An Efficient Resource Scheduling Algorithm for Cloud Computing Environment

Thesis submitted in partial fulfilment of the requirements for the award of degree of

Master of Engineering
in
Software Engineering

Submitted By
Chandan Malik
(Roll No. 801431006)

Under the supervision of:

Dr. Sushma Jain
Assistant Professor

and

Ms. Sukhchandan Randhawa
Lecturer

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

THAPAR UNIVERSITY

PATIALA – 147004

June 2016

Certificate

I hereby certify that the work which is being presented in the thesis entitled, "*An Efficient Resource Scheduling Algorithm for Cloud Computing Environment* ", in partial fulfillment of the requirements for the award of degree of Master of Engineering in *Software Engineering* submitted in Computer Science and Engineering Department of Thapar University, Patiala, is an authentic record of my own work carried out under the supervision of *Dr. Sushma Jain and Ms. Sukhchandan Randhawa* and refers other researcher's work which are duly listed in the reference section.

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

(Chandan Malik)

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

(Dr. Sushma Jain)
Assistant Professor,
CSED

(Ms. Sukhchandan Randhawa)
Lecturer,
CSED

Countersigned by

(Dr. Maninder Singh)
Head
Computer Science and Engineering Department
Thapar University
Patiala

(Dr. S. S. Bhatia)
Dean (Academic Affairs)
Thapar University
Patiala

Acknowledgement

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life. It is a great privilege to express my gratitude and admiration towards my respected supervisor **Dr. Sushma Jain,** Assistant Professor and **Ms. Sukhchandan Randhawa,** Lecturer, Computer Science & Engineering Department. This work would not have been possible without the encouragement and able guidance of them. I also thank my supervisor for their time, patience, discussions and valuable comments. Their enthusiasm and optimism made this experience both rewarding and enjoyable. I am truly grateful to them for extending their total co-operation and understanding whenever I needed help and guidance from them. I am also heartily thankful to **Dr. Maninder Singh**, Associate Professor and Head, Computer Science & Engineering Department and **Dr. Rupali Bhardwaj**, PG coordinator, for motivation and providing uncanny guidance and support throughout the preparation of the thesis report.

I will be failing in my duty if I do not express my gratitude to **Dr. S. S. Bhatia**, Senior Professor and Dean of Academic Affairs, for making provisions of infrastructure such as library facilities, computer labs equipped with net facilities, immensely useful for the learners to equip themselves with the latest in the field.

I am also thankful to the entire faculty and staff members of Computer Science and Engineering Department for their direct-indirect help, cooperation, love and affection, which made my stay at Thapar University memorable. Last but not least, I would like to thank my family for their wonderful love and encouragement, without their blessings none of this would have been possible.

Chandan Malik

(801431006)

Abstract

Cloud computing is an emerging technology and is getting popular because of its advantages. Cloud computing is a computing paradigm which is used to provide software, infrastructure and platform as a service to the users via internet. Cloud permits the sharing of resources which are geographically dispersed by the users as per their requirements. Thus cloud provides that scaled environment which can satisfy the changing requirements of the users.

One of the challenging problems in the area of Cloud computing is to manage the cloud resources efficiently. The objective is to allocate resources to user's requests such that user requests are completed in minimum time and resources are also used efficiently.

To solve the cloud scheduling problem, many researchers have proposed different scheduling algorithms. Based on Harmony Search, a new metaheuristic scheduling approach is proposed in this thesis and its performance is evaluated on various parameters. Harmony Search is a music-based metaheuristic optimization algorithm inspired by improvisation process of musicians. Proposed approach give better results in terms of makespan when compared with FCFS, Random Allocation (RA) algorithm and Constrained Random Allocation (CRA) algorithm. Also the proposed approach outperforms Genetic Algorithm in terms of convergence time of the algorithm and average waiting time of the cloudlets.

Contents

List of Figures

List of Tables

Chapter 1
Introduction

This chapter introduces the basic aspects of Cloud Computing, its evolution and covers related technologies like Utility Computing, Grid Computing and Cluster Computing. The chapter thereafter highlights the various research challenges in the area of Cloud Computing and describes the structure of thesis towards the end of the chapter.

1.1 Background

Need of computing resources are increasing rapidly with widespread usage of computer and Internet and with the growth in IT technologies. Investing more in resources is not economical for any organization. With Internet getting popular with time, a very elastic infrastructure is expected that can be scaled up or down with the changing requirements. This evolves the idea of Cloud. *Cloud Computing* [1] refers to the computing on the Internet, as opposed to computing on a desktop. Cloud Computing permits the sharing of resources by users at different geographical sites and provides access to resources which are geographically dispersed. Cloud offers services which are delivered on demand to the customers over the Internet. Thus cloud provides that scaled environment which can satisfy the changing requirements of the customers.

1.2 Evolution of Cloud Computing

Cloud undergoes evolution phase by phase which includes Utility Computing, Grid Computing and Distributed Computing. Concept of cloud was first implemented in 1950's when large-scale mainframes were installed in a server room and were accessed by users through terminals. Terminals act like dumb machines which are used to send orders to mainframe where processing takes place and response is sent back to the terminals.

In the mid-1990's, Grid Computing emerged which facilitate users to obtain computing resources on demand. Grid is a type of parallel and distributed system that enables the sharing, selection and aggregation of geographically distributed

autonomous resources dynamically at runtime depending on their availability, capability, performance, cost and user's Quality of Service (QoS) requirements.

In 2002, Amazon launched Amazon Web Services (AWS) followed by EC2 in 2006 which offers a suite of on-demand computing services to the users. After that, all major IT giants including IBM, Google, Microsoft and Yahoo launched cloud solutions. Cloud service providers are increasing rapidly in number now. Comparison of Cloud with related technologies like Utility Computing, Grid Computing and Cluster Computing is described below [2], [6]:

1.2.1 Comparison with Utility Computing: Nature of leasing is the fundamental difference between these two computing paradigms. Both paradigms use a third party which will lease computing services but in cloud user is unaware of the policies implemented at provider's side and source of the services etc. Utility Computing offers a straightforward rental service where user is fully aware about the source of services. Moreover virtualization is supported by cloud but not by Utility Computing.

1.2.2 Comparison with Grid Computing: Cloud resources are distributed geographically but are controlled centrally whereas management of resources in grid is decentralised and each node administrates its resources. Therefore cloud services are more reliable as the resources are managed centrally. Moreover, no particular billing model exists for grid like "Pay-per-use" in case of Cloud Computing. Grid services are usually used by Research institutes.

1.2.3 Comparison with Cluster Computing: Cluster differs from above computing paradigms by a fact that computers in a cluster are connected by LAN whereas resources of cloud and grid are widely scaled and geographically dispersed. Another way to differentiate Cluster from Cloud and Grid is that resources within cluster are tightly coupled and machines in a cluster have similar hardware configuration where as resources of Cloud and Grid are loosely coupled and their hardware configuration can differ.

1.3 Cloud Computing

Many definitions of Cloud Computing are proposed in terms of features offered by this computing paradigm. Some of the standard definitions of Cloud Computing are as

follows:

- Rajkumar Buyya defines Cloud Computing as "A Cloud is a type of parallel and distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and presented as one or more unified computing resources based on Service Level Agreements (SLAs) established through negotiation between the service provider and consumers" [3].
- National Institute of Standards and Technology (NIST) defines Cloud Computing as "Cloud Computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications and services) which can be rapidly provisioned and released with minimal management effort or service provider interaction. This Cloud model promotes availability and is composed of five essential characteristics, three service models, and four deployment models" [4].

1.4 Cloud Computing Characteristics:

Cloud Computing possesses following characteristics [5]-[7]:

- **Cost reduction:** Cloud provides "on-user-demand" computational resources which are shared among the cloud users and are paid as-per-use of these services. This computing paradigm helps in reducing cost as users have to pay only for the services they have used.
- **Elasticity:** Cloud offers a rapid provisioning of services, thus provides a flexibly of leasing and releasing services whenever required. Cloud provides services that can be easily scaled up or down with the changing user requirements.
- **Geographical independence**: Cloud Computing permits the sharing of resources by the users at different geographical sites and provide access to resources which are geographically dispersed. Cloud services are accessed via Internet, thus a user irrespective of its geographical location can easily access cloud resources.
- **Pay-as-you-go:** Cloud services are paid as per their use. Thus a user will pay only for the services he has used.

- **Maintenance:** Cloud applications are easy to maintain because applications don't require installation on user's machine. Maintenance is transparent and is done by the service provider.
- **Reduced IT infrastructure cost:** Cloud offers infrastructure services, thus allowing users to avoid purchase of infrastructure for every single need. This computing paradigm thus helps organizations by reducing their cost of infrastructure acquisition and maintenance.
- **Easy access:** Cloud services are web based and can be easily accessible with devices like desktops, laptops and mobiles etc. having Internet connection as shown in Fig. 1.1 [7].

Fig. 1.1: Cloud services easily accessed by various devices

1.5 Cloud Computing Service models

Cloud offers three services namely *Infrastructure as a Service (IaaS), Platform as a Service (PaaS)* and *Software as a Service (SaaS)* as shown in Fig. 1.2.

- **Infrastructure as a Service (IaaS):** IaaS offers storage, processing, networks and other computing resources to cloud users. IaaS users can install and run operating system, applications and software on the infrastructure that can be dynamically scaled up and down based on the requirements. Users can rent virtual machines or even datacenter. Amazon's EC2 is an example of IaaS [6], [8].

- **Platform as a Service (PaaS):** PaaS providers provide a computing platform which typically includes operating systems, programming language run-time environment, databases and web servers so that cloud users can develop their applications on this platform. PaaS is an application development platform supporting complete software lifecycle. PaaS can hosts complete application or an application-in-progress. Windows Azure, Google app Engine etc. are typical examples of PaaS [6], [8].

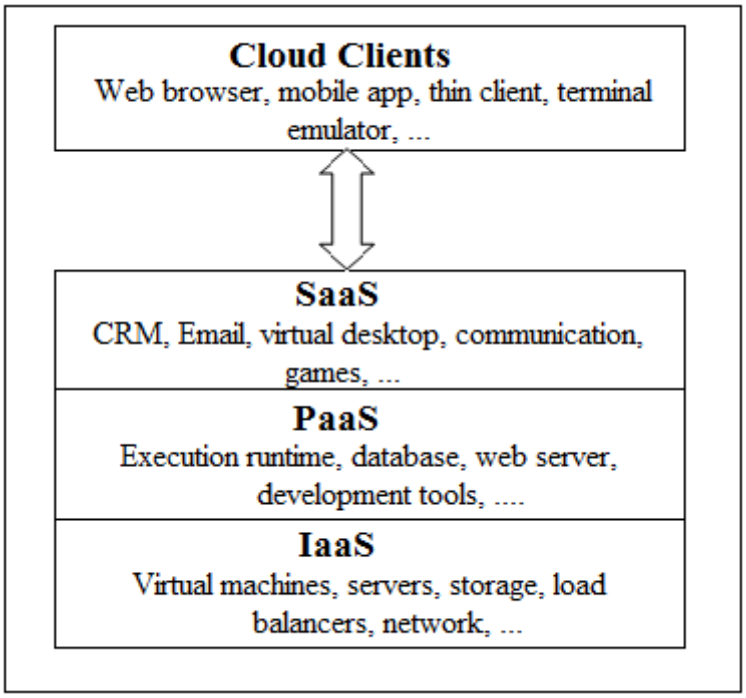


Fig. 1.2: Cloud Service models

- **Software as a Service (SaaS):** SaaS provides software or applications which are accessed by cloud users using interfaces such as web browsers over the Internet. It eliminates the need of users installing software on their machines and users can access them directly from the cloud. Cloud datacenter hosts the application. Another benefit of SaaS is that it eliminates the licensing and version compatibly issues for the users. SalesForce.com, Google Docs, Google Mail etc. are examples of SaaS [6], [8].

1.6 Deployment models of Cloud:

There are four deployment models of cloud namely *Private Cloud, Community Cloud, Hybrid Cloud* and *Public Cloud* as shown in Fig. 1.3. [6], [8]:

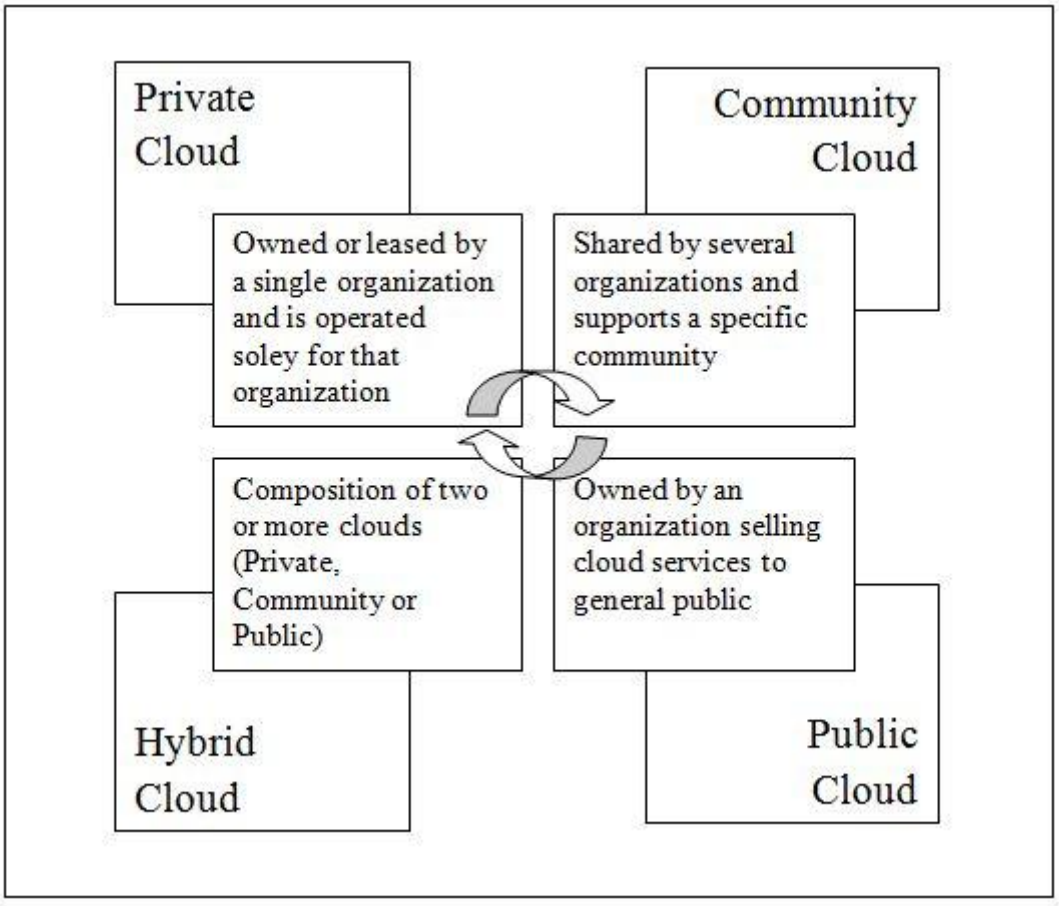


Fig. 1.3: Cloud Deployment Models

- **Private Cloud:** Private Cloud is a deployment model which is limited to an organization only. Private cloud is preferred by those organizations which need accurate control over their data. This model is efficient as resources are dedicated for an organization only. Security, accurate control over data, minimum data transfer cost and reliability are the other benefits of Private cloud.
- **Community Cloud:** Community Cloud is a deployment model in which cloud services are shared by a community having similar concerns i.e. services are shared by group of organizations. . This model is different from Private Cloud, where cloud services are dedicated to an organization only. Economical scalability is the main benefit of Community Cloud.
- **Public Cloud:** This deployment model is the most dominant model where resources like storage, applications and computation etc. are available for public via Internet. Cloud services are available publicly and service provider

has full control over the policies, charging model etc. e.g. Sun Cloud, Google App Engine etc

- **Hybrid Cloud:** Hybrid cloud is a cloud deployment model which is obtained by combining two or more cloud models like private, public and community cloud. These cloud models are integrated through standard protocols. Hybrid cloud offers benefits of multiple deployment models. This model is more flexible than private and public cloud as it extends capabilities of these models but certain issues of interoperability and standardization prevails within this deployment model.

1.7 Research Challenges in Cloud Computing:

Cloud Computing is a new technology which is getting evolved because of its advantages. This computing paradigm is in its early stage and there are various challenging research issues which are described below [6]-[9]:

- **Energy Management**: Energy consumption by the data centers must adhere to the environment standards and government regulations. Designing data centers which are energy efficient pose as a big challenge in cloud computing. Designing energy efficient data centers can be accomplished by using server consolidation and energy aware job scheduling.
- **Server Consolidation:** Server consolidation is used to maximise resource utilization and minimize energy consumption at cloud data centers. Virtual Machine (VM) migration technique is used to combine VMs which are running on various underutilized servers onto a single server and turning off unused servers to conserve energy. The main challenge is to consolidate servers optimally.
- **Security:** Security is another research issue in the area of cloud. In cloud, resources like computing, applications and storage etc. are provided by the third party. So, it arise the issue of security as user's data is stored on an infrastructure which is owned by a third party and user has no privilege to implement security checks. Even in Private Cloud, cloud users can manage security settings remotely without knowing whether it's implemented or not.
- **VM Migration:** VM migration technique help in balancing load across data centers. It also improves the response time in data centers. Another advantage

of VM migration is avoiding hotspots. VMs must be transferred effectively and this arise another challenge in cloud.

- **Resource Scheduling in Cloud:** Another research challenge in cloud environment is resource scheduling in cloud. The main challenge is to allocate resources to user's requests such that the user requests can be completed in minimum time. Scheduling policies are implemented at various levels in cloud. Research issues of resource scheduling are discussed in detail in the subsequent chapters.

1.8 Structure of thesis:

Rest of the thesis report is organized as follows:

Chapter 2: This chapter covers the state of art for the Resource scheduling problem in Cloud.

Chapter 3: This chapter describes the gaps find during literature survey along with the problem statement.

Chapter 4: This chapter discusses the proposed approach in detail.

Chapter 5: This chapter covers the tools used for implementation and implementation results.

Chapter 6: This chapter concludes the work and gives future directions.

Chapter 2

Literature Review

This chapter covers the state of art for the Resource scheduling problem in Cloud Computing environment.

Devipriya and Ramesh [10] proposed improved max-min strategy for resource scheduling in which expected execution time was used as scheduling criteria. Idea was to assign tasks having maximum expected execution time to resource with minimum completion time (slowest resource). Presented approach gave better results in terms of makespan compared to RASA and original max-min algorithm.

Ming and Li [11] proposed Max-Min Spare Time (MMST) algorithm which was based on Max-Min algorithm for cloud task scheduling. Presented approach mapped tasks with minimum spare time to the resource having maximum execution time. MMST algorithm outperformed the Max-Min algorithm in terms of load balancing and resource utilization.

Etminani and Naghibzadeh [12] proposed a Min-Min Max-Min Selective algorithm which was based on Min-Min and Max-Min algorithms. Based on the standard deviation of the expected completion time of tasks on the resources, presented approach select among these two algorithms. Experimental results proved that the presented approach outperforms the traditional Max-Min and Min-Min algorithms in terms of throughput.

Sindhu and Mukherjee [13] proposed a bi-objective genetic algorithm for cloud scheduling problem which minimised the makespan and improved processor utilization. In order to produce initial population, presented approach used LCFP, SCFP, MCT and random algorithm which was followed by crossover and mutation operations and compared these four approaches on various parameters.

Omara and Arafa [14] focussed on the modification of standard Genetic algorithm to get optimized results. Author proposed two algorithms, Critical Path Genetic Algorithm (CGPA) and Task Duplication Genetic algorithm (TDGA) which was based on the principles of Genetic Algorithm. CGPA reschedules critical path nodes

and focused on minimization of makespan and balancing load. TDGA employed a Genetic Algorithm which was based on task duplication and focused on minimizing the communication delay and which thus reduced overall execution time.

Zheng *et al.* [15] proposed Parallel Genetic Algorithm which improved the convergence time of algorithm and resource utilization as compared to standard Genetic Algorithm. Factors which were considered in this new approach were chromosome representation, fitness function design, migration method in order to improve the performance.

Singh and Sahu [16] proposed cloud scheduling solution using Compact Genetic Algorithm (CGA) which mimicked order-one behaviour of Simple Genetic Algorithm. In this approach population was represented as probability distribution and during each iteration CGA selected individuals based on the probabilities specified in probability vector. Probability vector was updated after evaluating the individuals. Presented approach resulted in minimum completion time while executing tasks.

Kumar and Verma [17] suggested Improved Genetic Algorithm. Idea was based on generating better initial population as compared to generating the population randomly. The proposed approach used Max-Min and Min-Min algorithm for generating initial population followed by crossover and mutation operators. Improved genetic algorithm gave optimal solution in terms of resource utilisation and makespan.

Guo-ning *et al.* [18] combined the basic ideas of Genetic Algorithm and Simulated Annealing and proposed an approach which include initial population generation followed by crossover, mutation and annealing operations. Algorithm takes into consideration QoS requirements of tasks and scheduling was done in accordance with QoS requirements.

Liu *et al.* [19] introduced a new approach for cloud scheduling problem which was an integration of GA and ACO. Proposed approach used global search ability of GA and then converted it into the initial pheromone of ACO which finally resulted in an optimal solution through positive feedback feature of ACO. Proposed approach outperformed the standard Genetic Algorithm and standard ACO in terms of convergence time of algorithm.

Shakya and Prajapati [20] proposed Genetic Algorithm for grid environment which minimised the makespan and efficiently utilized the resources. GA searches different regions of the solution space and results an optimal solution.

Hoang *et al.* [21] proposed a theory which was based on ACO and Particle Swarm Optimization (PSO) heuristic approaches with the objective of minimizing cost of system, satisfying QoS constraints, and providing maximum profit to the SaaS providers. Presented approach allocated low-cost resources to requests in order to minimize cost for the users and also combined Myprofit algorithm for maximizing profit of SaaS providers by utilizing resources to their fullest.

Liu *et al.* [22] suggested a combination of Genetic Algorithm and ACO for grid systems to improve the efficiency of task scheduling and also balance the load. Proposed approach made use of the advantages of ACO which includes robustness, potential parallelism and positive feedback mechanism to attain a set of solutions which are further operated by selection, crossover and mutation operations and these new set of solutions was fed as input to the next iteration of ACO, The process gets repeated until the termination condition occurs.

Chen *et al.* [23] proposed a Genetic Algorithm with multiple fitness values. Idea was to use a hybrid genetic algorithm combined with knapsack problem with multiple fitness values to improve the resource utilization of cloud resources and conserves energy.

Tawfeek *et al.* [24] presented ACO for cloud environment. Based on insect metaphor, ACO was used to solve optimization problem. ACO uses pheromone table to store the search results up to current iteration from the initial stage. Based on the values stored in pheromone table, ACO performs transition, evaluation and determination operation. Results were compared with FCFS and round-robin algorithms and presented approach outperforms FCFS and round-robin in terms of makespan.

Chen *et al.* [25] suggested improved ACO which combined the advantages of ACO and Greedy Algorithm which shortened the scheduling time and implement system load balancing. For setting the parameters of ACO, average random, Gaussian random, manual methods were used. Proposed approach was compared with FCFS and greedy on same simulation parameters and results evaluated were better.

Wen *et al.* [26] combined advantages of ACO and PSO. Idea was to eliminate the shortcoming of ACO to get trapped in the local optimum. Improved algorithm used ACO for finding the initial solutions using update pheromone followed by PSO to get more efficient solutions which were operated by crossover and mutation operation to avoid getting trapped into local optimum. Proposed approach improved the resource utilization and convergence time of the algorithm.

Mahamud and Nasir [27] proposed enhanced ACO scheduling algorithm. It combined Max-Min algorithm in Grid environment which minimised the makespan. Algorithm was based on local pheromone update and trail limits. The amount of pheromone for the assigned resource will be reduced during the update in local pheromone trail. It will ensure, for the other ants, assigned resource to become less desirable. While the trail limit, which is the allowed range of the pheromone strength, was limited to maximum and minimum trail strength.

Khalili and Babamir [28] suggested PSO based dynamic scheduling algorithm which minimized the makespan. Inertia weight adjusted the local and global search capability of the algorithm. Different inertia weight strategies were used during implementation and results shows that PSO algorithm with Linear Descending Inertia Weight (LDIW) strategy gave optimal results.

Abdi *et al.* [29] suggested a modified Particle Swarm Optimization (PSO) for cloud resource scheduling problem. Modified PSO approach used shortest job to fastest processor algorithm (SJFP) for generating initial population instead of generating the initial population randomly. Results proved that the proposed approach minimised makespan as compared to PSO and Genetic Algorithm.

Liu *et al.* [30] proposed a scheduling approach in grids using a fuzzy PSO in which position and velocity of particles were represented in fuzzy matrices. It generated an optimal schedule dynamically which minimized the makespan and improved resource utilization. Performance of fuzzy PSO was evaluated and compared with Simulated Annealing and Genetic Algorithm.

Sharma and Kuila [31] presented a new heuristic approach for scheduling in cloud environment called Dependency Task first (DTF). Idea of this algorithm was to consider dependencies among the tasks and schedule resources accordingly which in

turn minimized the makespan. Proposed approach was compared with Bounded Number of Processors (BNP) class of scheduling algorithms - Modified Critical Path (MCP), Earliest Time First (ETF) and Dynamic Level Scheduling (DLS).

Zhan *et al.* [32] conducted a survey on the existing resource scheduling algorithms for cloud at various layers, considered various objectives of scheduling and compared various existing approaches.

Tsai and Rodrigues [33] reviewed existing metaheuristic resource scheduling algorithms, describe and compared these metaheuristic approaches for scheduling problem in cloud.

Existing resource scheduling algorithms in cloud and grid environment are summarised in Table 2.1

Table 2.1: Resource scheduling algorithms in Cloud and Grid environment

Author & Year	Algorithm/ Technique	Scheduling Parameter	Description	Enviro -nment
S. Devipriya (2013) [10]	Improved Max-Min	Makespan	Tasks were assigned having maximum expected execution time to resource with minimum completion time	Cloud
Gao Ming (2012) [11]	Max-Min Spare time	Resource Utilization, Load balancing	Tasks were mapped with minimum spare time to resource having maximum execution time	Cloud
Kobra Etminani (2007) [12]	Min-Min Max-Min Selective Algorithm	throughput	Based on standard deviation of expected completion time of tasks, Max-Min or Min-Min selected	Grid
S.Sindhu (2013) [13]	Biobjective Genetic Algorithm	Makespan, Resource utilization	Four GA were compared with different initial population generation methods	Cloud

Fatma A. Omara (2009) [14]	Critical Path Genetic Algorithm, Task Duplication Genetic Algorithm	Makespan, Load Balance	1.CGPA rescheduled critical path nodes and modifications were added to efficiently use processor time and balance load 2. TDGA was based on task duplication and minimizes communication delay	Cloud
Zhongni Zheng(2011) [15]	Parallel Genetic algorithm	Convergence time, Resource Utilization	Factors such as chromosome representation, fitness function design, migration method were considered	Cloud
Prateek Kumar Singh (2014) [16]	Compact Genetic Algorithm	Makespan	Individuals were selected based on the probability specified in the probability vector	Grid
Pardeep Kumar (2012) [17]	Improved Genetic Algorithm	Makespan, Resource Utilization	Generate initial population using Min-Min and Max-Min followed by crossover and mutation operator	Cloud
Gan Guo-ning (2010) [18]	Genetic Simulated Annealing Algorithm	QoS	Scheduling was based on QoS requirents of tasks using Genetic Simulated Annealing Algorithm	Cloud
Chun-Yan Liu (2014) [19]	GA-ACO algorithm	Converging time of algorithm	GA-ACO converted optimised solutions generated by GA into pheromone of ACO and used positive feedback of ACO to get optimized solution	Cloud

Subarna Shakya (2015) [20]	Genetic Algorithm	Makespan, resource utilization	Scheduling was based on initial solution set generation followed by selection and crossover operators	Grid
Ha Nguyen Hoang(2016) [21]	ACACO, ACPSO	Cost minimization, QoS	Scheduling was done by allocating tasks to resources with less cost and Myprofit algorithm was used to maximize the profit of SaaS providers.	Cloud
Jing Liu (2008) [22]	Ant Colony Genetic Algorithm	Makespan, Load balancing	Idea was to combine GA and ACO heuristics to improve the scheduling performance	Grid
Shi Chen (2012) [23]	GA with multiple fitness	Resource utilization, Energy conservation	Scheduling was done using a hybrid GA combined with knapsack problem with multiple fitness values	Cloud
Medhat A. Tawfeek (2013) [24]	Ant Colony Optimization	Makespan	Using pheromone table, transition, evaluation and determination operations were performed	Cloud
Hongwei Chen (2013) [25]	Improved Ant Colony Optimization algorithm	Makespan, Load balancing	Parameters of ACO were set using average random, Gaussian random, manual methods	Cloud
Xiaotang Wen (2012) [26]	Combined ACO-PSO algorithm	Resource Utilization, Convergence speed	Scheduling was done using combined ACO-PSO approach	Cloud

Ku Ruhana Ku-Mahamud (2010) [27]	Enhanced ACO algorithm	Makespan	Idea was based on local pheromone update, PV matrix update, trail limits	Grid
Azade Khalili (2015) [28]	PSO-Based dynamic scheduling algorithm	Makespan	Different inertia weight strategies used with PSO	Cloud
Solmaz Abdi (2014) [29]	Modified PSO	Makespan	Initial population was generated using SJFP algorithm followed by evaluating fitness of particles, updating the position and velocity matrices	Cloud
Hongbo Liu (2009) [30]	Fuzzy PSO	Makespan, Resource utilization	fuzzy matrices were used to represent velocity and position of particle	Grid
Suruchi Sharma (2015) [31]	Dependency Task first (DTF	Makespan	Scheduling was done by considering the dependencies among the tasks	Cloud
ZHI-HUI ZHAN (2015) [32]	Survey on Cloud Resource Scheduling algorithms	QoS, Load Balance, Energy conservation, Cost Effectiveness etc.	Survey paper	Cloud
Chun-Wei Tsai (2014) [33]	Survey on metaheuristic cloud scheduling algorithms	Makespan, Load balance, Energy conservation etc.	Survey paper	Cloud

Chapter 3

Problem Statement

Previous chapter reviews various existing resource scheduling algorithms in cloud environment. This chapter analyzes the gaps which are found during the literature review of the existing resource scheduling algorithms and describes the problem statement.

3.1 Barriers in the existing research work

Previous chapter cover various existing cloud scheduling algorithms but these algorithms have certain issues or loopholes which are necessary to be considered while designing a new cloud scheduling algorithm. Among all the research papers studied, parameters like makespan, convergence time of algorithm, resource utilization, energy consumption, QoS requirements and load balancing are the major issues of concern. The gaps which have been analysed in the existing resource scheduling algorithms are as follows:

- Proposed algorithms do not consider the minimization of makespan and waiting time of requests in the suggested methodologies.
- Existing metaheuristic approaches proposed for the cloud resource scheduling problem do not usually considers the convergence time of the algorithm.
- Parameters such as priority, deadline, reliability, scalability, bandwidth etc. are usually ignored.
- Dependencies among the requests are usually not considered in the existing scheduling algorithms.
- Proposed algorithms for the scheduling problem do not consider the real time cloud environment where user's requests and cloud resources changes dynamically.
- Existing resource scheduling algorithms are not usually suitable for large-scale scheduling.
- Scheduling in cloud environment is to be done considering various objectives which favour the service providers as well as cloud users but proposed algorithms usually consider one or two objectives.

3.2 Problem Statement

One challenging problem in the area of Cloud Computing is to manage the cloud resources efficiently. The main objective is to allocate resources to user's requests in such a way that the user requests are completed in minimum time with minimum response time involved for the requests and algorithm converges quickly towards the optimal solution. The traditional scheduling problem can be defined as problem of finding an optimal solution to schedule given set of tasks $T=\{T_1, T_2, T_3, T_4 \ldots . T_n\}$ onto given set of machines $M= \{M_1, M_2, M_3, M_4 \ldots . M_n\}$ while considering the predefined constraints.

In context to thesis, Cloud scheduling problem can be defined as a problem of finding an "optimal" mapping C: $T \times R \rightarrow F^2$ which includes assignment of M tasks, $T= \{T_1, T_2, T_3 \ldots . T_M\}$ onto N cloud resources $R= \{R_1, R_2, R_3 \ldots . R_N\}$ such that the given objectives $F= \{F_1, F_2\}$ are minimised, where F_1 denotes makespan and F_2 represents average waiting time of the requests. Mathematically, these objectives can be represented as:

$$\begin{aligned} & F_1 = C(s), \\ & C(s) = max(C_i), \\ & i = 1, 2, 3 \ldots \ldots M \end{aligned} \tag{1}$$

where s is the candidate solution, and let $C(s)$ represents the completion time of the solution, C_i denotes the completion time of i^{th} task and M represents the number of requests.

$$\begin{aligned} & F_2 = W(s), \\ & T(s) = \sum_{i=1}^{M} W_i \\ & W(s) = T(s) / M \end{aligned} \tag{2}$$

where s is the candidate solution, let $W(s)$ represents average waiting time of the tasks, $T(s)$ represents total waiting time of the solution, W_i represents waiting time of the i^{th} task.

Fitness value of the solution will be calculated as:

$$\begin{aligned} & f = x_1 * F_1 + x_2 * F_2 \\ & where\, x_1, x_2 \in [0,1] \; and \; x_1 + x_2 = 1 \end{aligned} \tag{3}$$

Chapter 4
Proposed Work

This chapter covers the proposed Harmony Search based approach for the problem discussed in previous chapter and compare the proposed algorithm with *First-Come-First-Serve(FCFS) algorithm, Random Allocation (RA) algorithm, Constrained Random Allocation (CRA) algorithm and Genetic Algorithm (GA)* in terms of various parameters such as makespan, waiting time of cloudlets and convergence time of algorithm etc.

4.1 Harmony Search

Harmony Search (HS) is a music-based metaheuristic optimization algorithm which was first introduced by Geem *et al.* in 2001 [34]. This algorithm is already used for solving various optimization problems. Harmony search follows the following steps:

Algorithm 1: Harmony Search algorithm

begin

Define objective function $f(x)$, $x=(x_1,x_2, \ldots,x_n)^T$

Initialise harmony memory with random harmonies(solutions)

Define *Pitch Adjustment Rate*(r_{pa}), *pitch limits, bandwidth*

Define *Harmony Memory Considering Rate*(r_{accept})

while(*i<Maximum number of iterations*)

Identify the worst harmony

Improvise a new harmony by considering r_{accept} , *pitch limits*

Adjust pitch to get new harmonies by considering r_{pa}

Accept new harmonies if better

end while

Find the current best solutions

end

Fig. 4.1: Pseudo code for Harmony Search algorithm

Fig 4.1 represents the pseudo-code for the Harmony Search. The algorithm uses the following terminology for the various parameters:

r_{accept}: Harmony memory considering rate

r_{pa}: Pitch adjustment rate

Consider an optimization problem expressed in eq. 4:

$$min\ f(x) = (x_1 - 1)^2 + (x_2 - 2)^4 + (x_3 - 3)^2 + 3 \quad (4)$$

This is a minimization problem, objective of which is to find values of variables x_1, x_2, x_3 for which function value will be least. Considering global minimum, solution vector of the problem is (1, 2, 3) for which the function value will be minimum. However, Harmony Search uses another way to find the solution. Harmony search algorithm begins with random set of solutions. In the Fig. 4.2.a, Harmony memory is initialized with certain random solutions and these solutions are stored in the sorted order based on the objective function.

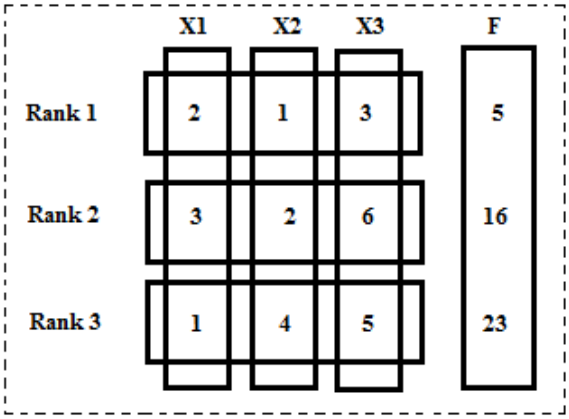


Fig. 4.2.a: Initial Harmony memory

Next, the new harmony (1,4,3) is generated by considering existing solutions from the harmony memory as x_1 chooses {1} from {2,3,1}; x_2 chooses {4} from {1,2,4} and x_3 chooses {3} from {3,6,5}.

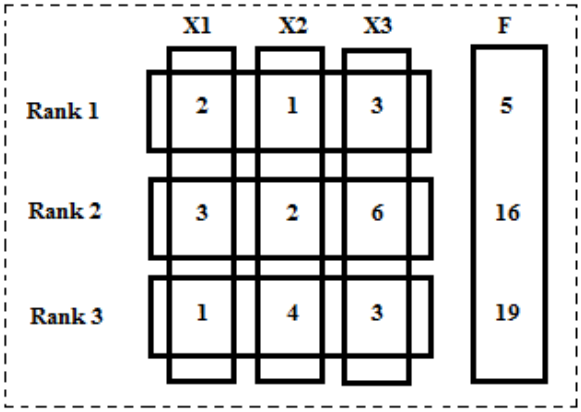


Fig. 4.2.b: Subsequent Harmony memory

Since the function value for the new harmony is 19, harmony (1, 4, 3) is included in the harmony memory replacing (1, 4, 5) which is the worst harmony, as shown in Fig. 4.2.b.

Next, harmony search improvises a new solution (1, 1, 3) whose function value is 4, which results in replacement of harmony (1, 4, 3) from the harmony memory as shown in Fig. 4.2.c.

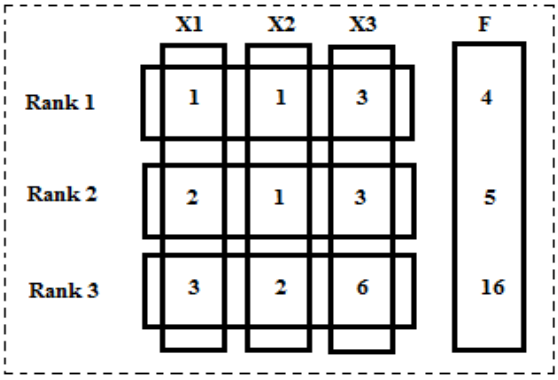


Fig. 4.2.c: Subsequent Harmony memory

Finally, Harmony Search generates harmony (1, 2, 3) which is the global minimum with function value 3. Obviously, the above approach assumes that every part of the global solution exists initially in harmony memory, which might not be the case always. Thus Harmony Search uses two parameters. Firstly Harmony Search uses Harmony Memory Considering Rate (HMCR) also represented by accepting rate $r_{accept} \in [0,1]$. HMCR of 0.85 means that during the improvisation process algorithm chooses value of component variable from harmony memory with a probability of 85%. Low HMCR results in the selection of only few harmonies from HM and thus solution converges slowly. If the rate of HMCR is too high, almost all the solutions are generated from HM which causes other harmonies not be explored well, which probably result in wrong solutions. Thus the typical value of HMCR is between 0.7 to 0.95.

Second parameter used in Harmony Search is Pitch Adjustment Rate (PAR) denoted by r_{pa}. By adjusting the pitch slightly, a new solution is produced close to the parent solution. Low pitch adjustment rate will cause the algorithm not to explore the entire solution space. On the other hand, having a high PAR value will slow down the convergence of algorithm as algorithm will behave similar to random search. Thus, PAR usually ranges between 0.1~0.5. E.g. PAR of 0.2 indicates that neighbouring

value will be chosen with a probability of 20%. Suppose that the possible range of values for a component variable is {A, B, C, D, E, F}, HMCR is 0.85, PAR is 0.10 and HM includes {A, C, E}. During the improvisation process of Harmony Search algorithm, the algorithm chooses the component value from {A, C, E} with 85% probability and value from {A, B, C, D, E, F} with 15% probability and {C} can be shifted to {B} or {D} with a probability of 10% when {C} is chosen. Flowchart of Harmony search is shown in Fig. 4.3 [35].

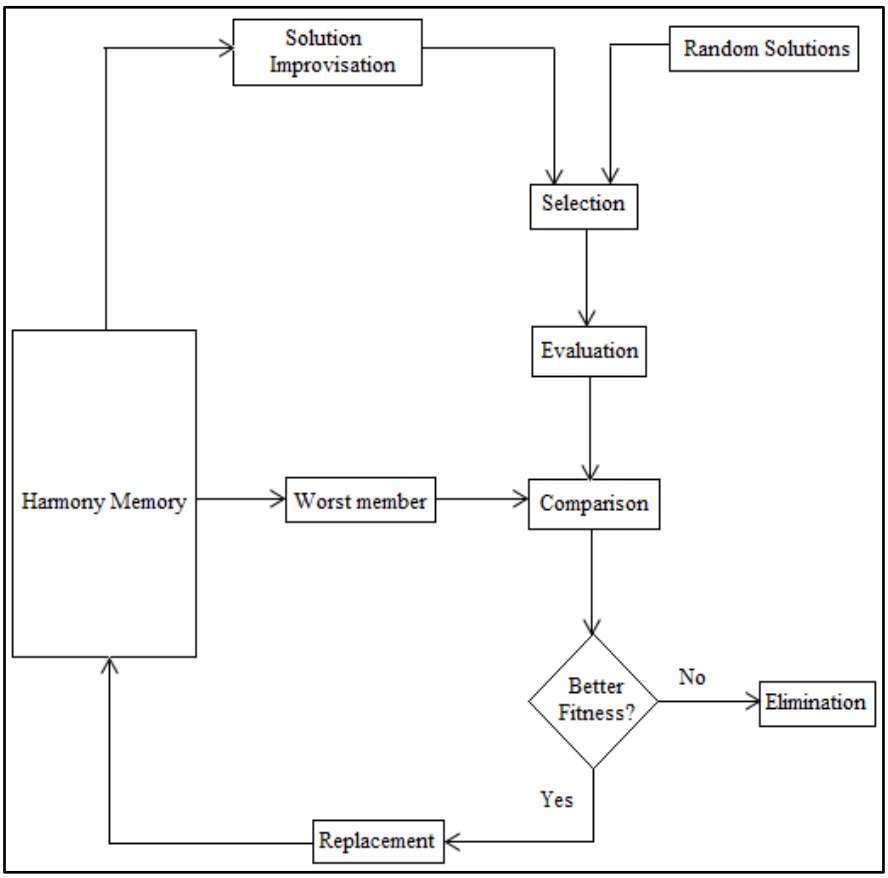


Fig. 4.3: Flowchart of Harmony Search

4.2 First-Come-First-Serve (FCFS) algorithm

First-come-first-Serve (FCFS) algorithm is a traditional algorithm which serves the requests in the order of arrival of requests. FCFS in cloud computing environment allocates VM's to cloudlets in the order of their arrival.

4.3 Random Allocation (RA) algorithm

Random Allocation algorithm is a cloud resource scheduling algorithm which allocates virtual machine to cloudlets in a random order. This algorithm sometimes

results the best solution and sometimes ends up in a worst possible solution since there is no constraint imposed on the allocation of virtual machines to cloudlets. Pseudo code for RA algorithm is described in Fig. 4.4.

Algorithm 2: Random Allocation algorithm

```
begin
        while(every cloudlet is not assigned VM )
                Select a cloudlet not binded to VM
                Select a VM  from VM set randomly
                Allocate selected VM to cloudlet
        end while
        Scheduling solution available
end
```

Fig. 4.4: Pseudo code for Random Allocation algorithm

Loop will terminate when every cloudlet is allocated a virtual machine. Since the algorithm performs VM allocation in a random manner, it might results in uneven load distribution which ultimately causes inefficient resource utilization. Thus algorithm gives results which ranges anywhere between best and worst possible solution and also causes poor resource utilization. Problem of uneven load distribution is handled in Constrained Random Allocation algorithm.

4.4 Constrained Random Allocation (CRA) algorithm

This algorithm forces even distribution of load among virtual machines by imposing a constraint that every virtual machine can process cloudlets up to a limit. Algorithm follows a RA policy for allocating virtual machines to cloudlets but a check is performed prior to binding a virtual machine with a cloudlet that the assigned virtual machine doesn't exceed the limit of cloudlets it can process. This threshold value is calculated by dividing the number of cloudlets with total number of virtual machines and taking the ceil value. This limit check ensures equal distribution of load on virtual machines and ensures efficient resource utilization. Steps followed by CRA algorithm is described in Fig. 4.5.

Algorithm 3: Constrained Random Allocation algorithm

```
begin
    Calculate limit, limit = ceil(Number of cloudlets / Number of virtual machines )
    while( every cloudlet is not assigned VM )
        Select a cloudlet not binded to VM
        Select a VM  from VM set randomly
        Check if selected VM reaches calculated limit
        if (selected VM not reaches limit)
            bind cloudlet to VM
        else
            Linearly find a VM with cloudlet limit not reached
            Allocate selected VM  to cloudlet
        end if
    end while
    Scheduling solution available
end
```

Fig. 4.5: Pseudo code for Constrained Random Allocation algorithm

Loop will terminate when every cloudlet is allocated a virtual machine. This algorithm gives better results in terms of makespan and resource utilization as compared to RA algorithm due to the even load distribution among virtual machines. To reduce the complexity, if the limit of randomly selected virtual machine is set to threshold value, then instead of generating random number again and again which is time consuming if nearly all the virtual machines are limit set, a linear approach is followed. This algorithm doesn't consider waiting time of cloudlets in the solution. The outcome of the algorithm may involves assignment of light cloudlets i.e. cloudlet having less workload to efficient machines and heavy cloudlets i.e. cloudlet having higher workload to machines with less efficiency which overall increases the makespan. These issues are considered and eliminated in proposed Harmony Search based approach.

4.5 Proposed Harmony Search based algorithm

Harmony search is a music based metaheuristic algorithm which is proposed to solve the resource scheduling problem in Cloud Computing environment [34], [35].

Harmony search based algorithm gives possibly the best optimised scheduling solution, converges quickly and also tries to minimise the response time of cloudlets. Harmony Search is used to solve various existing problems e.g. Constrained minimization problem, Travelling Salesman Problem (TSP) etc. The proposed Harmony search based algorithm for the cloud resource scheduling problem begins with a random set of solutions called *harmonies* which are generated using RA algorithm in which VMs are assigned to cloudlets randomly. Next a loop will iterate which improvise a new harmony considering HMCR and PAR in each iteration and replace it with the worst harmony. The fitness value for the harmonies is calculated by considering makespan and average waiting time of the cloudlets in the ratio x_1: x_2 such that x_1 and x_2 are in between 0 and 1 and x_1 and x_2 sums to 1. The pseudo-code for the proposed harmony search based algorithm is represented in Fig. 4.6.

Algorithm 4: Harmony Search algorithm for cloud resource scheduling problem

begin

Objective function: *minimise(makespan, average waiting time)*

Initialise harmony memory with random harmonies(solutions)

Define harmony search algorithm parameters, r_{accept} ,r_{pa} , *pitch limits*

Calculate fitness value, *fitness= 1 / (x_1*makespan + x_2*average waiting time)*

where $x_1, x_2 \in [0,1]$ & $x_1+x_2=1$

while *(i<Maximum number of iterations)*

Identify the worst harmony

Improvise a new harmony, calculate fitness of improvised harmony

Replace with worst harmony, if fitness of improvised harmony is better

end while

Optimal/suboptimal scheduling solution available

***e*nd**

Fig. 4.6: Pseudo code of harmony search based algorithm for cloud scheduling problem

The loop will terminate if the best harmony in the harmony memory doesn't change in last certain iterations, the value of which depends on harmony memory size. HMCR ranges between 0.7~0.95 and PAR value varies between 0.1~0.5. In each iteration, a new harmony is improvised by accepting best harmonies from the harmony memory and adjusting the components of new harmony. Harmony search takes advantage of

previous work done by considering harmony memory. Advantage of Harmony search is that it improvises a new harmony by considering all the existing harmonies in harmony memory and each component is considered independently during the improvisation. Consider an optimisation problem for resource scheduling in cloud with 2 VMs and 4 cloudlets. The execution time of cloudlets on VMs is given in Table 4.1.

Table 4.1: Execution time of cloudlets on virtual machines

Cloudlet	VM_1	VM_2
1	1	2
2	2	4
3	4	2
4	6	3

The algorithm begins with a set of random solutions as shown in Table 4.2, where VM_1, VM_2 denotes virtual machines and C_1, C_2, C_3, C_4 represents cloudlets. Since prime objective is to minimize the makespan, thus during the calculation of fitness, x_1 is given more weightage compared to x_2. Here we are taking x_1 as 0.7 and x_2 as 0.3. In each iteration worst harmony is identified and is replaced by a new harmony if the fitness value of new harmony is better than the worst harmony.

Table 4.2: Initial Harmony memory

S.No	Harmony		Makespan	Average Waiting time	Fitness value
1.	VM_1	C_1 C_3	7	1.25	0.18
	VM_2	C_2 C_4			
2.	VM_1	C_1 C_4	7	1.25	0.18
	VM_2	C_2 C_3			
3.	VM_1	C_2 C_4	8	1.00	0.169
	VM_2	C_1 C_3			
4.	VM_1	C_3 C_4	10	1.25	0.135
	VM_2	C_1 C_2			

Table 4.2 represents harmony memory with four harmonies along with their fitness values. Next a new harmony is improvised from the best existing harmonies which is further included in the harmony memory replacing the worst harmony as shown in Table 4.3. Finally harmony search will terminate if the best harmony in the harmony memory doesn't change in last certain iterations.

Table 4.3: Subsequent Harmony memory

S.No	Harmony		Makespan	Average Waiting time	Fitness value
1.	VM_1	C_3 C_2	6	1.50	0.21
	VM_2	C_1 C_4			
2.	VM_1	C_1 C_3	7	1.25	0.189
	VM_2	C_2 C_4			
3.	VM_1	C_1 C_4	7	1.25	0.189
	VM_2	C_2 C_3			
4.	VM_1	C_2 C_4	8	1.00	0.169
	VM_2	C_1 C_3			

For resource scheduling problem, cloudlets are considered as component variables and VM ids are considered as pitch limits. Pitch adjustment in scheduling problem corresponds to swapping VMs among two cloudlets of a solution. Proposed harmony search results possibly the best optimal solution for the resource scheduling problem. The convergence pace is accelerated because of the consideration of entire harmony memory during the improvisation process not just one or two harmonies. The proposed Harmony search based approach is compared with Genetic algorithm in terms of convergence time of algorithm.

4.6 Genetic algorithm

Genetic Algorithm (GA) is a metaheuristic algorithm which is used to solve the scheduling problem in cloud [17]. The pseudo-code of GA is described in Fig. 4.7.

Algorithm 5: Genetic algorithm for cloud scheduling problem

begin

Initialise population using random solutions

Define genetic algorithm parameters, *mutation and crossover rate*

while *(i<maximum number of iterations)*

Select better solutions from existing solutions for the next generation

Perform crossover operation based on crossover rate

Mutate the resulting off springs based on mutation probability

Evaluate fitness of new chromosomes

end while

Optimal/suboptimal scheduling solution available

***e*nd**

Fig. 4.7: Pseudo code of genetic algorithm for cloud scheduling problem

GA begins with random solutions called *chromosomes* which are generated randomly and there fitness value is evaluated. Next a loop will iterate which involves selection of candidates for the next generation which are further operated by crossover and mutation operation. Crossover operator exchanges information between the chromosomes where as mutation operator helps in escaping from the local optimum. Children replace their parents if their fitness value is better. Crossover operation is performed between two individuals i.e. only two individuals are considered during the improvisation process whereas Harmony search considers entire harmony memory during improvisation which thus results in better convergence pace. Moreover harmony search considers each component individually during improvisation but GA does not consider that.

Chapter 5
Simulation Results

This chapter covers the discussion of tools used for simulating the cloud environment and various implementation results. Chapter begins with discussion of CloudSim toolkit and ANT build tool followed by parameter settings of CloudSim used to simulate the cloud environment. Chapter ends with the discussion of results.

5.1 Requirement of Cloud simulators:

Deployment and configuration requirements of cloud applications differ from one application to other. It's a challenging task to evaluate the performance of Cloud related algorithms on real cloud environment. So the better solution is to simulate the cloud environment and evaluate the performance of algorithms on these simulated environments. The advantages of using simulated environment are as follows:

- Cost is not involved for the required infrastructure and services to test and evaluate new policies.
- Evaluating performance on simulated environment before is always better than directly deploying it on real cloud environment.

Evaluating the performance of cloud related algorithms under different load conditions, with different resource allocation policies and varying resource capabilities on real cloud environment incurs a lot of cost and thus simulation environment is preferred. If such platforms are not available, researchers need to depend on theoretical approaches which results in erroneous results and outcomes. So these simulated platforms like CloudSim eliminates the need of evaluating cloud related policies on real cloud environments and thus reduces the cost involved in infrastructure and environment setup of real cloud environment for evaluating new policies. Netbeans, CloudSim and Ant build tool are used for implementing the proposed approach. These tools are discussed in the next section.

5.2 CloudSim:

CloudSim is a simulation toolkit for simulating the Cloud environment [36]. It provides classes for modelling VMs, data centers, computational resources and

management policies like scheduling, load balancing etc. CloudSim is used to evaluate new strategies in the cloud such as Scheduling algorithms, Load balancing policies etc. CloudSim offers the following features:

- Support for simulating Cloud Computing environment including VMs, data centers and cloudlets etc.
- Provides platform for modelling the resource provisioning and allocation policies.
- Provides flexibility of switching between space-shared and time-shared allocation policies.

5.2.1 Design of CloudSim

CloudSim is a simulator whose building blocks are classes. Fig. 5.1[36] represents the class diagram of CloudSim simulator. The brief description of classes of CloudSim simulator is as follows:

- **Datacenter**: Datacenter class models the infrastructure services provided by the cloud service provider.
- **DatacenterBroker**: DatacenterBroker class act as a broker representing user which identifies suitable cloud service providers and negotiates with them for the resources which will satisfy the QoS requirements of users.
- **VirtualMachine**: VirtualMachine class models VM instance, which runs inside a Host and Host component is responsible for VM management during its lifetime. Host instantiate VM's and allocates CPU cores to VM's according to the scheduling policy defined at Host level.
- **Cloudlet**: Cloudlet class models the application and its complexity is represented by the computational requirements of the cloudlet.
- **BWProvisioner**: BWProvisioner class models the bandwidth provisioning policy to VM's running inside a host.
- **MemoryProvisioner**: MemoryProvisioner class is an abstract class that models the memory provisioning policy to VMs running inside a host. This component represents physical memory allocation policy among VMs. A new VM can be deployed on a host only if host has sufficient amount of free memory as requested for new VM creation.

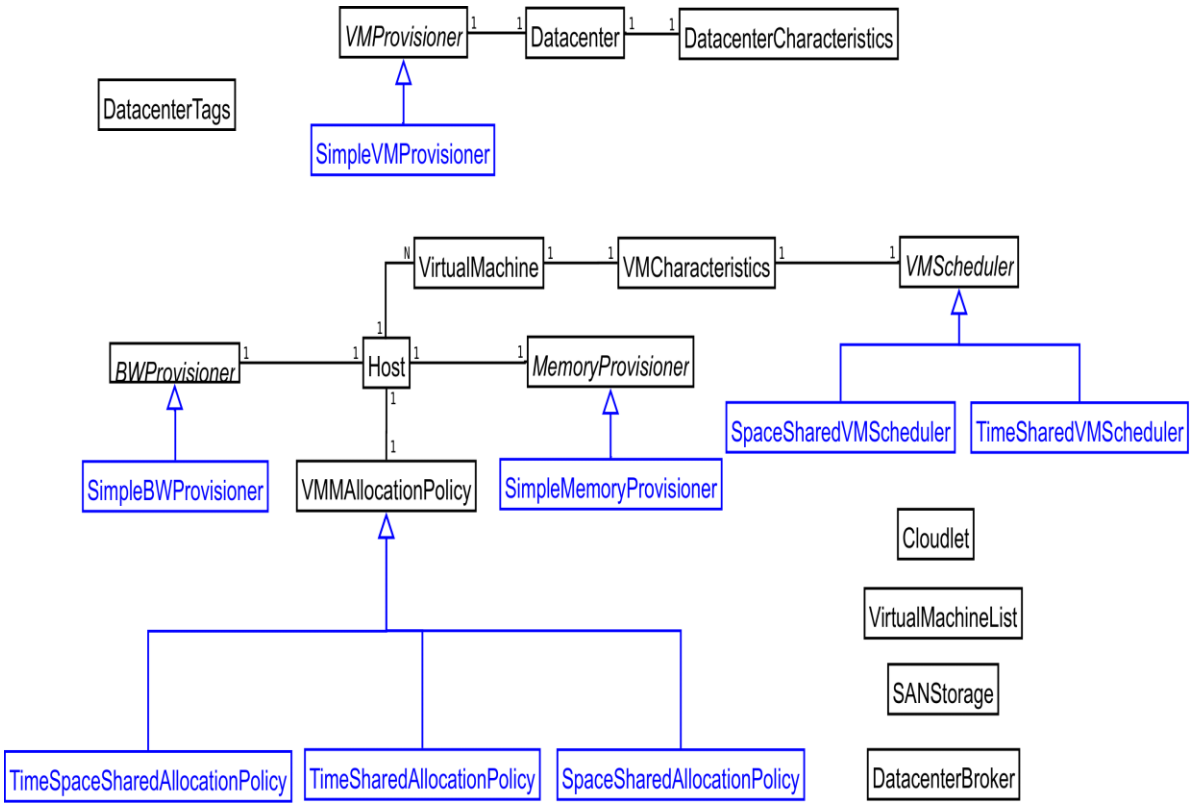


Fig. 5.1: Class Diagram of CloudSim simulator

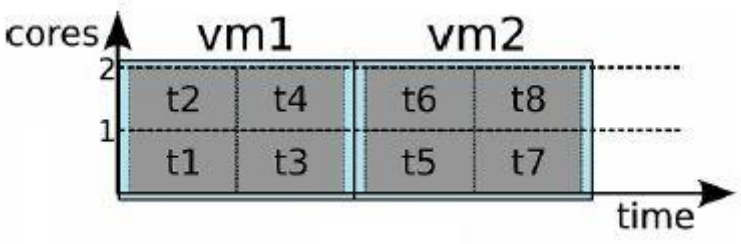
- **VMProvisioner**: VMProvisioner is an abstract class which models the allocation policy for assigning VMs to Hosts. The responsibility of the VMProvisioner class is to identify a suitable host within the datacenter which can satisfy storage, memory and availability requirement of VM deployment. The default SimpleVMProvisioner policy implemented in CloudSim allocates VMs to the first available host within the datacenter that can satisfy the VM deployment requirements and further allocation is done in sequential order. New provisioning policy can be implemented in this class.
- **VMAllocationPolicy**: VMAllocationPolicy is an abstract class. It models the allocation policies of assigning processor cores to VM.

5.2.2 Scheduling levels in CloudSim

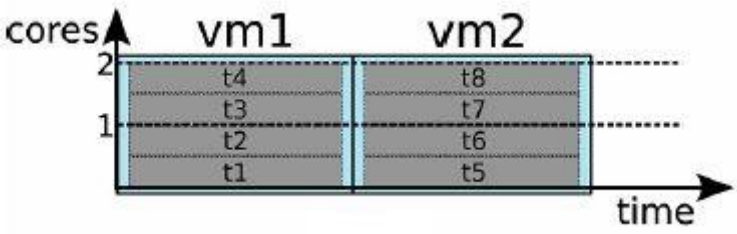
Datacenter comprises set of hosts and is accountable for managing VMs. In cloud, *Host* represents a physical computing node, which is configured in terms of its storage capacity, scheduling policy, processing capacity in Million Instructions Per Second (MIPS). CloudSim models resource provisioning at two levels: *Host level, VM level.* At host level, host specifies the fraction of the processor element to be assigned to

each VM running on it. This scheduler is also called *VmScheduler*. The allocation policy decides exactly how much processing capacity is to be allocated to a virtual machine. Two default policies are available at Host level: *xSpaceShared, xTimeShared*. SpaceShared policy includes allocation of specific CPU cores to specific VMs. Timeshard policy allows dynamic distribution of CPU cores to virtual machines i.e. on-demand allocation. Each host implements an allocation policy that can be TimeShared or SpaceShared or any other custom made policy for allocating CPU cores to VMs.

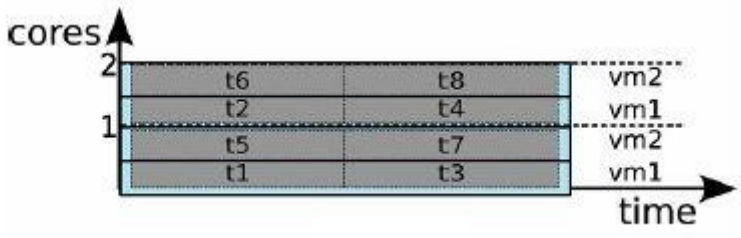
At VM level, VM shares the allotted resources among cloudlets scheduled to run on this VM. This scheduler is also called *CloudletScheduler*. Two default policies are available in CloudSim: xSpaceShared, xTimeShared which can be used in any combination at Host and VM level. To understand the scheduling policies implemented at Host and VM level, consider a scheduling scenario, a Cloud system with one host hosting two virtual machines, with each VM require two CPU cores each and eight task units to be executed on this system with tasks t1 - t4 to be executed on VM_1 and tasks t5 – t8 to be executed on VM_2. Various combinations of scheduling policies at Host and VM level are described below in Fig. 5.2 [36].



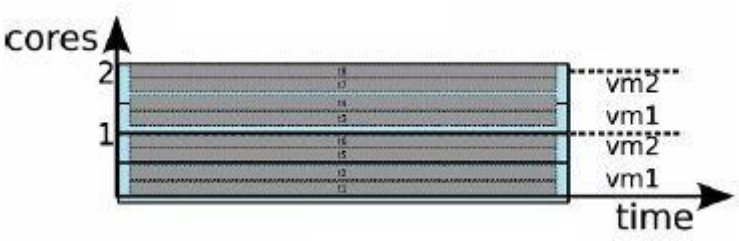
(a)



(b)



(c)



(d)

Fig. 5.2: Effects of different scheduling policies: (a) SpaceShared policy at Host and VM level, (b) SpaceShared policy at Host level and Timeshared at VM level, (c) TimeShared policy at Host level and SpaceShared at VM level, (d) TimeShared policy at Host and VM level.

Fig. 5.2(a) shows SpaceShared policy at Host and VM level. For simulation, requirement of each VM is two CPU cores and thus a single VM can run at any instance of time. Once VM_1 finishes execution of tasks only then VM_2 can be assigned CPU elements. Each task requires one CPU core, thus only two tasks can run simultaneously at VM level and other tasks are queued up.

Fig. 5.2(b) uses SpaceShared policy at Host level for allocating CPU cores to VM's and TimeShared policy at VM level. During lifetime of a VM, all the allocated tasks share the time by context switching until their completion. This policy reduces the waiting time for tasks but affects their completion time.

Fig. 5.2(c) shows TimeShared policy at Host level for allocating CPU cores to VMs and SpaceShared policy at VM level. CPU cores are allocated to VM's in time slices and within a VM these time slices of CPU cores are allocated to tasks in SpaceShared manner. Since the allocation policy used at VM level is SpaceShared thus only one task can be assigned the CPU core slice.

Fig. 5.2(d) uses TimeShared policy at Host level and VM level. Thus CPU cores are allocated to VM's in time shared manner i.e. each VM will be getting slices of CPU cores and at VM level these slices of cores are further time-shared among tasks. Thus there is no need to maintain queues at Host level and VM level.

5.3 Apache Ant tool:

Apache Ant is a java based tool used for building java applications [37]. Build file of Apache Ant is written in xml and is thus easy to understand and portable. Ant

identifies the dependencies among class files, compile them, assemble, test and build jar library. Features of Ant:

- Ant is platform independent and can perform platform specific tasks.
- Ant build files are xml files and are portable, easy to understand.
- Ant is used for automating the repetitive complicated tasks.
- Ant can integrate with IDEs like NetBeans and can be invoked from command prompt as well.
- Ant can also be used for building non-java applications.

Ant tool is thus used for compiling the modified source files of CloudSim and building the jar library.

5.4 NetBeans:

NetBeans is an open-source and free IDE which support programming languages like HTML, Java, JSP, Java Script, XML and PHP etc. [38]. NetBeans supports platforms like Windows, OS X and Linux. NetBeans provides built-in support of library classes and users can simply import these packages and thus need not to code from the scratch. Performance of proposed Harmony Search based algorithm is evaluated using Netbeans IDE and Cloudsim simulator.

5.5 Implementation Details:

The proposed Harmony Search based algorithm for cloud resource scheduling problem is implemented using CloudSim simulator by modifying its source code. The default policy used for allocating VMs to cloudlets in CloudSim is FCFS which is implemented in *DatacenterBroker* class. In the further section, implementation details of FCFS algorithm, RA algorithm, CRA algorithm, GA and proposed Harmony search based algorithm are discussed and interpreted.

5.5.1 Implementation details of FCFS algorithm:

FCFS is a traditional scheduling algorithm which serves the requests in the order of their arrival. For cloud resource scheduling problem, this approach allocates VMs to cloudlets in the order of their arrival. FCFS is default implemented in CloudSim. Simulation parameters of CloudSim used for the implementation of FCFS algorithm are described in Table 5.1:

Table 5.1: Simulation parameters of Cloudsim

Entity Type	Parameters	Value
Task (Cloudlet)	Cloudlet length	2500-25000 MI
	Number of tasks	10 – 30
Virtual Machine	Total Number of VMs	4
	MIPS	250-1000
	RAM	512 MB
	Bandwidth	1000
	Cloudlet Scheduler	SpaceShared
	Number of PE's	1 - 4
Datacenter	VmScheduler	TimeShared

Results are evaluated by considering fixed number of VMs and varying the cloudlets and are summarized below in Table 5.2:

Table 5.2: Makespan with varying Number of Cloudlets in FCFS

VMs fixed: 4	**Cloudlets**		
	10	20	30
Makespan	70	210	230

5.5.2 Implementation details of RA algorithm:

Random Allocation (RA) algorithm is a cloud resource scheduling algorithm which allocates VMs to cloudlets in a random order. This algorithm sometimes results in the best solution and sometimes ends up in the worst possible solution since there is no constraint imposed on the allocation of VMs to cloudlets. To implement RA algorithm, *submitCloudlets()* method of *DatacenterBroker* class is updated. After updating the source code, Ant tool is used for compiling and building the jar file. Simulation parameters used for the implementation of RA algorithm are described in Table 5.1.

Table 5.3: Makespan with varying Number of Cloudlets in RA

VMs fixed: 4	**Cloudlets**		
	10	20	30
Makespan	40 - 150	80 - 330	105 - 420

Since allocation is random i.e. VMs are allocated to cloudlets randomly, so results differ for each iteration. Results are evaluated by considering fixed number of VMs and varying the cloudlets and are summarized in Table 5.3.

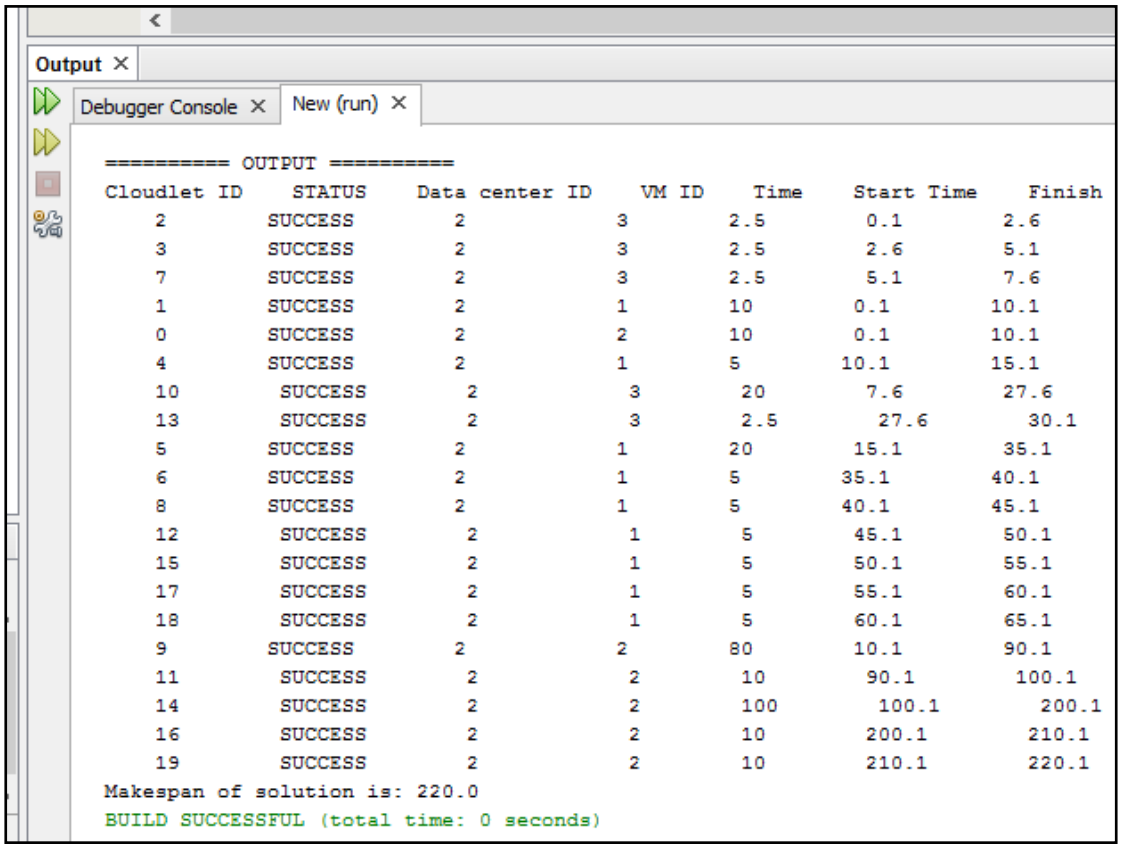


Fig. 5.3: Snapshot of console output for Random Allocation algorithm

Fig. 5.3 is a snapshot of output using RA policy as scheduling algorithm. Since the allotment of VMs to cloudlets is random, it can be seen out of twenty cloudlets, nine cloudlets are assigned a single VM with VM id 1. The results in terms of makespan vary greatly. Therefore algorithm gives results which ranges anywhere between best and worst possible solution and also causes poor resource utilization. The problem of uneven load distribution is handled in CRA algorithm which is discussed in next section.

5.5.3 Implementation details of Constrained Random Allocation algorithm:

This algorithm forces even load distribution among VMs by imposing a constraint that every VM can process cloudlets to a threshold. This threshold value is calculated as eq. 5:

$$z = ceil\ (a / b) \quad (5)$$

where z denotes threshold value, a represents number of cloudlets and b denotes number of VMs. This threshold value ensures even load distribution on VMs and ensures efficient resource utilization. To implement CRA algorithm, *submitCloudlets()* method of *DatacenterBroker* class is updated. After updating the source code, Ant tool is used for compiling and building the jar file. The results of this approach are evaluated by varying the number of cloudlets and taking fixed number of VMs. Simulation parameters in terms of VMs, Cloudlets and Host are described in Table 5.1. The results obtained after simulating CRA algorithm, in terms of makespan are summarized in Table 5.4.

Table 5.4: Makespan with varying Number of Cloudlets in CRA

VMs fixed: 4	**Cloudlets**		
	10	20	30
Makespan	40 - 100	80 - 210	100 - 300

Fig. 5.4 is a snapshot of output using CRA algorithm as scheduling policy for assigning VMs to cloudlets.

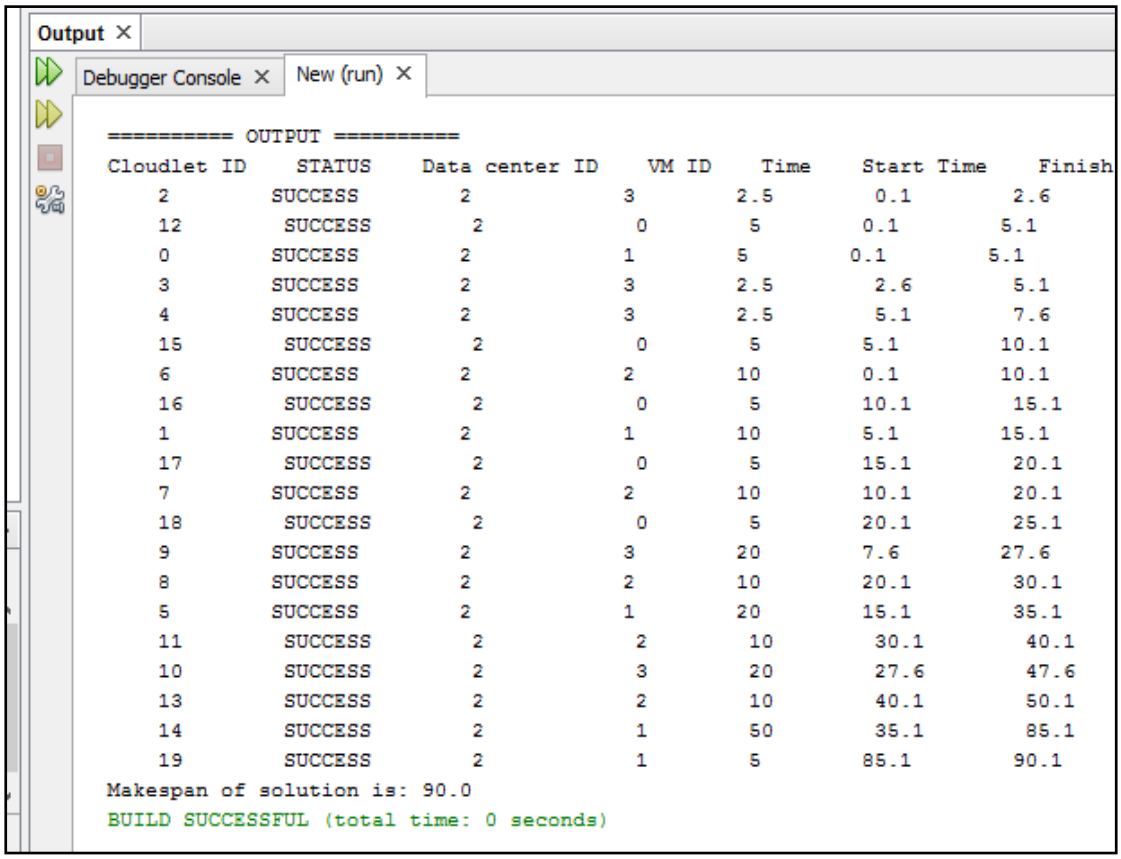


Fig. 5.4: Snapshot of console output for CRA algorithm

It can be seen from Fig. 5.4, allocation of VMs among cloudlets is evenly balanced and each VM is allocated to five cloudlets. Therefore, this algorithm gives better results in terms of makespan and resource utilization. This algorithm doesn't consider waiting time of cloudlets in the solutions. The outcome of the algorithm may involve assignment of light cloudlets to efficient machines and heavy cloudlets to less efficient machines which overall increases the makespan. The heavy and light cloudlets are distinguished in terms of load and complexity. These issues are considered and are eliminated in proposed Harmony Search based approach.

5.5.4 Implementation details of Harmony Search based algorithm:

Proposed Harmony Search based algorithm gives possibly the best optimised scheduling solution, converges quickly and also tries to minimise the waiting time for cloudlets. To implement the proposed algorithm, initial population is generated using the RA algorithm and new harmonies are improvised by considering HMCR and PAR which finally causes the algorithm to end up with optimal/suboptimal solution in terms of makespan and waiting time. The results of this approach are evaluated by varying the number of cloudlets and taking fixed number of VMs. Simulation parameters in terms of VM, Cloudlets and Host are described in Table 5.1 and parameters specifically used for Harmony search are described in Table 5.5.

Table 5.5: Simulation parameters of Harmony Search

Parameter	Value
r_{accept}	0.85
r_{pa}	0.10
Harmony Memory Size	10
Pitch Limits	0 - 3

The results obtained after simulating harmony search based algorithm, in terms of makespan are summarized in Table 5.6.

Table 5.6: Makespan with varying Number of Cloudlets in HS based algorithm

VMs fixed: 4	Cloudlets		
	10	20	30
Makespan	25 - 40	55 - 65	65 - 80

This approach gives possibly the best scheduling solution. The optimality of solution depend upon termination condition of the loop. The termination condition is based on various factors like harmony memory size etc.

To compare the results, each algorithm is run 10 times and average of makespan is considered. The evaluated results are summarised in Table 5.7.

Table 5.7: Comparison of FCFS, RA, CRA and proposed approach in terms of makespan

VMs fixed: 4		Cloudlets		
		10	**20**	**30**
Makespan	FCFS	70	210	230
	Random Allocation	79.5	150.25	194
	Constrained Random Allocation	54	116	150
	Harmony Search	30	60	70

Results presented in Table 5.7 are further evaluated in Fig. 5.5.

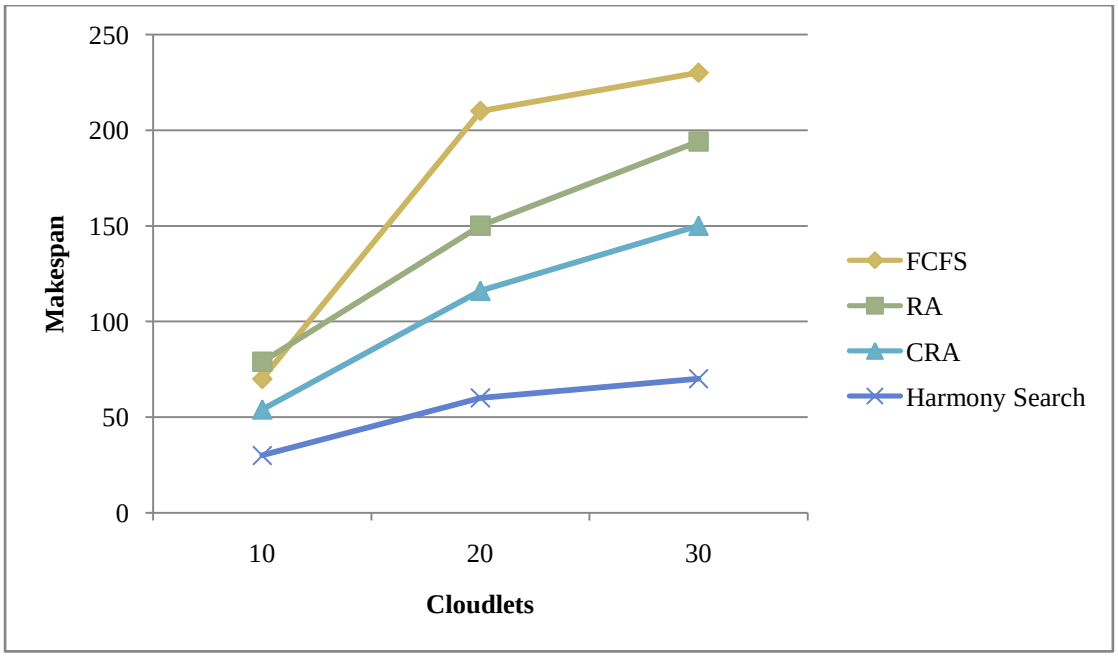


Fig. 5.5: Comparison of FCFS, RA, CRA and proposed approach in terms of makespan

Based on above results presented in Fig. 5.5, it can be evaluated that Harmony search based approach outperforms FCFS, RA and CRA algorithm in terms of makespan as performance parameter. Also this approach considers the waiting time of the tasks and solution is optimal in terms of makespan as well as average waiting time.

Harmony search begins with random set of solutions and slowly converges towards the optimal solution. The other parameter which is considered for comparison purpose is convergence time of algorithm. To evaluate the convergence time of algorithm, proposed Harmony Search based approach is to be compared with some another heuristic approach and we are thus using *Genetic Algorithm (GA)* for the comparison purpose.

5.5.5 Implementation details of Genetic algorithm:

GA is a metaheuristic algorithm which is implemented to solve the resource scheduling problem in cloud environment. The initial population is generated randomly using RA algorithm which is further operated by selection, crossover and mutation operator to find the optimal or suboptimal solution for the scheduling problem. Tournament Selection is used as used as selection operator, One-point crossover is used for crossover operation and Simple swap is performed for mutation purpose in GA. Simulation parameters used in the implementation of genetic algorithm are described in Table 5.8.

Table 5.8: Parameter settings of GA algorithm

Parameter	Value
Population size	20
Selection Type	Tournament Selection
Crossover Type	One-point Crossover
Mutation Type	Simple swap
Crossover Probability	0.7
Mutation probability	0.1
Fitness function	Makespan

The algorithm is implemented using CloudSim simulator and NetBeans IDE. Simulation parameters of Datacenter, Host and Virtual machine are described in Table 5.9. The results are evaluated by considering fixed number of VMs and varying the cloudlets.

Table 5.9: Simulation parameters of Cloudsim

Entity Type	Parameters	Value
Task (Cloudlet)	Cloudlet length	2500-7500MI
	Number of tasks	10 - 50
Virtual Machine	Total Number of VMs	10
	MIPS	250-1000
	RAM	512 MB
	Bandwidth	1000
	Cloudlet Scheduler	SpaceShared
	Number of PE's	1 - 4
Datacenter	VmScheduler	TimeShared

The results obtained after simulating the GA in terms of convergence time of algorithm are summarised in Table 5.10.

Table 5.10: Convergence time of GA with varying number of cloudlets

VMs fixed: 10	**Cloudlets Varying**		
	10	25	50
Convergence time (in sec.)	15.2	24.1	37.1

To compare the proposed Harmony Search based algorithm in terms of convergence time with Genetic Algorithm, proposed approach is simulated using the same parameters as described in Table 5.9. Specific parameters of Harmony Search are described in Table 5.11.

Table 5.11: Parameter settings of Harmony Search algorithm

Parameter	Value
r_{accept}	0.85
r_{pa}	0.10
Harmony Memory Size	20
Pitch Limits	0 - 9

Since the main objective is to minimise the makespan, x_1 is given more weightage as compared to x_2 while calculating fitness of a harmony and results are evaluated by

giving a 70:30 weightage to makespan and average waiting time of cloudlets. Results of Harmony Search based algorithm, in terms of convergence time are summarised in Table 5.12.

Table 5.12: Convergence time of Harmony search based algorithm with varying number of cloudlets

VMs fixed: 10	Cloudlets Varying		
	10	25	50
Convergence time (in sec.)	5	8.5	15

Results of Harmony Search based algorithm and Genetic algorithm in terms of convergence time of algorithm are compared in Fig. 5.6.

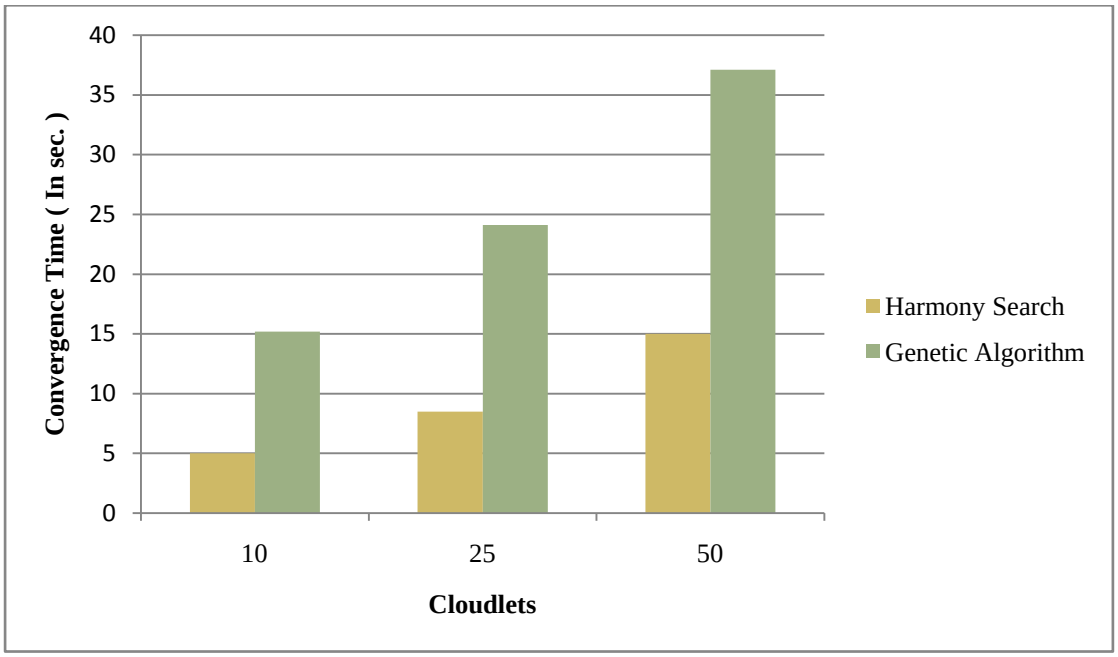


Fig. 5.6: Comparison of convergence time of Genetic algorithm and Harmony Search

Genetic algorithm selects two individuals and performs crossover and mutation operation i.e. G.A improvise a new solution by considering only two individuals from the population whereas HS improvises a new solution by considering all the existing harmonies. This causes the HS to converge quickly over Genetic Algorithm. Another important factor responsible for the delayed conversion of GA is the process it follows which involves Selection, Crossover and mutation operations which is time consuming.

5.6 Extended Results

Results can be further extended by analysing Harmony Search based approach and Genetic Algorithm for cloud resource scheduling problem.

5.6.1 Analysing effect of cloudlets on the convergence time

Effect of number of cloudlets on the convergence time in HS and GA is analysed in this section. Parameter settings used for simulating HS are described in Table 5.9 and Table 5.11 and settings used for simulating GA are described in Table 5.8 and Table 5.9. Results are evaluated by considering fixed number of VMs and varying the cloudlets and are summarised in Fig. 5.7.

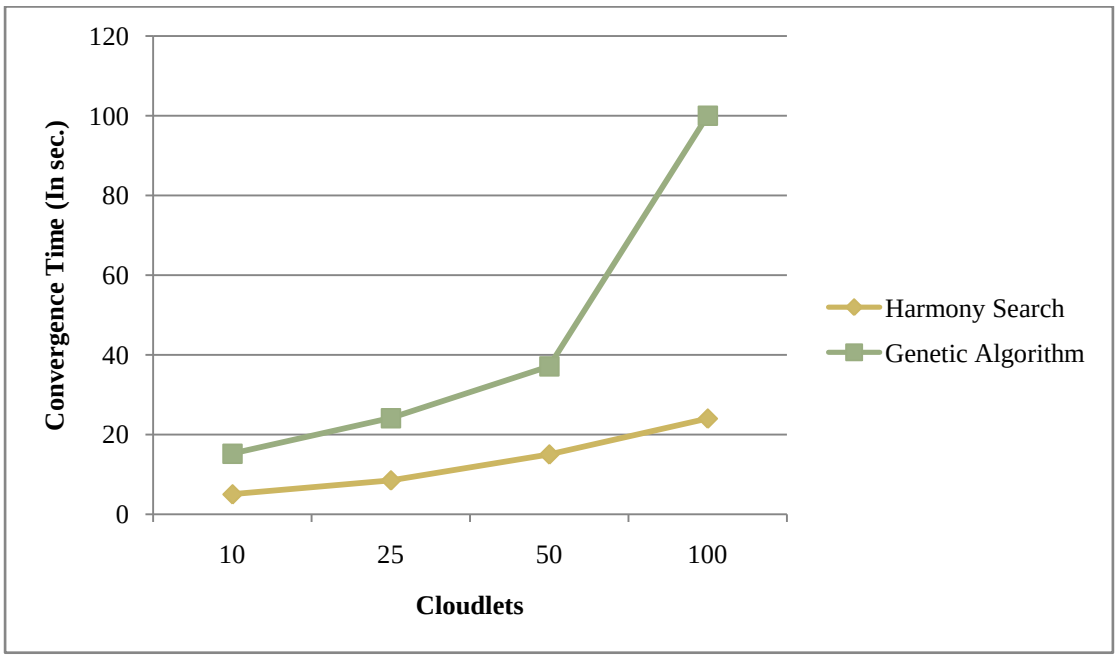


Fig. 5.7: Effect of cloudlets on the convergence time of GA and HS based approach

From Fig. 5.7, it can be evaluated that convergence time of H.S follows a linear growth with cloudlets whereas convergence time of GA increases rapidly with increase in the number of cloudlets. Improvisation process of H.S considers all the existing harmonies whereas GA considers only two individuals at a time during improvisation. This causes GA to take more time with increase in problem size. Also the process followed by GA is time consuming which will impact the convergence time of the algorithm.

5.6.2 Analysing effect on waiting time of cloudlets

This section covers the impact on the average waiting time of the cloudlets during the

scheduling process in H.S based approach and Genetic algorithm. Proposed approach considers average waiting time of cloudlets and makespan while calculating fitness of the harmony whereas standard genetic algorithm evaluates fitness only on the basis of makespan. Parameter settings used for simulating HS are described in Table 5.9 and Table 5.11 and settings used for simulating GA are described in Table 5.8 and Table 5.9. Results in terms of average waiting time of cloudlets are summarised in Table 5.13.

Table 5.13: Comparison of average waiting time in GA and HS based proposed approach

VMs fixed: 10		Cloudlets Varying		
		10	**25**	**50**
Average waiting time of cloudlets	Genetic algorithm	1.04	4.42	10.08
	Harmony search algorithm	0.69	3.88	8.9

Results of Harmony Search based algorithm and Genetic algorithm in terms of average waiting time of cloudlets are compared in Fig. 5.8.

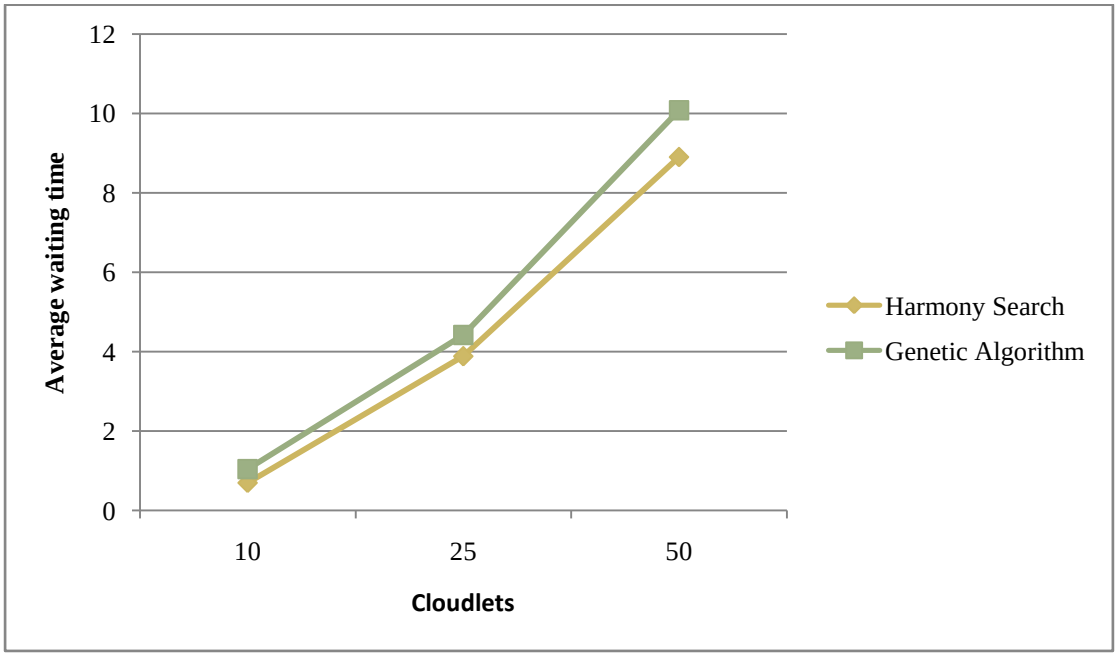


Fig. 5.8: Comparison of average waiting time in GA and HS based approach

From Fig. 5.8, it can be derived that the proposed HS based approach outperforms the standard GA in terms of average waiting time of cloudlets.

Chapter 6

Conclusion and Future Scope

This chapter concludes the thesis work and give directions how this work can be further extended.

6.1 Conclusion

Cloud Computing is an emerging paradigm used to provide services to the users via Internet. It's a challenging problem to manage cloud resources efficiently. Objective is to allocate resources to user's requests such that user requests are completed in minimum time and resources are used efficiently. In this thesis, gaps in the existing resource scheduling algorithms have been analysed and an efficient resource scheduling algorithm for Cloud Computing environment has been proposed. Proposed approach is evaluated on parameters like makespan, average waiting time and convergence speed of algorithm and is found to be effective. Proposed algorithm is implemented on CloudSim simulator and experimental results show that proposed approach is effective in terms of makespan, average waiting time and convergence speed of algorithm.

6.2 Thesis Contribution

Based on Harmony search, a new metaheuristic scheduling approach has been proposed to minimise the makespan, waiting time of cloudlets and converging time of algorithm. Proposed approach is successfully implemented on CloudSim simulator and results conclude that proposed approach is better than existing scheduling algorithms.

6.3 Future Scope

Work proposed in this thesis can be extended as:

- Proposed approach can be extended by using Max-Min and Min-Min algorithm to generate an initial population instead of taking random harmonies.
- Priorities of tasks can also be considered as a factor for calculating fitness

value along with makespan and waiting time of cloudlets.

- Dependency among the tasks can be considered further to extend the work.
- Parameters like deadline, energy consumption, budget restrictions, dependency constraint etc. can also be considered for extending the proposed work.
- Proposed work can be extended by considering a real time cloud environment where requests arrive dynamically at run time.
- Proposed work can be further extended by making the algorithm adaptive to the dynamic and varying QoS requirements of the cloud users.

References

[1] E. Qaisar, "Introduction to cloud computing for developers: Key concepts, the players and their offerings", in Proceedings of *IEEE TCF Information Technology Professional Conference*, 2012.

[2] F. Magoules, J. Pan and F. Teng, "Overview of cloud computing," *Cloud computing*, Boca Raton: CRC Press, pp. 1-17, 2013.

[3] R. Buyya, C. Yeo and S. Venugopal, "Market-Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities," in Proceedings of *10th IEEE International Conference on High Performance Computing and Communications*, Dalian, pp. 5-13, 2008.

[4] P. Mell, and T. Grance,"The NIST Definition of Cloud Computing," *National institute of Standards and Technology*, 2009.

[5] O. Brian, T. Brunschwiler and H. Dill, "White Paper Cloud Computing," *Swiss Academy of Engineering*, 2013.

[6] Q. Zhang, L. Cheng and R. Boutaba, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications*, Springer, vol. 1, no. 1, pp. 7-18, 2010.

[7]2016.[Online].Available:https://www.catcyfence.com/itsecurity/wpcontent/uploads/2012/01/cloud-computing.jpg.[Accessed: 15- Jun- 2016].

[8] T. Dillon, C. Wu and E. Chang, "Cloud Computing: Issues and Challenges," in Proceedings of *24th IEEE International Conference on Advanced Information Networking and Applications*, Perth, pp. 27-33, 2010.

[9] R. Buyya, "Introduction to the IEEE Transactions on Cloud Computing," *IEEE Transactions on Cloud Computing*, vol. 1, no. 1, pp. 3-21, 2013.

[10] S. Devipriya and C. Ramesh, "Improved Max-min heuristic model for task scheduling in cloud," in Proceedings of *International Conference on Green Computing, Communication and Conservation of Energy (ICGCE)*, Chennai, pp. 883-888, 2013.

[11] Z. Qian, L. Cao, W. Su, T. Wang and H. Yang, "An Improved Algorithm Based on Max-Min for Cloud Task Scheduling," *Recent Advances in computer Science and Information Engineering*, Springer, vol. 125, pp. 217-223, 2012.

[12] K. Etminani and M. Naghibzadeh, “A Min-Min Max-Min selective algorithm for grid task scheduling,” in Proceedings of *3rd IEEE/IFIP International Conference in Central Asia on Internet*, Tashkent, pp. 1-7, 2007.

[13] S. Sindhu and S. Mukherjee, “A genetic algorithm based scheduler for cloud environment,” in Proceedings of *4th International Conference on Computer and Communication Technology (ICCCT)*, Allahabad, pp. 23 – 27,2013.

[14] F. Omara and M. Arafa, “Genetic algorithms for task scheduling problem,” *Journal of Parallel and Distributed Computing*, Elsevier, vol. 70, no. 1, pp. 13-22, 2009.

[15] Zhongni Zheng, Rui Wang, Hai Zhong and Xuejie Zhang, “An approach for cloud resource scheduling based on Parallel Genetic Algorithm,” in Proceedings of *3rd International Conference on Computer Research and Development*, Shanghai, pp. 444 – 447, 2011.

[16] P. Singh and N. Sahu, “Task Scheduling in Grid Computing Environment Using Compact Genetic Algorithm,” *International Journal of Science, Engineering and Technology Research (IJSETR)*, vol. 3, no. 1, pp. 107-110, 2014.

[17] P. Kumar and A. Verma, “Scheduling using improved genetic algorithm in cloud computing for independent tasks,” in Proceedings of *International Conference on Advances in Computing, Communications and Informatics - ICACCI '12*, Kerala, pp. 137-142, 2012.

[18] G. Gan, T. Huang and S. Gao, “Genetic simulated annealing algorithm for task scheduling based on cloud computing environment,” in Proceedings of *International Conference on Intelligent Computing and Integrated Systems*, Guilin, pp. 60 – 63, 2010.

[19] Liu, C. Zou and P. Wu, "A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization in Cloud Computing", in Proceedings of *13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, Xian Ning, pp. 68 – 72, 2014.

[20] S. Shakya and U. Prajapati, “Task scheduling in Grid computing using Genetic Algorithm,” in Proceedings of *International Conference on Green Computing and Internet of Things (ICGCIoT)*, Delhi, pp. 1245-1248, 2015.

[21] H. Hoang, S. Le Van, H. Maue and C. Bien, “Admission Control and Scheduling Algorithms Based on ACO and PSO Heuristic for Optimizing Cost in Cloud

Computing," *Recent Developments in Intelligent Information and Database Systems*, Springer, vol. 642, pp. 15-28, 2016.

[22] C. Liu, C. Zou and P. Wu, "A Task Scheduling Algorithm Based on Genetic Algorithm and Ant Colony Optimization in Cloud Computing," in Proceedings of *13th International Symposium on Distributed Computing and Applications to Business, Engineering and Science*, Xian Ning, pp. 68 – 72, 2014.

[23] S. Chen, J. Wu and Z. Lu, "A Cloud Computing Resource Scheduling Policy Based on Genetic Algorithm with Multiple Fitness," in Proceedings of *IEEE 12th International Conference on Computer and Information Technology*, pp. 177-184, 2012.

[24] M. Tawfeek, A. El-Sisi, A. Keshk and F. Torkey, "Cloud task scheduling based on ant colony optimization," in Proceedings of *8th International Conference on Computer Engineering & Systems (ICCES)*, Cairo, pp. 64 – 69, 2013.

[25] Chen, L. Xiong and C. Wang, "Cloud Task Scheduling Simulation via Improved Ant Colony Optimization Algorithm," *Journal of Convergence Information Technology(JCIT)*, vol. 8, pp. 1139-1147, 2013.

[26] X. Wen, M. Huang and J. Shi, "Study on Resources Scheduling Based on ACO Allgorithm and PSO Algorithm in Cloud Computing," in Proceedings of *11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science*, Guilin, pp. 219 – 222, 2012.

[27] Ku-Mahamud and H. Nasir, "Ant Colony Algorithm for Job Scheduling in Grid Computing," in Proceedings of *Fourth Asia International Conference on Mathematical/Analytical Modelling and Computer Simulation*, Kota Kinabalu, pp. 40 – 45, 2010.

[28] A. Khalili and S. Babamir, "Makespan improvement of PSO-based dynamic scheduling in cloud environment," in Proceedings of *23rd Iranian Conference on Electrical Engineering*, Tehran, pp. 613 – 618, 2015.

[29] S. Abdi, S. Motamedi and S. Sharifian, "Task Scheduling using Modified PSO Algorithm in Cloud Computing Environment," in Proceedings of *International Conference on Machine Learning, Electrical and Mechanical Engineering (ICMLEME'2014)*, pp. 37-41, 2014.

[30] H. Liu, A. Abraham and A. Hassanien, "Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm," *Future Generation Computer Systems*, Elsevier, vol. 26, no. 8, pp. 1336-1343, 2009.

[31] S. Sharma and P. Kuila, “Design of Dependable Task Scheduling Algorithm in Cloud Environment,” in Proceedings of *Third International Symposium on Women in Computing and Informatics - WCI '15*, pp. 516-521, 2015.

[32] Z. Zhan, X. Liu, Y. Gong, J. Zhang, H. Chung and Y. Li, “Cloud Computing Resource Scheduling and a Survey of Its Evolutionary Approaches,” *ACM Computing Surveys*, vol. 47, no. 4, pp. 1-33, 2015.

[33] C. Tsai and J. Rodrigues, “Metaheuristic Scheduling for Cloud: A Survey,” *IEEE Systems Journal*, vol. 8, no. 1, pp. 279-291, 2014.

[34] Zong Woo Geem, Joong Hoon Kim and G. Loganathan, “A New Heuristic Optimization Algorithm: Harmony Search,” *SIMULATION*, vol. 76, no. 2, pp. 60-68, 2001.

[35] X. Wang, X. Gao and K. Zenger, “The Overview of Harmony Search,” *SpringerBriefs in Computational Intelligence*, Springer, pp. 5-11, 2014.

[36] R. Calheiros, R. Ranjan, A. Beloglazov, C. De Rose and R. Buyya, “CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms,” *Software Practise and Experience.*, vol. 41, no. 1, pp. 23-50, 2010.

[37] “Apache Ant,” *Wikipedia*, 2016. [Online].Available: https://en.wikipedia.org/wiki/Apache_Ant. [Accessed: 15- Jun- 2016].

[38] “NetBeans,” *Wikipedia*, 2016. [Online].Available: https://en.wikipedia.org/wiki/NetBeans. [Accessed: 15- Jun- 2016].

Video Presentation

Video Presentation Link ---- https://youtu.be/9C51dBTdc98

List of Publications

1. Chandan Malik, Sushma Jain and Sukhchandan Randhawa, "Resource Scheduling in Cloud using Harmony Search", International Conference on Inventive Computation Technologies (ICICT 2016),IEEE Explore,2016, Coimbatore, August 27, 2016**[Accepted]**
2. Chandan Malik, Sushma Jain and Sukhchandan Randhawa, "An efficient Resource Scheduling algorithm for Cloud Computing environment" **[To be communicated]**

report_-_Copy_2.docx

ORIGINALITY REPORT

13%	7%	10%	3%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	periyaruniversity.ac.in Internet Source	<1%
2	Submitted to Thapar University, Patiala Student Paper	<1%
3	www.academypublisher.com Internet Source	<1%
4	Teng, . "Overview of cloud computing", Chapman & Hall/CRC Numerical Analy & Scient Comp Series, 2012. Publication	<1%