Optimized Dynamic Data Replication for an Efficient Placement of Data in a Cloud Environment

Abraham Anbesu Abay

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February 2023

Adama, Ethiopia

Optimized Dynamic Data Replication for an Efficient Placement of Data in a Cloud Environment

Abraham Anbesu Abay

Advisor: Dr. Ravindra Babu

A Thesis Submitted to the Department of Computer Science and Engineering

School of Electrical Engineering and Computing

Presented in Partial Fulfillment of the Requirement for the Degree of Master's in Computer Science and Engineering

Office of Graduate Studies

Adama Science and Technology University

February 2023

Adama, Ethiopia

Declaration

I declare that this thesis entitled "**Optimized Dynamic Data Replication for an Efficient Placement of Data in a Cloud Environment**" is my own work and has not been submitted to any university for a similar purpose. The references used in this proposal are duly recognized by proper citations.

Abraham Anbesu	_____	_____
Name of student	Signature	Date

Recommendation

I, the advisor of this thesis, hereby certify that I have read the revised version of the thesis entitled **"Optimized Dynamic Data Replication for an Efficient Placement of Data in a Cloud Environment"** prepared under my/our guidance by Abraham Anbesu Abay submitted in partial fulfillment of the requirements for the degree of Mater's of Science in Computer Science and Engineering. Therefore, I recommend the submission of revised version of the thesis to the department following the applicable procedures.

_____	_____	_____
Major Advisor	Signature	Date

Approval Page of M.Sc. Thesis

I, the advisor of the thesis entitled **"Optimized Dynamic Data Replication for an Efficient Placement of Data in a Cloud Environment"** and developed by Abraham Anbesu Abay hereby certify that the recommendation and suggestion given by the board of examiners are appropriately incorporated into the final version of the thesis.

Dr. Ravindra Babu	_____	_____
Major Advisor	Signature	Date

Approval of Board of Examiners

We, the undersigned, members of the Board of Examiners of the final open defense by Abebe Abraham Anbesu have read and evaluated his thesis entitled **"Optimized Dynamic Data Replication for an Efficient Placement of Data in a Cloud Environment"** and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirement of the Degree of Masters Computer Science and Engineering.

_____	_____	_____
Chairperson	Signature	Date
_____	_____	_____
Internal Examiner	Signature	Date
_____	_____	_____
External Examiner	Signature	Date

Finally, approval and acceptance of the thesis is contingent upon submission of its final copy to the Office of Postgraduate Studies (OPGS) through the Department Graduate Council (DGC) and School Graduate Committee (SGC).

_____	_____	_____
Department Head	Signature	Date
_____	_____	_____
School Dean	Signature	Date
_____	_____	_____
Office of Postgraduate Studies, Dean	Signature	Date

Contents

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

LIST OF ACRONYMS

ABC	Artificial Bee Colony
CDR2S	Cost Aware Dynamic Re-replication and Re-balancing Strategy
CSP	Cloud Service Provider
EABC	Enhanced Artificial Bee Colony
EFS	Enhanced Fast Spread
EFS	Enhance Fast Spread
GB	Giga Byte
GCS	Google Cloud Storage
GFS	Google File System
HDFS	Hadoop File System
HEABC	Hybrid Enhanced Artificial Bee Colony
IaaS	Infrastructure-as-a-Service
IBM	International Business Machine
IDE	Integrated Development Environment
ITBDF	Improved Time-Based Decay Function
IWD	Intelligent Water Drop
MIPS	Millions of Instructions Per Second
MOABC	Multi Objective Artificial Bee Colony
MO-ACO	Multi Objective Ant Colony Optimization
NIST	The National Institute of Standard and Technology
OS	Operating System.
PaaS	Platform-as-a-Service
PE	Processing Element
PSO	Particle Swarm Optimization
QoS	Quality of Service
RAM	Random Access Memory
S3	Amazon Simple Storage Service
SaaS	Software-as-a-Service
SLA	Software level agreement
SLO	Service Level Objective

ACKNOWLEDGEMENT

First and foremost, I want to express my gratitude to God for giving me the time and strength to finish this thesis. next to that, I want to thank my advisor, Dr. Ravindra Babu, for his dedicated support and guidance. Next, I would like to thank the Director of Distributed System and Cloud Computing SIG Dr.-Ing. Frezewd Lemma, for his enormous support and follow up. Additionally, I want to thank all of my closest friends for helping me with this job by contributing their knowledge and suggestions. Finally, many thanks to all participants that took part in the study and enabled this research to be possible.

ABSTRACT

In cloud computing, it is important to maintain high data availability and the performance of the system. In order to meet these requirements, the concept of data replication is used. Data replication achieves the goal of effectively replicating the same data to different locations with zero loss of information in the event of a zero-downtime failure. Dynamic data replication strategies in the cloud (providing a runtime location for replicas) need to optimize key performance metrics parameters such as average response time, availability, cost, and load balance. The problem is, as the number of replicas of a data file increases, the data availability and the performance also increase, but at the same time, the cost of creating and maintaining new replicas also increases. This problem needs to find appropriate the replica to be selected and create the replica data based on its availability. Then place the replica by optimizing different parameters such as cost, distance, load balancing, etc. Because of this, the thesis proposes an Optimized Dynamic Data Replication for an Efficient Placement of Data in a Cloud Environment, which is a hybrid of Enhanced artificial bee colony (EABC) and Particle swarm optimization (PSO). These algorithms have certain drawbacks. ABC algorithm is the simple technique for data replication but it has the problem of inefficient resource utilization. PSO has a problem with execution time, and throughput when the environment is heterogeneous. The key aim of this thesis is to minimize the average response time and the replication cost, to reduce waiting time and data transmission time, and balance load. The proposed strategy is compared with another data replication approach in order to performance evaluation. A detailed performance evaluation study has been done to validate the proposed strategy. The proposed algorithm was tested and simulated on Cloudsim. Experimental results compared to different familiar algorithms. Based on the experiment results, the proposed strategy has reduced replication cost by 13.85% and waiting time by 18.37% compared to DCR2S, It has also reduce waiting time by 6% and cost reduction by 8.2 %, outperforming GA. And also reduced waiting time by 5.2% and replication cost reduction by 6.1% compared to ABC, and reduced waiting time by 4.7% and replication cost reduction by 5.7%, outperforming PSO.

Key Words: *Cloud Computing, Data Replication, Dynamic Data Replication, Replica Placement, data availability, system performance, replication cost*

CHAPTER ONE

1. Introduction

1.1. Background of the Study

Cloud computing is a large-scale parallel and distributed computing system (Ananthi & Hariganesh, 2015). The main purpose of cloud computing is to serve a simplified and proficient on-demand network access along with service to a pool of shared virtualized processing assets based on a pay-as-you-go agreement (Buyya et al., 2008). Recently, cloud computing has become an attractive and well-established solution for data storage, processing, and distribution (Shorfuzzaman & Masud, 2019). It provides on-demand, elastic computing and data storage resources without the large upfront investment normally required for traditional data center deployments. Cloud computing makes storage and computing resources available as a service without location restrictions (Shorfuzzaman & Masud, 2019). The main aim for the clients putting away the information in the cloud is to protect the information and recoup it at whatever point required. Any failure in the server ought not to bring about the loss of data (Sasikumar & Vijayakumar, 2020).

Cloud storage is a data storage model based on a virtualized infrastructure where data is stored in logical pools and physical storage is extended to various servers (Sun, 2012). Logical pools contain data that can be accessed by multiple users simultaneously. Cloud storage is purchased from a third-party cloud provider that owns and operates data storage capacity and makes it available over the Internet on a pay-as-you-go model. These cloud storage providers manage capacity, security, and durability to make your data accessible to applications around the world.

High availability, high fault tolerance and high efficiency access to cloud data centers where failures are normal rather than exceptional are significant issues, due to the large-scale data support. Data replication reduces user latency, speeds up data access, and increases data availability by providing users with all the different copies of the same service in a consistent state (Sun, 2012). Data replication is considered the most viable solution for improving access efficiency. Replication creates an exact copy of the original data in the same data center or remote location. It not only provides data availability, but also improves load balancing, fault

tolerance, and scalability (Waseem et al., 2021). Additionally, job execution time, bandwidth consumption, and performance are minimized. The services offered by the cloud incorporated infrastructure flexibility, cost control, faster application deployment, data adaptation of cloud resources to real needs, and improve profitability. There are two types of replications. (1) Static replication – is realized through a data resource pool, the number of replicas is historically based. However, this is not effective and you do not have to duplicate the entire block of data. (2) Dynamic replication - is a method of replicating data according to the various requirements of cloud users and as the environment changes (Nannai John & Mirnalinee, 2020). Still there are major concerns arising in the process of replication like which data should be replicated, when the replication should be done, where the replica should be placed, how many new replicas should be created, which data should be provided to the user and whether the load is balanced in storage.

Cloud storage and computing are widely used and recognized by many organizations. Optimally storing data in the cloud requires an optimal replication strategy for your system to achieve the goal of storing information in case an event occurs (Waseem et al., 2021). Therefore, to maintain the availability and general performance of the system, replicas are created and placed near to users. (Salem et al., 2020). Once a particular replica is established, it increases in availability and performance, but other sites increase the load, additional fees and higher cost to users. In fact, cloud providers aim not only to meet tenant requirements, but also to generate profits. Achieve tenant performance expectations without sacrificing provider profits and manage resource elasticity with pay-as-you-go pricing model, are the fundamentals of cloud systems.

1.2. Motivation

Nowadays, so much organizations are adopting and run their businesses through cloud services over all the world. One of an interesting service that is offered by CSPs is Storage-as-a-Service. Even if the privacy is the biggest threat, cloud storage has several advantages for the users: Cost, Accessibility, recovery, syncing and updating, security, etc. As various organizations around the world move to cloud IT services, huge amounts of data are generated and uploaded to cloud storage every day. One of an interesting mechanism in cloud storage in order to increase the availability of data and high performance is data replication.

Traditional systems such as distributed and parallel systems, peer-to-peer systems, and grid systems achieve goals such as achieving acceptable performance while ensuring good data availability, especially when data is distributed around the world. That is a big challenge for service providers. In this context, as a known technique, data replication can improve data availability, reduce data access costs, and improve fault tolerance. However, replicating data on all nodes is an impractical solution because it not only runs out of limited storage space, but also consumes significant bandwidth.

In cloud data storage, still there are an interesting problem that are related with data replication. In general, the main iteration issues arise of data replication between data availability, cost, and performance (Shakarami et al., 2021). Frequently used data should be replicated to multiple locations to increase data availability and performance. That way, users can easily access it from nearby sites. Other issues and challenges include data consistency, downtime during the creation of new replicas, maintenance overhead, and poor performance. Dynamic data replication strategies make decisions based on resource availability and current access patterns. The main issues are replica selection (data to duplicate) and replica placement (location). In addition to these two major replication issues, replication time (when to replicate) and replica quality (number of replicas to replicate) are also related issues. Based on these replication issues we consider to solve replica placement which is interrelated (interdependent) with other potential issues in cloud data replication. So, to ensure data availability and increase system performance, cost minimization, reduce latency and response time in cloud must be achieve by appropriate replica placement strategy.

1.3. Statement of the Problem

In a cloud-based replication, the data files are split into multiple blocks over the distributed data center. The goal is to have multiple copies (replicas) of the same data on different distributed data nodes. However, network-dependent factors within data-intensive applications cause node failures in large-scale cloud storage systems (Waseem et al., 2021). These network factors include bandwidth, node failures, and unreliable networks. If the node containing the data file fails, the entire data file is lost. Therefore, there is always a need for data availability by means of copying the original data file on numerous nodes across different data centers. Maintaining high data availability and system performance is critical in cloud computing. In practice, increasing the

number of replicas of a data file increases data availability and performance, but also increases the cost of creating and maintaining new replicas.

One of the major issues in cloud-based replications is replica selection. To meet the user requirement, such as to reduce the waiting time and increase the data access, replica selection must be addressed in cloud replications effectively. In adverse conditions, if early replication of a data file is done, or if the replica selection is not done efficiently, both conditions will lead unnecessary utilizing an extra storage space consumption and will increase its associated storage cost. We used a particular popular data for replica selection. The replica factor is one of the key factors of replica placement. The main two issues in the replica placement are how to determine the replica factor and how to select the optimal data node for storage of replica. Replica placement algorithm are categorized into two basic types, as static replica placement algorithm and dynamic replica placement algorithm. Static replica placement algorithm generates replica and selects data node at the initialization of the cloud storage system, while dynamic replica algorithm selects the optimal data node dynamically to store replica based on current available data. To determine an efficient replica placement, we design better optimization algorithm to replica placement followed by selection.

When considering new replication needs, one of the most important issues is determining an efficient location for transferring replicas. The proposed optimized dynamic data replication algorithm that provides better replica placement is considered the cost of transferring replica and DC, least cost path, bandwidth consumption, data transmission rate, and ensure replica availability while maintaining efficient access times.

In this research, we have designed an optimized dynamic data replication placement using Hybrid Enhanced Artificial Bee Colony optimization algorithm which decide to which replica file to replicate on an efficient data node to ensuring the accessibility of data. This approach minimizes latency and response time (increase system performance), and reduces the costs.

1.4. Research Questions

RQ1: How do optimize data replication by having efficient replica placement strategy in cloud environment?

RQ2: Which parameters should be optimized to reduce response time and increase access time to data replicas?

RQ3: How to evaluate dynamic data replication and placement algorithm in a cloud environment?

1.5. Objectives of the Study

1.5.1. General Objective

The general objective of the proposed study is to develop a system that finds optimal location for data placement through optimized dynamic data replication and placement strategy.

1.5.2. Specific Objectives

The following specific tasks have been performed to achieve the above general goals:

- Designing a strategy for replica placement to place the replica on the node nearest to user in order to reduce waiting time and balancing load.
- Identifying a suitable cloud architecture that meets the latency and cost optimization data replication algorithm requirements.
- Identifying a suitable simulator for experimentation based on the proposed data replication requirements over cloud environment.
- Comparing the waiting time and cost optimizing data replication algorithm's performance to existing approaches.

1.6. Significance and Beneficiary of the study

1.6.1. Significance of the Study

The proposed research has benefited all cloud computing users and cloud service providers. The proposed technique would provide the user with quick and flexible access to data. It has enabled the cloud service provider to meet the needs of clients in accordance with service level agreements. In other words, if a well-designed data replication strategy is not deployed, the service provider may be unable to meet the expectations of the customer.

This research work is beneficial to improving service quality by maintaining data availability using cost, bandwidth consumption and latency reduction.

1.6.2. Beneficiary of the study

The beneficiaries of the research are the following:

- **The Cloud end-users:** cloud end users can access the data within minimum time. A good data replication placement strategy will satisfy user requirements by improving system performance within the budget.
- **The Cloud Service Providers:** providers can easily utilize the storage services and the become more cost effective due to optimal replica placement of data, decrease data transmission time and load balancing of the system.
- **Future researchers and academicians:** The proposed algorithm can be used by researchers and academicians. It is also used at the university level for research and as an academic reference.

1.7. Expected outcome of the Study

Finally, this work should satisfy cloud service providers and end-users with ensuring the system performance by maintain data availability that increase quality of service. The result of this work has been to optimize the replica placement of data replica availability while access time.

1.8. Scope and Limitations

The main purpose of this research is to create optimized dynamic data replication placement algorithm. The way to improve data replication placement is by provide better replica selection and placement that must be considered to data availability, cost reduction, response time and bandwidth consumption while maintaining efficient access times. The propose algorithm implemented and test in cloud simulation tool, it doesn't consider real deployment environment due to its expensiveness.

1.9. Thesis Organization

Following are how the remaining portions of this study are structured: In addition to a thorough explanation of the concepts of cloud computing, cloud storage, and dynamic data replication, the second chapter looks at the literature review and related research that have already been done on the cloud environment in relation to data replication and placement. Chapter Three provides a detailed explanation of the proposed solution's design as well as the architecture adapted to satisfy the design requirements. The Chapter Four compares the proposed algorithm to existing approaches in terms of user budget, data transmission time, and response time using the Cloudsim simulator. In Chapter Five, the conclusion and future works of the research work have been discussed.

CHAPTER TWO

2. Literature Review and Related Work

2.1. Overview of Cloud Computing

Cloud computing is based on the concept of dynamic provisioning, which applies not just to services, but to computing capability, storage, networks, and information technology (IT) infrastructure in general. Resources are made available over the Internet and delivered on a pay-as-you-go basis by cloud computing providers. The utility-oriented nature of cloud computing expressed in (Buyya et al., 2008) A cloud is a type of parallel distributed system consisting of a collection of interconnected and virtualized computers that are dynamically provisioned and represented as one or more unified computing resources based on service-level agreements established through negotiation between the service provider and consumers.

As *the U.S. National Institute of Standards and Technology (NIST)* (Osorio et al., 2006) 'Cloud computing is 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'.

Cloud computing offers the illusion of infinite pool of highly reliable, scalable, and flexible computing, storage, and network resources in a pay-per-use manner. These resources are typically categorized as Infrastructure as a Service (IaaS), where Storage as a Service (StaaS) forms one of its critical components.

2.1.1. Cloud Computing Architecture

All concrete realizations of cloud computing can be organized in a layered view that covers the entire stack, from hardware appliances to software systems. Cloud resources are used to provide the "computing power" necessary to provide the service. This layer is often implemented using data centers with hundreds or thousands of nodes stacked together. Cloud infrastructures can be heterogeneous in nature because they can be built using a variety of resources such as clusters and networked PCs. Additionally, database systems and other storage services can also be part

of the infrastructure. The physical infrastructure is managed by the core middleware, the objectives of which are to provide an appropriate runtime environment for applications and to best utilize resources.

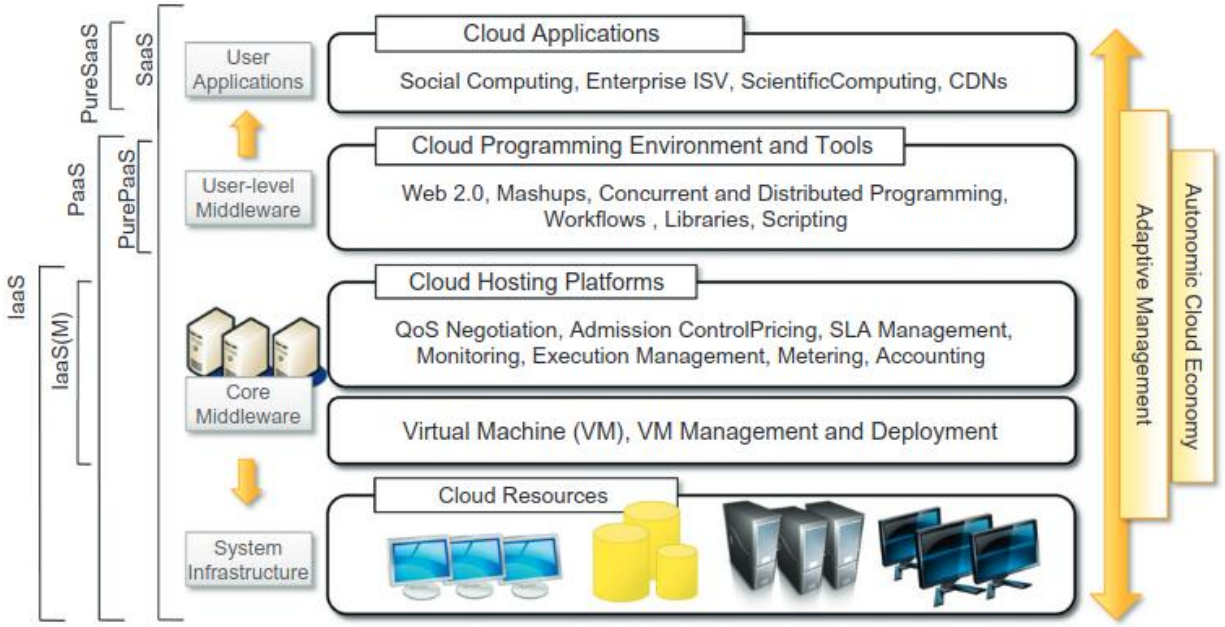


Figure 2.1 Layered Cloud Computing Architecture (Buyya et al., 2009)

2.1.2. Virtualization and Cloud Computing

Virtualization technology is one of the fundamental components of cloud computing, especially when it comes to infrastructure-based services. Virtualization allows you to create a secure, customizable, and isolated execution environment for running applications without affecting other users' applications, even if the applications are untrusted (Osorio et al., 2006). Virtualization plays a key role in cloud computing because it enables the right level of customization, security, isolation, and manageability that is fundamental to delivering IT services on demand. Virtualization technology is primarily used to provide configurable computing environments and storage.

2.1.3. Cloud Services

Cloud services are infrastructure, platforms, or software hosted by third parties and made available to users over the Internet. There are three basic types of as-a-service solutions: IaaS, PaaS, SaaS. Each facilitates the flow of user data from a front-end client to the cloud service provider's systems over the Internet, but varies by deployment.

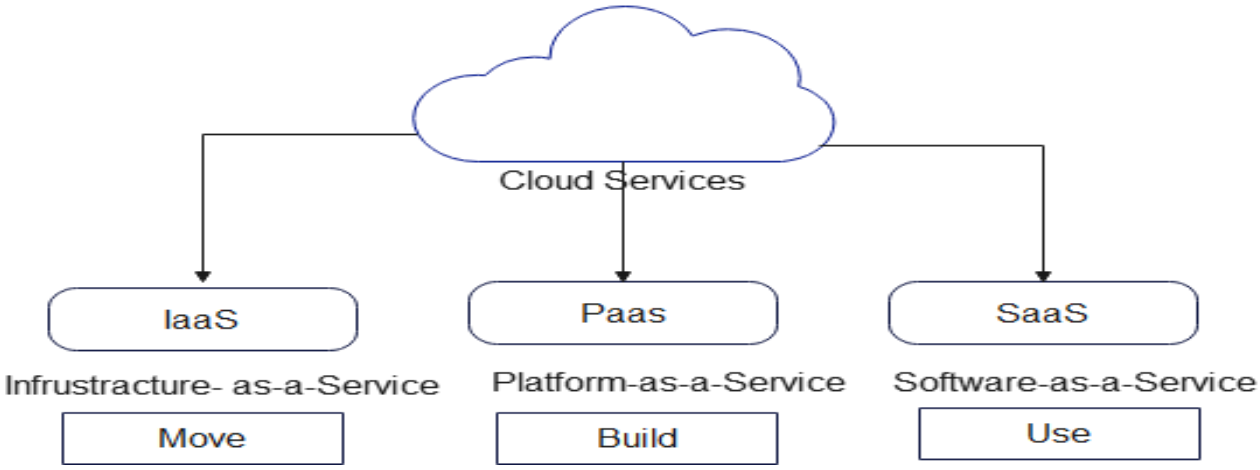


Figure 2.2 cloud services

I. Infrastructure as a Service

Infrastructure as a service is a cloud computing offering that vendor affords a customer access to hardware and software resources that can be storage, networking, servers, and other supporting resources including operating systems which afterward customers run their application and platforms within provided infrastructure (Rajan, 2012). These allow the client to pay only for infrastructure such as processing and storage that might be scalable depending on demands.

II. Software as a Service

Software as a Service allows the customer to use software applications that reside remotely over cloud infrastructure which means users do not install, configure and upgrade the application on their local devices but access it through a web browser (e.g., web-based Email) or an API. This removes the cost of hardware purchase, provisioning, and repairs, as well as software configuration, installation, and deployment (Buyya et al., 2008).

III. Platform as a Service

In this type of cloud service, cloud providers provide everything that developers need to build an application such as development tools, operating systems, and other infrastructures. All necessary management of tools for development will take place behind the scenes at cloud providers so that developers do not bother to update, install and configure the development environment (Gibson et al., 2012). The services offered by PaaS to its users are operating

system, database management system, server software, storage, network access, hosting, tools for design and development (Buyya et al., 2008).

2.1.4. Types of Clouds Deployment Model

Cloud deployment model describes how it operates and who gets access to hosted cloud resources depending on user needs that require a reduction in their capital expenditure. The deployment model represents the category of cloud environment based on user requirement, size, and access as well as describes cloud purpose and nature. There are four types of cloud deployment models such as private cloud for a private organization, public cloud for general users, hybrid cloud, and community cloud (Buyya et al., 2008).

Figure 2.3 Cloud deployment model

1. **Public Cloud-based**

The public cloud is the first expression of cloud computing. These are implementations of a standard view of cloud computing, where the services provided are made available to anyone, anytime, anywhere via the Internet. Structurally, they are distributed systems, most likely consisting of one or more interconnected data centers where specific services of the cloud are implemented. All customers simply need to register with the cloud provider, enter their access and billing data, and use the services offered.

Historically, public clouds were the first cloud class to be implemented and delivered. It provides a solution that minimizes IT infrastructure costs and serves as a smart option for handling peak local infrastructure loads. A fundamental feature of public clouds is multi-

tenancy. Public clouds are intended to serve a large number of users rather than a single customer.

2. **Private Clouds**

A private cloud is a virtual distributed system built on a private infrastructure that provides dynamic provisioning of computing resources to internal users. Instead of a pay-as-you-go model like the public cloud, there may be other systems that account for cloud usage and charge proportionally to different divisions or divisions of an organization. A private cloud has the advantage of keeping your core business in-house by relying on your existing IT infrastructure and reducing maintenance once the cloud is established. Security concerns are less important in this scenario because sensitive information never leaves the private infrastructure. In addition, private clouds can serve different user groups, making better use of existing IT resources (De Chaves et al., 2011).

3. **Hybrid Clouds**

It is a heterogeneous distributed system that from a private cloud that integrates additional services or resources from one or more public clouds. Because of this reason, it's also called heterogeneous clouds. A hybrid cloud allows organizations to leverage their existing IT infrastructure, manage sensitive information on-premises, and scale naturally by provisioning and freeing up external resources when they are no longer needed. Security concerns are limited to the public part of the cloud only.

4. **Community Clouds**

A community cloud is a distributed system created by integrating services from different clouds to meet the specific needs of an industry, community, or business sector.

2.1.5. Characteristics of Cloud Computing

National Institute of Standards and Technology states the cloud computing definition as: - "Cloud computing is a model that enables convenient, on-demand network access to a shared pool of configurable computing resources such as networks, servers, storage, applications, and services, quickly with minimal administrative effort or service provider interaction Can be

provisioned and shared". Five critical characteristics of cloud computing listed with the aid of NIST are, "on demand self-service, broad network access, resource pooling, rapid elasticity and measured service". The difference between cloud and other large-scale distributed computing platforms can be summarized as follows.

1. **On –demand**

Computer values like email, applications, network or server service can be presented without requiring human cooperation with each service provider. Cloud service providers providing on demand self-services comprises Amazon Web Servicing (AWS), MS, Google and International Business Machines (IBM and Salesforce.com.

A user can unilaterally provision computing capabilities, such as network storage, as needed automatically without requiring human interaction with each service's provider.

2. **Broad network connection**

Cloud capabilities are accessible over the network and accessed through normal mechanisms that further use by heterogeneous thin or thick client platforms such as mobile phones, laptops and Personal Data Assistant.

3. **Resource Pooling**

The provider's computing resources are pooled together to serve many consumers using multiple tenant model, with distinct physical and virtual resources dynamically assigned and reassigned according to customer requirement. The resources consist of amongst others storage, processing, memory, network radio bandwidth, virtual machines and email services. The pooling together of the resource builds economies of scale.

4. **Rapid Elasticity**

Cloud services could be quickly and elastically provisioned, in some cases automatically, to quickly scale out and quickly released to instantly scale in. To the consumer, the capabilities available for provisioning often appear to be huge and can be purchased in any quantity at any time.

5. **Measured service**

Their source usage can be measured, composed, and noted given that transparency for both the provider and consumer of the utilized service. Cloud computing services apply a metering ability which enables to control and optimize resource use. It is just related to air time; electricity or municipality water IT services are owed per usage metrics – pay per use.

2.2. Cloud Storage

Storage as a Service (StaaS) is a key component of cloud computing by providing the vision of a virtually infinite pool of storage resources. It supports a variety of cloud data store classes in terms of availability, scalability, ACID (Atomicity, Consistency, Isolation, and Durability) properties, data models, and price options. Application providers deploy these storage classes across different cloud-based data stores to address the challenges posed by relying on a single cloud-based data store, as well as to improve availability, improve response times, and It shortens the time and improves cost efficiency.

Cloud computing provides on demand services to cloud users, and one among them is storage. Nowadays, various applications produce terabytes or even petabytes of data that need to be efficiently analyzed, stored and processed (Wheeler & Winburn, 2015). Cloud storage is a service model in which data is transferred and stored on a remote storage system where it is maintained, managed, backed up, and made available to users over a network (usually the Internet). As data grows rapidly and needs to be kept safer and longer, organizations need to integrate how they manage and use it. Now we have the opportunity to store all our data on the internet.

A new paradigm of distributed cloud computing provides a virtualized pool of computing resources. Storage, applications, memories, computing power, and services are examples of these resources that customers can request when they need those (Shakarami et al., 2021). Cloud services can be divided into three models: IaaS (Infrastructure-as-a-Service), PaaS (Platform-as-a-Service), and SaaS (Software-as-a-Service).

Cloud storage is based on a virtualized storage infrastructure with accessible interfaces, near-instantaneous elasticity and scalability, multi-tenancy, and metered resources. Cloud-based data is stored in logical pools on various commodity storage servers located on-premises or in data centers managed by third-party cloud providers. Today, several cloud providers offer storage as

a service, such as Amazon S3, Google Cloud Storage (GCS), and Microsoft Azure, However, traditional storage vendors such as Dell EMC, Hewlett Packard Enterprise, Hitachi Data Systems, IBM, and NetApp have also products for both large and small business owners, including a self-service cloud portal for provisioning and usage monitoring.

2.3. Data Replication in Cloud Computing System

Data replication, a well-known technique in distributed systems, is the primary mechanism used in the cloud to reduce user latency, increase data availability, and minimize bandwidth consumption of cloud systems by offering the user different replicas with a coherent state of the same service (Sun, 2012). Data replication has been commonly used in traditional systems such as Peer-to-peer and data grid system (Mokadem & Hameurlain, 2020). Nowadays, data replication is mainly adapted to manage a large amount of data in a distributed way. Data replication increases the data access speed and reduces the access latency along with the data availability enhancements (Shakarami et al., 2021). There are two fundamental issues related to data process replication in cloud storage clusters. Adequate number of data copies and arrangement of copies to speed up system tasks.

Data replication is a very well-known data management technique that has been commonly adopted by many traditional systems, including database management systems (DBMS) (Jimenez-peris, 2014), parallel and distributed systems (Özsu & Valduriez, 2011) mobile systems (Rocha et al., 2015) and other large-scale systems including P2P (Xhafa et al., 2015) and data grid systems.

Generally, performance, availability, response time, costs, reliability, energy and bandwidth are the metrics of data replication in cloud computing. Replicas' selection, replica's placement, replica removal, load balancing, integrity checking, replication time, consistency models, security policies, and QoS/QoE/SLA are the factors of data replication in cloud environment (Shakarami et al., 2021).

Cloud providers offer seemingly infinite number of resources to meet ever-increasing storage and computational needs of the tenants by benefiting from filling datacenters with commodity hardware (Millán Tejedor & Esfandiari, 2014). Elastic scaling of abstracted resources also opened the doors for the cloud providers to offer an economy-based service model. These seemingly

infinite resources are rented to tenants as a utility with pay-as-you-go pricing model (Philip Chen & Zhang, 2014).

In a typical cloud environment, where frequent access requests are placed on a large-scale data, having low response time and high availability is crucial for the tenants. As in many systems, a number of data replication strategies have been proposed to improve the service quality through SLA-awareness in the cloud systems (Alami Milani & Jafari Navimipour, 2016). Part of the proposed SLA-aware data replication strategies are tenant-centric (Long et al., 2014). Some propose replication strategies that focus on minimizing consumption of a certain cloud resource, e.g., bandwidth, while others deal with replication of databases to satisfy the SLA without considering monetary impact of the replication decisions (Sousa & Machado, 2012).

Data replication is an integral part of the cloud and many distributed data management systems on the cloud such as Google File System (GFS) and Hadoop File System (HDFS) already implement some form of replication. As an example, HDFS creates triple replicas in a rack-aware fashion to enhance both the performance and availability. However, as the scale of the cloud systems enlarge as the number of datacenters and their geographical disparities increase to the scale of Internet, it is necessary for data replication strategies to address this trend.

Arguably the most widely considered objective by data replication in the cloud is the satisfaction of a desired level of availability. Availability of data sets often depends on the availability of the nodes hosting them or their system load, as overloaded nodes will not be able to serve requests. Increasing availability of data therefore be satisfied by increasing the number of replicas accordingly.

It is also possible to achieve the desired quality of service in terms of availability by calculating the minimum redundancy compared to heterogeneous cloud resources (Pamies-Juarez et al., 2011). In these heterogeneous environments, availability calculations are performed relative to each cloud location with different levels of reliability. This reliability heterogeneity also determines how many replicas a cloud site can reliably host as well.

Modeling data placement strategies to improve availability can also be accomplished with a custom economic cost model. Some have suggested an auction model for implementing replication placement policies in large cloud storage environments (Zhang et al., 2014). If the

required level of availability cannot be maintained, a bidding process will be conducted to determine the placement of new copies. The bid price depends on several properties of the node, including probability of failure, network bandwidth, and available storage capacity. Increasing availability by replicating popular data closer to the locations that originate the greatest number of requests is also a frequently researched problem. Some strategies predict future access based on past access records and propose mechanisms to preemptively replicate data to meet increased demand (Ridhawi et al., 2015). Moreover, some other strategies deal with data popularity that peaks for a short amount of time, and then tapers off (Qu & Xiong, 2012). To cope with the rapid increase in popularity, these data replication strategies react quickly and change replication configurations accordingly. Increasing data locality based on popularity also reduces network consumption.

(Wang & Wu, 2009) Deal with replica creation and placement in cloud storage systems dealing with balanced load. Authors aim to minimize costs of the service providers while maximizing storage utilization for users. They measure quality of service and calculate an importance metric of data sets. What to replicate is then determined according to the importance metric. Replica placement is performed in a way that maximizes data transfer volume per unit expenditure.

An interesting objective for data replication is to specifically target some expenditure that is particularly interesting in the cloud. In this respect, some data replication strategies (Boru et al., 2015) focus on minimizing energy consumption and communication delays through data replication. These strategies take into account data access frequency for regular replica reconfiguration. During this periodical assessment, the replication strategy predicts a future value of data sets that include energy and bandwidth demand. According to this estimation, a suitable placement is found for the replicas to minimize their power and bandwidth consumption in the following time periods.

Minimizing cost of resources can also be considered in a heterogeneous cloud context (Y. Mansouri et al., 2019). Datacenters in different locations can have varying costs for storage, network etc. for placing replicas. A replication strategy should take advantage of this heterogeneous pricing while responding with replication to future variations in workloads. As a common theme in many other distributed environments, reducing network transfers between datacenters through replication to both ensure performance due to faster access to data and

provider costs by utilizing more expensive network links less frequently is also present in the cloud systems context (Vulimiri et al., 2015).

Since SLA requires only the satisfaction of a given level of a quality-of-service metric, e.g., availability, performance; ideally, the provider should aim to supply the promised service quality, without trying to overachieve. In other words, pursuing best performance for the tenant is unrealistic and most likely a wasteful endeavor for the provider. Instead, the provider should focus on delivering an acceptable quality of service that will maximize its profits. Therefore, in cloud systems, major questions of what to replicate, when to replicate, how many replicas to create and where to place these replicas must be answered in such a way to satisfy the SLA in an economically feasible way (Computing et al., 2016).

2.3.1. Techniques of Cloud Data Replication

Data replication techniques in clouds are broadly labeled into two basic categories (Waseem et al., 2021), which include static replication mechanism and dynamic replication mechanism, and their summary is presented in Table 1.

Table 2.1 Static data replication versus dynamic data replication.

	Static Data Replication	Dynamic Data Replication
Brief Description	A predefined set of replicas and host nodes are the key factor to achieve the data distribution at multiple sites. Decide on the location of the replication nodes during the design phase.	A key factor in achieving multiple site data distribution is an adaptive method that can automatically create and omit replicas based on user behavior and network topology. Determines the location of replication nodes at run time.
Key Features	This strategy is closely related to a deterministic policy in which host nodes and replication numbers are pre-determined and very highly characterized.	By default, the dynamic strategy creates and removes replicas based on changes in storage capacity, bandwidth, and user access patterns (adaptive in nature).
	A static replication strategy is always easy to implement because the number of replicas is constant.	Since the number of replicas is variable (based on heterogeneous workloads), these strategies are not trivial to implement.
	To keep the maximum number of active service replicas, a random policy should be supported.	Intelligent in nature, dynamic data replication is designed to make intelligent decisions about where to store data based on currently available information.
Drawbacks	They are used less in real scenarios because of their predetermined nature.	It is very difficult to control and accumulate the runtime information of all the data nodes in a complex cloud setup.
	More active service replicas guarantee higher performance, but performance cannot be achieved at high operating costs.	It takes a lot of effort to effectively keep data files consistent.

2.4. Nature-Inspired Metaheuristics Algorithms

Each meta-heuristic algorithm makes a trade-off between randomization and local search. It should be noted that there is no single definition of heuristics and metaheuristics in the literature; some authors use the terms "heuristics" and "metaheuristics" interchangeably (Sedighizadeh & Mazaheripour, 2018). However, recently, there has been a tendency to refer to all stochastic algorithms involving randomization and local search as "metaheuristics." This concept is also applied here. Randomization is an efficient method for moving from local to global search. As a result, almost all metaheuristic algorithms can be used for global optimization.

Exploration and exploitation are the two most important parts of any metaheuristic algorithm. Exploration involves searching the entire search domain and obtaining different solutions, whereas exploitation involves finding the best available solutions in any domain, applying knowledge about the selected solution, focusing the search (local search) in that domain, and selecting the best candidate solutions (Hasançebi et al., 2010). Although randomization is useful for solutions in the exploration phase, it prevents the local optimum from being hit. The exploration stage allows the algorithm to search the solution domain more efficiently. In other words, the search area is explored to find the best solution; the neighborhood of that solution is searched; meanwhile, the search continues to find the best solutions (Hasançebi et al., 2010).

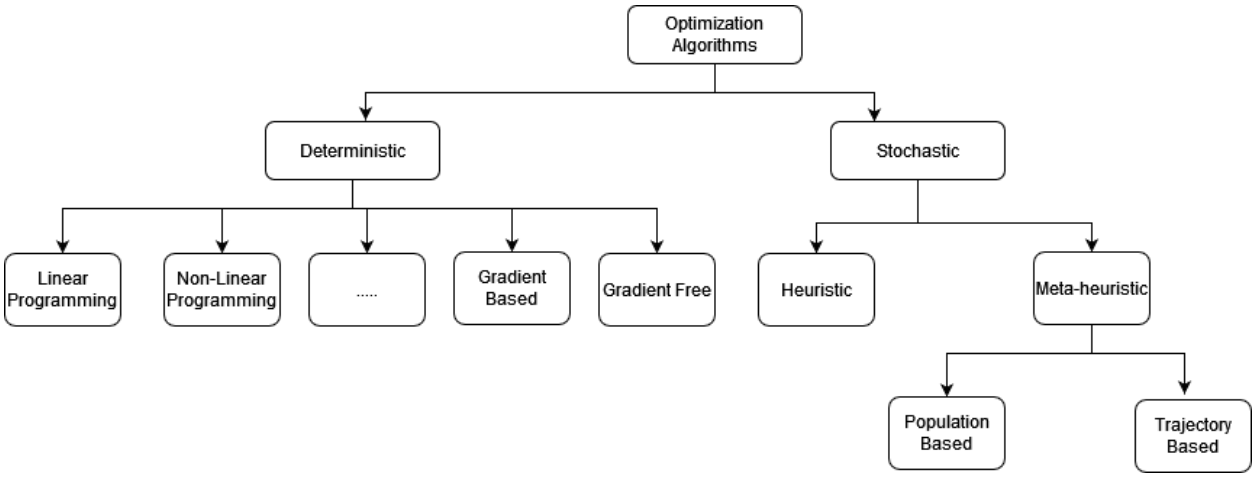


Figure 2.4 Classification of Optimization Algorithm (Kakandikar & Nandedkar, 2018)

There are many metaheuristics algorithms such as Genetic Algorithm, Artificial Bee Colony algorithm, Ant Colony Optimization, Grey Wolf Optimization algorithm, Particle Swarm Optimization algorithm, and etc.

2.4.1. Swarm Intelligence-based Algorithms

Swarm intelligence refers to collective intelligence. Biologists and natural scientists have studied social insect behaviors that can efficiently solve complex problems such as finding the shortest route between nests and food source or organizing their nests (Saka et al., 2013). These insects are very demanding as individuals, but by interacting with each other and their environment, they are prodigious swarm performers. Various swarm behaviors used in have been simulated with numerical optimization techniques. There is different swarm intelligence–based algorithms; ant colony optimization, particle swarm optimization, artificial bee colony algorithm, glowworm algorithm, firefly algorithm, cuckoo search algorithm, bat algorithm, and hunting search algorithm and etc.

Artificial Bee Colony Optimization

The ABC algorithm is proposed by Karaboga in 2005, and the performance of ABC is analyzed in 2007 (Karaboga & Basturk, 2008). The ABC algorithm was developed by examining the behavior of real bees in finding a food source called nectar and passing information about that food source to bees in the hive. In ABC, artificial agents are defined and divided into three types: employed bees, onlooker bees, and scout bees. Each plays a different role. Employed bees stay at the food source and remember the environment of the source. viewers get the information of food sources from the employed bees in the hive and select one of the food sources to gather the nectar; and the scout is responsible for finding new food, the new nectar, sources.

This emergent intelligent behavior in foraging bees can be summarized as follows (Akay & Karaboga, 2012a):

1. During the early foraging stages, bees begin to explore the environment randomly to find food sources.

2. After finding a food source, the bees become employed foragers and begin utilizing the food source found. An employed bee returns to the hive with nectar and unloads it After uploading the nectar, she can return directly to the source she discovered, or perform a dance in the dance area to share information about the source. If her source is exhausted, she becomes a scout and starts to randomly search for a new source.
3. Onlooker bees waiting in hives watch dances announcing beneficial springs and choose the location of the source site depending on the frequency of a dance proportional to the quality of the source.

Particle Swarm Optimization Algorithm

Particle Swarm Optimization is a random optimization technique (Juneja & Nagar, 2017). In the PSO algorithm, a group of particles in the search space work together to optimize a fitness function, much like a flock of birds might when foraging for food in the wild. The particles evaluate their quality or fitness at a given position in the search space after being distributed at random. Each particle then shifts to a new position that provides a higher fitness than the prior position for a predetermined number of iterations. Based on the history of each particle's best past and present locations in comparison to the best positions reached by other particles in the swarm, plus some random disturbances, this movement is determined (Lv et al., 2018). With a set number of particles cooperating, the swarm therefore finds in following iterations the fitness function solution that is most optimal in the problem space. Depending on the algorithm's intended use, the fitness or objective function in PSO is a performance evaluation criterion. A mathematical framework is typically used to describe the performance criterion in order to quantify the system performance attained through a performance index. The n particles that make up the fundamental particle swarm optimization technique each represent a potential solution to the fitness function in the D-dimensional search space. Three factors have an impact on the particle's status.

- Its own inertia.
- Personal most optimal position.
- Swarm’s most optimal position.

An Enhanced Artificial Bee Colony Optimization

An enhanced Artificial Bee Colony (ABC) optimization algorithm, which is called the Interactive Artificial Bee Colony (IABC) optimization, for numerical optimization problems, was proposed by (Journal & Computing, 2009). Onlooker bees should travel directly to the picking coordinates indicated by the employed bees and assess fitness values near the original artificial bee colony algorithm to reduce computational complexity. Therefore, ABC's exploration capacity in a single zone is limited. Based on the framework of the ABC, the IABC was introduces the concept of universal gravitation into the consideration of the affection between employed bees and the onlooker bees. By assigning different values of the control parameters, the IABC's gravitational pull should be included for varying numbers of employed bees and individual onlooker bees. Therefore, the exploration ability is redeemed about on average in the IABC.

2.5. Experimentation Tools

Cloud computing and fog computing simulation tools are virtual environments that are designed and developed to resemble physical environment aspects in order to reduce the expense of purchasing and configuring physical devices for developing and simulating research work. Cloudsim, Cloud Analyst, and iFogSim are the most frequently used simulators. Well-defined environments are very important for performing empirical evaluations of algorithms, as experiments cannot be performed in a real environment. The proposed approach is tested using Cloud Simulator tool which runs on eclipse IDE with a different number of independent tasks and different number of VMs.

I. CloudSim

Is an open-source java-based simulation tool with multiple layers for modeling, programming, and cloud-based simulation experiments. The GRID laboratory at Melbourne University created and developed it (Buyya et al., 2009) Cloudsim offers a variety of policies for virtual machine allocation, network connectivity within elements, processor core flexibility, and cloudlet management (Suryateja, 2016). The provision of computing resources, virtual machine configuration, data center administration, and user task management are all managed by classes in cloudsim. Researchers can use this simulator to create and develop cloud-related solutions without wasting time and money configuring physical devices. The main advantages of using

Cloudsim for initial performance testing are (i) time efficiency: implementing a test environment for cloud application delivery requires very little effort and time and (ii) flexibility and ease of use: developers can model and test the performance of their application services in heterogeneous cloud environments (such as, Amazon EC2, Microsoft Azure) with few programming and deployment requirements (Buyya et al., 2009).

In other words, it provides cloud computing environment modeling and simulation capabilities. Cloudsim claims that the user attempts to submit his requests in the form of cloudlets. Each cloudlet has its own file size, number of instructions to be executed, and so on. These cloudlets will be submitted to the broker to be scheduled onto VMs. The ability to create broker-driven policies gives Cloudsim an advantage. In cloudsim, the defined class, VM, represents the virtual machine that can be created on the hosts. The broker is responsible for creating hosts and assigning each VM to a different host.

Cloud Analysts

Cloud Analyst is a simulation tool that is built on top of the Cloudsim tool set and is used to model large scale Internet and Internet application behaviors in cloud and fog computing environments (Calheiros et al., 2011). The java programming language is used to create it. As a development environment, you can use any Java IDE, such as NetBeans or Eclipse. It adds additional extensions to the Cloudsim toolkit.

Application users: Cloudlets are used as traffic generators with user-configurable behaviors. **Internet:** network delays are used to define the transfer of tasks from IoT devices to fog nodes and cloud servers. Bandwidth constraints are used to model the uplink and downlink data transmission rates.

Service Brokers: This component manages a data center's CPU and virtual machines.

Graphical User Interface (GUI): The Cloud Analyst simulator has a graphical user interface (GUI) for setting simulation input and behavior. It is used to display experiment results and to allow users to repeat tests.

Saving simulations and results: The simulator has a feature that allows you to store simulations and results as pdf files.

Region: This component allows you to assess the proximity and distance between fog nodes and data centers.

Cloudsim was chosen for modeling of the suggested algorithm for this work because of its very extensive capabilities to help the assessment of cost and delay on cloud environments. Cloudsim simulator was chosen for this purpose since it supports both computing environments.

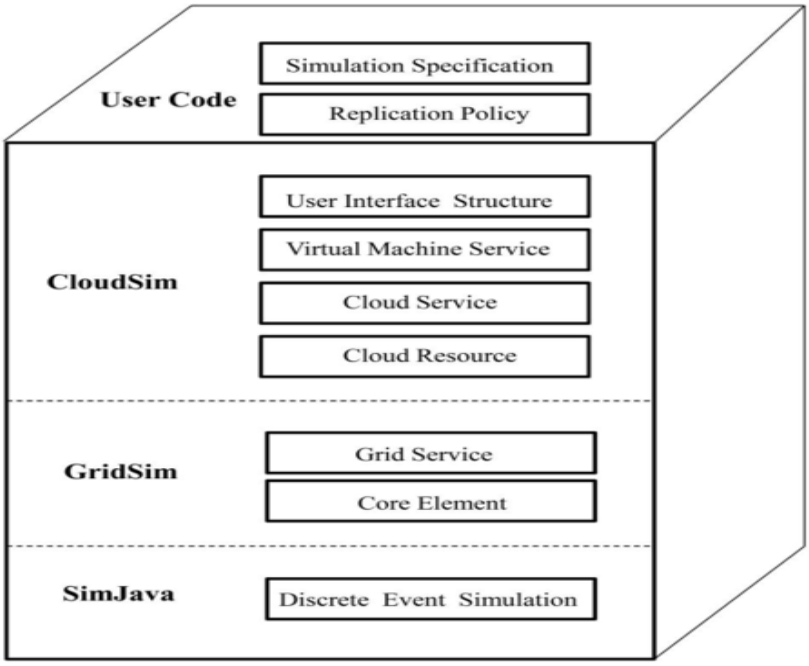


Figure 2.5 Layered CloudSim Architecture (N. Mansouri & Javidi, 2018)

2.6. Related Works

In (N. Mansouri et al., 2021) a CSO-based approach for secure data replication that determines suitable data center for new replica by designing a smart fuzzy inference system has been proposed. To obtain a higher level of security, they consider partition each popular file into several fragments with different sizes based on the ability of data centers. To do that, these fragments are stored based on the T-coloring concept to prevent an attacker from determining the locations of the fragments. Consequently, SDR protects the data file without any encryption technique since each data center has a single fragment of a particular file and no meaningful data are achieved in a successful attack. The proposed algorithm was evaluated with CloudSim

toolkit. The main reason is that SDR successfully balances the trade-offs among objectives by the fuzzy system.

(Janpet & Wen, 2013) designed a data replication strategy to minimize data access time by finding the shortest access path to data objects. They model access frequency, delay and replication budget to find the closest, most suitable node for replica placement. Replication budget is predefined and it is only used as a limiting factor for the users in such a way to regulate number of replicas. A detailed economic relationship between the users and the cloud provider is not addressed. The experimental study shows that by placing data objects closer to the nodes with high access frequency, response time is improved. In (Wei et al., 2010) a dynamic distributed cloud data replication algorithm CDRM is proposed. The CDRM is designed on the HDFS platform, the data replica placement is based on the capacity and location according to workload changing and node capacity, and the lower bound of the number of replicas is dynamically determined according to the availability requirement.

Based on users' access patterns, six different replica placement strategies are proposed in (Doğan, 2009). These are: No replication, Best Client, Cascading, Plain caching, Caching plus Cascading and Fast Spread. The ''No Replication'' strategy is to place all data files only at the root node of the hierarchical data structure with no replication. The ''Best Client'' strategy is to place the replicas of a data file at the client generating maximum number of requests to this data file. However, it is not efficient for all the access patterns. On the other hand, the ''Cascading'' strategy is to place the replicas in the direction and the path of the best client node using top-down approach. But it is efficient only when the locality of access patterns is known. Further, this strategy is to be used if the main focus is to reduce access latency. The ''Plain Caching'' strategy is to place the copy of the data file locally at the requesting client node. The ''Caching plus Cascading'' strategy is combination of the ''Cascading'' and the ''Plain Caching'' strategies. To reduce bandwidth consumption, the ''Fast Spread'' strategy is used, wherein the replicas are placed at all the nodes throughout the path of the client node whenever it requests to access the data file. This one is the best in case of random-access patterns. Since the prediction of user access patterns and the workload is very difficult to predict in advance. In (Bsoul, Al-khasawneh, et al., 2011), an advanced version of the ''Fast Spread'' strategy, namely the Enhanced Fast Spread (EFS) is proposed. The main drawback

of the ‘‘Fast Spread’’ strategy is that the replicas that are replaced may be more important than those replicas that are replacing them. Therefore, in EFS, the replacement is done keeping in mind the importance of the replicas. The importance of replicas is calculated using the information like the number of accesses and the frequency of these accesses along with the size of the replicas. All the six algorithms are dynamic replication algorithms and implemented in a distributed fashion.

A dynamic data replication strategy is proposed by (Gill & Singh, 2016) for a heterogeneous cloud environment. Their strategy keeps performance and availability at a desired level while optimizing the cost of replication. The cost of replication is calculated by the unit replication cost per datacenter and whether a replica is placed there. A metric called replica factor is used to calculate weighted access frequencies. Replica factor is compared with a threshold for making replication decision. Placement of replicas is performed with an optimization technique that gets lowest replication cost with highest value of replicas.

(Karuppusamy & Muthaiyan, 2016) have proposed a placement algorithm for data replication and to improve system availability in cloud environment. They have proposed dynamic replication and placement algorithms to improve the performance of the software system. Calculate the popularity level and replication factor to determine the appropriate file to duplicate. The most accessed files and the files that need to be duplicated are recognized and a copy of the file is made. Then place the replica using the proposed algorithm. Compare the performance of a technology with existing technology. Problems with this method are, first, the appropriate number of new replicas to create. As the number of new replicas increases, the maintenance cost of the system increases significantly. And the second, where to place new replicas to meet the requirements for successful system task execution and bandwidth consumption.

(Nannai John & Mirnalinee, 2020) they have proposed an optimization algorithm that manages replicas and thereby improves cloud storage performance. This novel dynamic data replication strategy is designed to improve the access efficiency of cloud storage. It is based on preserving multiple copies of a single item of data in geographically distributed locations, and then performing adaptive replication by shifting replicas among different regions so as to optimize access times while minimizing latency. This strategy also enables automated scaling according to user demand and allows for movement from caching/residential decisions. In addition, as

data replicas are frequently moved introducing high levels of redundancy can be achieved, thus improving reliability. By replicating items nearest to where they are used most often, overall performance can be significantly improved resulting in better user experience and improved system reliability. However, the strategy also has certain disadvantages such as increased costs due to replication, the possibility of conflicts in replicated data.

(Salem et al., 2020) have proposed an artificial bee colony algorithm for optimizing data replication in a cloud environment. This study presented costs in the cloud with regards to replication placement between data centers with multi-objective optimization. Artificial bee colonies are used to identify optimal replication placement by solving the low-cost issues that knapsack problem has used to solve this problem. Multi-objective optimization with the Artificial Bee Colony (MOABC) algorithm was used to achieve the required efficiency and lowest cost. The MOABC algorithm was tested with CloudSim.

(Awad et al., 2021) have proposed A Novel Intelligent Approach for Dynamic Data Replication in Cloud Environment. In this paper, two bio-inspired algorithms were proposed to improve both the selection and placement of data replicas in the cloud. The proposed algorithms for dynamic data replication are multi objective particle swarm optimization (MOPSO). The proposed algorithm, MOPSO is used to get the best selected data replica, depending on the most common ones and it is used to obtain the optimal placement of data replicas, depending on availability. The simulation used was CloudSim. The performance of these techniques was compared with variety of approaches such as Enhance Fast Spread (EFS), Genetic Algorithm (GA) and dynamic cost-aware re-replication and re-balancing strategy (DCR2S).

(Mohammadi & Navimipour, 2022) proposed a replica replacement to improve average access time and replica cost using fuzzy logic and Ant Colony Optimization algorithm. Ants can find the shortest path to discover the optimal node to place the duplicate file with the least access time latency. The fuzzy module evaluates the historical information of each node to analyze the pheromone value per iteration.

The following table shows a summary of each literature of review including their strength, weakness and ideas for which algorithm developed.

Table 2.2 Summary of literature review and related work

Authors and published year	**Main ideas**	**Limitation**	**Finding**	**Metrics**
(Gill & Singh, 2016)	A dynamic cost-aware replication strategy, which optimizes and identifies the least number of replicas that are required to maintain desired availability along with data reliability.	Low consistency rates, Low load balancing, and High response time.	When compared to other algorithms, their proposed algorithm used Low replication cost, High reliability, and High availability.	Reducing Cost,
(Nannai John & Mirnalinee, 2020)	They have proposed an optimization algorithm that manages replicas and thereby improves cloud storage performance by using Intelligent Water Drop Algorithm	Response time was not considered, low availability	Their proposed algorithm better bandwidth consumption, Enhanced storage space and number of Replications and consider load balancing.	Replication Cost
(Awad et al., 2021)	To improve both the selection and placement of data replicas in the	Energy inefficiency and Low load balancing.	It's used to get the best selected data replica, depending on the most common ones. and obtain the optimal placement of data	Replica Transferring

	cloud using two bio-inspired algorithms.		replicas, depending on data availability.	Cost and Distance
(Mohammadi & Navimipour, 2022)	To improves access to the information and increases the reliability of the system.	Low Load balancing, high bandwidth consumption	The study has been proposed an algorithm of replica replacement to improve average access time and replica cost using fuzzy logic and Ant Colony Optimization algorithm	Access Time and Replication Cost
(Mokadem & Hameurlain, 2020)	proposeD data replication strategies that meet tenant performance goals and provider interests in cloud data centers.	Low data availability, High response time	New replicas are created in a truly balanced manner only when a suitable replica placement node is heuristically discovered and the SLO_RT is met again while providing economic benefits to the provider.	Execution cost, load balance
(Tos et al., 2021)	a dynamic data replication strategy to satisfy both the response time objective and provider economic benefit in the cloud.	Low availability, medium load balancing	High consistency, Low bandwidth consumption, low storage cost.	Cost, bandwidth consumption

(Salem et al., 2020)	Swarm intelligence-based algorithm that is find least cost path to replica placement by optimizing the distance between DC.	High waiting time, low availability	In addition to using knapsack problem, they introduced some distribution methods. Select the popular replica and calculate the shortest path to place it.	Cost, Transmission time.
(Beigrezaei et al., 2021)	By proposing fuzzy inference system to avoids the imposed delays by considering a comprehensive set of significant parameters to improve the performance	High replication cost, Low response time, low availability	The proposed algorithm uses the fuzzy decision to select the best place for replication and the best replicas to delete	Access time, latency

CHAPTER THREE

3. Designing and Implementation

This chapter discusses research methodology, designing proposed solutions, algorithm system model, and implementation of the proposed solution.

3.1. Research methodology

In this section, the we discuss in more detail the research strategy, method, approach, the research measures or metrics, the research procedure or technique, the simulation system, the method of testing and evaluation, the algorithm used.

3.1.1. Research Strategy

This research is carried out by reviewing several relevant literatures work in cloud computing environments within the domain of data replication. To accurately collect the relevant resource, the author has analyzed written articles, research papers, a forum on the identified problem, and also visited the official technology sites to look around specifications, methodology, technology stack, conventions, and archetype. Additionally, some significant papers printed by specialists and authors are reviewed. In conduct this work, the following brief steps apply to the research strategy.

- Defined the identified problem clearly to be resolved.
- Requirements for the design of the proposed solution have been established.
- Collect the relevant resource by reading written articles, research papers, a dedicated forum on the problem found, visiting official technology sites to look at requirements, methods, application stacks, conventions, and models.
- Implement the proposed solution
- Prepare to conduct the experiment using various mechanisms that suit the problem identified.

- Document the results the analysis obtains

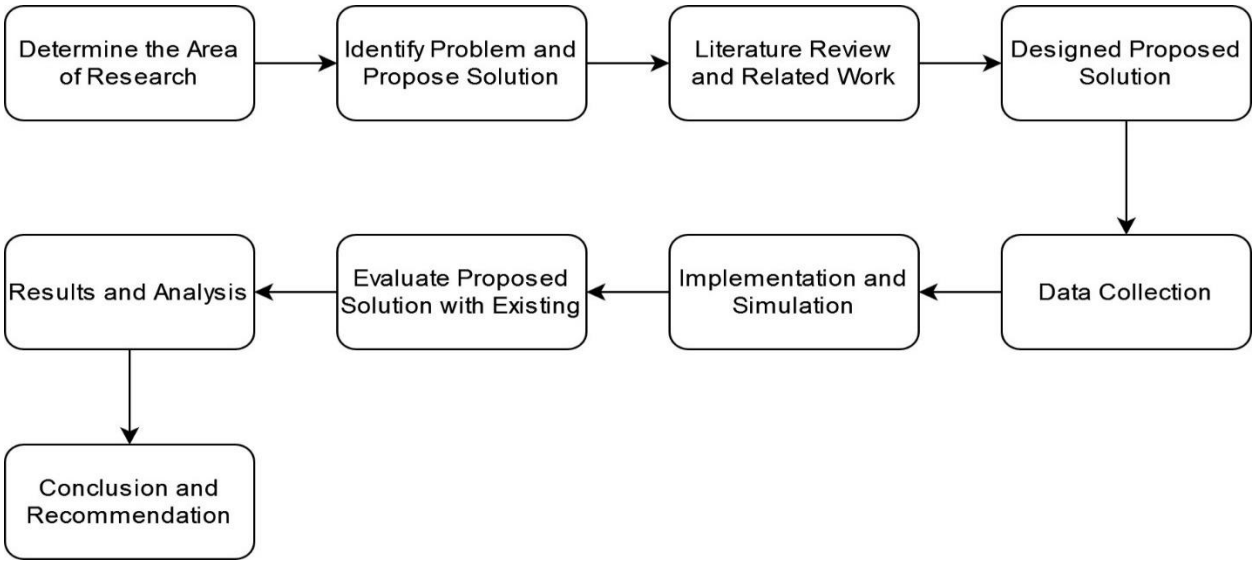


Figure 3.1 Procedure of the Study

3.1.2. Method

Quantitative research was held to fulfill the goals of the thesis. The main characteristic of this kind of research is, it is best suited for both the small dataset and the huge dataset, whereas its results are measurable and quantifiable. Its straightforward benefit, which also establishes its fundamental difference with qualitative research, is that quantitative data is more efficient, capable of testing hypotheses, by restricting the research scope and the feature of the responses of the participants, i.e., not subjective (interpretation of events by individuals may not be important).

Generally speaking, the reliability of quantitative research is based on the researchers' skills and abilities and also on the value of the data used, while the findings shall view as accurate because the results do not originate from the personal opinions and perceptions of the researcher.

3.1.3. Research Approach

The investigative approach followed for this research was the deductive one. According to this approach, researchers start exploring an identified phenomenon and testing whether that theory is valid under certain circumstances. The reason for the deductive approach was that it takes

into account the context in which the research effort is active, while it is also most suitable for small as well as large samples that produce qualitative data or results (the reliability of the research results cannot be questioned)

3.1.4. Simulation environment and Design tools

There are many simulation toolkits for cloud and fog environment, but the proposed task-scheduling algorithm is simulated by using the Cloudsim simulator toolkit. Because this toolkit is more appropriate with our selected requirements. Also, the reason behind choosing Cloudsim is that it allows cloud providers to test their services in a repeatable and controlled environment free of cost and to tune the performance bottlenecks before deploying on real-world Clouds. On the simulation side, the simulation environment permits different kinds of resource allocation under different load distributions.

Eclipse, a development environment, has been chosen as the best tools for implementing the proposed solution. Eclipse includes a Java Development Kit for Java developer. Wondershare EdrawMax is a design tool that allows software developers to create diagrams such as flowcharts diagrams. In the research, this tool is used to design and draw diagrams such as flowcharts and workflows. And Java programming language because Cloudsim framework is designed to work with JDK.

3.1.5. Research Metrics

This research will analyze the advanced techniques, and algorithms in order to design and evaluate the proposed algorithm in terms of selected metrics. The criteria to evaluate the proposed approach are: -

- Replication cost
- Waiting time

3.1.6. Testing and Methods of Evaluation

The proposed work or approach is tested and evaluated by using a given evaluation metric as explained above. Moreover, the projected approach is equated with other peer algorithms, and also the original algorithm to analyze whether the defined objective of the research is

succeeded or failed. Finally, the result is documented using the graph and other methods and the final document is prepared like this.

3.2. Designing Proposed Solution

As it already indicated, this study provides a detailed discussion of the optimized dynamic data replication placement algorithm's latest architecture, policy, design consideration, and parameters. The use of the hybrid EABC optimization algorithm to build a data replication design that fits the data availability while saving costs and mapping to virtual machines is extensively discussed. Appropriate replica file selection and determine efficient replica placement improves the quality of services such as availability, fault tolerance, cost, and access time.

While our algorithm performs, other factors like running at the background controlled by cloud system that use container orchestration technologies like Kubernetes or Docker Swarm to manage and control these background services and processes. The system also monitors resource utilization and usage of the cloud infrastructure to ensure that no other processes are consuming unnecessary resources that can slow down the replication process.

3.1.1. Dynamic Data Replication Architecture in the Cloud

The hierarchical cloud system architecture (Bsoul, Al-Khasawneh, et al., 2011, Shorfuzzaman et al., 2010), supports an efficient method for sharing data and computational and other resources. It usually consists of different tiers of data centers with different regions and sizes.

The super data centers in tier 0 will handle the data analysis in the intra domain and exchange data information among the inter domains. The main data centers are in tier 1, ordinary data centers are in tier 2, and users are in tier 3. The architecture minimizes the data access time and network load by creating and spreading replicas from the super data centers to main data centers, or to ordinary data centers. The super data center regularly collects and distributes global information.

A cloud data services system typically consists of scheduling brokers, replica brokers, and data centers. Scheduling Broker is the central management broker. Replication Manager contains general information about the replication sites in your data center.

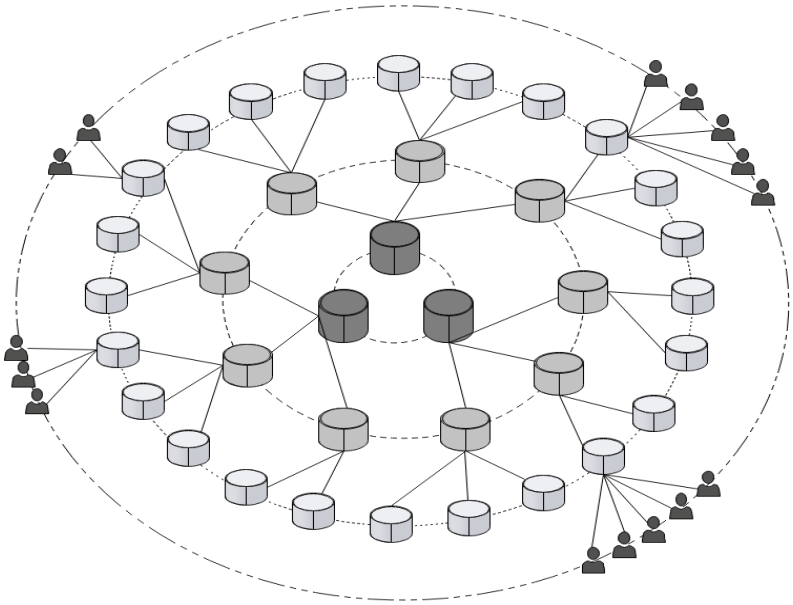


Figure 3.2 hierarchical cloud system topology

3.1.2. Main Components of the System Model

Data Centers

There are three types of different Data centers. Super DC, Main DC, and Ordinary DC. Every DC consists of different components, such that the VM differs from DCs, and so on. At the same time, it differs in terms of RAM, CPU, PE (Processing Element), etc. DCs differ in terms of the storage cost and number of available replicas.

Hosts

Host is a physical machine that can run multiple virtual machines on it. Each data center consists of multiple hosts.

Scheduling and Replica Broker

The scheduling broker is the central managing broker. The scheduler reads the service list, and uses the service description file as input to deploy each new service. The replica broker is used to manage the activities of the replica files. The control will first move from the user to the scheduling broker where it schedules the user request and pass the control to the replica broker. Here it indicates the replica files to be created and sends it the replica manager to decide where to place these replicas.

Replica Manager

The replica manager controls the overall operation of replication management system. It helps to create and manage replicas based on your needs. It increases the storage space and a directory to keep track of all the replicas and their locations. The directory information is configured by the replica manager. The log in replica manager is used to schedule the request and redirects the suitable replica copies. It holds the general information about the datacenter and replica location in the region. It provides replica services and manages the replica access, consistency, core replica creation, deletion and authentication.

Replica Catalog

Each newly created replica is registered in the replica catalog table. Replica catalog is also responsible for locating requested data and maintains the number of user bases, datacenter, replicas in region, number of requests at certain time period and availability. When each time, the site stores a new replica it sends a file register request to replica catalog and then replica catalog add this site to the list of sites and hold the replica. The catalog is then queried by applications to discover to discover the locations of available replicas of a particular replica location in each datacenter.

3.2. Proposed Dynamic Data Replication Placement Algorithm System Model

Under this section we describe how the proposed optimized dynamic data replication to determine efficient placement algorithm works.

3.2.1. Proposed Solution Process

The proposed solution is prepared up of a Swarm Intelligence Based hybridized of Particle swarm and Enhanced Artificial Bee Colony Optimization algorithms. In designing dynamic data replication placement over cloud environment, this approach ensures high data availability, cost reduction, access time, waiting time, and balanced load.

The operation of the system model step by step

i. User send a request, which is handled by scheduling broker

ii. Scheduling broker send task to the replica broker.

iii. Access history is retrieved from DCs replica manager.

iv. For the highest popular data replica new replica is created and stored on replica catalog

v. Based on the information about replicas in replica catalog table the HEABC algorithm start its operation.

vi. The destination data center information retrieved from replica manager, then HEABC algorithm optimize the replication cost.

vii. Output, data replica is move to the corresponding main/ordinary data center nodes.

viii. Replica block store to an efficient data center node.

ix. Then the next popular data replica processed.

x. Once the replica placement is completed, the result of the replication is combined with the data center broker that's visible to replica manager.

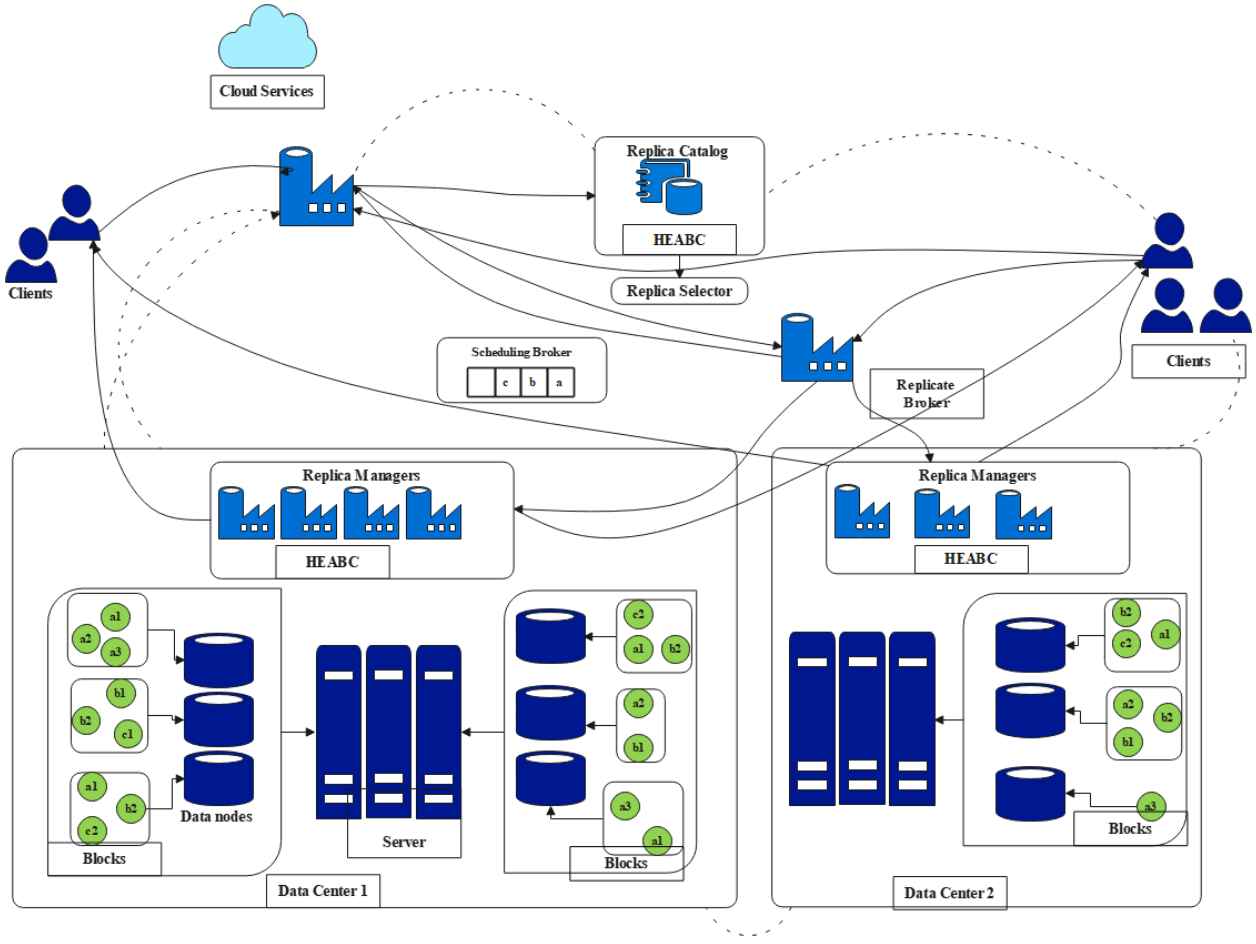


Figure 3.3 Proposed Architecture

3.3. The proposed Optimized Dynamic Data Replication Placement Algorithm

3.3.1. Proposed Architecture for Replica Selection and Placement Optimization

This section describes a proposed structural model for sharing data and arranging replication between nodes in the cloud. The proposed model framework describes data replication access and its placement among nodes in the cloud. Thus, to acquire the best access to selected nodes with the lowest cost and the shortest route between DCs. We used the heterogeneous system to optimize the data replication placement by using the statistical distribution among nodes in different ways. Each DC consists of different stages, such that the VM differs from DCs, and so on. At the same time, it differs in terms of RAM, CPU, PE, etc. DCs vary in cost and number of replicas available. Alternatively, you can access the data replicas and perform the desired optimized placement on the least-distance and least-cost path. All DCs are connected hierarchically and at the same time circularly at every DC level. Therefore, it is important to access replicas and place DCs in appropriate locations by following the guide bees, considering the most cost-effective route and low cost through the proposed system. Users are in the outward layer of the system, and they send tasks to data replication to access the best placement according to a shortest time, shortest route and lower cost via DCs in the Cloud.

Each datacenter can be offered in DC form $= \{dc_{1,}, dc_2, \ldots\ dc_n\}$, where n is a set of various DCs in the Cloud. It differs in the number of DCs, given super, main and ordinary datacenters. The system is heterogeneous by its nature. Hosts can be represented as $PM = \{pm_{1,}, pm_2, \ldots\ pm_x\}$, where x is a various pm set existing in DCs. A virtual machine can be presented as $VM = \{vm_{1,}, vm_2, \ldots\ vm_s\}$, where s is a VM set that is placed in PM. There are two aspects of VM work a time-shared and a space shared operation in the Cloud. Data file replication can be presented in $F = \{f_{1,}, f_2, \ldots\ f_y\}$ form, where y is a set of various data replicas that can be placed in DCs. The main storage unit is a block and can be presented as follows $B = \{b_{1,}, b_2, \ldots\ b_y\}$, where y represents a variable data replication stored set in DCs. General placed data file replication at a super datacenter is randomly distributed with other datacenters. Various values of probability $= pro(ba_j)$ can be saved in every DC by a replica catalog with every replica position in various DCs.

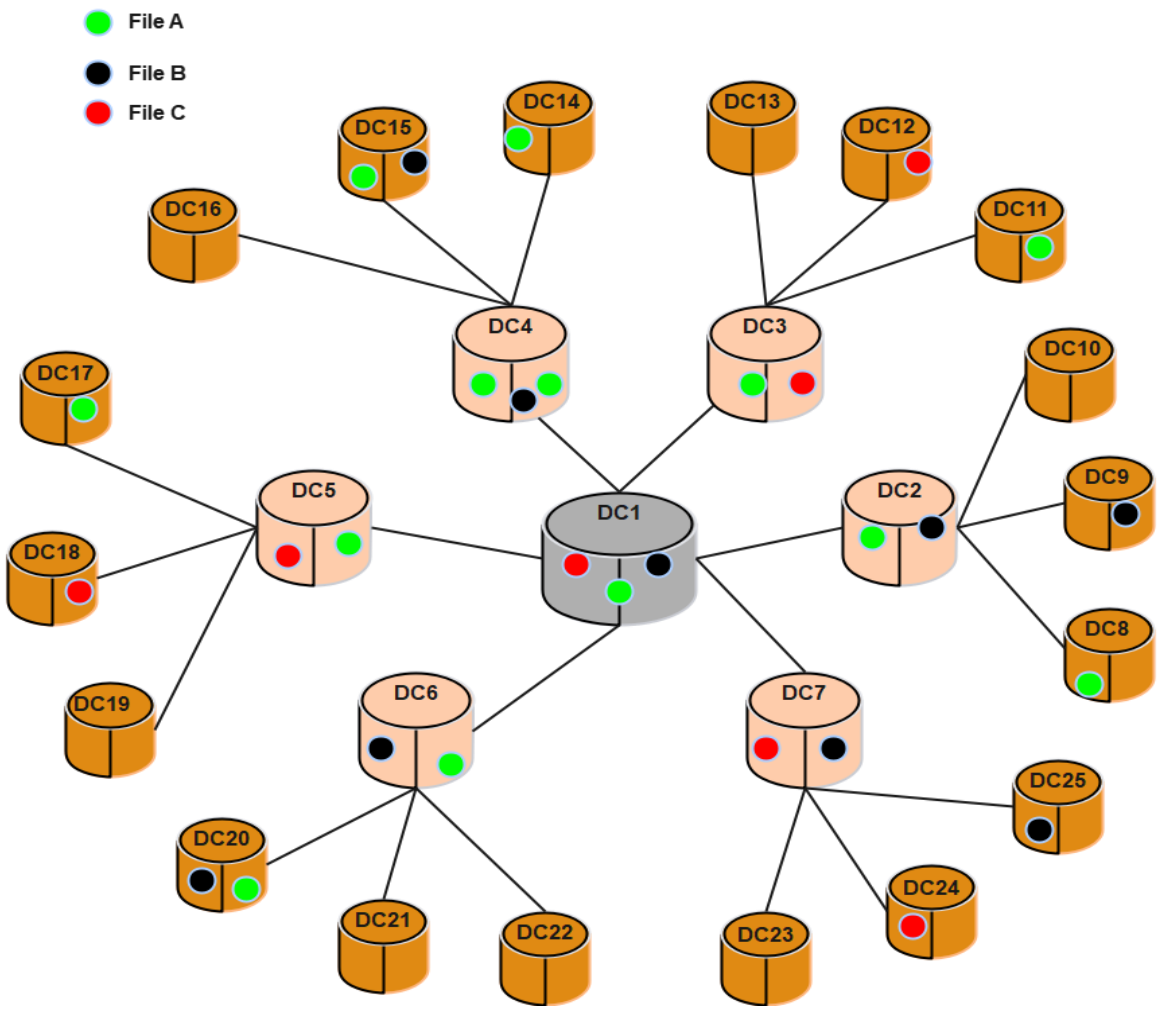


Figure 3.4 A scenario illustrating the replication of files at different data centers

Data File Availability - Availability can be defined as that achieved simply without any failure in nodes or data file replication. Users can access their files whenever they need them. Heterogeneous systems have different availability and unavailability of data files. Data file availability is accessible for each user, and the failure ratio is reduced, whereas the system is more available and reliable. Super data centers generally have highest cost and most reliable hardware; hence they provide highest block available probability. Ordinary data centers, on the other hand, have comparatively less cost and less reliable hardware; hence they provide less block available probability.

$$pro(ba_j)_{super_{DC}} > pro(ba_j)_{main_{DC}} > pro(ba_j)_{ordinary_{DC}} \qquad \textit{Equation 3.1}$$

$$pro(fa_k) = \begin{cases} (1 - \prod_{i=1}^{br_k}(1 - pro(ba_j)_i))^{n_k} & Case1 \\ \prod_{i=1}^{n_k}(1 - \prod_{i=1}^{br_k}(1 - pro(ba_j)_i)) & Case2 \end{cases} \qquad \textit{Equation 3.2}$$

$$\overline{pro(fa_k)} = \begin{cases} 1 - (1 - \prod_{i=1}^{br_k}(1 - pro(ba_j)_i))^{n_k} \\ 1 - \prod_{i=1}^{n_k}(1 - \prod_{i=1}^{b_k}(1 - pro(ba_j)_i)) \end{cases}$$ *Equation 3.3*

Whereas notations in **Eq. 3. 1 - 3. 3** are described,

$pro(ba_j)$	Probability of block availability
b	Blocks
$pro(fa_k)$	Probability of file availability
n_k	Number of blocks
br_k	Number of replicas of a data file
$\overline{pro(fa_j)}$	Probability of block unavailability
na_k	Number of access task have request
$\overline{pro(fa_k)}$	Probability of file unavailability
$Super_{DC}$	Super Data-Centers
$Main_{DC}$	Main Data-Centers
$Ordinary_{DC}$	Ordinary Data-Centers

To analyze the type of fragments and the access time to replication, the Improve Time-Based Decaying Function (ITBDF) is used to determine access data file replication, distinguishing high popularity over other files. Time Based Decaying Function (TBDF) is used to assign different weights or importance to the accesses of a data file at different intervals of time. Concept of temporal locality is used which states that the recently accessed data file has more probability of being accessed again in the future. Based on this concept, TBDF assigns higher weight to recent accesses to the data file compared to previous accesses to the data file. TBDF is able to decay the importance/weight of data file accesses as the time passes. For this, TBDF uses the concept of Exponential Decay Function.

TBDF is defined as,

$$TBDF\ (t_c,\ t_s) = e^{-(tc-ts)\lambda}$$ *Equation 3.4*

Where $\lambda \in \{1, 2, 3, \ldots\}$ and t_s is the start time and t_c is the current time.

As the value of λ increases, the decay rate also increases. **Eq. 3.5** can be written as

$$TBDF\ (t_c,\ t_s) = e^{-(\Delta t)\lambda}$$ *Equation 3.5*

Where $\Delta t = (t_c - t_s)$

whereas other notations are described as: t_c is the current time, t_s is the start time, λ is increasing value, e is the exponential function decay.

After analyzing the access details, the Replica Manager next determines the replica factor of each data file. The replica factor of a data file helps in deciding whether the data file should be replicated or not.

Replica Factor (RF_k) of a data file f_k is the ratio of the total number of weighted accesses to a data file within a time period, say (t_c, t_s), to the total number of requested data bytes by all the users.

System Replica Factor (RF_{sys}) is used to define a dynamic threshold value. It is defined as the ratio of the total weighted accesses and the total number of requested bytes of all the s different data files.

Replica Factor (RF_k) of a data file (f_k) and System Replica Factor (RF_{sys}) is given by

$$\boldsymbol{RF_k} = \frac{\sum_{t_i=t_s}^{t_c}\left(\boldsymbol{an_k}(t_i,\ t_{i+1}) * \boldsymbol{TBDF}(t_i, t_c)\right)}{\boldsymbol{br_k} * \sum_{i-1}^{nb_k} \boldsymbol{b_s}} \qquad \textit{Equation 3.6}$$

$$\boldsymbol{RF_{sys}} = \frac{\sum_{k=1}^{s}\left(\sum_{t_i=t_s}^{t_c}\left(\boldsymbol{an_k}(t_i,\ t_{i+1}) * \boldsymbol{TBDF}(t_i, t_c)\right)\right)}{\sum_{k=1}^{s}\left(\boldsymbol{br_k} * \sum_{i-1}^{n_k} \boldsymbol{bs_i}\right)} \qquad \textit{Equation 3.7}$$

Whereas,

$\boldsymbol{an_k}$ is the number of accesses, $\boldsymbol{br_k}$ is the number of replicas, $\boldsymbol{nb_k}$ is the number of blocks, $\boldsymbol{bs_i}$ is the size of a block.

Cost of Replication - The cost between different DCs is computed according to the data replica number within every DC. For access to data replica, having low cost and placement to the nearest users is important. Also, maintaining the total cost of the system within every DC through different routes via the Cloud is crucial, whether a budget is provided or not. It can be calculated using,

$$\boldsymbol{cost_k}(\boldsymbol{dc_s}) = \sum_{x=1}^{y}\left(\boldsymbol{cost}(\boldsymbol{dc_y}) * \boldsymbol{br_k}(\boldsymbol{dc_y})\right) \qquad \textit{Equation 3.8}$$

Minimum Distance of Replication -The shortest distance and the shortest route between DCs has been calculated. Thus, it is important to access the least cost path via DCs in the Cloud. Moreover, it is crucial to ensure access to the optimal data replica placement via DCs in the Cloud. It is calculated using **Eq. 3. 9** and **Eq. 3. 10** as follows:

$$\boldsymbol{Min} \sum_{i=1}^{n} \sum_{j=1}^{n} \boldsymbol{d_{ij} x_{ij}} \qquad \textit{Equation 3.9}$$

$$\sum_{i=1}^{n} \boldsymbol{n_i x_i} \geq \boldsymbol{k}, \quad \boldsymbol{x_i} \in \{\boldsymbol{0}, \boldsymbol{1}\}, (\boldsymbol{1} \leq \boldsymbol{i} \leq \boldsymbol{n}) \qquad \textit{Equation 3.10}$$

Knapsack Problem -The knapsack problem is NP-hard. Every item has a particular value and weight. The target is to maximize the resources in the bag with the constraint such that the carrying value of the bag shall not exceed the following **Eq. 3.11** and **Eq. 3. 12**.

$$\boldsymbol{Maximize} \quad \boldsymbol{px} = \sum_{j=1}^{n} \boldsymbol{p_j} \qquad \textit{Equation 3.11}$$

$$\boldsymbol{wx} = \sum_{j=1}^{n} \boldsymbol{w_i x_j} \leq \boldsymbol{v_i} \qquad \textit{Equation 3.12}$$

$x_j \in \{0,1\}, \mathrm{j} = 1,2,\ldots,\mathrm{n}$
$p = (p_1, p_2, \ldots, p_n)$
$w = (w_1, w_2, \ldots, w_n)$
$i = 1,2,\ldots,m$

Each object $j \in$ J has profit p_j and weight w_j in dimension i, where $1 < i < m$. Binary variable x_j indicates whether object j is included in the knapsack ($x_j = 1$) or not ($x_j = 0$). Notations of Knapsack problem are described as follows:

$\boldsymbol{w_j}$ weight of object j, $\boldsymbol{p_j}$ value of object j_s, $\boldsymbol{p_x}$ total value, $\boldsymbol{w_x}$ total weight, $\boldsymbol{j}$ item j, $\boldsymbol{p}$ value vector of all item j, $\boldsymbol{w}$ weight vector of all item j, **Set $\boldsymbol{j}$ of $\boldsymbol{n}$** object and knapsack with m dimension, $\boldsymbol{v_i}$ each dimension of the knapsack has a capacity v_i.

3.3.2. Artificial Bee Colony Optimization Algorithm

The ABC algorithm was developed by examining the behavior of real bees in finding a food source called nectar and passing information about that food source to bees in the hive. In ABC, artificial agents are defined and divided into three types: employed bees, onlooker bees, and scout bees. Each plays a different role.

Employed bees stay at the food source and remember the environment of the source. Onlookers receive information about food sources from bees busy in the hive and select one of the food sources to collect nectar. Scouts are responsible for finding new food, new nectar, and sources. The EABC algorithm process is expressed as:

Step 1. Initialization: Spray n_e percentage of the populations into the solution space randomly, and then calculate their fitness values, which are called the nectar amounts, where

n_e represents the ratio of employed bees to the total population. When these populations are placed in the solution space, they are called employed bees.

Step 2. Move the Onlookers: Calculate the probability of selecting a food source by the equation **Eq. 3.13**, select a food source to move to by roulette wheel selection for every onlooker bee and then determine the nectar amounts of them. The movement of the onlookers follows the **Eq. 3.14**.

Step 3. Move the Scouts: If the fitness value of the employed bees does not increase after a predetermined number of iterations called "$Limit$", their food source is abandoned and these deployed bees become scouts. The scouts are moved by the equation **Eq.3.15.**

Step 4. Update the Best Food Source Found So Far: Updates the best food sources found so far.

Memorize the best fitness values and locations found by bees.

Step 5. Termination Checking: Check if the number of iterations satisfies the termination criteria. If the termination condition is satisfied, terminate the program and output the results; otherwise go back to the Step 2.

$$\boldsymbol{p_i} = \frac{\boldsymbol{F(\theta_i)}}{\sum_{\boldsymbol{k=1}}^{\boldsymbol{S}} \boldsymbol{F(\theta_k)}} \qquad \textit{Equation 3.13}$$

where θ_i denotes the position of the i^{th} employed bee, S represents the number of employed bees, and p_i is the probability of selecting the i^{th} employed bee.

$$\boldsymbol{x_{ij}(t+1) = \theta_{ij} + \phi\left(\theta_{ij}(t) - \theta_{kj}(t)\right)} \qquad \textit{Equation 3.14}$$

where x_i denotes the position of the i^{th} onlooker bee, t denotes the iteration number, θ_k is the randomly chosen employed bee, j represents the dimension of the solution and $\phi(\bullet)$ produces a series of random variable in the range $[-1, 1]$.

$$\boldsymbol{\theta_{ij} = \theta_{ij_{min}} + r(\theta_{ij_{max}} - \theta_{ij_{min}})} \qquad \textit{Equation 3.15}$$

where r is a random number and $r \in [-1, 1]$.

We have selected ABC algorithm because it is relatively simple to use with three control parameters. These parameters are the food sources (S), iterations numbers, and threshold for converting the employed/onlooker bees into a scout (limit). In addition, ABC is more adaptable

to new problems than other swarm algorithms and shows better performance results for different types of problems (Akay & Karaboga, 2012b). The ABC algorithm has two main mechanisms for searching called exploration and exploitation (Rashedi et al., 2009). The purpose of the exploration mechanism is to guide the bees to reach the best local solutions. On the other hand, the exploitation focuses on guiding the algorithm to reach global optima (Civicioglu & Besdok, 2013). In this study, we used the enhanced ABC algorithm that balanced these two mechanisms.

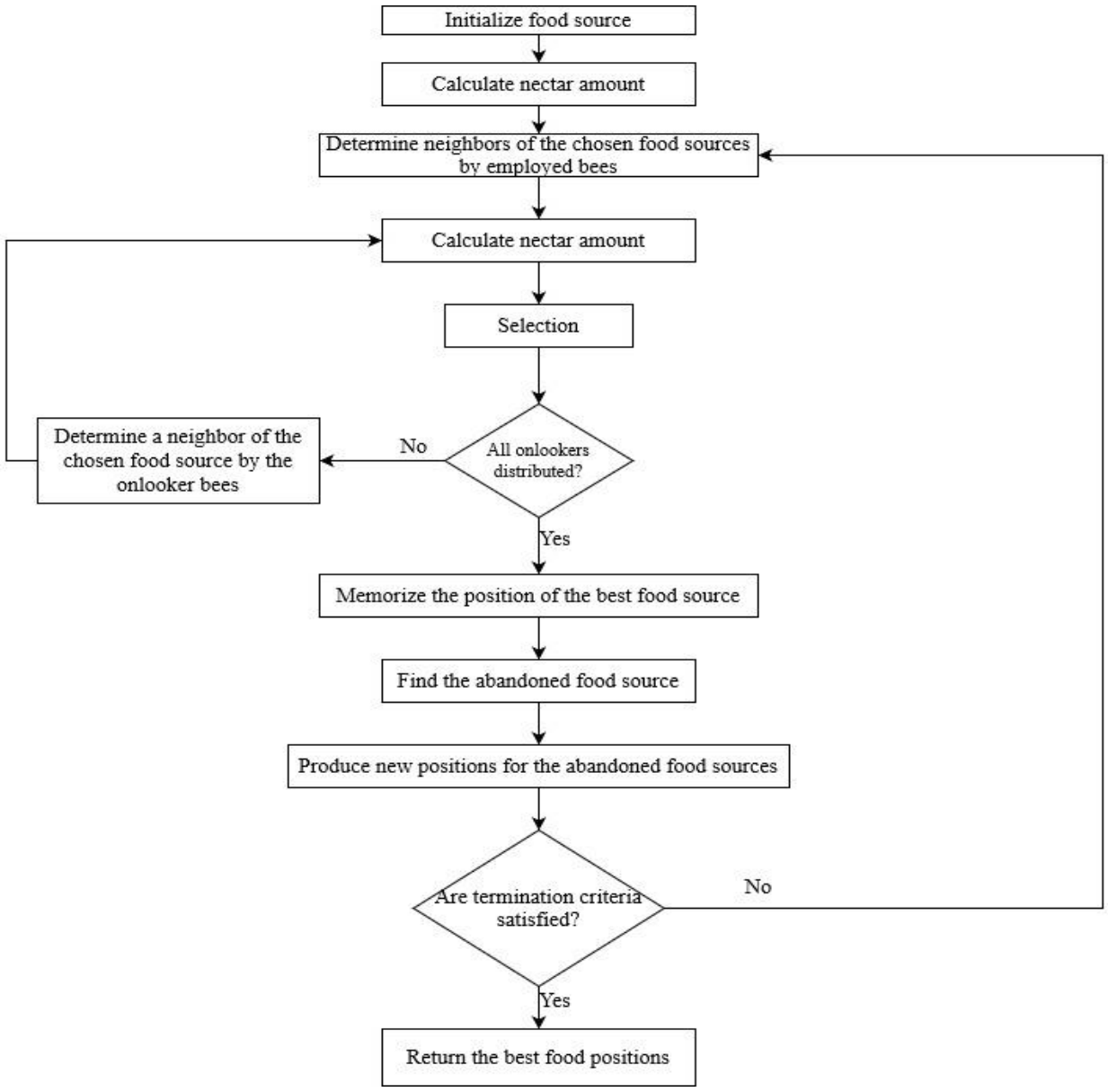


Figure 3.5 Artificial Bee Colony Algorithm flowchart (Akay & Karaboga, 2012b)

3.3.3. The Proposed Strategy using Particle Swarm Optimization (PSO)

PSO algorithm is one of the first algorithms used for solving continuous problems and is related to evolutionary strategy that is inspired by group behaviors of birds or fish for finding food. So, a group of birds are looking for food in random space. There is food and no bird knows where it is, only know the distance to it. One of the best strategies is to chase birds close to food. Observing the movements of the birds shows that all birds follow the leader bird with varying deviations. Changes in particle positions in the search space are influenced by the experience and knowledge of birds and their neighbors. The result of that modeling is the search for particles that tend to successful regions.

In this subsection we define PSO algorithm for data replication. For each particle with the best data replica, a fitness function tests the optimally chosen data replication. The velocity, position and inertia weight updates are shown in **Eq. 3.16 - 3.18** as follows:

$$\boldsymbol{V_{ij}^{k+1} = W.V_{ij}^{k} + C_1R_1\left(pbest_{ij}^{k} - X_{ij}^{k}\right) + C_2R_2\left(gbest_{ij}^{k} - X_{ij}^{k}\right)} \qquad \textit{Equation 3.16}$$

Where $\boldsymbol{V_{ij}^{k+1}}$ particle's new velocity, $\boldsymbol{V_{ij}^{k}}$ particles current velocity, $\boldsymbol{C_1, C_2}$ positive constants acceleration parameters, $\boldsymbol{pbest_{ij}^{k}}$ personal best position particle, $\boldsymbol{X_{ij}^{k}}$ position $\boldsymbol{i^{th}}$ in $\boldsymbol{j^{th}}$ swarm, $\boldsymbol{R_1, R_2}$ random numbers in the range **[0,1],** $\boldsymbol{gbest_{ij}^{k}}$ global best position particle.

$$\boldsymbol{X_{ij}^{k+1} = X_{ij}^{k} + V_{ij}^{k+1}} \qquad \textit{Equation 3.17}$$

Where $\boldsymbol{X_{ij}^{k+1}}$ new position of particle, **K** iteration population, $\boldsymbol{i \in 1, 2, 3, \dots, m}$ m is members number in an iteration, $\boldsymbol{j \in 1, 2, 3, \dots, d}$ d is the size of the swarm.

$$\boldsymbol{w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} * iter} \qquad \textit{Equation 3.18}$$

Where, $\boldsymbol{w}$ inertia weight**,** $\boldsymbol{w_{max}}$ max value of inertia weight**,** $\boldsymbol{w_{min}}$ min value of inertia weight, $\boldsymbol{iter_{max}}$ maximum number of iterations**,** $\boldsymbol{iter}$ current iteration number.

The proposed PSO algorithm is introduced in the following Algorithm. The fitness function is shown in **Eq. 3.16 – 3.18**.

The Proposed PSO for Selecting Data Replicas

1. **Input:** Size α of population, Number of iterations, Datacenters, Data availability, Improved Time-Based Decay Function
2. **Output:** Selected $P_{position} \leftarrow (Optimal_{BestReplica}, Optimal_{TotalExecutionTime}, Optimal_{Cost})$
3. **begin**
4. Define Values of parameters, Size of Pop, Num of Iterations and Num of Particles
5. Initialize set values of particle swarm (Num of Particles and Num of Iterations);
6. Initialize un/availability probabilities;
7. Initialize the replica by cost and time;
8. **Repeat**
9. for $j = 1$ to α **do**
10. **foreach** data replication in DC
11. **do**
12. Calculate fitness function
13. Update velocity
14. Update position
15. $P_{velocity} \leftarrow$ Random velocity ()
16. $P_{position} \leftarrow$ Random position ()
17. $P_{best} \leftarrow P_{position}$
18. **if** $\alpha \leq 0$ **then**
19. Exploitation
20. **Else**
21. Exploration
22. Select best data replication
23. **end if**
24. Calculate the ITBDF of data replication;
25. Calculate the replica factor the data replication;
26. Calculate the cost of data replication;
27. **end for**
28. **end**
29. **until** reaching an iteration of the maximum number
30. **Return** the optimal best replica solution

3.3.4. The Proposed Strategy using Enhanced ABC Algorithm

Apart from the common control parameters of population-based algorithms such as population size or colony size (SN) and maximum generation number or maximum cycle number, the basic version of the Artificial Bee Colony algorithm has only one control parameter "limit" (MCN). The ABC algorithm's basic version is very efficient for multimodal and

multidimensional basic functions (Akay & Karaboga, 2012b). However, the algorithm's convergence rate is lower when working with constrained problems, composite functions, and some non-separable functions.

Frequency of the perturbation - One of the modifications in the ABC algorithm is controlling the frequency of perturbation. In the basic version of ABC, this frequency is fixed. In basic ABC, while producing a new solution, $\boldsymbol{v_i}$ changing only one parameter of the parent solution $\boldsymbol{x_i}$ results in a slow convergence rate. In order to overcome this issue, the ABC algorithm is modified by introducing a control parameter, modification rate (MR). By means of this modification, for each parameter $\boldsymbol{x_{ij}}$, a uniformly distributed random number, $(0 \leq R_{ij} \leq 1)$, is produced and if the random number is less than MR, then the parameter $\boldsymbol{x_{ij}}$ is modified as in the **Eq. 19**.

$$\boldsymbol{v_{ij}} = \begin{cases} \boldsymbol{x_{ij} + \phi_{ij}(x_{ij} - x_{kj}),} & \boldsymbol{if\ R_{ij} < MR,} \\ \boldsymbol{x_{ij},} & \boldsymbol{otherwise,} \end{cases} \qquad \textit{Equation 3.19}$$

where $k \in \{1,2,3,\ldots SN\}$ is randomly chosen index that has to be different from $\boldsymbol{i}$ and $\boldsymbol{MR}$ is the modification rate which takes value between **0** and **1.** A lower value of MR may cause solutions to improve slowly while a higher one may cause too much diversity in a solution and hence in the population.

A heterogeneous system and SI (swarm intelligence) in the form of EABC are used to find the shortest ways and reduced costs to access and place replicas across nodes. We execute the proposed system through the EABC algorithm to determine the best and shortest route to access data placed to users. It consists of three main features which are the cost, the distance and the knapsack of the process for the selection and placement of the replica at the best place via nodes. The EABC algorithm consists of three bee types: scout bees, onlooker bees and employed bees. Half are employed bees, and the other half are onlooker bees. Employed bees are responsible for using nectar resources and sending information to onlooker bees that wait in the cell. They are responsible for the quality of the appropriate site of two of the resources being used. Onlooker bees decide whether the scout bees at the new site obtain new food, depending on external and internal motives. The nectar quantity from food resources or the site corresponds with the nectar of bees. The principal steps of the algorithm can be summarized as follows:

I. Initialization of Food Source Sites

A search space is supposed in the environment where the cell contains sites of food resources. The algorithm randomly generates new locations with food sources that match the search space solution. The primary food sources' limits are produced within parameters according to **Eq. 20**:

$$\boldsymbol{x_{ij} = x_j^{min} + rand[0,1)(x_j^{max} - x_j^{min})} \qquad \textit{Equation 3.20}$$

$i = 1, 2, \ldots\ s_n,\ j = 1,2, \ldots d$, $\boldsymbol{s_n}$ number of food sources, $\boldsymbol{d}$ number of optimization parameters, $\boldsymbol{rand}$ random number in range [0, 1), $\boldsymbol{x_j^{min}}$ lower border in the j^{th} diminution of the problem space, $\boldsymbol{x_j^{max}}$ upper border in the j^{th} diminution of the problem space.

II. Sending Employed Bees to the Food Source Sites

Here every employed bee is connected with one site of only food sources so the number of sites of food sources in the solution area is equivalent to the number of employed bees. At the same time, it changes the locations of different food sources (solutions) in memory and finds food sources according to new visual information from local information. Its quality is estimated using **Eq. 21 & Eq. 24**:

$$\boldsymbol{v_{ij} = x_{ij} + \phi_{ij}(x_{ij} - x_{kj})} \qquad \textit{Equation 3.21}$$

whereas employed bee notations are described as follows

$\boldsymbol{x_{ij} - x_{kj}}$ Difference between parameter decrease, the perturbation on the position $\boldsymbol{x_{ij}}$ decrease, $\boldsymbol{x_i}$ Neighborhood of every food source site, $\boldsymbol{v_i}$ Food source is determined by changing one parameter of $\boldsymbol{v_i}$, j Random integer in the range , $[1, d]$, $\boldsymbol{k}\ k \in \{1, 2, \ldots, s_n\}$ is a random chosen index that has to be different from i, $\boldsymbol{\phi_{ij}}$ A uniformly distributed real random number in the range $[-1, 1]$, $\boldsymbol{x_i}$ & $\boldsymbol{v_i}$ Greedy selection is applied between x_i & v_i.

The difference between the values of $\boldsymbol{x_{kj}}$ and $\boldsymbol{x_{ij}}$ decreases. In addition, the $\boldsymbol{x_{ij}}$ site decreases, so the step length decreases adaptively. Searching for the best solution within the matter search space is an important case. If the value produced by this process exceeds the definite limit, the value shall be reset to acceptable values in the case of exceeding its limit to the definite limit; these values are: Minimize problem fitness value:

$$If\ x_i > x_i^{max} \quad then \quad x_i = x_i^{max}, \qquad \textit{Equation 3.22}$$

$$If\ x_i < x_i^{min} \quad then \quad x_i = x_i^{min}, \qquad \textit{Equation 3.23}$$

After producing x_{ij} within the specified limits and standards in the matter, the fitness for the solution is specified by using the following formula:

$$fitness_i = \begin{cases} \frac{1}{1+f_i} & if \quad f_i \geq 0 \\ 1 + abs(f_i) & if \quad f_i \leq 0 \end{cases} \qquad \textit{Equation 3.24}$$

Then, the best solution is selected depending on the fitness value that represents the nectar amount of the food sources at x_i and v_i. If the v_i source is higher than x_i regarding profit, employed bees store the new site in their memory and forget the old one. If there is no improvement in the new process, it saves the previous site in its memory, increases the counter by one in tests and offers zero otherwise.

III. Calculating Probability Values involved in Probabilistic Selection

After completing the search, an employee bee shares information, nectar quantity, sites and food sources with any onlooker bee in the area via a dance. There are two types of dances (dance round and waggle tail dance). The onlooker bee estimates information and nectar taken from the employed bee and selects sites and food sources that may connect with the nectar quantity. This step is an advantage in the ABC algorithm. This selection is estimated according to the fitness values in society. Selection may be made by a roulette system or other systems. The roulette system is shown in **Eq. 25**:

$$p_i - \frac{fitness_i}{\sum_{i=1}^{sn} fitness_i} \qquad \textit{Equation 3.25}$$

Where, p_i Probability value.

IV. Food Source Site Selection by Onlooker Bees based on the information Provided by Employed Bees

A real number is created in the range of 0 to 1 for each site. If the probable value p_i is in **Eq. 25**, the site is larger than the random number. Onlooker bees modify this site by using **Eq. 25** as employed bees. A greedy algorithm estimates the site because the onlooker bee forgets the old site and saves the new one in the improvement state, or the old site stays as it is in its memory if the solution is not improved. The EABC algorithm makes a test counter and

increases it by 1 each time, unless it is returned to 0. This process is repeated until the onlooker bee is distributed to all food sources.

V. Abandonment Criteria: Limit and Scout Production

After employed bees and onlooker bees complete and finish the search processes in their cycle, the EABC algorithm discovers a depleted source that can be abandoned and decides to dispense with them by updating the counter during a search test counter. If the counter is larger than the control standard, then the EABC algorithm can be known by the limit; hence, this source or site is depleted and must be abandoned. The EABC algorithm that is called the limit shall be employed. Its formula is shown in (26):

$$\boldsymbol{LimitValue} = (\boldsymbol{s_n} * \boldsymbol{mcn}) \qquad \textit{Equation 3.26}$$

Where $\boldsymbol{mcn}$ iteration number

Food sites abandoned by the bee are to be exchanged for another food source explored by scout bees during changes in the ABC algorithm and by producing a new site at random instead of the abandoned one. Employed bees are likely to be scouts. If it is supposed that there are more than the counters exceeding the limit value, in this case the highest one is selected programmatically.

IV. Zipf And Geometric Distributions

Zipf and Geometric Distributions can be compared with the data replication process distributions among datacenters. Data file replication is placed in DCs according to the users' tasks.

First: Zipf is used in a random distribution process of data file replica placements among DCs closer to users. It can be calculated by **Eq. 27** & **Eq. 28**.

$$\boldsymbol{p(f_i)} = \frac{\boldsymbol{1}}{\boldsymbol{i^{\alpha}}} \qquad \textit{Equation 3.27}$$

where $\boldsymbol{i} = 1, 2, \ldots, n$ and $\boldsymbol{\alpha}$ is a is a factor data replication distribution, $0 \leq \boldsymbol{\alpha} > \boldsymbol{1}$.

Zipf's Law is a widely observed empirical law, originally derived from linguistics, that states that the frequency of any word in natural language is inversely proportional to its rank in the frequency table. This means that the most frequent word will occur twice as often as the second

most frequent word, three times as often as the third most frequent word, and so on. In a cloud computing context, Zipf's Law can help explain how certain applications or services are used more frequently than others. By analyzing usage patterns of an application or cloud service and other metrics related to it such as latency, response time, etc., cloud providers can use Zipf's Law to better understand usage trends and optimize resources accordingly.

Second: The Geometric distribution represents a random distribution to solve optimal data file replica placements with various parameters. It can be calculated by the following equation:

$$\boldsymbol{p(i) = (1-p)^{i-1}.p} \qquad \textit{Equation 3.28}$$

Where $\boldsymbol{i} = 1, 2, \ldots, n$ and $0 < p < 1$. A wide p represents the file replication access.

The specific implementation steps for the Enhanced Artificial Bee Colony optimization algorithm:

1. Initialize the population of solutions $x_{ij}, i = 1 \ldots SN, j = 1 \ldots d, trial_i = 0$ $trial_i$ is the non-improvement number of the solution x_i used for abondend
2. Evaluate the population
3. Cycle =1
4. **Repeat**

   Produce a new food source, population for Employed bee
5. **for** $i = 1$ to SN **do**
6. Produce a new food source $\boldsymbol{v_i}$for the employed bee of the food source $\boldsymbol{x_i}$ by using **(Eq. 19)** and evaluate its quality
7. Apply a greedy selection process between v_i and x_i and select the better one
8. If solution x_i does not improve $trial_i = trial_i + 1$, otherwise $trial_i = 0$
9. **end for**
10. calculate the probability values $\boldsymbol{p_i}$ by **Eq. 21 (**in case of modified ABC **(Eq. 25))** for the solutions using fitness value

    Produce a new food source population for onlookers
11. $t = 0, i = 1$
12. **Repeat**
13. **If** $random < p_i$ **then**

14. Produce a new v_{ij} food source by (**Eq. 21**) (in case of modified ABC (**Eq.24**)) for
    onlooker bee
15. Apply a greedy selection process between v_i and x_i and select the better one
16. If solution x_i does not improve $trial_i = trial_i + 1$, otherwise $trial_i = 0$
17. $t = 0, i = 1$
18. **End if**
19. **Until** $t = SN$
    Determine Scout
20. **if** $\max(trial_i) > limit$ **then**
21. Replace xi with a new randomly produced solution by (**Eq. 20**)
22. **end if**
23. Memorize the best solution achieved so far
24. cycle = cycle+1
25. **until** (cycle = Maximum Cycle Number) (**Eq. 26**)

Enhanced Artificial Bee Colony Optimization Algorithm Pseudo Code

The Proposed Enhanced ABC optimization algorithm for the Placement of Data Replicas in a Cloud Environment

1. **Input:** Number of bees**,** Number of Iterations**,** Datacenters, Min Distance between Data centers
2. **Output:** Selected Optimally Best Data Replica Placement, Total Execution Time and Costs
3. **Begin:**
4. Define Values of parameters, Num of Iterations and Number of Bees;
5. Initialize availability and unavailability probabilities;
6. Initialize distance between DCs;
7. Initialize data replication cost and size;
8. Initialize optimal best data replication placement in DC solution;
9. **Repeat**
10. **For** i = 1 to (Num of Employed Bees/Onlooker Bees);
11. Cycle = cycle + 1;
12. Set all bees distribution in DC;
13. **End for**

14. **Repeat until stopping criterion is met**;
15. **For each** DC in current system;
16. Calculate desirability of the movement;
17. Calculate probability of the movement;
18. **If the** $p_i \ of \ v_i > x_i$ if available **then** select them directly or select greedy solutions;
19. **Else**
20. exploration
21. **End if**
22. **End for**
23. Set Onlooker Bee/ Global search/ update;
24. Set determine replica placement in DC;
25. Until all replicas are selected;
26. Until all replicas are placed;
27. **If** the space of storage the DC is insufficient;
28. **Then**
29. Apply the global update rule;
30. **Else if**
31. Delete small replica popularities;
32. **End if**
33. Until maximum number of iterations is reached or access solution is found;
34. Return the optimal data replication placement in the DC;

How to combine these two algorithms

1. Start by analyzing the computational complexity of both algorithms. Look at the time and resources required to execute each algorithm, and assess their current performance based on the results. Identify which variables are impacting their performance the most.

2. We formulate a hybrid model of ABC and PSO algorithms by combining elements from both algorithms that can improve overall performance. This could include adjusting the parameters for each algorithm's function or incorporating additional features that could lead to better results than either algorithm would provide on its own.

3. Test and validate of the hybrid model with experiments that familiarize us with its capabilities while allowing as to analyze its changes in performance compared to the original algorithms.

4. We evaluate how well the HEABC (new hybrid model) is performing against the original algorithms and other competing models we have tested it against.

3.3.5. Proposed Algorithm Flowchart

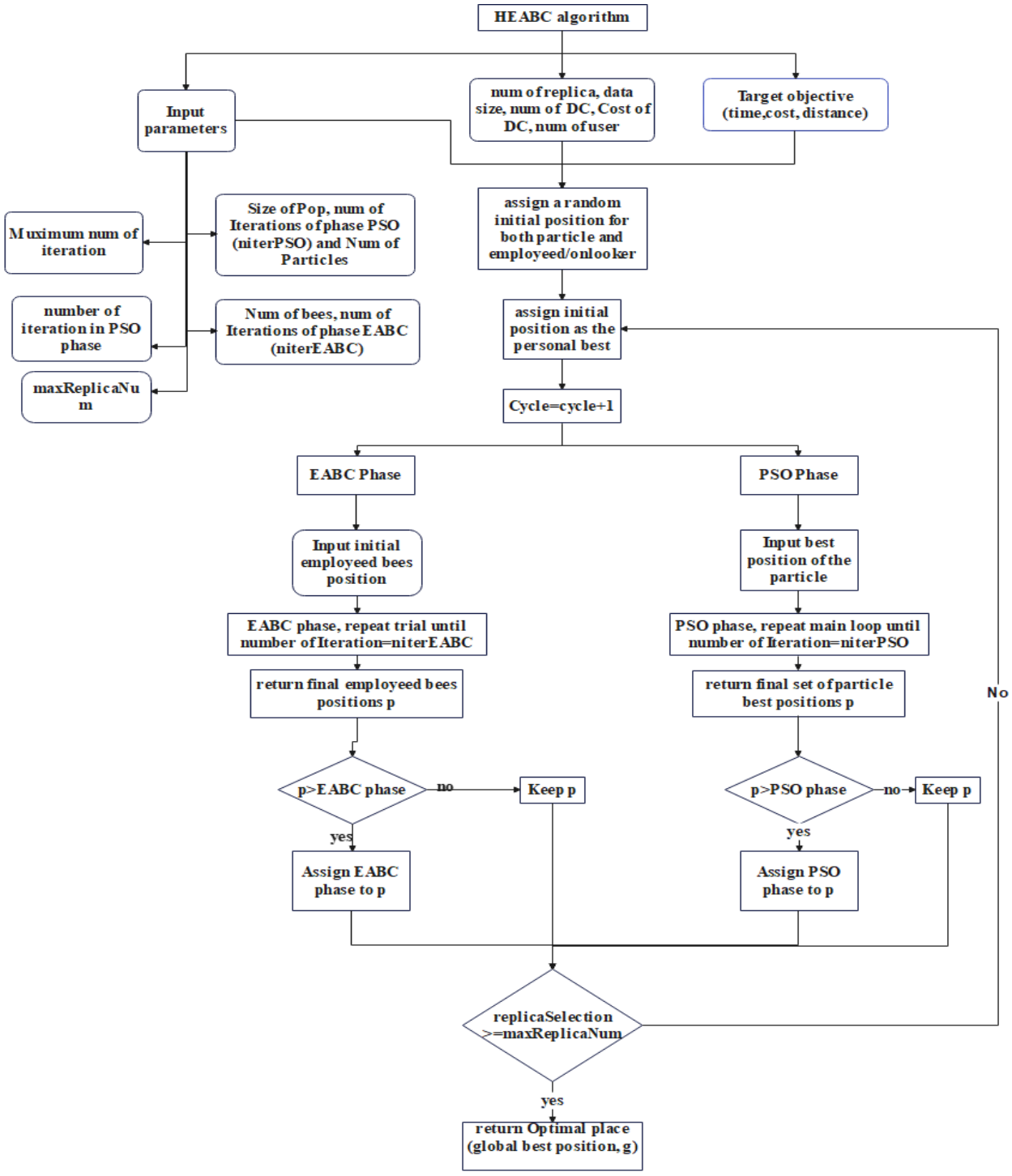


Figure 3.6 Proposed algorithm flowchart

CHAPTER FOUR

4. Experiment Results and Discussions

In this part we discuss the simulation setup and dynamic data replication placement simulation, as well as the data replication parameter, resource parameter, and outcome evaluation metrics including cost, transmission time, response time and number of data transmission. With a Hybrid Enhanced Artificial Bee Colony optimization algorithm, we were compared to other algorithms including Dynamic Cost-Aware Re-Replication and Re-balancing Strategy (DCR2S), Genetic Algorithm, and others. The comparison was conducted using the Cloudsim simulator to verify its effectiveness. The simulation was conducted using Java and the Eclipse development environment.

4.1. Research Simulation Tools

CloudSim enables researchers to focus on specific system design issues rather than worrying about low-level details associated with cloud-based infrastructures and services. It is a java-based toolkit that includes actors or entities that are nothing more than java classes that communicate with each other during the simulation process (Humane & Varshapriya, 2015). Cloudsim have the following common entity or actors which performed on action

- Virtual machine (VM) is logical machine on which application or task will be run.
- Virtual machine Generator (VMG) these generators the VMs on Cloudsim Simulator and submit all VMs to the broker.
- Data center (DC) this provisions the resources, and may have one or more than one hosts.
- Host is a physical machine on which logical machines are placed.
- Cloud Information Services (CIS) an entity which manages all registering of resources.
- Broker is an agent, who will submit the VMs in specific host and then bind cloud-lets, applications, processes on specific VMs and after execution of all the cloud-lets, the free VMs will be automatically destroyed.

- Data center Broker an agent who is responsible for communications between data center and VMs as well as Cloud-lets.
- Cloudlets are the task or application that run on VMs.
- Cloud-lets Generator will generate the cloud-lets and submit Cloud-let or task list to the broker.
- BW Provisioner the provision policy of bandwidth to VMs that are deployed on a Host component.
- Memory Provisioner represent the provision for allocation of memory to VMs.
- VM Allocation Policy This is an abstract class implemented by a Host component that models the policies (which may be space-shared or time-shared) required for allocating processing power to VMs.

These classes all have corresponding methods that improve the simulation process.

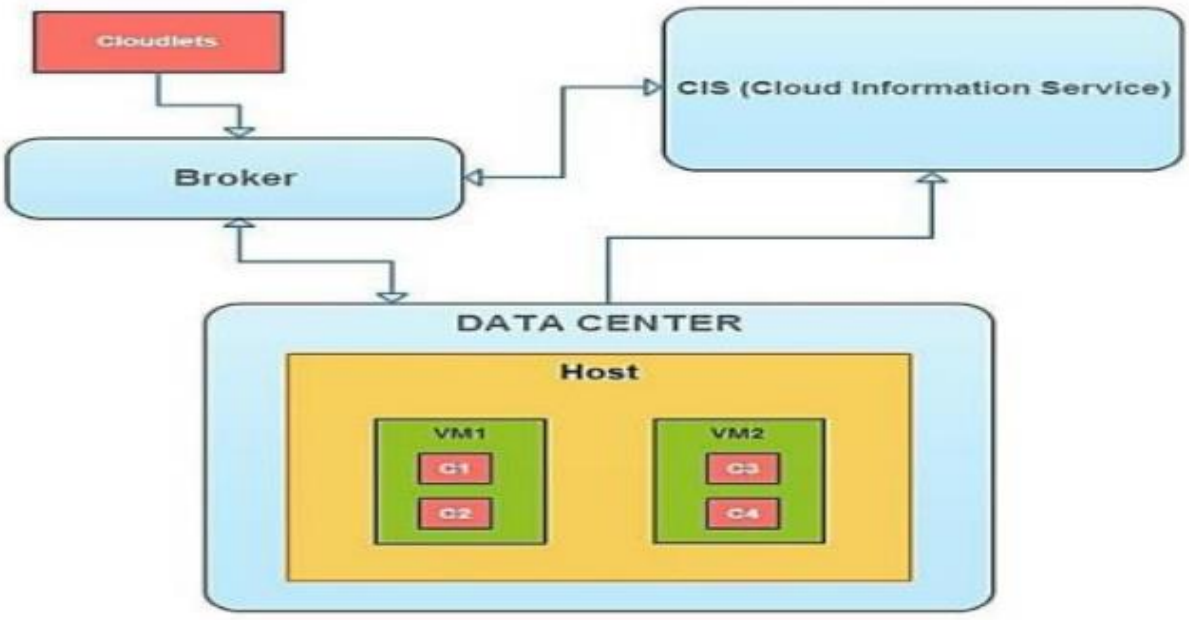


Figure 4.1 CloudSim Model (Humane & Varshapriya, 2015)

4.2. Simulation Model

Cloud Information Services (CIS), Data-center, Broker, Virtual Machine (VM1, VM2), Host,

and Cloud-lets are its entities (C1, C2, C3, C4). The CIS functions as a registry for all cloud resources. Before they can be used, all cloud resources must be registered with the CIS. There are one or more Hosts in the data center. The following configuration parameters are available for the Hosts: -Processing Element (PEs), RAM, Bandwidth (Bw), and so on. Because this is a virtualized environment, the Virtual Machines will always reside within the Host. The Cloudlets are executed on the VMs. A broker is a member of the data center broker class. It is the Broker's responsibility to submit tasks to the Data-center. The broker communicates with the CIS and requests information about registered resources. The data-center characteristics are then returned to the broker by CIS. The broker has Cloudlets (applications) that have yet to be assigned. Once the broker has received all of the information from CIS, it submits the Cloudlet List to the Data-center. CloudSim default allocation policies make the allocation decision (like Cloudlet Scheduler, VmAllocationPolicys VmScheduler etc.). The Hosts (which reside in the data center) schedule the VMs, and the VMs (which reside in the Host) schedule the Cloudlets.

4.3. Proposed Algorithm Evaluation

A parameter, such as a task attributes or virtual machine attribute, must be clearly described and configured in order to compare the proposed algorithm to other algorithms before evaluating the algorithm's performance.

4.3.1. Configuration details

The Cloud is formed to simulate different kinds of DCs with different structures. Moreover, 25 DCs are used. The system configurations are shown in Table 4.2. Each DC consists of a host that contains a set of VMs that provides blocks of available data replications. We created three different data placements for high datacenters. A total of 1,000 cloudlets are randomly checked for data replication jobs. PSO and EABC algorithms parameters are shown in Tables 4.1 and 4.3 respectively.

Table 4.1 PSO parameter

No.	Parameters	Values
1	No. of particles	50
2	C_1	2

3	C_2	2
4	R_1	[0 - 1]
5	R_2	[0 - 1]
6	w_{max}	0.9
7	w_{min}	0.4
8	No. of iteration	1000
9	w	1

Table 4.2 Simulation parameters of the configuration system

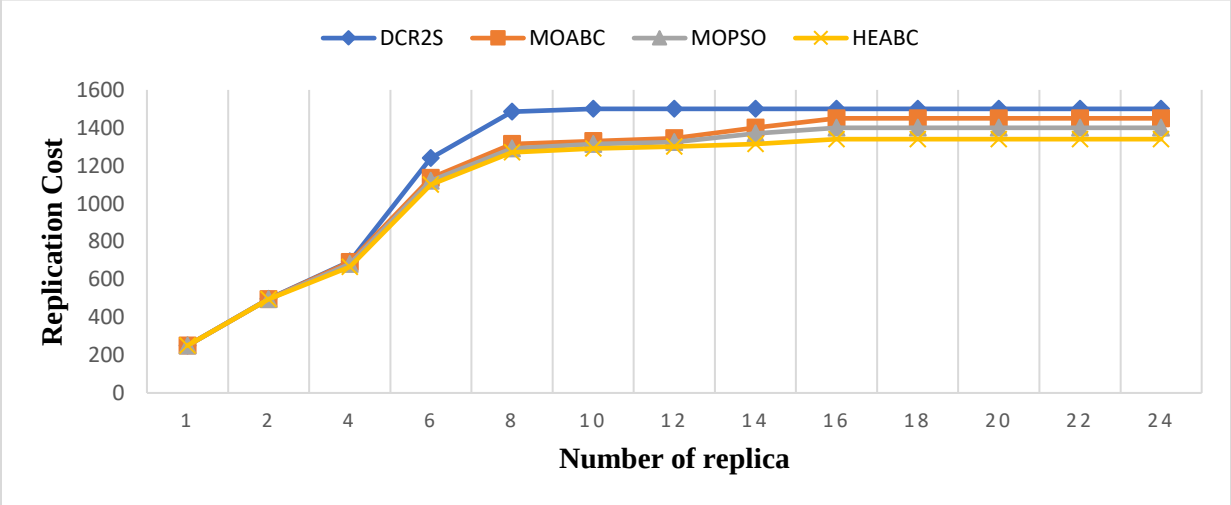
Resources	**Parameters**	**High Data-centers**	**Medium Data-centers**	**Low Data-centers**
Data Centers	No. of DCs (21)	1	6	18
	Cost of DCs	500	300	100
	No. of Hosts per DCs (110)	20	30	60
Hosts	How many Processing Elements per host?	12 – 16	4 – 8	1 – 4
	How many MIPS per Processing Element?	1000 – 2000	500 – 1000	100 – 500
	How much Bandwidth per Processing Element?	5 - 15 GB	2 - 4 GB	1 - 2 GB
Virtual Machine (VMs)	Number of VMs (470)	200	150	120
	MIPS	800	400	200
	Memory RAM	2 GB	1 GB	512 MB
	Bandwidth	10 GB	2GB	1 GB
	No. of Processing Elements	8-16	4-8	1-4
File	Using zipf and geometric distribution			
	A three different data files (3 files) are placed in the Cloud storage environment, with each size in the range of [0.1, 10] GB.			
	No. of files (3)	A	B	C
	Cost of replication	500	300	100
Cloudlet	Cloudlet task	1000 task		
	Length of task	1000 – 20000		
User	No. of users	10 – 50		
	Cloudlet Scheduler	Time & space shared		
	VM scheduler	Time & space shared		

Table 4.3 EABC algorithm parameter

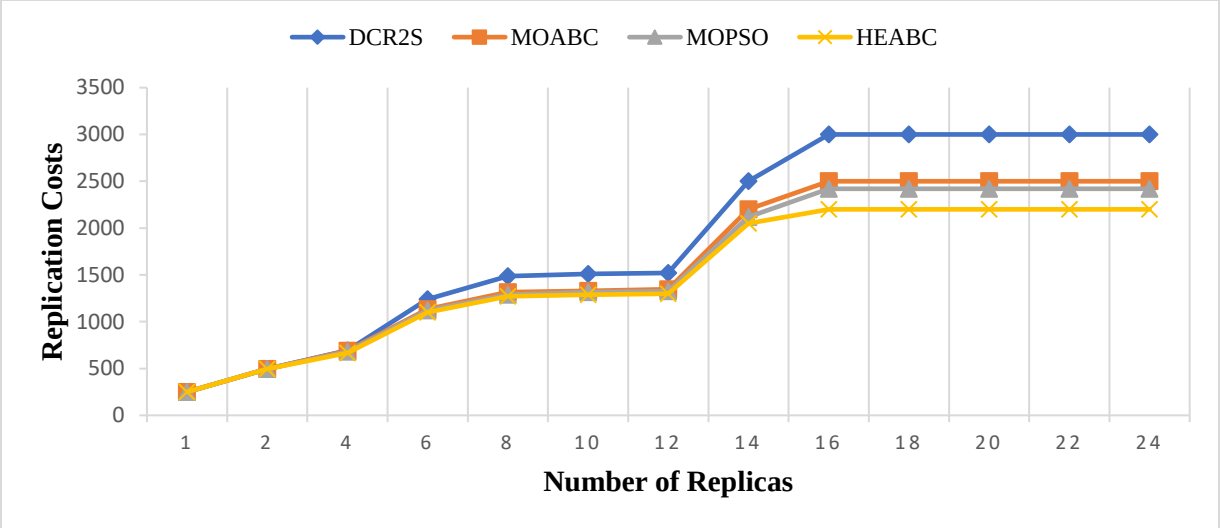
Parameter	Value
Number of employed bees	25
Number of onlooker bees	25
Number of scout bees	5
mcn = Max Iterations	1000

4.3.2. Comparing the proposed algorithm based on Cost of replication and number of replicas

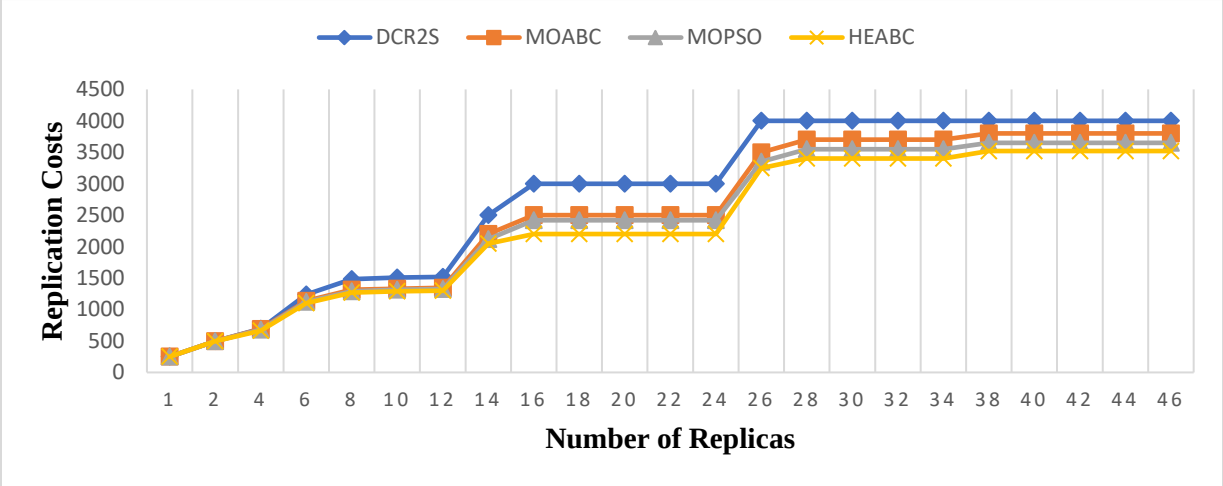
Figure 4.2 shows the comparison between the HEABC algorithm and two swarm intelligence-based algorithms (multi-objective particle swarm optimization and multi-objective artificial bee colony algorithm) and additional with mathematical model algorithm which is Dynamic Cost-aware Re-Replication and Re-balancing Strategy (DCR2S) algorithm regarding the total cost of replication when using the knapsack problem, which affects the increased replication number. Regardless, before DCR2S, when the requested numbers of data file replications increase, the total cost of replication is increased automatically. Therefore, when requesting the number of replicas, the costs of data replication increase at a high rate. Alternately, MOPSO and MOABC algorithms shows cost savings within the request number of replicas by users than DCR2S algorithm but it still not much better in data availability. For comparison of the MOPSO, MOABC and DCR2S algorithms in the process saving cost, we notice that the HEABC algorithm is superior to the others algorithm.



(a)



(b)



(c)

Figure 4.2 Cost of replication with the number of replicas for three types of budget cases

4.3.3. Selecting the optimally best replica

Figure 4.3 shows the use of the HEABC, MOPSO, MOPSO, EFS and DC2RS algorithms, showing the number of requested users, as well as their influence on cost. The number of Cloudlets determines data replication, and increases availability through DCs. Figure 4.5 shows access to the best data replica, which has the highest popularity through the HEABC algorithm to determine the optimal data replication. Based on ITBDF, which determines the

high popularity of data replica, the proposed HEABC algorithm is superior to other algorithms when accessing the optimal data replication and probability of data availability.

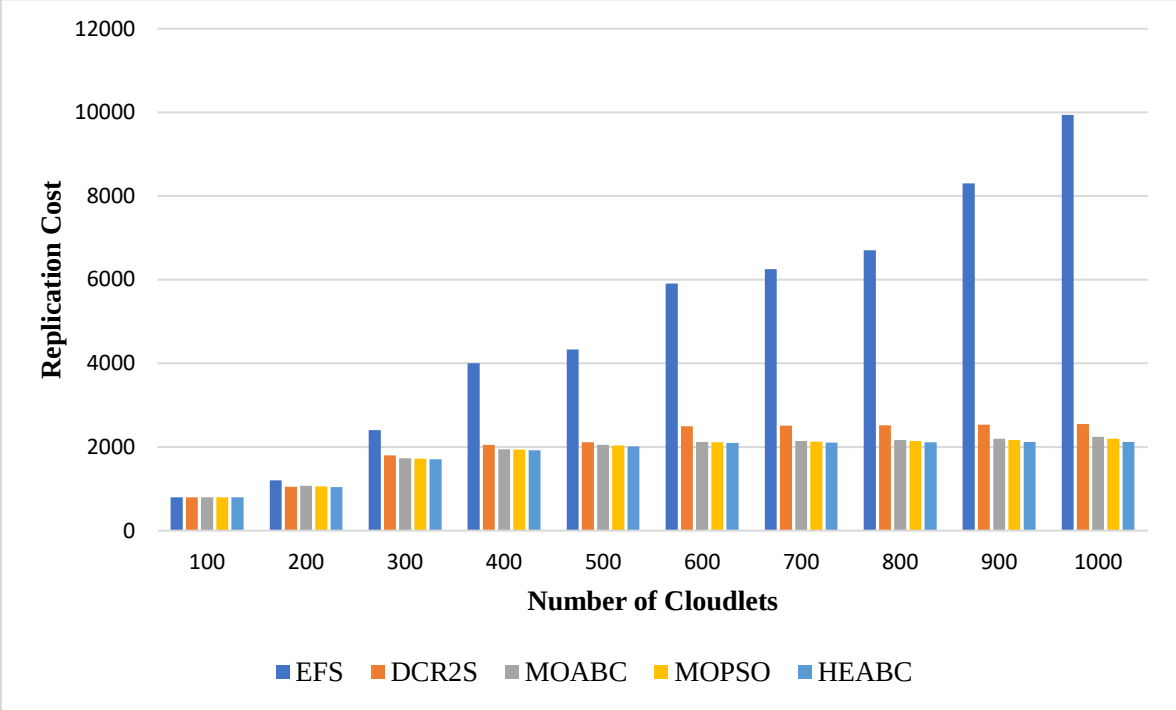


Figure 4.3 Cost of replication and number of cloudlets

Probability of file availability and number of replicas

In Figure 4.4, the effect of using HEABC algorithm on the probability of the file availability of a data file is shown when different values of the budget are considered. In general, as the number of replicas increases, the probability of availability also increases. Th available probability of a data file increases as the budget associated with the data file increases. As shown in Figure 4.4, when 23 replicas are there, then the available probability of a data file in case of budget = 3000 is larger than the available probability of that data file in case the budget is 1500. Similarly, when 46 replicas are considered, then the available probability of a data file in case of budget = 4000 is larger than the available probability of that file in case of budget = 3000. Thus, more available probability can be achieved from the same number of replicas when the budget value is more as compared to the available probability for the same number of replicas when the budget value is less. Hence, with the increase in the value of the budget, the available probability also increases with the increase in the number of replicas.

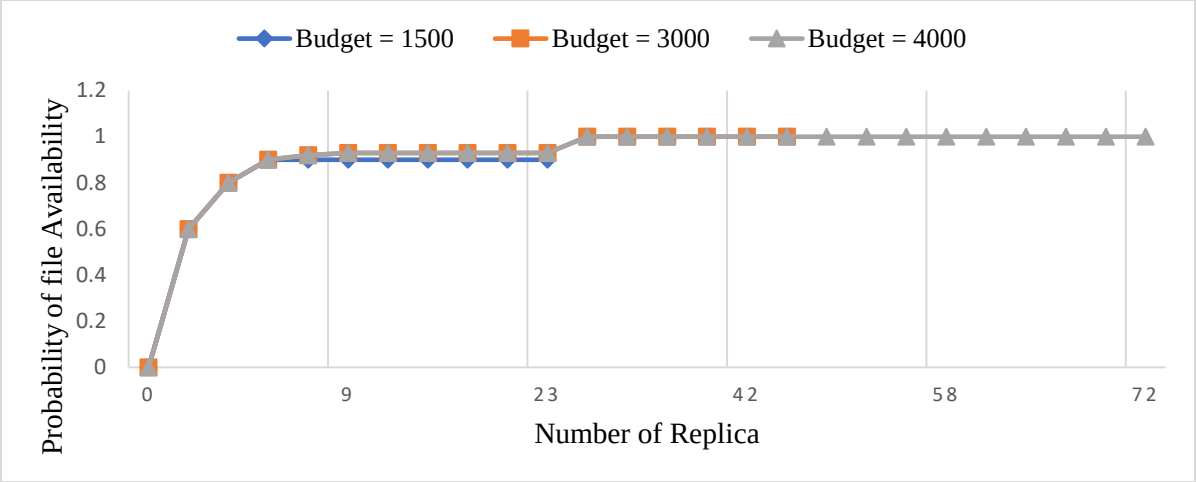


Figure 4.4 Probability of file availability and number of replicas

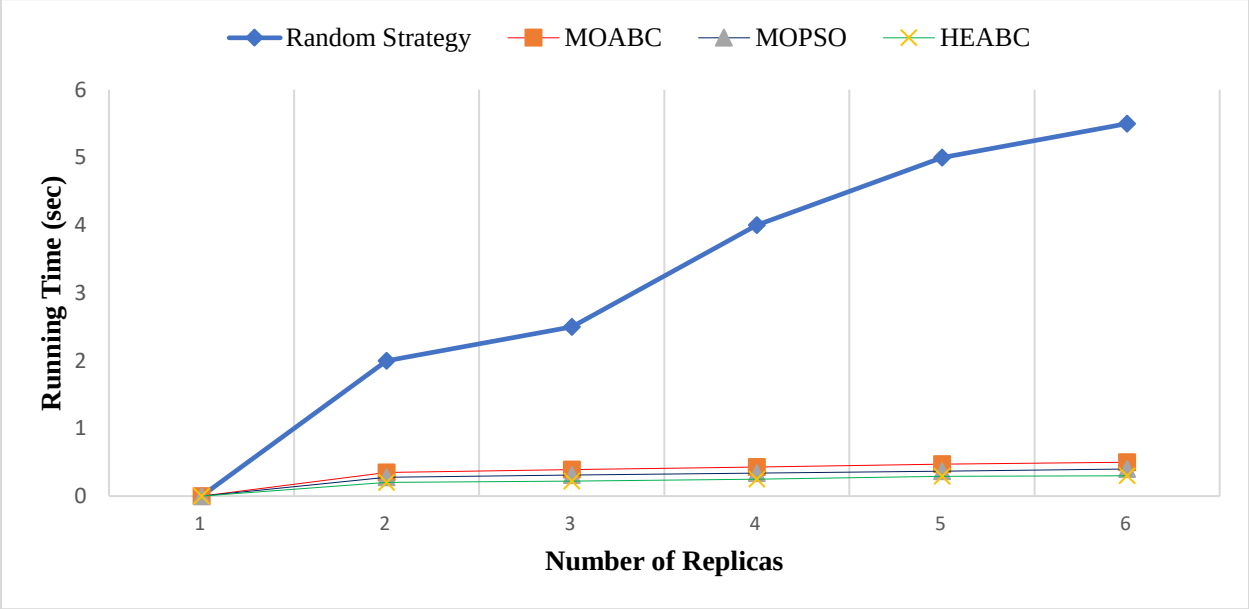


Figure 4.5 Running time for selecting optimal replica

4.3.4. The optimally best placement of replicas

The evaluation is done for many aspects, including the least-cost path and the distance to reach the optimal replica placement locations in DCs. Several approaches are implemented to assess the transmission process in DCs such as the number of tasks and number of nodes. In Figure 4.6 and 4.7 HEABC is used to determine the number of replications through DCs and the data transmission rate that reduces time and costs. The comparison of the results of our proposed algorithm with the other algorithms shows that our strategy is superior with respect to the time and costs of optimizing the placement of replicas.

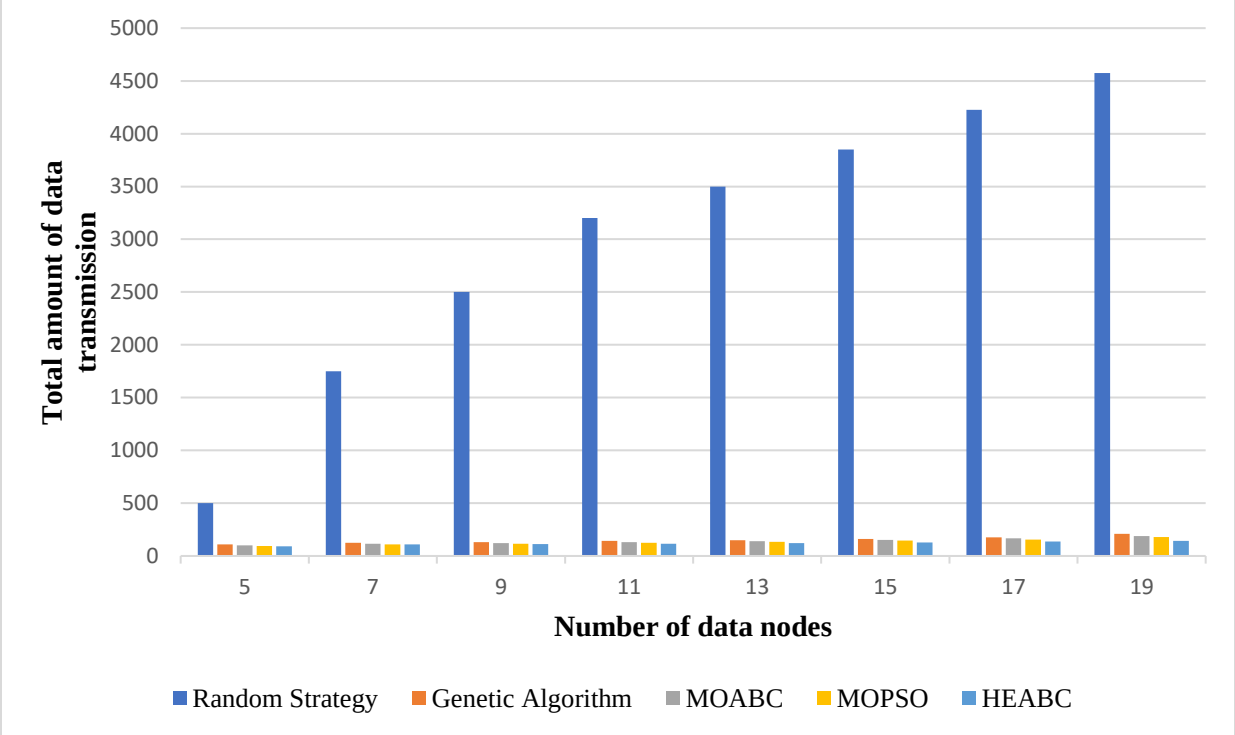


Figure 4.6 Total amount of data transmission time and data nodes

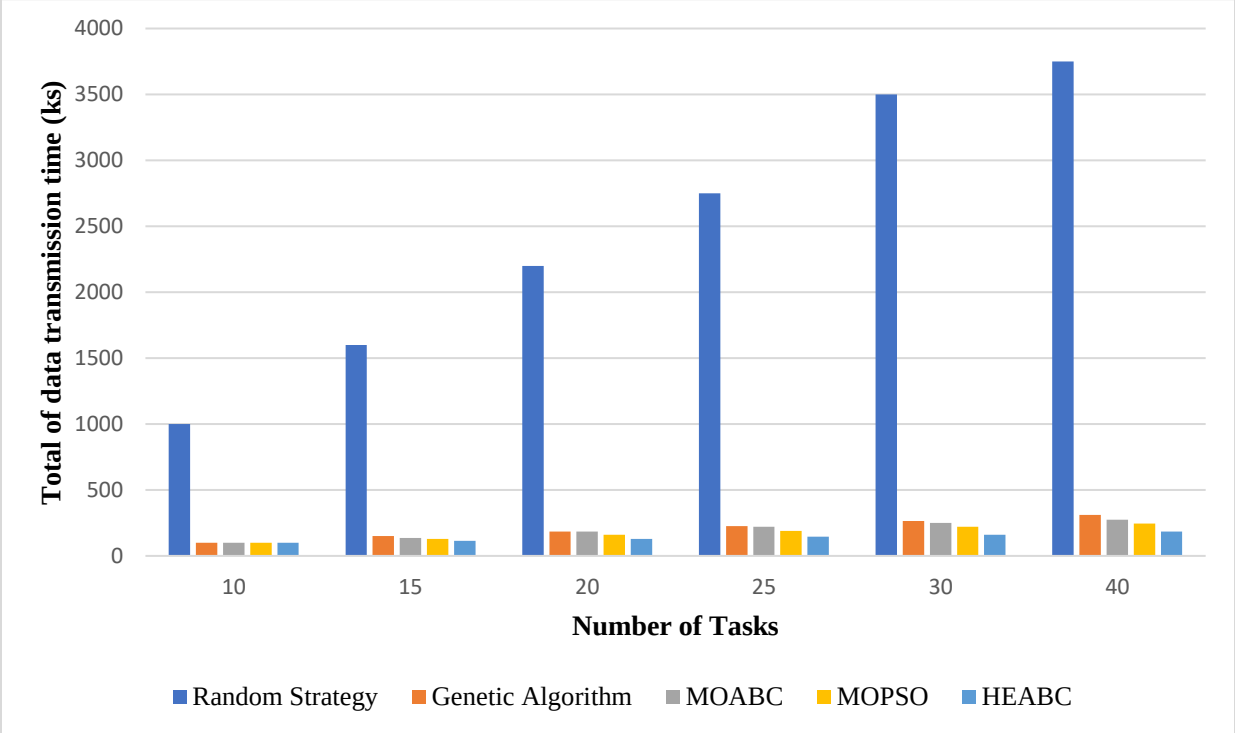


Figure 4.7 Total data transmission and number of tasks

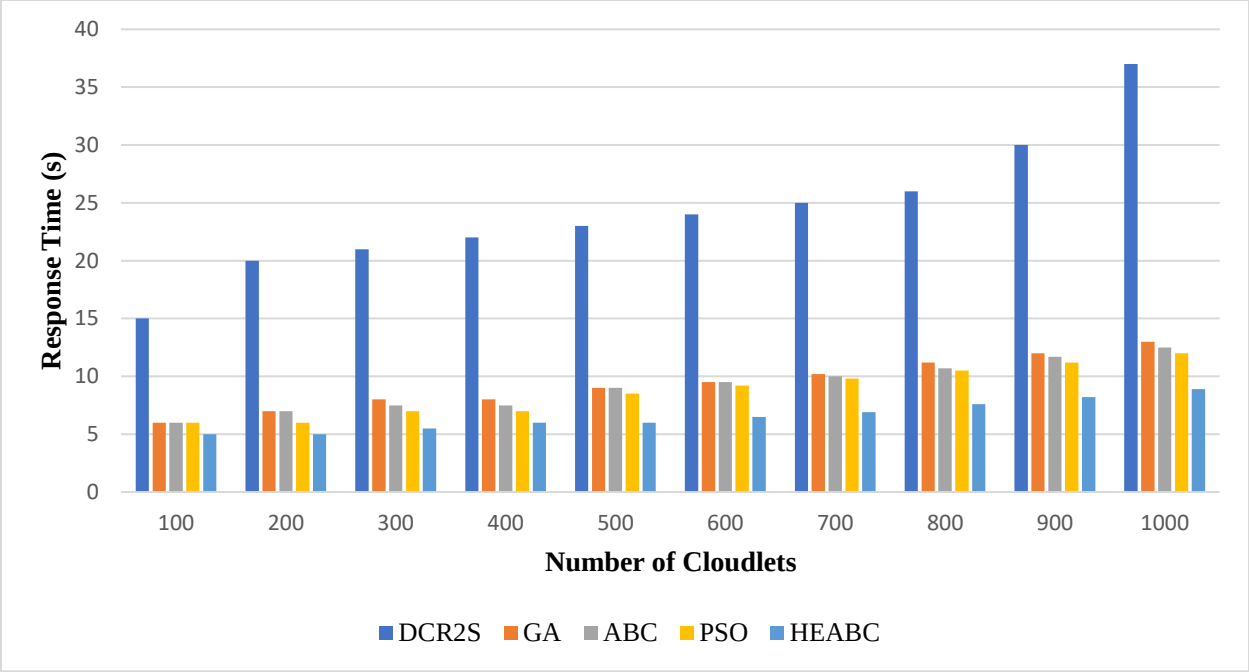


Figure 4.8 Response time for cloudlets

The proposed algorithm evaluation result summary

The proposed HEABC is a combination of particle swarm optimization and enhanced artificial bee colony algorithms. So, the technique is utilized to access the most common data using user request tasks to access data replicas in the least period of time for the lowest cost and maximum availability after several attempts according to the HEABC requirements. Then it is utilized to position or place replications at appropriate locations based on budget, time and distances between DCs. In addition, global search is improved by discovering a new search field. The suggested technique produces results in any period by constructing HEABC solutions for expense, availability, distance and time of data replication. The best location of data duplication will be maintained as the best global approach following many attempts to use the criterion.

The effectiveness of the proposed algorithm has been evaluated using multiple metrics. The average response times of the replication strategy are given. Based on the experiment results, the proposed algorithm has reduced replication cost by 13.85% and waiting time by 18.37% compared to DCR2S, It has also reduce waiting time by 6% and cost reduction by 8.2 %, outperforming GA. reduced waiting time by 5.2% and replication cost reduction by 6.1% ABC,

and reduced waiting time by 4.7% and replication cost reduction by 5.7%, outperforming PSO. Furthermore, As the number of user tasks to determine and placement replications increases, the average response time gradually increases but our strategy efficiently reduces the time.

Chapter summary

In this research work, the performance of hybrid enhanced artificial bee colony optimization algorithm has been evaluated with two other swarm intellegence based algorithms such as particle swarm optimization, artificial bee colony algorithm and genetic algorithm. In addition, cost-aware dynamic re-replication and re-balancing algorithm have been evaluated with the proposed algorithm in terms of waiting time reduction, replication cost reduction, data file availability during running time, and data transmission time with various number of replicas and cloudlets. As it can be noticed from bar charts, the proposed algorithm reduces waiting time and replication cost by placing the popular data file replica to the node of nearest to user that help to fast data access and to load balancing better than other approaches. On the cloudsim simulator, the effectiveness of the proposed approach was evaluated and compared to Genetic algorithm (GA), Particle Swarm Obtimization (PSO), Artificial Bee Colony (ABC) and Cost-aware Dynamic Re-replication and Re-balancing Strategy (CDR2S). As a result, we found that our algorithm outperformed the previously mentioned ones.

CHAPTER FIVE

5. CONCLUSION AND FUTURE WORK

This chapter summarizes the research work and additional work that needs to be done in the future in the area of data replication that will ensure cloud performance by increasing data availability in order to make cloud computing environments provide quality services enormous tenants.

5.1. Conclusion

In cloud computing environment failures are normal rather than exceptional. Replication strategies have been widely adopted in current cloud systems for data availability, reliability, and performance. The adaptation improves system resilience during disasters without any downtime. Therefore, the trend toward cloud replication strategies for storing and replicating large amounts of geographically distributed data requires an optimal replication strategy for acceptable performance. In this research, hybrid enhanced artificial bee colony optimization algorithm which derived from a combination Enhanced ABC and Particle Swarm Optimization has been proposed to reduce cost and waiting time. The proposed replica selection and placement optimization strategy provide more least-cost paths, better response times and replication costs within the budget.

Data replication and placement is an NP-hard problem that requires a meta-heuristics algorithm to get optimal solutions. In this research, we proposed hybrid enhanced artificial bee colony optimization algorithm that to increase data availability by placing data replica to appropriate node that is nearest users.

The proposed algorithm was compared using the cloudsim simulator to well-known algorithms such as Dynamic Cost-aware Re-replication and Re-balancing Strategy (DCR2S), Multi Objective Particle Swarm Optimization (MOPSO) and Multi Objective Artificial Bee Colony (ABC) Algorithm. we found that, when the distance between DCs and user budget is taken into account, the proposed algorithm reduces replication cost and access time more than other specified algorithms.

5.2. Contribution

The proposed HEABC algorithm has contributed better response time, low waiting to access the data, replication costs within the user budget and selected the best replica placement nearest to users. Once the replication cost exceeds the user budget, the replication algorithm optimizes the replication cost.

5.3. Future Work

Future research in cloud data replication and placement can focus on several areas. First, predict the size (quantity) of replica that should involve in replication at an interval of any given time. The other is an exploration of sophisticated techniques for automating the process, for example, using machine learning algorithms to identify patterns in data access and latency to determine optimal replication strategies. There is also potential for further study in exploring ways to balance the tradeoffs between security, privacy and performance when replication strategies are employed.

Special Acknowledgment

This research project is funded by Adama Science and Technology University under the grant number:

ASTU/SM-R/503/22

Adama, Ethiopia

REFERENCE

Akay, B., & Karaboga, D. (2012a). A modified Artificial Bee Colony algorithm for real-parameter optimization. *Information Sciences*, *192*, 120–142. https://doi.org/10.1016/j.ins.2010.07.015

Akay, B., & Karaboga, D. (2012b). *Artificial bee colony algorithm for large-scale problems and engineering design optimization*. 1001–1014. https://doi.org/10.1007/s10845-010-0393-4

Al Nuaimi, E., Al Neyadi, H., Mohamed, N., & Al-Jaroodi, J. (2015). Applications of big data to smart cities. *Journal of Internet Services and Applications*, *6*(1), 1–15. https://doi.org/10.1186/s13174-015-0041-5

Alami Milani, B., & Jafari Navimipour, N. (2016). A comprehensive review of the data replication techniques in the cloud environments: Major trends and future directions. *Journal of Network and Computer Applications*, *64*, 229–238. https://doi.org/10.1016/j.jnca.2016.02.005

Anderson, Z. (2014). Cassandra. *Dancing Times*, *105*(1252), 43. https://doi.org/10.1145/1773912.1773922

Awad, A., Salem, R., Abdelkader, H., & Salam, M. A. (2021). A Novel Intelligent Approach for Dynamic Data Replication in Cloud Environment. *IEEE Access*, *9*, 40240–40254. https://doi.org/10.1109/ACCESS.2021.3064917

Boru, D., Kliazovich, D., Granelli, F., Bouvry, P., & Zomaya, A. Y. (2015). Energy-efficient data replication in cloud computing datacenters. *Cluster Computing*, *18*(1), 385–402. https://doi.org/10.1007/s10586-014-0404-x

Bsoul, M., Al-Khasawneh, A., Abdallah, E. E., & Kilani, Y. (2011). Enhanced Fast Spread Replication strategy for Data Grid. *Journal of Network and Computer Applications*, *34*(2), 575–580. https://doi.org/10.1016/j.jnca.2010.12.006

Bsoul, M., Al-khasawneh, A., Eddien, E., & Kilani, Y. (2011). Journal of Network and Computer Applications Enhanced Fast Spread Replication strategy for Data Grid. *Journal of Network and Computer Applications*, *34*(2), 575–580. https://doi.org/10.1016/j.jnca.2010.12.006

Buyya, R., Ranjan, R., & Calheiros, R. N. (2009). *Modeling and Simulation of Scalable*

Cloud Computing Environments and the CloudSim Toolkit : Challenges and Opportunities. 1–11.

Buyya, R., Yeo, C. S., & Venugopal, S. (2008). Market-oriented cloud computing: Vision, hype, and reality for delivering IT services as computing utilities. *Proceedings - 10th IEEE International Conference on High Performance Computing and Communications, HPCC 2008*, 5–13. https://doi.org/10.1109/HPCC.2008.172

Civicioglu, P., & Besdok, E. (2013). *A conceptual comparison of the Cuckoo-search , particle swarm optimization , differential evolution and artificial bee colony algorithms*. https://doi.org/10.1007/s10462-011-9276-0

Computing, T., Computing, S., Computing, B. D., Congress, S. W., Tos, U., Mokadem, R., Hameurlain, A., Ayav, T., & Bora, S. (2016). *A Performance and Profit Oriented Data Replication Strategy for Cloud Systems*. https://doi.org/10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.55

DeCandia, G., Hastorun, D., Jampani, M., Kakulapati, G., Lakshman, A., Pilchin, A., Sivasubramanian, S., Vosshall, P., & Vogels, W. (2007). Dynamo: Amazon's highly available key-value store. *Operating Systems Review (ACM)*, 205–220. https://doi.org/10.1145/1294261.1294281

Doğan, A. (2009). A study on performance of dynamic file replication algorithms for real-time file access in Data Grids. *Future Generation Computer Systems*, *25*(8), 829–839. https://doi.org/10.1016/j.future.2009.02.002

Gill, N. K., & Singh, S. (2016). A dynamic , cost-aware , optimized data replication strategy for heterogeneous cloud data centers. *Future Generation Computer Systems*, *65*, 10–32. https://doi.org/10.1016/j.future.2016.05.016

Hasançebi, O., Çarbaş, S., Doğan, E., Erdal, F., & Saka, M. P. (2010). Comparison of non-deterministic search techniques in the optimum design of real size steel frames. *Computers and Structures*, *88*(17–18), 1033–1048. https://doi.org/10.1016/j.compstruc.2010.06.006

Humane, P., & Varshapriya, J. N. (2015). Simulation of cloud infrastructure using CloudSim simulator: A practical approach for researchers. *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials, ICSTM 2015 - Proceedings*, *May*, 207–211.

https://doi.org/10.1109/ICSTM.2015.7225415

Janpet, J., & Wen, Y. F. (2013). Reliable and available data replication planning for cloud storage. *Proceedings - International Conference on Advanced Information Networking and Applications, AINA*, 772–779. https://doi.org/10.1109/AINA.2013.125

Jimenez-peris, R. (2014). *Towards Large-Scale Replicated Databases*. *March 2007*, 2–4.

Journal, I., & Computing, I. (2009). *Enhanced artificial bee colony optimization*. *5*(12), 1–12.

Juneja, M., & Nagar, S. K. (2017). Particle swarm optimization algorithm and its parameters: A review. *ICCCCM 2016 - 2nd IEEE International Conference on Control Computing Communication and Materials*, *Iccccm*. https://doi.org/10.1109/ICCCCM.2016.7918233

Kakandikar, G. M., & Nandedkar, V. M. (2018). Engineering Optimization. In *Sheet Metal Forming Optimization*. https://doi.org/10.4324/9781315156101-4

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied Soft Computing Journal*, *8*(1), 687–697. https://doi.org/10.1016/j.asoc.2007.05.007

Karuppusamy, S., & Muthaiyan, M. (2016). An efficient placement algorithm for data replication and to improve system availability in cloud environment. *International Journal of Intelligent Engineering and Systems*, *9*(4), 88–97. https://doi.org/10.22266/ijies2016.1231.10

Lin, J. W., Chen, C. H., & Chang, J. M. (2013). QoS-aware data replication for data-intensive applications in cloud computing systems. *IEEE Transactions on Cloud Computing*, *1*(1), 101–115. https://doi.org/10.1109/TCC.2013.1

Long, S. Q., Zhao, Y. L., & Chen, W. (2014). MORM: A Multi-objective Optimized Replication Management strategy for cloud storage cluster. *Journal of Systems Architecture*, *60*(2), 234–244. https://doi.org/10.1016/j.sysarc.2013.11.012

Lv, X., Wang, Y., Deng, J., Zhang, G., & Zhang, L. (2018). Improved Particle Swarm Optimization Algorithm Based on Last-Eliminated Principle and Enhanced Information Sharing. *Computational Intelligence and Neuroscience*, *2018*. https://doi.org/10.1155/2018/5025672

Maheshwari, N., Nanduri, R., & Varma, V. (2012). Dynamic energy efficient data placement and cluster reconfiguration algorithm for MapReduce framework. *Future Generation*

Computer Systems, *28*(1), 119–127. https://doi.org/10.1016/j.future.2011.07.001

Mansouri, N., & Javidi, M. M. (2018). PT. *The Journal of Systems & Software*. https://doi.org/10.1016/j.jss.2018.05.027

Mansouri, N., Javidi, M. M., & Mohammad Hasani Zade, B. (2021). A CSO-based approach for secure data replication in cloud computing environment. *The Journal of Supercomputing*, *77*(6), 5882–5933. https://doi.org/10.1007/s11227-020-03497-3

Mansouri, Y., Nadjaran Toosi, A., & Buyya, R. (2019). Cost Optimization for Dynamic Replication and Migration of Data in Cloud Data Centers. *IEEE Transactions on Cloud Computing*, *7*(3), 715–718. https://doi.org/10.1109/TCC.2017.2659728

Millán Tejedor, R., & Esfandiari, S. (2014). Qué es NFV (Network Functions Virtualization). *Bit*, *196*, 18.

Mohammadi, B., & Navimipour, N. J. (2022). A Fuzzy Logic-Based Method for Replica Placement in the Peer to Peer Cloud Using an Optimization Algorithm. *Wireless Personal Communications*, *122*(2), 981–1005. https://doi.org/10.1007/s11277-021-08936-9

Mokadem, R., & Hameurlain, A. (2020). A data replication strategy with tenant performance and provider economic profit guarantees in Cloud data centers. *Journal of Systems and Software*, *159*. https://doi.org/10.1016/j.jss.2019.110447

Nannai John, S., & Mirnalinee, T. T. (2020). A novel dynamic data replication strategy to improve access efficiency of cloud storage. *Information Systems and E-Business Management*, *18*(3), 405–426. https://doi.org/10.1007/s10257-019-00422-x

Özsu, M. T., & Valduriez, P. (2011). Multidatabase Query Processing. In *Principles of Distributed Database Systems, Third Edition*. https://doi.org/10.1007/978-1-4419-8834-8_9

Pamies-Juarez, L., García-López, P., Sánchez-Artigas, M., & Herrera, B. (2011). Towards the design of optimal data redundancy schemes for heterogeneous cloud storage infrastructures. *Computer Networks*, *55*(5), 1100–1113. https://doi.org/10.1016/j.comnet.2010.11.004

Philip Chen, C. L., & Zhang, C. Y. (2014). Data-intensive applications, challenges, techniques and technologies: A survey on Big Data. *Information Sciences*, *275*, 314–347. https://doi.org/10.1016/j.ins.2014.01.015

Qu, Y., & Xiong, N. (2012). RFH: A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage. *Proceedings of the International Conference on Parallel Processing*, 520–529. https://doi.org/10.1109/ICPP.2012.3

Rajan, R. A. P. (2012). Evolution of Cloud Storage as Cloud Computing Infrastructure Service. *IOSR Journal of Computer Engineering*, *1*(1), 38–45. https://doi.org/10.9790/0661-0113845

Rashedi, E., Nezamabadi-pour, H., & Saryazdi, S. (2009). GSA : A Gravitational Search Algorithm. *Information Sciences*, *179*(13), 2232–2248. https://doi.org/10.1016/j.ins.2009.03.004

Ridhawi, I. Al, Mostafa, N., & Masri, W. (2015). Location-aware data replication in cloud computing systems. *2015 IEEE 11th International Conference on Wireless and Mobile Computing, Networking and Communications, WiMob 2015*, 20–27. https://doi.org/10.1109/WiMOB.2015.7347936

Rocha, Á., Correia, A. M., Costanzo, S., & Reis, L. P. (2015). New Contributions in Information Systems and Technologies. *Advances in Intelligent Systems and Computing*, *353*(December 2017), III–IV. https://doi.org/10.1007/978-3-319-16486-1

Saka, M. P., Dogan, E., & Aydogdu, I. (2013). Analysis of Swarm Intelligence-Based Algorithms for Constrained Optimization. *Swarm Intelligence and Bio-Inspired Computation: Theory and Applications*, 25–48. https://doi.org/10.1016/B978-0-12-405163-8.00002-8

Salem, R., Salam, M. A., Abdelkader, H., & Awad Mohamed, A. (2020). An Artificial Bee Colony Algorithm for Data Replication Optimization in Cloud Environments. *IEEE Access*, *8*, 51841–51852. https://doi.org/10.1109/ACCESS.2019.2957436

Sasikumar, K., & Vijayakumar, B. (2020). An efficient multi-objective model for data replication in cloud computing environment. *International Journal of Enterprise Information Systems*, *16*(1), 69–91. https://doi.org/10.4018/IJEIS.2020010104

Sedighizadeh, D., & Mazaheripour, H. (2018). Optimization of multi objective vehicle routing problem using a new hybrid algorithm based on particle swarm optimization and artificial bee colony algorithm considering Precedence constraints. *Alexandria Engineering Journal*, *57*(4), 2225–2239. https://doi.org/10.1016/j.aej.2017.09.006

Shakarami, A., Ghobaei-Arani, M., Shahidinejad, A., Masdari, M., & Shakarami, H. (2021).

Data replication schemes in cloud computing: a survey. In *Cluster Computing* (Vol. 24, Issue 3). Springer US. https://doi.org/10.1007/s10586-021-03283-7

Shorfuzzaman, M., & Masud, M. (2019). Leveraging a multi-objective approach to data replication in cloud computing environment to support big data applications. *International Journal of Advanced Computer Science and Applications*, *10*(3), 418–429. https://doi.org/10.14569/IJACSA.2019.0100354

Sousa, F. R. C., & Machado, J. C. (2012). Towards elastic multi-tenant database replication with quality of service. *Proceedings - 2012 IEEE/ACM 5th International Conference on Utility and Cloud Computing, UCC 2012*, 168–175. https://doi.org/10.1109/UCC.2012.36

Sun, D. (2012). *Modeling a Dynamic Data Replication Strategy to Increase System Availability in Cloud Computing Environments*. *27*(2), 256–272. https://doi.org/10.1007/s11390-012-1221-4

Suryateja, P. S. (2016). A Comparative Analysis of Cloud Simulators. *International Journal of Modern Education and Computer Science*, *8*(4), 64–71. https://doi.org/10.5815/ijmecs.2016.04.08

(Time to relax) Slater, Noah, Anderson, J. (n.d.).

Vulimiri, A., Curino, C., Godfrey, P. B., Jungblut, T., Karanasos, K., Padhye, J., & Varghese, G. (2015). Wanalytics: Geo-distributed analytics for a data intensive world. *Proceedings of the ACM SIGMOD International Conference on Management of Data*, *2015-May*, 1087–1092. https://doi.org/10.1145/2723372.2735365

Wang, Y. H., & Wu, I. C. (2009). Achieving high and consistent rendering performance of java AWT/Swing on multiple platforms. *Software - Practice and Experience*, *39*(7), 701–736. https://doi.org/10.1002/spe

Waseem, Q., Wan Din, W. I. S., Alshamrani, S. S., Alharbi, A., & Nazir, A. (2021). Quantitative analysis and performance evaluation of target-oriented replication strategies in cloud computing. *Electronics (Switzerland)*, *10*(6), 1–49. https://doi.org/10.3390/electronics10060672

Wei, Q., Veeravalli, B., Gong, B., Zeng, L., & Feng, D. (2010). CDRM: A cost-effective dynamic replication management scheme for cloud storage cluster. *Proceedings - IEEE International Conference on Cluster Computing, ICCC*, 188–196.

https://doi.org/10.1109/CLUSTER.2010.24

Wheeler, A., & Winburn, M. (2015). Data in the Cloud. In *Cloud Storage Security* (pp. 1–22). Elsevier. https://doi.org/10.1016/B978-0-12-802930-5.00001-0

Xhafa, F., Barolli, L., Barolli, A., & Papajorgji, P. (2015). *Modeling and Processing for Next- Generation Big-Data Technologies With Applications and Case Studies*.

Zhang, H., Lin, B., Liu, Z., & Guo, W. (2014). Data replication placement strategy based on bidding mode for cloud storage cluster. *Proceedings - 11th Web Information System and Application Conference, WISA 2014*, 207–212. https://doi.org/10.1109/WISA.2014.45

APPENDIX

A.1 DatacenterCreator

```
 public class DatacenterCreator {

public s t a t i c Datacenter createDatacenter ( String name) {

List <Host> hostList = new ArrayList<Host >() ;

List <Pe> peList = new ArrayList<Pe>() ;

i n t mips = 500;

for ( i n t i = 0; i < 10; i ++) {

Random rojb = new Random( ) ;

PeProvisionerSimple pProvisioner = new PeProvisionerSimple (mips + rojb .nextInt (4500) )
;

Pe core = new Pe( i , pProvisioner ) ;

peList . add ( core ) ;

}

i n t ram = 2048;

long storage = 1000000;

i n t bw = 10000;

for ( i n t i = 0; i < 10; i ++) {

Random rojb1 = new Random( ) ;

hostList . add (new Host ( i , new RamProvisionerSimple (ram + rojb1 . nextInt(100) ) ,new
BwProvisionerSimple (bw + rojb1 . nextInt (100) ) , storage + rojb1 . nextInt (100) ,

new VmSchedulerSpaceShared ( peList ) ) ) ; / / This i s our f i r s t machine

}String arch = "x86" ;

String os = "Windows" ;
```

String vmm = "Xen" ;

double time_zone = 10.0;

double cost = 3 .0 ;

double costPerMem = 0.05;

double costPerStorage = 0 .1 ;

double costPerBw = 0 .1 ;

LinkedList<Storage> storageList = new LinkedList<Storage >() ;

DatacenterCharacteristics c h a r a c t e r i s t i c s = new DatacenterCharacteristics(arch , os , vmm, hostList , time_zone ,cost ,costPerMem , costPerStorage , costPerBw) ;

Datacenter datacenter = null ;

t r y {

datacenter = new Datacenter (name, c h a r a c t e r i s t i c s , new VmAllocationPolicySimple(hostList) , storageList , 0) ;

} catch (Exception e) {

e . printStackTrace ( ) ;}

return datacenter ;

} }

A.2 Simulation

public Simulation(**int** cloudletSchedulerType, **int** numOfCloudlets, **int** numOfVMs, **int** brokerType, **int** fitnessType, Random rng, **boolean** silent, **int** highHeterogeneity) {

this.cloudletSchedulerType = cloudletSchedulerType;

this.numOfCloudlets = numOfCloudlets;

this.numOfVMs = numOfVMs;

this.brokerType = brokerType;

this.fitnessType = fitnessType;

```
this.rng = rng;
this.silent = silent;
VM_MIPS_POWERS = new double[numOfVMs];
for (int i = 0; i < numOfVMs; i++) {
if (highHeterogeneity == 1) {
VM_MIPS_POWERS[i] = rng.nextInt(901) + 100;
} else {
VM_MIPS_POWERS[i] = rng.nextInt(101) + 500;
}}
CLOUDLET_LENGTHS = new int[numOfCloudlets];
for (int i = 0; i < this.numOfCloudlets; i++) {
CLOUDLET_LENGTHS[i] = 1000 + rng.nextInt(1001);
}
ETC_MATRIX = new double[numOfCloudlets][numOfVMs];
for (int i = 0; i < numOfCloudlets; i++) {
for (int j = 0; j < numOfVMs; j++) {
ETC_MATRIX[i][j] = (double) CLOUDLET_LENGTHS[i] / VM_MIPS_POWERS[j];
}}}
private List<Cloudlet> cloudletList;
private List<Vm> vmlist;
private List<Vm> createVM(int userId, int numOfVMs) {
LinkedList<Vm> list = new LinkedList<Vm>();
long size = 1024;
```

```
int ram = 1024;
double mips;
long bw = 1024;
int pesNumber = 1;
String vmm = "Xen";
Vm[] vm = new Vm[numOfVMs];
for (int i = 0; i < numOfVMs; i++) {
CloudletScheduler cs = null;
switch (cloudletSchedulerType) {
cs = new CloudletSchedulerSpaceShared();
cs = new CloudletSchedulerTimeShared();
}
mips = VM_MIPS_POWERS[i];
Log.printLine("VM " + i + " MIPS: " + mips);
vm[i] = new Vm(i, userId, mips, pesNumber, ram, bw, size, vmm, cs);
list.add(vm[i]);
} return list; }
private List<Cloudlet> createCloudlet(int userId, int numOfCloudlets) {
LinkedList<Cloudlet> list = new LinkedList<Cloudlet>();
long length;
long fileSize = 300;
long outputSize = 300;
int pesNumber = 1;
```

```
UtilizationModel utilizationModel = new UtilizationModelFull();

Cloudlet[] cloudlet = new Cloudlet[numOfCloudlets];

for (int i = 0; i < numOfCloudlets; i++) {

length = CLOUDLET_LENGTHS[i];

Log.printLine("Cloudlet " + i + " length: " + length);

cloudlet[i] = new Cloudlet(i, length, pesNumber, fileSize, outputSize, utilizationModel,
utilizationModel, utilizationModel);

cloudlet[i].setUserId(userId);

list.add(cloudlet[i]);

}

return list;

}

private Datacenter createDatacenter(String name) {

List<Host> hostList = new ArrayList<Host>();

List<Pe> peList1 = new ArrayList<Pe>();

int mips = 1000;

for (int i = 0; i < numOfVMs; i++) {

peList1.add(new Pe(i, new PeProvisionerSimple(mips))); // need to store Pe id and MIPS
Rating

}

int hostId = 0;

int ram = 100 * 1024;

long storage = 100 * 1024;

int bw = 100 * 1024;
```

```
hostList.add(

new Host(

hostId,

new RamProvisionerSimple(ram),

new BwProvisionerSimple(bw),

storage,

peList1,

new VmSchedulerTimeSharedOverSubscription(peList1)
```

A.3 Sample Code of Hybrid Algorithm

```
public class HEABC{

 static int maxIteration = 50;

 static int swarmSize = 45;

 static int numDimensions = 10

 static int colonySize = 100;

 static int maxCycle = 20;

 static double lowerLimits [] = new double[numDimensions];

 static double upperLimits [] = new double[numDimensions];

public static void main (String args[]) {

ArrayList<Particle> swarmList= new ArrayList<Particle> ();

double bestCost, cost=0.0, pBest[]=new double[numDimensions], gBest[]= new
double[numDimensions] ;

for (int d=0; d<numDimensions; d++) {

pBest[d] = 0;

gBest[d] = Double.MAX_VALUE
```

```
 }
initializeSwarm(swarmList);
for (int i=0; i<maxIteration; i++) {
cost= colonySizeForAbc(swarmList);
if (cost < bestCost) {
setpbest(swarmList) }
updateGbst(swarmList, bestCost);  }   }
private static void initializeSwarm(ArrayList<Particle> swarm){
 Random randGenrtr= new Random();
for (int i= 0; i < swarmSize; i++ )      {
Particle particleInstance= new Particle(lowerLimits ,upperLimits);
for(int j=0; j < numOfParamters ;j++)          {
particleInstance.position [j] = (minParaBoundarys [j] + ((maxParBoundaryLimit )-
minParBoundarys- [j])randGenrator .nextDouble());
```

A.4. PSOAlgo to select replica

```
public PSOAlgo() {
Random r = new Random();
(int i=0 ; i<numParticles; i++) {
 for(int j=0;j<N;j++){
particle[i] = r.nextInt(2);
}
}
CostFunc = new double[numParticles][N] :
for (int p=0; p < numParticles; p++ ) {
float tempCost;
float GeneCostSequence;
```

```
int x[]=new int [N];
GeneCostSequence=0.0f:
for(int m=0; m< N ; m++) {
x[m]=particle[pnumParticles + m];
tempCost=CostFunctionCalculator(x);
 GeneCostSequence+= tempCost;
CostFunc pm=tempCost;   }  } }
 public static void main(String args){
PSOAlgo psObject = new PSOAlgo();
System.out.println("Optimal Replica: "+ Arrays.toString(psObject.gBest));
}
```