, we defined the fitness function whose objective is minimizing the time, execution cost and reducing the energy consumption considering deadline as QoS parameter.

$$\text{Fitness function } f(r_j) = \alpha * EET_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i s_p}} + \beta * EEC_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i s_p}} + \gamma * \mathcal{E}C_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i s_p}} \quad (1)$$

$$\alpha + \beta + \gamma = 1 \quad (2)$$

Where the value of α , β , and γ is lies between 0 to 1 i.e. $0 < \alpha < 1$, $0 < \beta < 1$, $0 < \gamma < 1$ and represent the weight assigned to execution time, execution cost and energy consumption.

(a) Execution Time

Our problem is based upon the multi-objective function therefore first objective is defined to minimize the total processing time of tasks $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \dots \mathcal{T}_n$ at virtual machine $r_1, r_2, r_3 \dots r_m$ in a schedule \mathcal{S}_p . First objective function is defined as

$$\text{Total processing time } (TPT_{\mathcal{T}_i}^{r_j}) = EET_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i s_p}} \quad (3)$$

$$\text{Formula to calculate the Execution time (ET) for task is } = \frac{\mathcal{L}(\mathcal{T})}{\rho_s * \eta} \quad (4)$$

Notation and their description are used in problem formulation and fitness function is shown in Table 5.2.

Notations	Description
$\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3, \dots, \mathcal{T}_n,$	Set of task request in a schedule \mathcal{S}
$r_1, r_2, r_3 \dots r_m$	Available cloud resources (number of virtual machine)
$\partial(\mathcal{T}_i)$	Deadline of the Task
$w_{\mathcal{T}}$	Window of tasks (number of task in a schedule)
\mathcal{S}_p	p^{th} schedule of upcoming workload
$\theta_{\mathcal{T}_i r_j}$	Matched resources list of tasks \mathcal{T}_i in schedule \mathcal{S}_p ($i = 1$ to n)
$TPT_{\mathcal{T}_i}^{r_j}$	Total processing time of tasks \mathcal{T}_i at cloud resources r_j
$ET_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}}$	Execution time of task \mathcal{T}_i on matched resources r_j
$EET_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}}$	Excepted execution of task \mathcal{T}_i at cloud resources r_j
$TTT_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}}$	Task Transfer time of task \mathcal{T}_i on resource r_j in the matched cloud resource
$\psi_{\mathcal{T}_i r_j}$	Binary decision variable such that $\psi_{\mathcal{T}_i r_j} = 1$ if \mathcal{T}_i is allocated to resource r_j
X_{ijk}	If task i is allocated to resource r_k from r_j then value is 1 otherwise 0
$WT_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}}$	Time spend by task \mathcal{T}_i in job queue before assigned to resource r_j in schedule
$TEC_{\mathcal{T}_i}^{r_j}$	Total cost of tasks $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3 \dots \mathcal{T}_n$ at cloud resources $r_1, r_2, r_3 \dots r_m$
C_{r_j}	Cost of virtual machine r_j per hour basis
$EC_{\mathcal{T}_i r_j}$	Execution cost of task \mathcal{T}_i at resource r_j
$TTC_{\mathcal{T}_i r_j}$	Task transfer cost from one resource to other resource
$EEC_{\mathcal{T}_i r_j}$	Excepted execution cost of tasks at resources
$PT_{\mathcal{T}_i}^{\rho_j}$	Processing time of task \mathcal{T}_i at virtual machine ρ_j
M_i	Memory required by task $\mathcal{T}_i r_j$
W_{r_j}	Workload at resource r_j
$\mathcal{E}C_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}}$	Energy consumption by resources r_j for tasks \mathcal{T}_i in schedule \mathcal{S}_p
$\mathcal{E}C_{\mathcal{M}ax}$	Energy consumption when resource is completely utilized (approximately 100%)
$\mathcal{E}C_{\mathcal{M}in}$	Energy consumed by resources when they are ideal or low utilization (0 to 5 % only)

There are many task in a schedule \mathcal{S}_p so calculate the execution time of tasks (\mathcal{T}_i) at virtual machine r_j in cloud environment using the equation 5.

$$\text{Execution time } ET_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} = \sum_{i=1}^n \psi_{\mathcal{T}_i r_j} * \frac{L(\mathcal{T}_i)}{p_{s r_j} * \eta} \quad (5)$$

Task is allocated based upon the scheduling algorithm but there are two problems can occur after scheduling: one is scheduling algorithm can allocate more task to some virtual machine and less task to other virtual machine i.e. some virtual machine may be in overloaded or underloaded condition, depends upon the capacity of the virtual machine. Second one is, if scheduling algorithm allocate same number of task to all the running virtual machine but upcoming tasks or applications are in heterogeneous nature so it may be possible that all the high computation-intensive tasks is allocated to some virtual machine and low intensive tasks is allocated to other machine i.e. again there is possibility of overload and underloaded of virtual machine in cloud environment. Therefore to solve such types of problem, transfer the task from overloaded to underloaded virtual machine. Task transfer time and capacity of virtual machine is calculated using the equation 6 & 7.

$$TTT_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} = \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^m X_{ijk} * \frac{VDE_{jk}}{B_{jk}} \quad (6)$$

$$Cap_{r_j} = p_s * \eta + B_{r_j} \quad (7)$$

Where VDE_{jk} represent the volume of data (may be more than one task) exchange between virtual machine r_j & r_k and can be calculate using the formula

$$VDE_{jk} = \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^m X_{ijk} * L(\mathcal{T}_i) \quad (8)$$

Where Cap_{r_j} represent the capacity of virtual machine and W_{r_j} represent the available workload at virtual machine. Equation 9 ensures that total load assigned at virtual machine should not be more than the capacity of virtual machine. Constraint (10) ensures that ψ_{ij} is a binary variable it has the value either 0 or 1.

$$\sum_{i=1}^n \psi_{ij} W_{r_j} \leq Cap_{r_j}, \quad \forall i \in n \ \& \ \forall j \in m \quad (9)$$

$$\psi_{ij} \in [0,1] \quad \forall i, j \quad (10)$$

$$\text{Waiting time } (WT_\rho) = ST_i - TA_i \quad (11)$$

$$\text{Where } |\theta_{\mathcal{T}_i \mathcal{S}_p}| \leq m \quad (12)$$

$EET_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}}$ is the sum of execution time ($ET_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}$) task transfer time ($TTT_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}$) and waiting time ($WT_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}$) where $WT_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}$ is the time wait by the processor in job queue before assigned the task \mathcal{T}_i to resource r_j in schedule \mathcal{S}_p as shown in equation 11. Waiting time should be minimum for better performance. Where $\mathcal{J}A_i$ task arrival time and $\mathcal{S}\mathcal{T}_i$ is the start execution time of task \mathcal{T}_i at VM. We have assumed that all virtual machine are in running/active state. Matched resource list generated by scheduler (in equation 12) for each task \mathcal{T}_i consist of less than or equal to total number of running virtual machine (available cloud resources).

(b) Execution Cost

The goal of second objective function is to optimize the total execution cost ($TEC_{\mathcal{T}_i}^{r_j}$) of tasks \mathcal{T}_i at resources r_j in particular schedule \mathcal{S}_p . We defined the second objective function in equation 13 that reduce the execution cost

$$\text{Min } TEC_{\mathcal{T}_i}^{r_j} = EEC_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} \quad (13)$$

$$EC_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} = ET_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} * C_{r_j} \quad (14)$$

$$TTC_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} = TTT_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} * C_{r_j} \quad (15)$$

$$EEC_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} = EC_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} + TTC_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} \quad (16)$$

Where C_{r_j} represent the execution cost of cloud resources per hour basis. Resource may be memory intensive or CPU computation intensive. Here we are using two types of CPU intensive resources one is high computation intensive other is low computation intensive.

(c) Energy consumption

The objective of the cloud service provider is to reduce the energy consumption. Energy model is presumed on the basis that resource utilization has a linear relationship with energy consumption [23]. Energy consumption is represented by the equation 17.

$$\mathcal{E}C_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} = \mathcal{E}C_{DC} + \mathcal{E}C_{\mathcal{T}ransceivers} + \mathcal{E}C_{Memory} + \mathcal{E}C_{Extra} \quad (17)$$

Where $\mathcal{E}C_{DC}$ represents energy consumption by datacenters, energy consumption by switching equipment is represented by $\mathcal{E}C_{\mathcal{T}ransceivers}$, energy consumption by memory (primary as well as secondary) is represented by $\mathcal{E}C_{Memory}$ and energy consumption by other devices like fans,

current conversion loss etc. is represented by $\mathcal{E}C_{\text{extra}}$. Maximum energy (approximately 90%) is consumed by the computing devices specially CPU. $\mathcal{E}C_{t,i}(r)$ represents the energy consumption at given time t as shown in equation 18.

$$\mathcal{E}C_{t,i}(r) = q * \mathcal{E}C_{\text{Max}} + (1 - q) * \mathcal{E}C_{\text{Max}} * \mathcal{R}_u \quad (18)$$

$\mathcal{E}C_{\text{Max}}$ represent the maximum energy consumption when cloud resources is 100% utilized (fully utilized), if resource is in idle condition then it consumed fraction of energy that is represented by q . Completely idle servers consumed approximately 70% of their peak power. \mathcal{R}_u represent utilization of cloud resources. Therefore we need an efficient scheduling algorithm that distributes the tasks in effective way so that resource utilization should be maximum. \mathcal{R}_u Change over the time when task is allocated to virtual machine or de-allocated from virtual machine. For any resource r_j at a given time t, resource utilization (ResU_t) is defined in equation 19.

$$\text{ResU}_t = \sum_{i=1}^n \mathcal{E}C_{t,i}(\mathcal{R}_u(t))dt \quad (19)$$

Where n is the total number of tasks running at time t in a schedule. Actual energy consumption is calculated by equation 20.

$$\mathcal{E}C_{\text{actual}} = (\mathcal{E}C_{\text{max}} - \mathcal{E}C_{\text{min}}) * \text{ResU}_t + \mathcal{E}C_{\text{min}} \quad (20)$$

Where $\mathcal{E}C_{\text{max}}$ is energy consumption when resource is 100% utilized. $\mathcal{E}C_{\text{min}}$ is minimum energy consumption when resource is in ideal condition or low utilized.

Subject to:

$$r_j \in \theta_{\mathcal{J}_i \mathcal{S}_p}, \quad \psi_{\mathcal{J}_i r_j} = 1 \quad (21)$$

$$r_j \notin \theta_{\mathcal{J}_i \mathcal{S}_p}, \quad \psi_{\mathcal{J}_i r_j} = 0 \quad (22)$$

$$PT_{\mathcal{J}_i}^{r_j} > TA_i + ET_{\mathcal{J}_i r_j} \quad \forall i \in n \ \& \ \forall j \in m \quad (23)$$

$PT_{\mathcal{J}_i}$ is the processing time of task \mathcal{J}_i at cloud resource r_j . TA_i is task arrival time, if it is known and certain then problem is static otherwise problem is dynamic. $ET_{\mathcal{J}_i r_j}$ is the execution time of task \mathcal{J}_i at resource r_j . Equation 23 indicates that a task can't be started before its time.

$$PT_{i,j} \geq PT_{i-1,j} + ET_{\mathcal{J}_i r_j} \quad (24)$$

Equation 24 represent that a task start to execute at a resource only if when previous task has been completed on that particular resource.

$$\sum_{j=1}^m \psi_{i,j} = 1, \quad \forall i \in n \ \& \ \forall j \in m \quad (25)$$

Equation 25 represent that a task is allocated to only one virtual machine.

$$\sum_{i=1}^n \psi_{i,j} RM_i \leq \mathcal{M}_{memj}, \quad \forall i \in n \ \& \ \forall j \in m \quad (26)$$

RM_i represent the memory required by task \mathcal{T}_i at virtual machine r_j and equation 26 state that available memory of virtual machine should be larger than the memory requested by tasks. User submits the tasks with their deadline in cloud environment. Goal of service provider is to execute maximum tasks before the deadline so that task rejection ratio as well as SLA violation remains minimum. The time required to complete the task on available cloud resource is expressed in equation 27 and task will be assigned to resource that satisfied the condition shown in equation 28. Initial workload $W_1, W_2, W_3 \dots W_m$ at virtual machine is 0 i.e. W_{r_j} is 0 at the starting.

$$RT_{\mathcal{T}_i r_j} = \partial(\mathcal{T}_i) - W_{r_j} \quad (27)$$

$$ET_{\mathcal{T}_i r_j} \leq RT_{\mathcal{T}_i r_j} \quad (28)$$

Here W_{r_j} represent the available workload at virtual machine before assigned the task \mathcal{T}_i .

Workload W_j at virtual machine r_j is calculated by equation 29 [14].

$$W_{\mathcal{T}_i} = \frac{\text{Number of task } \mathcal{T}_i \text{ at } r_j}{\text{Service rate of } r_j} \quad (29)$$

Service rate of virtual machine \mathcal{T}_i . can be defined in equation 30

$$S = p_s * \eta \quad (30)$$

Total load (L) at all available virtual machine can be calculated by equation 31

$$L = \sum_{j=1}^m W_{r_j} \quad (31)$$

After allocate the tasks to virtual machine workload at virtual machine is increased and calculate it by equation 32.

$$W_{r_j} = W_{r_j} + ET_{\mathcal{T}_i r_j} \quad (32)$$

Throughput of the system is calculated using the equation 33

$$\text{Throughput } (\Gamma) = \frac{\text{number of tasks completed successfully}}{\text{Total processing time}} \quad (33)$$

Suppose 100 tasks are completed in 80sec then throughput of the system is $100/40=2.5$

Task unable to meet deadline (Task rejection ratio) is also measure the performance of the algorithm. It is calculated by equation 34

$$\text{Task rejection ratio } (\mathcal{T}_i) = \frac{\text{No.of rejected tasks}}{\text{Total number of tasks}} * 100 \quad (34)$$

Total time taken by a virtual machine r_j to complete the entire assigned task is calculated using the equation 35

$$MST_{r_j} = \sum_{T_i \in \theta_{r_j}} EET_{T_i r_j} \quad (35)$$

Makespan time or total time to complete all the tasks are successfully in a schedule is represented by equation 36. To calculate the makespan time select the resource (virtual machine) among all the running virtual machine that has the maximum execution time after the completion of all tasks

$$\text{Makespan} = \max \{MST_{r_j}\} \quad (36)$$

5.6 Particle Swarm Optimization (PSO)

PSO is population based optimization algorithm which was proposed by Eberhart and Kennedy [30] in 1995 that mimic the behavior of animal swarm, such as bird flock and fish school which each member in a swarm is called a particle. PSO have been applied in many research and scientific application (model classification, function optimization, machine study, neural network training etc.) due to distinguishing characteristics [31-32] like as (i) Consist of limited number of parameters, there is no need to calculate the overlapping and mutation (ii) simple and easy enumeration, (iii) it is attractive because there are few parameters to adjust, (iv) being free from derivation, (v) sensitivity move towards the fitness function and parameters, (vi) less dependency at initial parameter, (vii) relatively faster convergence and cheaper way rather than other meta-heuristic algorithm like GA, ABC, ACO etc. (h) high precision solutions. Population in PSO algorithm is defined as the total number of particles in a problem space D and population is initialized randomly. Position of each particle represents a possible solution. Every particle is associated with corresponding velocity that helps the particle to move on to best position based upon the own experience and experience of its neighbors. Velocity of each particle is updated in each time step to find out the two best positions personal best p^{Best} and global best g^{Best} . PSO equation (37) represents that particle movement for each iteration t

$$V_{id}^{t+1} = w * V_{id}^t + c_1 r_1 (p^{Best}_{id}^t - X_{id}^t) + c_2 r_2 (g^{Best}_{id}^t - X_{id}^t) \quad (37)$$

Where t+1 represent the current instruction, i represent the number of particles and d denotes the dimension of the particle in PSO algorithm. c_1 is cognitive learning factor and c_2 is social interaction coefficient. r_1 and r_2 are random number in the range between 0 to 1 in t^{th} iteration. w is Inertia factor that is used to maintains the balance between exploration and exploitation.

Velocity $V_i^{t+1}(j)$ is bounded by threshold limit (velocity range should not be outside the search space) shown in equation 38.

$$V_{id}^{t+1} = \begin{cases} V_{\max}, & \text{if } (V_{id}^{t+1}) > V_{\max} \\ V_{\min}, & \text{if } (V_{id}^{t+1}) < V_{\min} \end{cases} \quad (38)$$

After updating the velocity, particle position is updated using velocity equation and each value of new position should not exceed the range of $[X_{\max}, X_{\min}]$

$$X_{id}^{t+1} = X_{id}^t + V_{id}^{t+1} \quad (39)$$

p^{Best} of the particle is updated using the equation 40.

$$p_{id}^{Best^{t+1}} = \begin{cases} X_{id}^{t+1}, & \text{if } f(X_{id}^{t+1}) < p_{id}^{Best^t} \\ p_{id}^{Best^t}, & \text{otherwise} \end{cases} \quad (40)$$

Fitness function f returns the optimal g^{Best} fitness value. After finding the $p_{id}^{Best^{t+1}}$ of each particle, we find $g_{id}^{Best^{t+1}}$ by comparing all the calculated value of $p_{id}^{Best^{t+1}}$.

5.6.1 PSO working methodology

PSO working principle is divided into four steps: initialization, update the particle velocity and position, fitness function calculation and optimal solution as shown in Fig. 5.2 [21]. Firstly

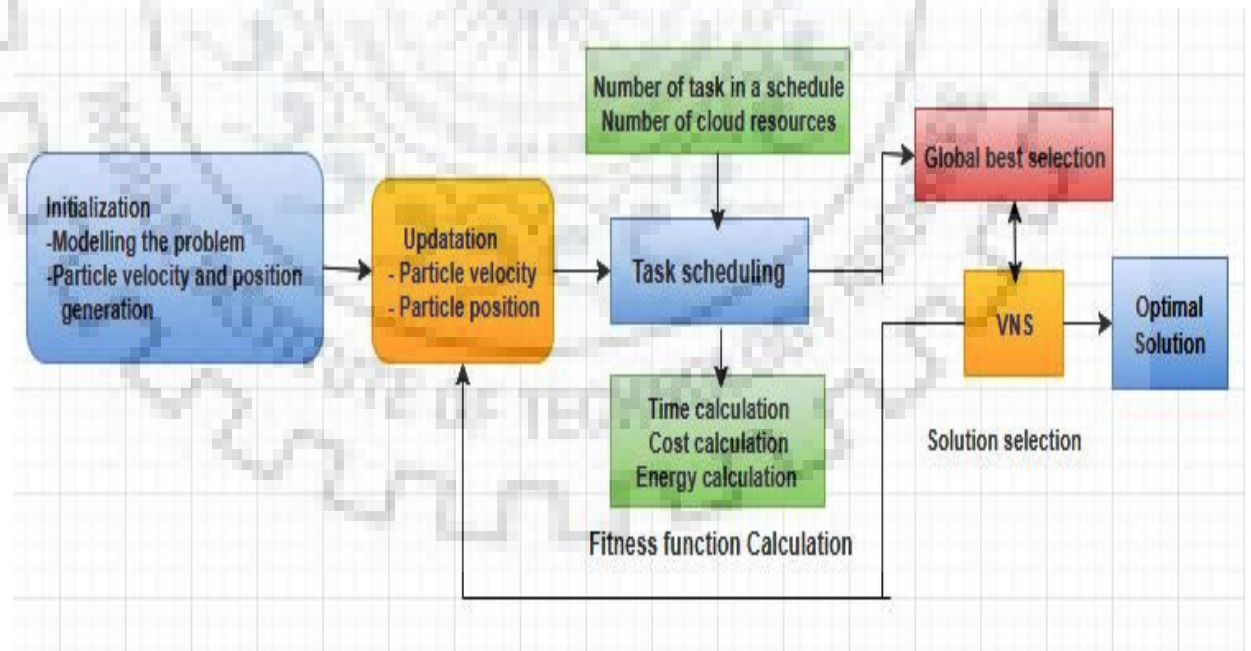


Figure 5.2 PSO working methodology

particle is designed to represent all constraints that are used to calculate the value of fitness function. After that velocity and position of the particle is initialized randomly. Velocity and

position of the particle is updated in each iteration based upon the value of fitness function and comparing with their neighbor. Tasks are schedule by PSO algorithm and calculate the value of fitness function is calculated based upon the number of upcoming task in a schedule and available cloud resources. Tasks are independent in nature. Last process is to select the optimal solution of the problem using PSO algorithm. Each particle has contained p^{Best} but aim of the algorithm is to find the global or optimal solution g^{Best} . Therefore variable neighbor search (VNS) approach is used to find the optimal solution. VNS process improve the fitness function through flow back of algorithm goes to second step and this process is continue until given number of iteration has been finished.

5.7 Modified proposed PSO

To find the optimal solution of NP-Complete problem (like task scheduling, 0-1 knapsack, airplane scheduling, travelling salesman etc.) using the PSO algorithm required two factor in efficient way that affect the performance of the algorithm, one is to maintain the proper balance between exploration and exploitation for finding the optimal solution rapidly. Another one is jump out of local optima in case of premature convergence.

5.7.1 Exploration and Exploitation

Exploration mechanism searches the whole space rather than specific regions and help the algorithm to find out the optimal solution. Exploitation focuses on retrieving the best solution from a specific area of the search space. The traditional inertia weight adaptation mechanism does not provide a good balance between exploration and exploitation. To improve the search capability, velocity updation strategy is used in this chapter. Hence we created the velocity randomly in the range V_{min} to V_{max} at the starting. Time should be longer for stronger search capability at the early stage when particle keep high velocity and time should be also longer at the latter stage when particle keep smaller velocity. Search capability is improved (better exploration) when particles keep high velocity at the early stage and the search accuracy is also improved (stronger exploitation) while particles keep smaller velocity at the latter stage. This approach avoids the prematurity and divergence types of problems from PSO to a large extent. We are using APSO-VI algorithm to modify the PSO that provides nonlinear ideal average velocity to control the search process [33]. Firstly we calculate average absolute velocity of particle using equation 41

$$V_{avg}(t) = \frac{1}{n.d} \sum_{i=1}^n \sum_{j=1}^d V_{id}^t \quad (41)$$

Where V_{id}^t represents the velocity of i^{th} particle and V_{avg} is average absolute velocity. Size of the particle search space is reflected by the size of average velocity. A high velocity of particle implies larger search space of population and possibility of strong exploration ability to keep high diversity and avoid the local optimal trap while low velocity of particle implies smaller search space of population and strong exploitation ability to improve the accuracy of the solution. Ideal average velocity is defined in equation 42

$$V_{ideal}(t) = V_{start} * \frac{1 + \cos(t\pi/T_{end})}{2} \quad (42)$$

Where V_{start} represents the starting ideal velocity of particle and we calculate using the equation 43

$$V_{start} = (X_{max} - X_{min})/2 \quad (43)$$

X_{max} and X_{min} represent the maximum and minimum value of search space. V_{ideal} is ideal average velocity of particle.

$$T_{end} = .95 * t_{max} \quad (44)$$

Where t_{max} represents the maximum number of iterations.

Value of w is depend upon the values of V_{avg} and V_{ideal} . If value of V_{avg} is greater than or equal to the value of V_{ideal} then w switches to low value otherwise w switches to higher value.

$$\text{If } V_{avg}(t) \geq V_{ideal}(t+1) \text{ then } w(t+1) = \text{maximum}(w(t) - \delta w, w_{min}) \quad (45)$$

$$V_{avg}(t) < V_{ideal}(t+1) \text{ then } w(t+1) = \text{minimum}(w(t) + \delta w, w_{max}) \quad (46)$$

Where $w_{max} = 0.9$, $w_{min} = 0.3$ and δw is 0.1.

5.7.2 Flow chart of modified PSO algorithm

We have added constraint in proposed PSO based algorithm keeping in mind that if position of the particle is exceed the defined boundary. Further one more constraint is added in this proposed PSO algorithm, task should be assigned to each virtual machines from the entire task window (schedule) i.e. virtual machine should not be in ideal condition like existing PSO based algorithm [18] in Table 4 where $vm2$ is in ideal mode. Flowchart of the modified PSO algorithm is shown in Fig. 5.3. The steps of modified PSO algorithm are as follows.

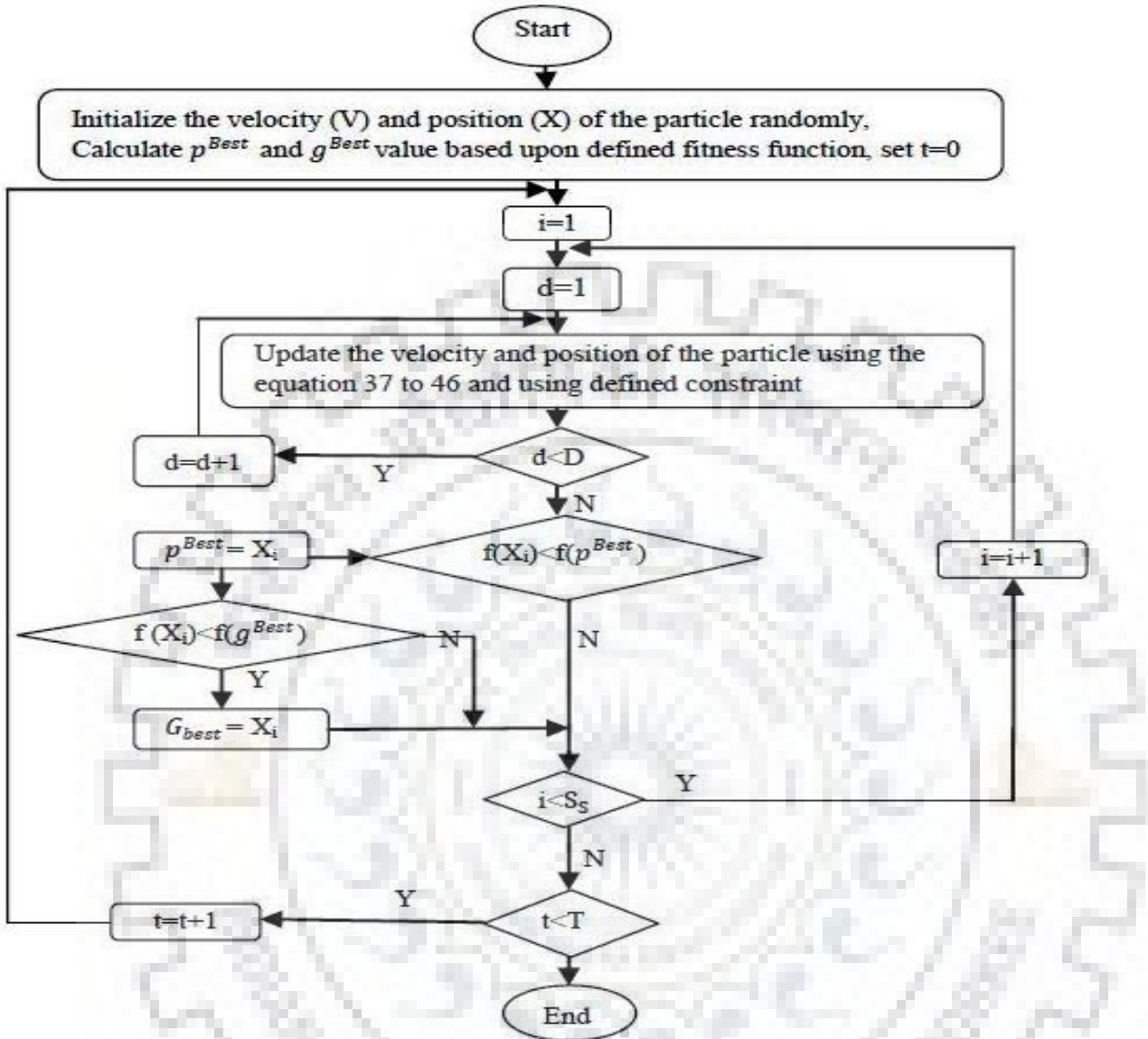


Figure 5.3 Flowchart of the modified PSO (proposed) algorithm

- a. Generate the initial population
- b. Position and velocity of each particle is generated randomly.
- c. For $i=1$ to population size
- d. Calculate the p^{Best} and g^{Best} based upon fitness function where p^{Best} represent the best position of the particle achieve by itself and g^{Best} denote the global best (optimal value) among all the particles.
- e. For $t=1$ to t_{max} (maximum iteration)
- f. For $i=1$ to population size
- g. Calculate V_{avg} and V_{ideal} using the equation 41 & 42

- h. Calculate the value of inertia weight w using the equation 45 & 46.
- i. Update the velocity of particle (V_{id}^{t+1}) using ($V_{id}^t, p^{Best}, g^{Best}$)
- j. Update the position $X_{id}^{t+1} = X_{id}^t + V_{id}^{t+1}$
- k. If current fitness value is less than ($f(X_{id}^{t+1}) < p_{id}^{Best^t}$) the particles best previous fitness value then update $p_{id}^{Best^{t+1}} = X_{id}^{t+1}$
- l. If current fitness value (g^{Best}) of the particle is better than all its neighborhood fitness function value then update the g^{Best} . g^{Best} is final optimal solution of the problem.

5.7.3 Proposed PSO algorithm for multi-objective scheduling

Standard encoding scheme of PSO cannot be applied in scheduling problem directly. Therefore we need to find suitable mapping between scheduling problem and positions of particles. There are some key steps to model our problem with PSO: first one is how to relate (encoded) or map the problem of scheduling (NP Complete problem) with meta-heuristic optimal searching algorithm like PSO. Particle is generated randomly in PSO but question is that what a particle represents in optimal task scheduling problem and how it map with the problem. Last one is how it optimizes the defined fitness function and objective function.

Therefore we are relating our problem with the PSO algorithm and defining the meaning and dimension of the particles. Each particle is representing the window of task (collection of tasks or workflow) and dimension of the particle is represented by total number of tasks in a window. Position of the particle is defined by coordinate system that is determined by particle dimension. The dimension of a particle will determine the coordinate system used to define its position in space. Position of the 1 D particle is specified by 1 coordinate, 2D particle is specified by 2 coordinate, 5D particle is specified by 5 coordinate and so on. A particle contains the 10 tasks shown in Table 5.3 i.e. particles is 10 D and position of the particle is determined by ten coordinates. Cloud resources (VM) are available to process the tasks and determine the range of particle movement in search space. Particle coordinate range value is equal to number of available cloud resource in the pool. Example of particle position encoding and mapping tasks with available recourses is shown in Table 5.3 & Table 5.4. The value of each coordinate denotes the resource index in particle position Table 5.3 and represent the cloud resource is assigned to the tasks by that particular coordinate. Hence Table 5.3 provide

the encode mechanism to map the upcoming tasks with cloud resources. This mapping (tasks with resources) is shown in Table 5.4.

Table 5.3 Position of the particles

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

Table 5.4 Mapping task with cloud resources

$T_1 \rightarrow r_2$	$T_2 \rightarrow r_3$	$T_3 \rightarrow r_4$	$T_4 \rightarrow r_2$	$T_5 \rightarrow r_2$	$T_6 \rightarrow r_1$	$T_7 \rightarrow r_3$	$T_8 \rightarrow r_0$	$T_9 \rightarrow r_4$	$T_{10} \rightarrow r_0$
-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	-----------------------	--------------------------

We have created 5VM which have the coordinate value from 0 to 4 and particle coordinate represent the corresponding index of the tasks. Coordinate C_1 represent the task 1 and value of the coordinate C_1 is 2 indicate that task T_1 is allocated to resource r_2 Coordinate C_2 represent the task 2 and value of the coordinate C_2 is 3 indicate that task T_2 is allocated to resource r_3 . Coordinate C_3 represent the task 3 and value of the coordinate C_3 is 4 indicate that task T_3 is allocated to resource r_4 . On the same way process is continues until the entire coordinate has not been finished. The main objective of proposed PSO algorithm is to find the best mapping between task and resource that can optimize the defined fitness function and produce the optimal solution of scheduling problem. Particle try to find out new solution at each iteration (based upon new velocity and position) if previous is poor (not optimal) than new solution. Proposed PSO algorithm is shown in Fig. 5.4.

Brief explanation of the proposed PSO algorithm is follows:

- Particles position and velocity are initialized randomly with in defined range.
- Generated workload for each schedule S_p
- Generated virtual machine with different processing capability.
- Once controller node obtained the information about tasks and resources, proposed algorithm schedule the tasks based upon the fitness function.
- Calculate the value of p^{Best} based upon fitness function, if calculated value of p^{Best} is optimal then replace new calculated value of p^{Best} with previous value of p^{Best} .

Proposed PSO based multi-objective scheduling algorithm

1. **Input:** Number of tasks in a schedule \mathcal{S}_p and available cloud resources (virtual machine)
2. **Output:** Optimize the parameters time, cost and energy based upon the defined fitness & objective function
3. **Start**
4. S_S =Swarm Size (number of particle in population)
5. Number of iteration is represented by t
6. V_R =Range of Random Velocity
7. V_{id}^t represents the Particle velocity
8. X_{id}^t Represent the Position of Particle
9. P_R =Random position of the particle
10. p^{Best} represents the personal best or local best of the particle
11. g^{Best} represents the global best of the particle among all the particle
12. **For** i=1 to S_S //initialize the particle of the swarm
13. **do**
14. $V_{id}^t \leftarrow V_R$ (randomly initialize the velocity of particles)
15. $X_{id}^t \leftarrow P_R$ (randomly initialize the position of particles)
16. Generate the schedule \mathcal{S}_p that contain the tasks $\mathcal{T}_1, \mathcal{T}_2, \mathcal{T}_3 \dots \mathcal{T}_n$ of random length with defined range and deadline of the tasks $\partial(\mathcal{T}_i)$ is also generated.
17. Generate the cloud resources $r_1, r_2, r_3 \dots r_m$ that contain different MIPS.
18. Schedule the task \mathcal{T}_1 to resources r_j
19. $\theta_{\mathcal{T}_1 \mathcal{S}_p} \leftarrow (EET_{\mathcal{T}_1 r_j \in \theta_{\mathcal{T}_1 \mathcal{S}_p}}, EEC_{\mathcal{T}_1 r_j \in \theta_{\mathcal{T}_1 \mathcal{S}_p}}, \mathcal{E}C_{\mathcal{T}_1 r_j \in \theta_{\mathcal{T}_1 \mathcal{S}_p}})$
20. Check the condition, task is allocated to all the virtual machine or not,
21. If any virtual machine r_j is ideal then reschedule the task, otherwise go to step 23
22. Calculate p^{Best} based upon fitness function using the equation 1
23. $f(r_j \in \theta_{\mathcal{T}_1 \mathcal{S}_p}) \leftarrow \alpha * EET_{\mathcal{T}_1 r_j \in \theta_{\mathcal{T}_1 \mathcal{S}_p}} + \beta * EEC_{\mathcal{T}_1 r_j \in \theta_{\mathcal{T}_1 \mathcal{S}_p}} + \gamma * \mathcal{E}C_{\mathcal{T}_1 r_j \in \theta_{\mathcal{T}_1 \mathcal{S}_p}}$
24. $p^{Best} \leftarrow X_{id}^t$
25. Repeat the while loop until maximum iteration ($t_{max} \neq \emptyset$)
26. **do**
27. **For** i=1 to population size
28. Calculate V_{avg} and V_{ideal} based upon the formula given in equation 41 & 42
29. Calculate the value of inertia weight w using the equation 45 & 46 based upon the value of V_{avg} and V_{ideal}
30. $p^{Best^{t+1}} \in r_j \leftarrow \min(f(X_{id}^{t+1}), p^{Best^t})$
31. Calculate $\mathcal{RT}_{\mathcal{T}_i r_j} = \partial(\mathcal{T}_i) - W_{r_j}$
32. **if** $ET_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}} \leq RT_{\mathcal{T}_i r_j \in \theta_{\mathcal{T}_i \mathcal{S}_p}}$ then
33. Assigned the task \mathcal{T}_i to resources r_j in schedule \mathcal{S}_p and workload at the virtual machine increase
34. $W_{r_j} = W_{r_j} + ET_{\mathcal{T}_i r_j}$
35. **if**
36. $f(g^{Best}) \geq f(p^{Best})$
37. then
38. $g^{Best} \leftarrow p^{Best}$
39. Return the value of g^{Best}
40. Velocity of the particle is updated using equation 37
41. $V_{id}^{t+1} = w * V_{id}^t + c_1 r_1 (p^{Best^t} - X_{id}^t) + c_2 r_2 (g^{Best^t} - X_{id}^t)$
42. $X_{id}^t(k+1) \leftarrow$ Update the particles position from (V_{id}^t, X_{id}^t)
43. Repeat the step until all the task has not been finished in schedule
44. **End**

Figure 5.4 Steps of proposed PSO based algorithm for scheduling algorithm

- Resource monitor node monitor the status of the virtual machine continuously and forward the information to controller node that decide task would be allocated to resource or not.
- Calculate the value of V_{avg} and V_{Ideal} using the equation and define the new value of w for better exploration and exploitation.
- If execution time of the task is less than defined deadline then controller node allocates the task to resources.
- Our aim is to find out optimal g^{Best} based upon the defined fitness function. Compare the fitness value of p^{Best} with g^{Best} . If fitness value of p^{Best} is better than g^{Best} then assigned it to g^{Best} .
- This process is continuing until all tasks have not been schedule to available virtual machine in optimal way or terminates the loop of max iteration.
- Proposed PSO scheduling algorithm optimize the parameters execution time, makespan time, task rejection ratio, execution cost, energy consumption and throughput while considering deadline as constraint.

5.8 Analysis, Comparison and Simulation Results

We choose cloudsim simulator to test the performance of the developed PSO based algorithm. There are some features provided by cloudsim like as, it provides modeling and simulation environment for large scale cloud computing including datacenters, host and virtual machines of different configuration etc. cloudsim contains the service broker that is used for provisioning and de-provisioning of cloud resources.

Table 5.5 VM properties

VM Id	VM MIPS	VM Image size	Memory	No. Of cpu	Hypervisor
0	280	1000	512	1	kvm
1	230	1000	1024	1	kvm
2	200	1000	512	1	kvm
3	300	1000	1024	1	kvm
4	250	1000	1024	1	kvm

Virtual machine is a processing entity in cloud environment that is controlled by hypervisor such as KVM, XEN etc. Each virtual machine (VM) contains the parameter like VM Id, CPU

speed in MIPS, and number of core per CPU, image size of virtual memory, main memory and hypervisor as shown in Table 5.5. Further we have generated cloudlet (Task) with their parameters like Task Id, Length (random task length of range 100000 MI to 300000 MI).

5.8.1 Execution Time Calculation

One aim of the developed PSO based algorithm is to minimize the execution time of tasks submitted to the cloud resource allocation model. We have analyzed and investigated the performance of the developed algorithm by varying the number of tasks and virtual machines in each schedule, range of the virtual machine MIPS is 200 to 300. Further generated 10 tasks randomly of length range 100000 MI to 300000 MI and all the tasks are independent in nature. Range of the tasks is extended from 10 to 100 and virtual machines are extended from 5 to 50 to analysis (test) the performance of the developed algorithm. Simulation has been run for more than 200 times at different number of task with random length and results are found using the space shared policy [34] in cloudsim. A task of window is sending continuously after a time interval for analysis and test the performance of the developed algorithm. Simulation details of task, virtual machine, PSO parameters and calculated execution time is shown in Table 5.6.

Table 5.6 Detail of task, VM, PSO parameters and calculated execution time of algorithms

Schedule	VM	Task	No. of Particle	Iteration	Exe_Time Proposed PSO	Exe_Time PSO[18]	Exe_Time HoneyBee[14]	Exe_Time Min-Min[12]
S_1	5	10	50	100	111.19045	115.3014	123.5464	129.15554
S_2	5	20	50	500	234.6504	242.9382	254.26564	269.75087
S_3	10	20	100	500	236.6916	251.14016	268.12436	279.80435
S_4	10	30	100	500	356.3048	372.7792	390.3942	413.10694
S_5	10	50	100	500	527.5738	555.5568	567.73287	594.95942
S_6	20	50	100	500	528.3474	557.1864	566.51344	593.55664
S_7	50	100	100	500	1149.1836	1196.3745	1295.7964	1308.3122
S_8	100	200	100	500	2417.9718	2492.6994	2566.4415	2581.6713
S_9	200	500	200	500	6297.8423	6428.9696	6548.7741	6629.79664
S_{10}	400	1000	200	500	12821.208	12972.158	13294.727	13322.452

Execution time is increased as number of task is increased in cloud environment. Calculated execution time shown in Fig. 5.5 is the average execution time of 10 schedules. To test the

performance of the developed algorithm it is compare with meta-heuristic algorithm PSO [18], honey bee [14], traditional algorithm min-min [12] and it is observed that calculated execution time by developed modified PSO based algorithm is approximately less than 4% from PSO [18] algorithm, approximately less than 10% from honey bee and approximately 20% less than from min-min algorithm for 10 tasks. Same procedure is repeated for all the tasks and observed that PSO [18] algorithm take approximately 4% to 6% more time, honey bee [14] take approximately 10% to 15% more time and min-min[12] algorithm take approximately 14% to 20% more time to execute the tasks rather than developed modified PSO based algorithm.

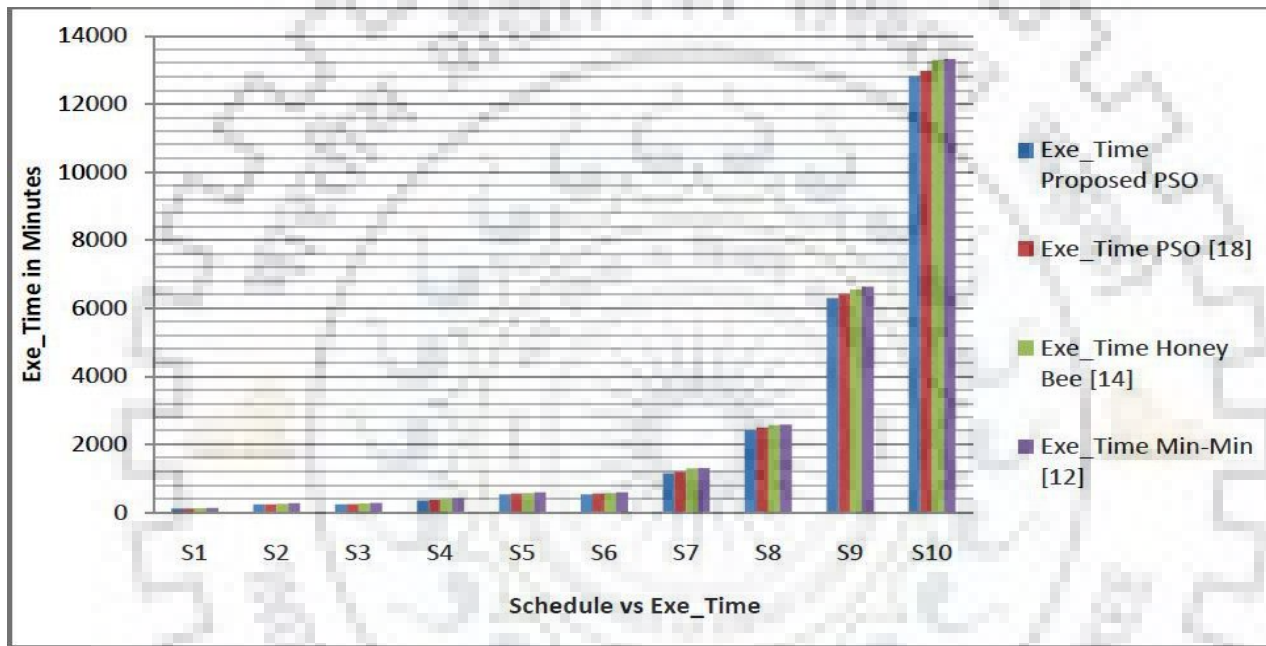


Figure 5.5 Execution time comparisons proposed PSO vs. PSO, Honey Bee and Min-Min

5.8.2 Makespan Time calculation

We have generated fourteen different schedules to analysis the performance of the developed PSO based algorithm. Number of virtual machine range is extended from 10 to 500 and task range is extended from 50 to 1000 in generated schedules. Different value of PSO parameters (like swarm size and iteration) is taken for each schedule as shown in Table 5.7. When number of tasks is increase at the virtual machines then makespan time is also increase. Fig. 5.6 represents the makespan time comparison between developed PSO based algorithm and other algorithms like PSO [18], Honey Bee [14] and min-min [12] for fourteen schedules. Computational results proved that developed algorithm reduce the makespan time of tasks up to 10% from existing PSO, 20% honey bee, 18.8 % min-min algorithms.

Table 5.7 Detail of task, VMs, PSO parameters and calculated makespan time of algorithms

VM	VM	Task	No. of Particle	Iteration	Proposed PSO	PSO[17]	Honey Bee[16]	Min-Min[7]
S_1	10	50	50	100	71.0964	74.2468	78.1547	82.3564
S_2	20	50	50	100	45.96	47.7966	51.17904	54.1328
S_3	50	100	50	500	48.5857	50.2848	55.9772	56.8553
S_4	50	200	50	500	83.5127	86.85174	92.1375	94.1355
S_5	50	500	50	500	186.2388	192.10043	206.8644	211.9728
S_6	50	1000	50	500	350.627	356.122	372.446	378.921
S_7	100	200	50	500	56.7412	58.70482	64.0732	68.4352
S_8	100	500	100	500	119.427	123.142	132.7132	136.779
S_9	100	1000	100	500	203.529	208.762	222.852	220.117
S_{10}	200	500	100	500	79.7782	81.778	88.9321	87.5852
S_{11}	200	1000	100	1000	129.6398	132.6382	139.924	144.462
S_{12}	300	1000	100	1000	102.2971	104.9928	112.2031	118.6642
S_{13}	400	1000	100	1000	86.994	89.741	96.0332	99.8841
S_{14}	500	1000	100	1000	78.886	80.470	87.298	93.207

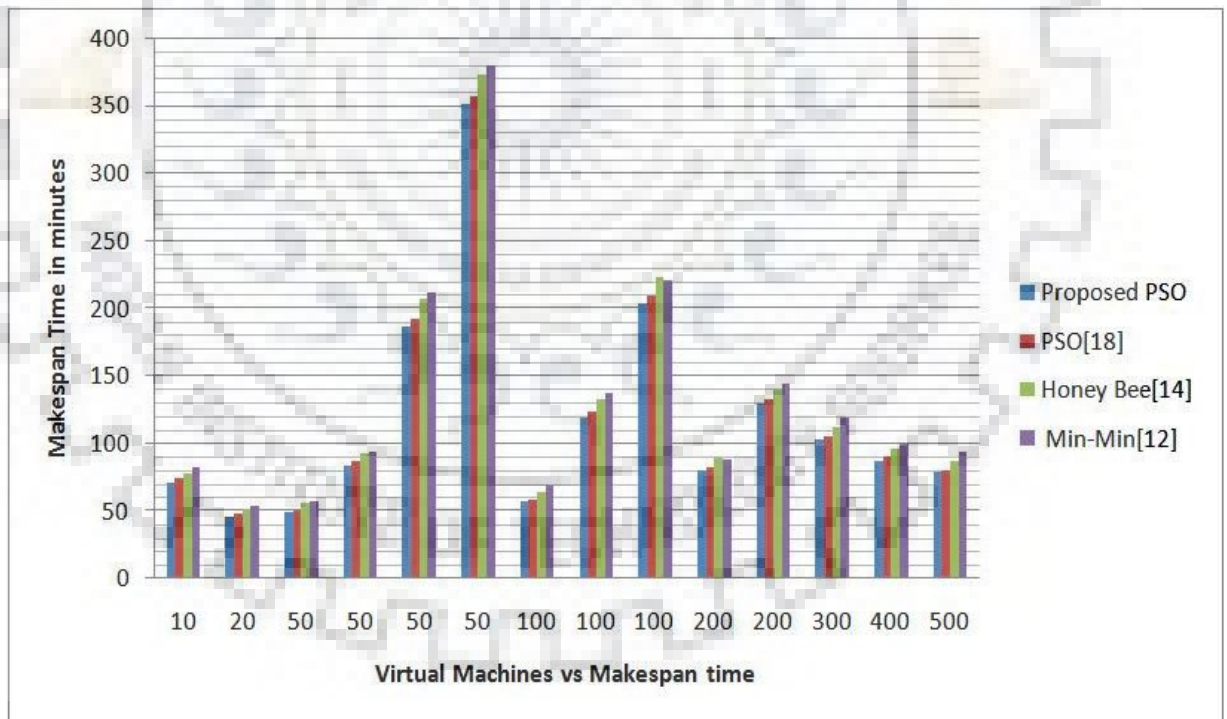


Figure 5.6 Makespan time comparisons between proposed PSO vs. PSO, Honey Bee and Min-Min algorithm

5.8.3 Task rejection ratio

Task rejection ratio is calculated using the equation 34. It is the ratio of number of task getting rejected versus total number of tasks submitted to the controller node per schedule. If a task is spending more time in waiting queue than its deadline then task is rejected and task rejection ratio is increased i.e. SLA violation is increase. Waiting time of the task should not be more than its deadline hence it decrease the SLA violation. Waiting time of the task is calculated using the equation 11.

Table 5.8 Detail of task, VM, PSO parameters and calculated task rejection ratio of algorithms

Schedule	VM	Task	No. of Particle	Iteration	Proposed PSO	PSO[18]	Honey Bee[14]	Min-Min[12]
S_1	20	100	100	100	10.600	11.00	12.80	12.40
S_2	100	500	100	300	17.799	18.528	20.04	21.86
S_3	200	1000	200	500	18.700	19.799	21.928	23.453
S_4	300	1000	200	500	7.5999	8.1923	9.887	9.4862
S_5	400	1000	200	500	3.2996	3.600	4.8829	5.0470
S_6	500	1000	200	500	1.29942	1.6749	1.8742	1.8711

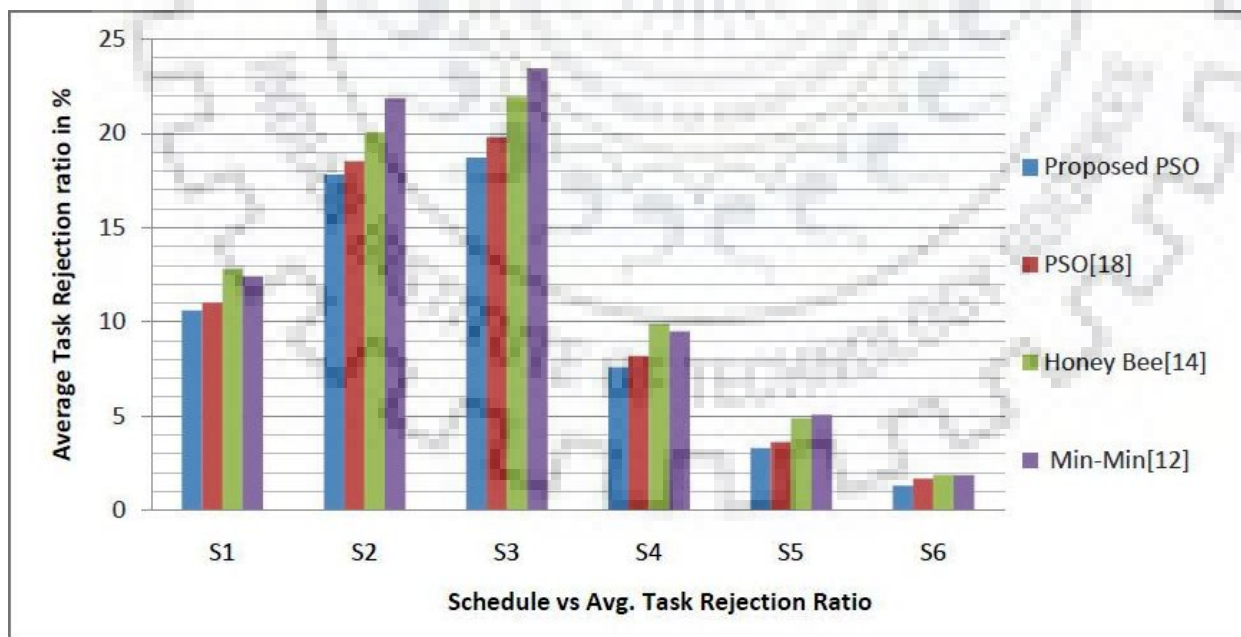


Figure 5.7 Task rejection ratio comparisons between proposed modified PSO vs. PSO, Honey Bee and Min-Min algorithm

Task rejection ratio is calculated with the help of equation 27 & 28. We have generated six different schedules to calculate the task rejection ratio and analyze the performance of the algorithm. Deadline of the tasks is generated randomly within the range 3000 to 5000 second. Table 5.8 represent the parameters taken in each schedule to test the performance of the developed modified PSO algorithm and comparison the performance of the algorithm with other existing algorithms. As number of tasks is increases at the virtual machine task rejection ratio is also increase because possibility of occurring of overloaded or underloaded of virtual machine is increased. Task processing time of overloaded machine is more than the balanced virtual machine because task transfer time is also added in processing time of overloaded machine. Therefore processing time becomes more than the deadline value and task rejection ratio is increases. Calculated results shown in Fig. 5.7 prove that developed modified PSO based algorithm has lowest task rejection ratio compare to others algorithms ((up to 8% from existing PSO, 23% honey bee, 19.6 % min-min algorithms)) in all the schedules.

5.8.4 Execution cost calculations

The objective of modified PSO based algorithm is to schedule the tasks at cloud resource in such a manner that execution cost is minimum. There is always a conflict between time and cost. Our proposed research work solves the trade-off issue between time and cost by assigning an appropriate weighting to α , β and γ .

Table 5.9 Low computation-intensive virtual machine cost details

VM MIPS	CPU	RAM	Storage	Cost
200-250	1 vCPU	1 GB	2 GB	.25\$/hour

Table 5.10 High computation-intensive virtual machine cost details

VM MIPS	CPU	RAM	Storage	Cost
250-300	1 vCPU	4 GB	10 GB	.5\$/hour

Divided the virtual machines instance in two groups (low computational and high computational) based upon the configuration of virtual machines as shown in Table 5.9 and Table 5.10. Fourteen different schedule are generated to analyzed and test the performance of the developed algorithm in cloud environment shown in Table 5.11. It has been observed that developed PSO based algorithm execute the task of each schedule in lowest cost compare to

other existing algorithm [12] [14][18] (up to 8% from existing PSO, 20% honey bee, 25% min-min algorithms) as shown in Fig. 5.8. PSO based algorithm [18] allocates the task to the resources but don't given the guarantee of optimal task allocation.

Table 5.11 Simulation details of Task, VM, no. of particle, iteration and calculated execution cost of algorithms

Schedule	VM	Task	No. of Particle	Iteration	Proposed PSO	PSO[17]	Honey Bee[16]	Min-Min [7]
S_1	10	50	50	100	4.25	4.50	5.00	5.25
S_2	20	50	50	100	5.25	5.75	6.50	7.25
S_3	50	100	50	500	12.50	13.50	14.75	16.50
S_4	50	200	50	500	20.00	21.25	24.50	25.25
S_5	50	500	50	500	40.75	41.50	46.75	48.50
S_6	50	1000	50	500	77.50	80.75	87.25	89.50
S_7	100	200	50	500	28.75	30.25	33.25	36.75
S_8	100	500	100	500	51.50	53.75	57.75	56.25
S_9	100	1000	100	500	89.50	92.25	98.75	101.25
S_{10}	200	500	100	500	69.0	72.25	78.75	81.25
S_{11}	200	1000	100	1000	105.50	108.75	114.50	118.00
S_{12}	300	1000	100	1000	125.50	128.50	136.0	135.75
S_{13}	400	1000	100	1000	143.75	147.25	158.25	162.50
S_{14}	500	1000	100	1000	157.75	163.25	172.25	177.50

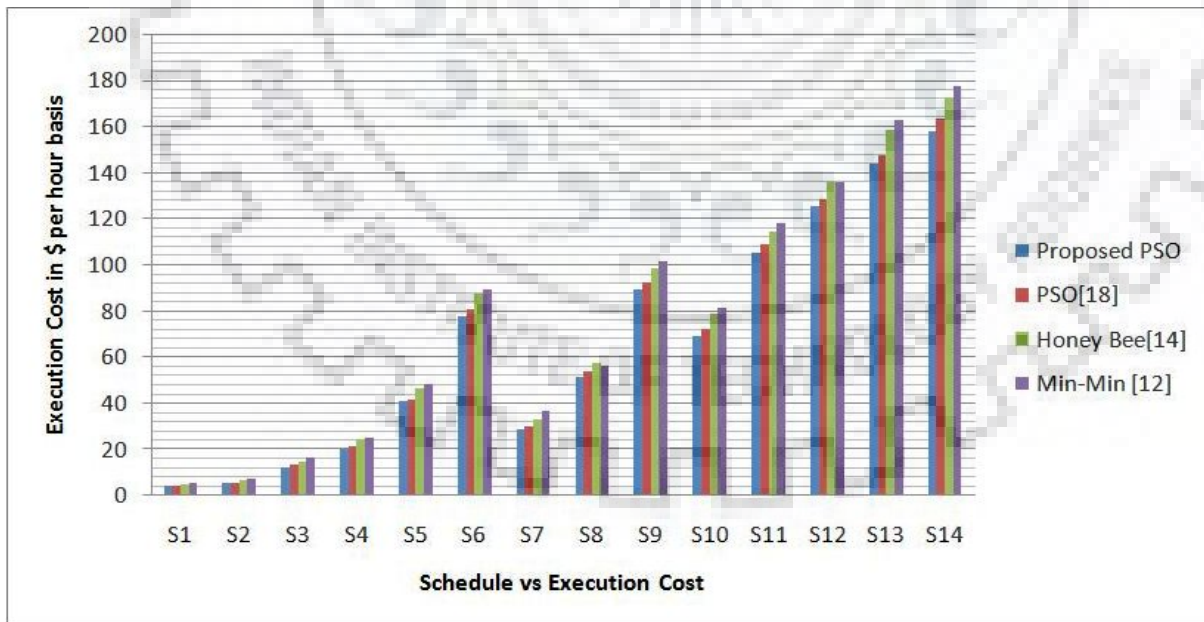


Figure 5.8 Execution cost comparison between proposed PSO vs. PSO, Honey Bee and Min-Min algorithm

There are some virtual machines which are in ideal condition (vm2 in Table 4) due to which

execution cost is increased. Further we have created six different schedules to solve the trade-off issue between time and cost as shown in Table 5.12. Makespan time and execution cost is calculated at fixed number of tasks (1000 tasks) using proposed modified PSO algorithm where range of virtual machines is vary from 50 to 500. It is observed from Fig. 5.9 proved that with the increase number of virtual machines makespan time is decreasing but execution cost is increasing. Therefore trade-off is required between both the parameters i.e. 250 numbers of virtual machines is required to execute the 1000 tasks.

Table 5.12 Simulation details of Task, VM, calculated makespan time and execution cost

Schedule	Virtual Machines	Number of Tasks	Makespan Time	Execution Cost
S_1	50	1000	350.627	77.5
S_2	100	1000	203.529	89.5
S_3	200	1000	129.6398	105.5
S_4	300	1000	102.2971	125.5
S_5	400	1000	86.994	143.75
S_6	500	1000	78.886	157.75

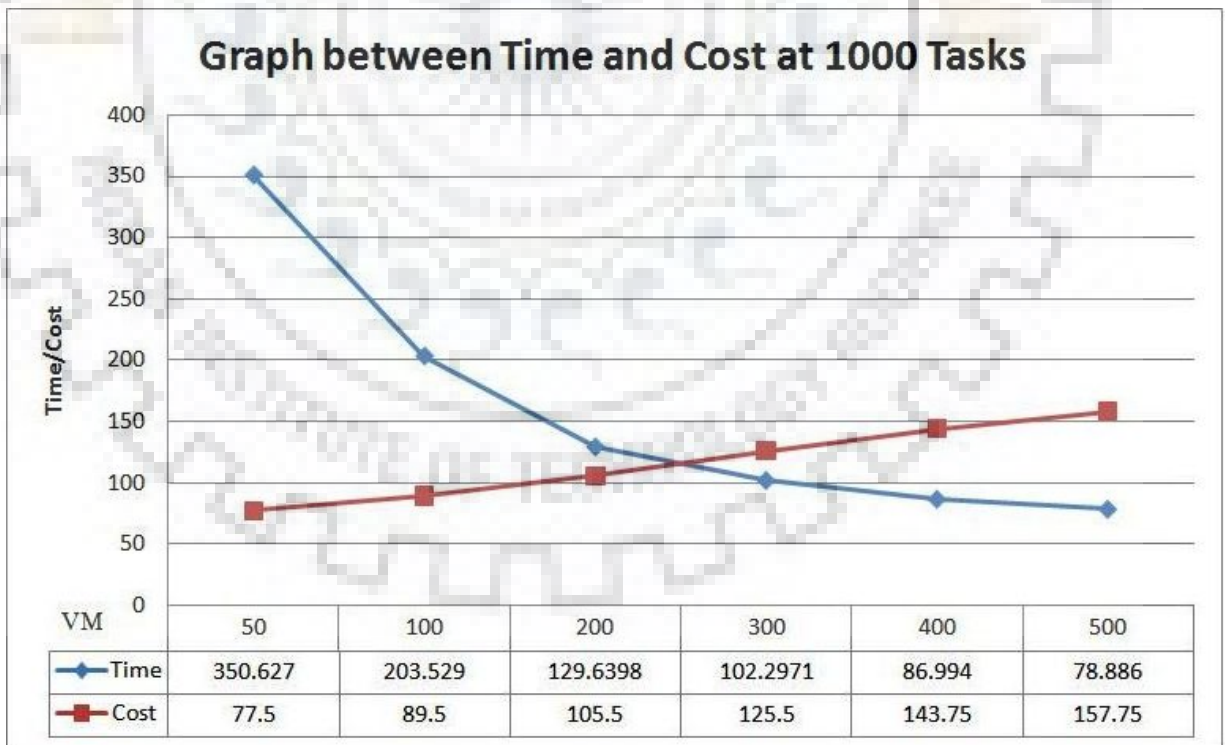


Figure 5.9 Results of makespan time and execution cost at 1000 tasks using proposed PSO

5.8.5 Throughput

We have analyzed and investigated the performance of the developed PSO based algorithm for the parameter throughput. It is calculated using the equation 33. Ten different schedules are generated to test the task execution rate per hour of the algorithms as shown in Table 5.13. Calculated results (shown in Fig. 5.10) proved that developed modified PSO based algorithm outperforms than other existing algorithm in the paper [12][14][18].

Table 5.13 Simulation details to calculate throughput of the algorithms

Schedule	VM	Task	No. of Particle	Iteration	Throughput Proposed PSO	Throughput PSO[18]	Throughput Honey Bee [14]	Throughput Min-Min[12]
S_1	5	10	50	100	5.40549	5.21739	4.85672	4.64558
S_2	5	20	50	500	5.11399	4.93694	4.72440	4.44856
S_3	10	20	100	500	5.08472	4.77897	4.48776	4.28944
S_4	10	30	100	500	5.05617	4.82871	4.61438	4.35729
S_5	10	50	100	500	5.69259	5.40540	5.29100	5.05050
S_6	20	50	100	500	5.67294	5.38212	5.30147	5.06829
S_7	50	100	100	500	5.22193	5.01672	4.63320	4.58715
S_8	100	200	100	500	4.96483	4.81540	4.67653	4.64936
S_9	200	500	200	500	4.76417	4.66708	4.58774	4.52556
S_{10}	400	1000	200	500	4.67982	4.62534	4.51331	4.50382

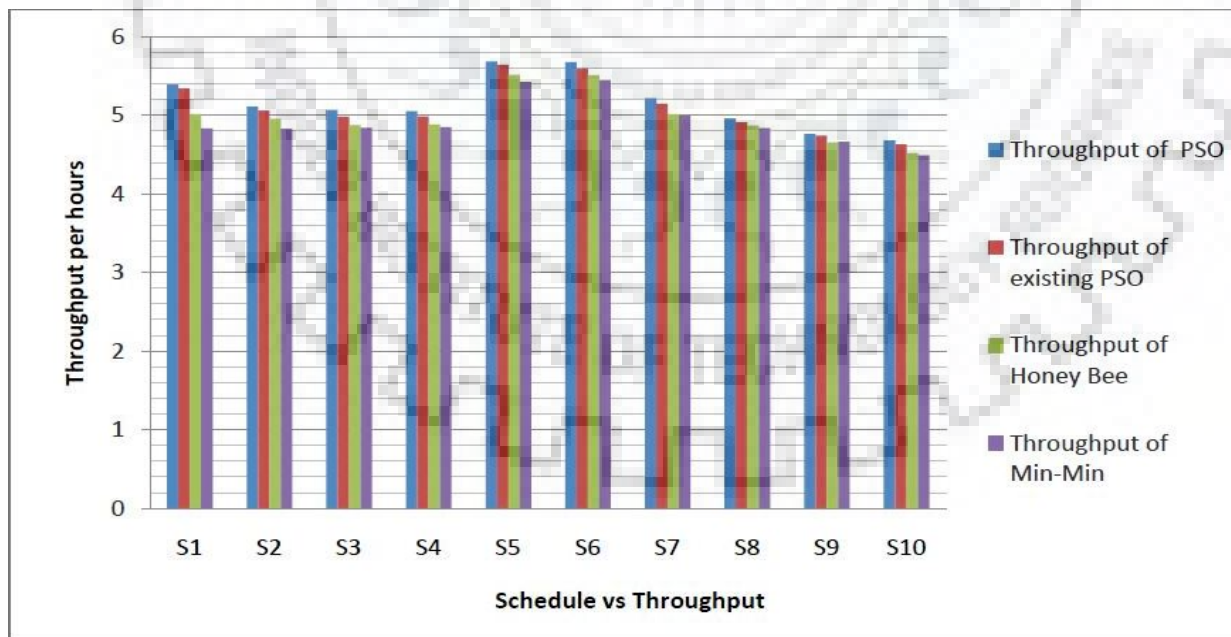


Figure 5.10 Throughput comparisons between proposed PSO vs. PSO, Honey Bee and Min-Min algorithm

Each schedule contains the number of task and range of task is extended from 10 to 1000. Developed PSO based algorithm executes more tasks in an hour's (average 5.4 task per hour) compare to others algorithm PSO [18] execute 5.21 tasks per hour, honey bee execute 4.85 tasks per hour and min-min execute 4.64 tasks per hour for the workload 10 tasks. We have calculated throughput for all the tasks and found that developed PSO based algorithm execute approximately 5.2 tasks per hours while PSO [18] execute 4.9 tasks per hours, honey bee execute approximately 4.7 tasks per hour and min-min execute 4.6 tasks per hour. When number of tasks is increase in cloud environment throughput is slightly decrease because processing time is increase. Table 5.13 and Fig. 5.10 results prove that developed PSO based algorithm has better throughput than other existing algorithm in the entire schedule and all the condition.

5.8.6 Energy consumption

Cloud computing data centers consume huge amount of electrical energy due to which cost and CO_2 emissions is increasing day by day. Energy consumption is a challenging research problem in the field of cloud environment. To reduce the energy consumption, we choose the workstation (server) configuration from Table 1[35] HP ProLiant G4 and HP ProLiant G5 and create the 4 virtual machine in each workstation.

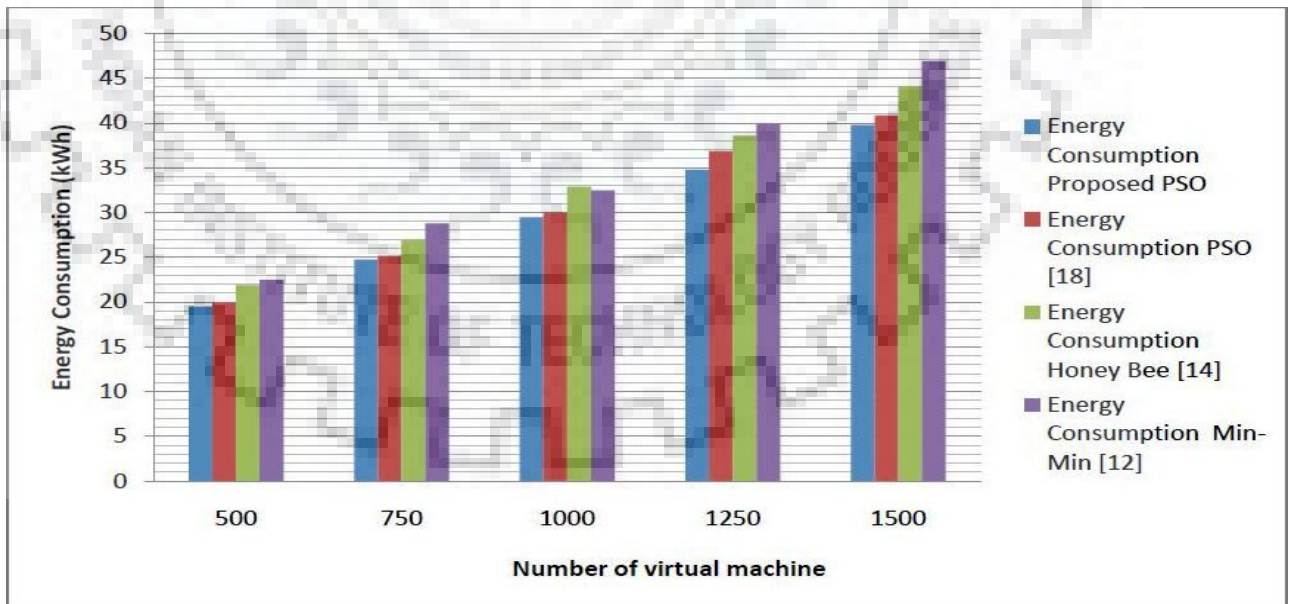


Figure 5.11 Energy Consumption comparisons between proposed PSO vs. PSO, Honey Bee and Min-Min algorithm

Resource utilization based energy consumption description is taken from Table2 [35]. There is always trade-off between profit and energy consumption therefore trade-off solution is required to solve this issue. Resource utilization should be maximum to reduce the energy consumption in cloud environment because ideal resource consumes approximately 70% energy of total utilization of resources.

We have calculated execution time of each virtual machine, after that we find the makespan time of each virtual machine. Resource utilization of each virtual machine is calculated with the help of makespan time. Further to calculate the energy consumption 5000 tasks of random length between 100000MI to 300000MI are generated and 500 virtual machines of different configuration are generated to process the tasks in cloud environment with the help of PSO parameters (number of particle is 100 and number of iteration is 1000). Energy consumption is increased with the increase in number of virtual machine as shown in Fig. 5.11. Number of virtual machine is increased from 500 to 1500 to analyze the performance of the developed PSO based algorithm in five different schedules. Calculated results (shown in Fig. 5.11) proved that developed modified PSO algorithm utilize the cloud resource maximum and reduce the energy consumption in comparison to existing algorithm PSO [18] (up to 7%), Honey Bee [14] (up to 12 %), min-min [18] (up to 18%).

5.9 Summary

There are a few task scheduling algorithms in cloud environment that considered either execution cost or execution time or both the parameters but none of the algorithm considered time, cost and energy simultaneously. In this chapter, we proposed an efficient scheduling technique which play an important role in minimizing the energy consumption and optimize the others parameters like execution cost and time. There is always a trade-off between profit and energy consumption, so a trade-off solution is required. We have designed and discussed the resource allocation model with its components for cloud environment. We developed modified PSO based scheduling algorithm that schedule the tasks at cloud resources in efficient way and optimize the parameters (execution time, makespan time, task rejection ratio, throughput, execution cost and energy consumption) based upon fitness function considering deadline of tasks as quality of service parameter. Modified PSO based algorithm is tested at cloudsim simulator and experimental results shown in Figs. 5.5 to Fig. 5.11 which proved that developed algorithm decrease the execution time, optimize the execution cost and minimize the energy

consumption in efficient way rather than existing PSO, Honey Bee, min-min algorithm in all the conditions. The proposed algorithm can also be tested in future by using a private cloud like OpenNebula.

5.10 References:

- [1] M. Kumar, K. Dubey and S.C. Sharma, "Elastic and flexible deadline constraint load Balancing algorithm for Cloud Computing," in *Procedia Computer Science*, vol. 125, pp. 717-724, India, 2018.
- [2] US EPA ENERGY STAR Program, Report to congress on server and data center energy efficiency, Public law, pp. 109–431, 2007.
- [3] W. Forrest, How to cut data centre carbon emissions? Website, December 2008. Available: <http://www.computerweekly.com/Articles/2008/12/05/233748/how-to-cut-data-centre-carbon-emissions.htm>.
- [4] L. Barroso and U. Holzle, "The case for energy proportional computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.
- [5] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "Recent advancements in resource allocation techniques for cloud computing environment: a systematic review," *journal of Cluster Computing*, vol. 20, no. 3, pp. 2489-2533, Dec. 2016.
- [6] M.Kumar and S. C. Sharma, "Priority Aware Longest Job First (PA-LJF) algorithm for utilization of the resource in cloud environment," in *International conference on Computing for Sustainable Global Development (INDIACom)*, pp. 415-420, New Delhi, India, 2016.
- [7] M. Kumar and S. C. Sharma, "Dynamic load balancing algorithm for balancing the workload among virtual machine in cloud computing," in *Procedia Computer Science*, vol. 115, pp.322-329, Cochin, India, 2017.
- [8] K. Dubey, M. Kumar and M. Chandra, "A Priority Based Job Scheduling Algorithm Using IBA and EASY Algorithm for Cloud Metascheduler," in *International Conference on Advances in Computer Engineering and Applications*, pp. 66-70, Ghaziabad, India, 2015.
- [9] M. Malawski, G. Juve, E. Deelman and J. Nabrzyski, "Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds," in *International Conference on High Performance Computing, Networking, Storage and Analysis*, USA, Nov. 2012.

- [10]. H. Ren, Y. Lan and C. Yin, "The load balancing algorithm in cloud computing environment," in *International Conference on Computer Science and Network Technology*, pp. 925-928, Changchun, China, Dec. 2012.
- [11]. J. Bhatia, T. Patel, H. Trivedi and V. Majmudar, "HTV Dynamic Load Balancing Algorithm for Virtual Machine Instances in Cloud," in *International Symposium on Cloud and Services Computing*, pp. 15-20, Mangalore, KA, 2012,.
- [12]. H. Chen, F. Wang, N. Helian and G. Akanmu, "User Priority Guided Min-Min Scheduling Algorithm For Cloud Computing," in *national conference on Parallel Computing Technologies (PARCOMPTECH)*, pp. 1-8, Bangalore, India, Oct. 2013.
- [13]. H. Chen, X. Zhu, H. Guo, J. Zhu, X. Qin and J. Wu, "Towards energy-efficient scheduling for real-time tasks under uncertain cloud computing environment," *Journal of System Software*, vol. 99, pp. 20–35, Jan. 2015.
- [14]. D. Babu and P. Venkata, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol.13, no. 5, pp. 2292–2303, May 2013.
- [15]. E. Pacini, C. Mateos and C. G. Garino, "Balancing throughput and response time in online scientific Clouds via Ant Colony Optimization (SP2013/2013/00006)", *Advances in Engineering Software*, vol. 84, pp. 31-47, June 2015.
- [16]. J. T. Tsai, J. C. Fang and J. H. Chou, "Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm," *Computer Operation Research*, vol. 40, no. 12, pp.3045-3055, Dec. 2013.
- [17]. M. Masdari, F. Salehi, M. Jalali, and M. Bidaki, "A survey of PSO based scheduling algorithms in cloud computing," *Journal of Network and System Management*, vol. 25, no. 1, pp. 122–158, Jan. 2016.
- [18]. F. Ramezani and F. K. hussain, "Task-based System Load Balancing in cloud computing using Particle Swarm Optimization," *International Journal of Parallel Programming*, vol. 42, no. 5, pp. 739-754, Oct. 2013.
- [19] T. Somasundaram and K. Govindarajan, "CLOUDRB: A framework for scheduling and managing High-Performance Computing (HPC) applications in science cloud," *Future Generation Computer System*, vol. 34, pp. 47-65, Oct. 2014.
- [20]. A. Verma, S. Kaushal, "Bi-Criteria Priority based Particle Swarm Optimization workflow scheduling algorithm for cloud," in *Recent Advances in Engineering and Computational Sciences (RAECS)*, pp. 1–6, Mar. 2014.

- [21]. N. Netjinda, B. Sirinaovakul and T. Achalakul, "Cost optimal scheduling in IaaS for dependent workload with particle swarm optimization," *Journal of Supercomputing*, vol. 68, no. 3, pp. 1579–1603, 2014.
- [22]. S. Yassa, R. Chelouah, H. Kadima and B. Granado, "Multi-objective approach for energy-aware workflow scheduling in cloud computing environments," *The Scientific World Journal*, 2013. doi:10.1155/2013/350934.
- [23] S. Singh, I. Chana, M. Singh and R. Buyya, "SOCCER: self-optimization of energy-efficient cloud resources," *Cluster Computing*, vol. 19, no.4, pp. 1787–1800, 2016.
- [24] S. S. Gill, I. Chana, M. Singh and R. Buyya, "CHOPPER: an intelligent QoS-aware autonomic resource management approach for cloud computing," *Cluster Computing*, pp. 1-39, 2017.
- [25] R. Aldmour, S. Yousef, M. Yaghi, S. Tapaswi, K. K. Pattanaik, and M. Cole, "New cloud offloading algorithm for better energy consumption and process time," *International Journal of System Assurance Engineering and Management*, vol. 8, no. 2, pp. 730-733, 2017.
- [26] Z. Zhou, J. Abawajy, M. Chowdhury, Z. Hu, K. Li, H. Cheng, A. A. Alelaiwi and F. Li, "Minimizing SLA violation and power consumption in Cloud data centers using adaptive energy-aware algorithms," *Future Generation Computer Systems*, 2017. <https://doi.org/10.1016/j.future.2017.07.048>.
- [27] M. Shojafar, C. Canali, R. Lancellotti, and J. Abawajy, "Adaptive computing-plus-communication optimization framework for multimedia processing in cloud systems." *IEEE Transactions on Cloud Computing*, 2016.
- [28] S. K. Mishra, D. Puthal, B. Sahoo, S. K. Jena, and M. S. Obaidat, "An adaptive task allocation technique for green cloud computing," *The Journal of Supercomputing*, 1-16, 2018.
- [29] S. K. Mishra, D. Puthal, B. Sahoo, P. P. Jayaraman, S. Jun, A. Y. Zomaya, and R. Ranjan, "Energy-Efficient VM-Placement in Cloud Data Center," *Sustainable Computing: Informatics and Systems*, 2018. <https://doi.org/doi:10.1016/j.suscom.2018.01.002>
- [30] J. Kennedy, R.C. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, 1995.
- [31] Q. Bai, "Analysis of particle swarm optimization algorithm," *Computer and Information Science*, vol. 3, no. 1, pp. 180–184, Feb. 2010.

[32] N. Singh and R. Arya, "A novel approach to combine features for salient object detection using constrained particle swarm optimization," *Pattern Recognition*, vol. 47, no. 4, pp.1731–1739, April 2014.

[33] G. Xu, "An adaptive parameter tuning of particle swarm optimization algorithm," *Applied Mathematics and Computation*, vol. 219, no. 9, pp. 4560-4569, 2013.

[34] Sindhu HS, "Comparative analysis of scheduling algorithms of Cloudsim in cloud computing," *International Journal of Computer Applications*, vol. 97, no. 16, 2014.

[35] A. Horri, M. Mozafazi and G. Dastghaibfard, "Novel resource allocation algorithms to performance and energy efficiency in cloud computing," *The journal of Supercomputing*, vol. 69, no.3, pp. 1445-1461, june 2014.



CHAPTER-6

CONCLUSIONS AND FUTURE WORK

This chapter is dedicated to provide conclusive remarks on overall work done under this study. It also provides a quick review of the results obtained in this study as well as we discuss some of the future directions for cloud services in better way. The research work presented in this thesis has broadly focused on improving the services of cloud computing by using heuristic and meta-heuristic algorithm.

6.1 Conclusions

In this thesis, we monitored the computing resources continuously to tackle the problem of scheduling, load balancing, elasticity, scalability and optimize the QoS parameters like execution time, makespan time, throughput, task rejection ratio, cost optimization and energy consumption for better cloud services using PSO based algorithm considering deadline as constraints. The objective of cloud service provider is to maximize its profit and revenues with extreme resource utilization, while cloud users want to pay minimum amount for services by minimizing the SLA violations. Energy consumption is also play an important role in the field of cloud computing, because there is conflict between energy consumption and time. So we need a trade-off solution that optimizes both the parameters simultaneously by providing appropriate weight. We used modified PSO based algorithm (binary PSO as well as continuous PSO) to solve the discrete as well as continuous problem in the field of cloud computing and used the heuristic algorithm to provide the load balancing with elasticity.

The results obtained in various studies as reported in chapter 3 to 5 of the thesis along with the prominent features of the algorithms are summarized below:

- We have proposed an algorithm that monitored the virtual machines continuously and provides the load balancing with elasticity (resource provisioning and deprovisioning) based upon last optimal k-interval. The computational results proved that the developed algorithm optimize the parameters like makespan time, execution time, task meet with deadline ratio considering deadline as constraint in better way than min-min, SJF, FCFS algorithms.

- Secondly we have proposed dynamic transfer function based modified binary PSO (TF_p-BPSO) algorithm to solve the discrete problem by providing better exploration in the starting phase and exploitation in the last phase. Computational results proved that developed algorithm reduce the execution time, makespan time and increase the throughput in better way than existing algorithm like first come first serve and BPSO.
- Lastly we have proposed resource allocation model for processing the applications efficiently optimize execution cost, time and reduce the energy consumption of cloud data centers considering deadline as constraint using PSO based scheduling algorithm. It has been observed that developed algorithm reduces the execution time, execution cost, task rejection ratio, energy consumption and increase the throughput in comparison to PSO, honey bee and min-min algorithm.
- Analyses of above algorithms have been tested using cloudsim simulator.

6.2 Future Work

The present study can be extended in future in the following directions as mentioned below:

- Dynamic threshold value may be used for load balancing and elasticity decision.
- Machine learning algorithm can be used to predict the upcoming data rate or application requests for better scalability in cloud computing.
- The present study has mainly focused at deadline as constraint but in future we recommended for further research in the prioritization of resource allocation in relation to the finite available resources.
- The developed algorithm can be modified to improve the others QoS parameters like reliability, availability, mean time to failure, degree of imbalance, performance, SLA violation and response time for better cloud services.
- Developed PSO based algorithm can be tested in future at Montage, EpiGenomics, CyberShake, LIGO, SIPHT realistic workflows that are used for diverse scientific applications in cloud computing.
- Virtual machines migration techniques can be used in future for reducing the energy consumption of cloud datacenters.
- Hybrid meta-heuristic algorithm can be used in future to optimize the QoS parameters in better way.