STATISTICS, OPTIMIZATION AND INFORMATION COMPUTING
Stat., Optim. Inf. Comput., Vol. **14**, July 2025, pp 264–281.
Published online in International Academic Press (www.IAPress.org)

Optimizing Data Replication in Cloud Computing Using Firefly-Based Algorithm for Selection and Placement

B.Hafiz, Heba Abdelrahman *, BenBella S.Tawfik, Hosam E.Refaat

Department of Information System, Faculty of Computers and Informatics, Suez Canal University, Egypt

Abstract The rapid adoption of cloud computing has driven extensive research into data replication methods and their practical applications. Data replication is a vital process in cloud systems, ensuring data availability, improving performance, and maintaining system stability. This is especially crucial for data-intensive applications that require the distribution and sharing of large volumes of information across geographically dispersed centers. However, managing this process presented significant challenges. As the number of data replicas increases and they are distributed across multiple locations, the associated costs and complexity of maintaining system usability, performance, and stability also rise. In this study, we initially randomized the distribution of data replication files across the cloud infrastructure to simulate a realistic scenario where data already exists within the system before the application of replication algorithms. This approach allowed the algorithms to optimize the replication process based on the initial data distribution and adapt to the evolving demands of incoming workloads. To address the challenges of dynamic data replication in cloud environments, this paper introduced two algorithms: the Firefly Optimization Algorithm for Data Replica Selection (FFO-S) and the Firefly Optimization Algorithm for Replica Placement (FFO-P). A detailed simulation study was performed using the CloudSim platform to assess the effectiveness of the proposed FFO-S and FFO-P algorithms. The simulation environment was designed to closely emulate real-world cloud infrastructures, ensuring the practical applicability of the results.

Keywords Cloud Computing; Data Replication; Replicated Data Selection; Replicated Data Placement; Optimization Algorithms; CloudSim Simulation

DOI: 10.19139/soic-2310-5070-2317

1. Introduction

Cloud computing is a technology that provides on-demand access to computing resources such as servers, storage, and applications over the internet. It enables users to store, manage, and process data in remote data centers rather than relying on local systems. This environment supports scalability, flexibility, and cost efficiency, allowing users to access resources as needed without requiring large infrastructure investments. Within this cloud infrastructure, various computing resources—servers, storage, networks, and software—are available on-demand and can be accessed by users or applications as required [1, 2]. The cloud computing model allows users to: Access Resources On-Demand: Users can utilize computing power, storage, and software from the cloud without needing to own or maintain the underlying infrastructure. Pay-as-You-Use: Users only pay for the resources they consume, avoiding upfront investments in hardware and software. Scale Dynamically: The cloud can automatically adjust resources up or down based on user needs, offering flexibility and elasticity. Leverage Shared Infrastructure: Multiple users share the same cloud resources, benefiting from economies of scale. Outsource IT Management: Cloud providers manage maintenance, updates, and security of the underlying infrastructure, relieving users of IT responsibilities [3, 4]. In essence, the cloud model abstracts the complexities of the underlying infrastructure, enabling users to focus on core applications and services while the cloud provider manages computing resources.

*Correspondence to: Heba Abdelrahman (Email: hebaabdelrahman815@gmail.com).

ISSN 2310-5070 (online) ISSN 2311-004X (print)

This model emphasizes key characteristics of cloud computing: on-demand access, pay-per-use, scalability, shared resources, and outsourced IT management. The cloud environment has played a pivotal role in advancing Artificial Intelligence (AI) technologies. Several benefits made the cloud environments valuable for AI applications including Scalable Computing Power(SCP): Cloud platforms provide virtually limitless computing resources, including powerful graphics processing units (GPUs) and specialized hardware, facilitating the training and deployment of complex AI models. Access to Big Data: Cloud storage allows AI systems to utilize large, diverse datasets hosted on the cloud, driving the development of data-intensive machine learning algorithms. Elastic Scalability: The cloud's ability to dynamically scale computing resources enables AI systems to handle fluctuating workloads and adapt to changing demands. Distributed Processing: Cloud infrastructure supports distributed processing, accelerating AI training and inference tasks through parallelization. Simplified Deployment: Cloud platforms simplify the deployment and management of AI-powered applications, allowing developers to concentrate on model development rather than infrastructure management[5, 6, 7]. AI techniques, In-turn are employed to enhance various aspects of Cloud Computing: Resource Optimization: AI algorithms optimize the allocation and utilization of cloud resources, improving efficiency and cost-effectiveness. Predictive Maintenance: AI models analyzed cloud infrastructure telemetry to predict and prevent potential failures, enhancing service reliability. Automated Management: AI-driven automation streamlines the provisioning, scaling, and maintenance of cloud resources, reducing administrative overhead. Intelligent Load Balancing: AI-based load-balancing algorithms dynamically distribute workloads across cloud resources, ensuring optimal performance. Enhanced Security: AI-powered threat detection strengthens cloud security by identifying anomalies and potential cyber threats. The synergistic relationship between AI and cloud computing has led to the emergence of "AI-as-a-Service" (AIaaS) offerings, where cloud providers deliver AI-powered services and tools to developers and businesses, further accelerating AI innovation. Cloud environments frequently host applications that handle large volumes of data, often requiring data sharing and dissemination across geographically dispersed locations. To ensure data availability, integrity, and scalability, cloud systems commonly employ replication techniques [8, 9, 10, 11, 12, 13]. Replication within cloud-based clusters ensures data consistency and accuracy across multiple nodes. This is achieved using specialized protocols that manage data reads and writes across replicated copies. Existing research has explored both static and dynamic replication approaches. Regardless of the method, three key challenges must be addressed: What data should be replicated? When should replication occur? Where should new replicas be placed? Addressing these questions effectively is essential for successful data replication in cloud environments, as the decisions made significantly impact the performance, reliability, and cost-effectiveness of data-intensive cloud applications[14].

1.1. Motivations

Data replication is essential in the cloud computing environment, yet implementing efficient and reliable replication remains a challenge. Increasing the number of replicas across locations can enhance availability, but it also incurs significant costs. To address this, researchers have framed data replication as an optimization problem. AI-driven dynamic replication strategies have outperformed traditional static methods in terms of reliability, efficiency, and cost-effectiveness. By leveraging optimization algorithms, cloud systems can determine optimal data replication configurations, balancing factors such as access latency, replica availability, and maintenance costs. This adaptive approach allows cloud infrastructures to deliver high-quality data processing services more efficiently and reliably.

1.2. Contributions

This study introduced two intelligent, dynamic data replication algorithms FFO-S for data replica selection and FFO-P for data replica placement. These algorithms optimize the replication process across cloud data centers. Simulated using CloudSim and evaluated against existing strategies including, Multi-Objective Particle Swarm Optimization(MO-PSO), Dynamic Cost-aware Re-replication and Re-balancing Strategy(DCR2S), Enhance Fast Spread(EFS), Genetic Algorithm (GA), Ant-Colony Optimization(ACO), Genetic adaptive Selection Algorithm (GASA), Replica Selection and Placement(RSP), Dynamic Replica Selection Ant Colony Optimization(DRSACO), the proposed methods demonstrate superior replication efficiency, reduced costs,

minimized bandwidth consumption, and enhanced data availability. By utilizing the Firefly Optimization Algorithm FFO-S and FFO-P offer an effective solution for dynamic data replication in the cloud, balancing replication costs, resource utilization, and data accessibility.

1.3. Paper Organization

This paper is organized as follows: 2 presents related work on data replication in cloud environments. 3 outlines the system model for data replication in the cloud. 4 introduces the proposed algorithms, FFO-S and FFO-P. 5 describes the simulation configuration used for evaluation. 6 discusses and analyzes the results of the proposed algorithms. 7 addresses the performance evaluation metrics. 8 concludes the study and highlights directions for future research.

2. Related Work

The research literature contained numerous studies on data replication techniques in cloud environments, exploring methods to address challenges like data availability, reliability, and performance. This body of work provided a foundation of knowledge, guiding the development of advanced replication solutions and helping researchers and architects identify opportunities for further innovation, as follows:

- A. Awad et al. proposed two bio-inspired algorithms - Multi-Objective Particle Swarm Optimization (MO-PSO) and Multi-Objective Ant Colony Optimization (MO-ACO) - to enhance both the selection and placement of data replicas in the cloud environment [15].

- Laila Bouhouch et al. introduced a combined strategy of data placement and dynamic data replication management to efficiently store datasets and reduce data transfer costs. The proposed approach considered data center characteristics, dataset-task dependencies, and storage capacity to determine optimal replication decisions[16].

- X. Sun et al. conducted a systematic review of data replication techniques in IoT environments, categorizing them into static, dynamic, and distributed approaches. The study highlighted the strengths and limitations of existing strategies, identified gaps in research, and proposed future directions for more effective data replication methods to enhance reliability, fault tolerance, and accessibility in IoT systems. [17].

- A. Sharma et al. proposed an optimized data replication strategy aimed at balancing data availability and system performance in cloud environments. The approach focuses on minimizing the number of replicas while ensuring reliable access and maintaining system efficiency, specifically within a multi-tiered cloud framework[18].

- M.Javidi et al. introduced the Hierarchical Data Replication Strategy (HDRS) to enhance performance in cloud computing. HDRS dynamically managed replicas by creating, placing, and replacing them based on file popularity and system load, optimizing storage use while meeting QoS requirements. Evaluations using CloudSim showed HDRS effectively reduces response times and bandwidth usage, improving overall system efficiency.[19].

- D. Rambabu et al. proposed an improved correlation strategy-based approach for task scheduling and data replication in cloud environments. The model integrates a Self-adaptive Dwarf Mongoose Optimization (SADMO) algorithm for optimized job scheduling and replica placement, achieving better efficiency in terms of bottleneck value, migration cost, VM load, and replication performance[20].

- Yahia et al. reviewed optimization challenges in cloud computing, focusing on threats like task scheduling, security, and energy efficiency. They emphasized the growing use of nature-inspired algorithms for solving complex, non-linear problems due to their simplicity and efficiency [21].

- H. Yu. The study proposed an Improved Particle Swarm Optimization (IPSO) algorithm to enhance resource scheduling efficiency in cloud computing. Using CloudSim, simulation experiments showed that IPSO outperformed traditional PSO by avoiding premature convergence, balancing virtual machine loads, and scaling efficiently as task numbers increased[22].

-P. Pirozmand et al. The study proposed an Improved Particle Swarm Optimization (IPSO) algorithm using Multi-Adaptive Learning to enhance task scheduling in cloud computing. It outperformed other methods in makespan, load balancing, and efficiency, achieving faster and optimal solutions.[23].

- Y. Ebadi et al. proposed an energy-aware data replication technique using tabu search and particle swarm optimization (PSO). The approach effectively determines the number of newly placed data replicas[24].

- N. Mansouri et al. introduced a dynamic data replication approach that leverages the 80/20 rule to optimize data access. This approach was evaluated against ADRS and RSP algorithms and was found to achieve the highest accuracy[25].

- M. Zheng et al. proposed a Deep Reinforcement Learning-based data replica placement scheme(BRPS), which optimizes latency, reliability, and load in edge–cloud environments. The scheme used Double Deep Q-Network (DDQN) to place replicas efficiently, dynamically adapting to changes in network conditions while ensuring system reliability and load balance. Experimental results showed significant improvements in latency and memory utilization compared to existing methods. [26].

- T. Yuan et al. proposed two methods for partially and fully replicated systems. These approaches aim to achieve causal consistency when executing tasks in stable, large-scale distributed networks. The two methods realize causal consistency for both partial and complete data replications[27].

- F. Prity et al. introduced a comprehensive review of swarm intelligence optimization techniques for task scheduling in cloud computing. The study analyzed various swarm-based algorithms, compared their performance metrics, and discussed simulation tools, highlighting their potential to enhance resource allocation and system performance while addressing future research challenges[28].

- N. Mansouri et al. illustrated A fuzzy-based self-defense algorithm for replication was introduced, focusing on six optimization objectives: availability, service time, load, energy consumption, latency, and centrality. This approach has yielded positive results across all these objectives[29].

The paper compared various data replication techniques and strategies, focusing on replica selection and placement. Previous studies had limitations, often overlooking the potential of AI (Artificial Intelligence). In contrast, the proposed FFO-S and FFO-P algorithms utilized AI to achieve optimal replica access, minimize retrieval costs, and serve the maximum number of user tasks. This study represented a significant advancement over earlier replication strategies in cloud environments.

3. System Model

This part exposes the architecture of the proposed model for data replica selection and placement in a cloud computing environment. The model adopts a heterogeneous approach through AI algorithms to enhance the efficiency of data replica selection and placement. This approach is particularly suitable because the datacenters on this model have varying storage capacities and different types of processors. Unlike a homogeneous method, which uses uniform elements and a single approach, a heterogeneous method combines diverse tools and methodologies, making it more adaptable to the differing characteristics of each datacenter. This diversity results in a more robust and efficient solution, as it can leverage the strengths of each element. The heterogeneous approach is better suited to address the specific needs and variations within the cloud environment.

DESCRIPTION OF THE MODEL:

3.1. Data Centers

The data centers(DCs)are represented as DCs=$\{dc_1, dc_2, dc_3, ..., dc_x\}$,where x denotes the number of data centers. The proposed scheme, illustrated in Figure1, showcases a hierarchical arrangement of data centers across different levels. At the first level are high-tier data centers, which are more centralized and offer superior data access, greater storage capacity, higher output, and a larger number of physical machines (hosts) and virtual machines(VMs) compared to other data centers. These high-tier data centers serve, at the top of the hierarchy. The second level consists of mid-tier data centers, which contain fewer components than the high-tier data centers but still offer substantial capabilities and resources. At the third and last level are the low-tier data centers, which have fewer components than mid-tier centers and provide more limited resources and functionalities. By structuring data centers into hierarchical levels based on their capacities and resources, the proposed scheme facilitates efficient resource allocation and management across the system. Each data center(DC) consists of physical machines(hosts), denoted as Ps=$\{p_1, p_2, p_3, ..., p_y\}$, where y represents the number of hosts in the system. Each

Table 1. Parameters and Variables of the model.

$DCs = (dc_i)$	dc_i represents the ith data center
Ps=(p_i)	p_i represents the ith physical machine
VMs=(vm_i)	vm_i represents the ith virtual machine
B=(b_i)	b_i represents the ith block
T=(t_i)	t_i represents the ith task
R=(r_i)	r_i represents the ith data replica
Ds=(d_i)	d_i represents the ith data file
nB_k	Number of Blocks per data file d_k
nBR_k	Number of Replica per data file d_k
$p(BA)_j$	Block Availability Probability
$p(FA)_k$	File Availability Probability
$p\left(\overline{FA_k}\right)$	File unavailability Probability
nA_k	Number of data file accesses
BS_i	Size of Block

host contains virtual machines(VMs), represented as VMs=$\{vm_1, vm_2, vm_3, \ldots, vm_z\}$, where z is the number of virtual machines in the system. Each virtual machine is composed of blocks, denoted as B=$\{b_1, b_2, b_3, \ldots, b_m\}$, where m is the number of blocks, which serve as the fundamental storage units for data files. Each DC also has its own storage capacity, represented as C=$\{c_1, c_2, c_3, \ldots, c_x\}$, where x is the number of data centers in the system.

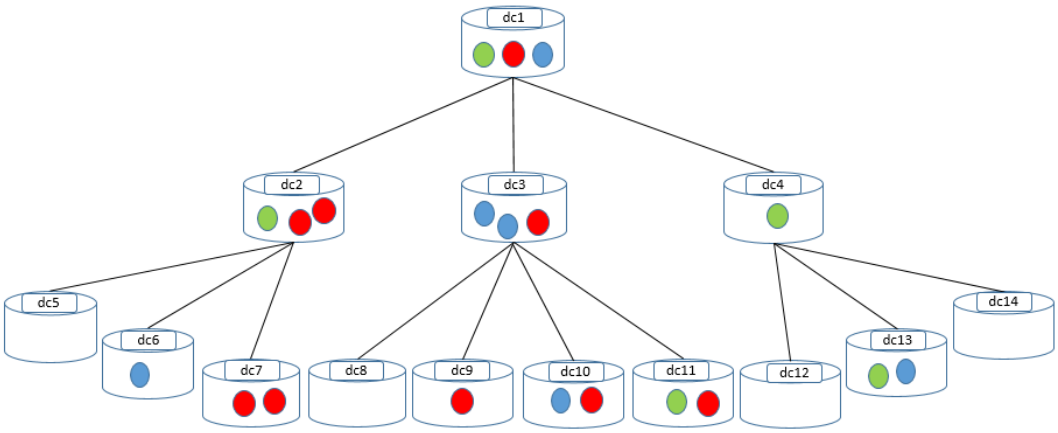

Figure 1. The Proposed Architecture

3.2. Tasks

Which is represented as $T = \{t_1, t_2, t_3, \ldots, t_n\}$, where n is the total number of tasks in the system. Our approach assumes a fixed and predetermined number of tasks to be submitted. Each task is associated with a specific set of required data files necessary for its execution. By considering the datasets needed for each task in advance, we can streamline the assignment process, efficiently allocating the appropriate data files to their corresponding tasks. The

fixed number of tasks allows for effective planning and optimization of the data file assignment strategy, ensuring that the necessary data file is available for the successful execution of each task.

3.3. Data Replication

In cloud computing, data replication is essential for ensuring high availability and efficient resource utilization across data centers (DCs), which have hosts and virtual machines(VMs) with varying storage capacities, distances, and latencies. Achieving optimal data replication requires balancing time, cost, and storage to provide users with the most accessible and cost-effective paths. Our strategy focuses on selecting and placing replicas, which is represented as R=$\{r_1, r_2, r_3, \ldots, r_s\}$, where s is the number of replicas in the system within DCs to optimize data placement and enhance accessibility. By dynamically generating data file copies as needed, we ensure that tasks in any given DC can access the most suitable source, thereby improving task performance and overall system efficiency. This approach involves continuously monitoring dataset movements, selecting the appropriate DC based on task requirements, and replicating data on demand. To support this, a replica catalog is maintained, recording the location of each replica and storing probability values $p(BA)$ for each DC. This ensures balanced resource utilization and optimal distribution of replicas across the network, maximizing data availability while minimizing costs.

3.4. Data File Availability

Data files are represented as Ds=$\{d_1, d_2, d_3, \ldots, d_l\}$, where l is the number of data files in the system. Data file availability refers to the system's ability to consistently provide accurate services, ensuring that users can access complete replicas of the data upon request. This is a critical concern for cloud consumers, as servers may become inaccessible due to data loss from failed replication or network malfunctions within the cloud infrastructure. To improve data availability, replicas can be stored across multiple (DCs). Additionally, within each DC, multiple replicas can be allocated to ensure redundancy across various DC blocks. High-tier DCs offer superior storage access and stability, resulting in higher availability for the data blocks they contain. However, these high-tier DCs are associated with increased costs. In contrast, low-tier DCs may offer reduced costs but exhibit lower reliability and availability. The determination of which data files to use can be calculated using Equations 1 to 4 as follows:

$$p(\text{BA}_j)\text{High-tier DC} > p(\text{BA}_j)\text{Mid-tier DC} > p(\text{BA}_j)\text{Low-tier DC} \tag{1}$$

The data file availability probability $p(DAk)$ can be calculated using these equations :

$$p(\text{FA}_k) = \prod_{i=1}^{nB_k} \left(1 - \prod_{i=1}^{nBR_k} (1 - p(\text{BA}_j)_i)\right) \tag{2}$$

$$p\left(\overline{\text{FA}_k}\right) = 1 - \left(\prod_{i=1}^{nB_k} \left(1 - \prod_{i=1}^{nBR_k} (1 - p(\text{BA}_j)_i)\right)\right) \tag{3}$$

Let's denote the BA probability as follows:

$$\begin{aligned} \text{High-tier DC} &: 0.6 \leq p(\text{BA}_j) < 0.9, \\ \text{Mid-tier DC} &: 0.3 \leq p(\text{BA}_j) < 0.6, \\ \text{Low-tier DC} &: 0 \leq p(\text{BA}_j) < 0.3 \end{aligned} \tag{4}$$

4. Proposed Model For Data Replication

The proposed diagram integrates two algorithms, $FFO - S$ and $FFO - P$, which are designed for selecting and placing identical data replicas. The functional structure of the proposed system is characterized by three key attributes: access time, data availability, and cost. Geometric distribution and compressed files are employed to

facilitate the distribution of data replicas across the cloud environment. The computational complexity of FFO-S and FFO-P was analyzed to ensure their feasibility in large-scale environments. The time complexity is

$$O(n^2)$$

, where n is the number of replicas, while space complexity scales with the number of tasks and replicas. Parallel processing techniques and adaptive parameter tuning were explored to reduce computational overhead. These optimizations ensure that the algorithms remain efficient even in high-load scenarios.

4.1. THE PROPOSED FFO-S FOR DATA SELECTION

The Firefly Optimization Algorithm for Data Replica Selection $(FFO-S)$ addressed the challenge of efficiently selecting and managing data replicas in cloud computing environments. This algorithm is inspired by the natural behavior of fireflies, where they are attracted to brighter counterparts based on a defined objective function, which may include minimizing replication costs, reducing access time, and optimizing resource utilization across data centers. In $FFO-S$, each firefly represents a potential solution for replica selection. The light intensity of each firefly is indicative of the solution's quality and is determined by a combination of factors, including storage costs, transfer costs, and data access time. The algorithm iteratively moves fireflies toward superior solutions, driven by their attractiveness and a degree of randomness, using the following equations: Light Intensity (Objective Function) : The light intensity of each firefly represents its fitness function of replica selection and is calculated using Equation(5):

$$I_i = \frac{1}{\text{Cost}_i + \text{Access Time}_i + \text{Distance}_i} \tag{5}$$

Table 2. Parameters and Variables of $FFO-S$ Algorithm.

I_i	The light intensity of firefly i
$Cost_i$	Represents the cost of selecting a replica
$AccessTime_i$	The time to access the replica
$Distance_i$	The distance (or latency) from the requesting task to the replica
$X_i(t)$	The position (possible replica location) of firefly i at time t
β_0 = 1	The attractiveness at distance $distance$ =0
y = 1	The light absorption coefficient
r	The distance between fireflies
α = 0.5	A randomization parameter

This function ensures that replicas with lower costs, faster access times, and shorter distances (in terms of network latency) are given higher priority. Attraction between Fireflies: The attractiveness (β) of a firefly is a function of its distance from another firefly. The attractiveness decreases with distance and is defined by Equation(6):

$$\beta(r) = \beta_0 e^{-yr^2} \tag{6}$$

Movement Equation: calculated using Equation(7) was The fireflies in the replica selection process move towards better solutions based on their attractiveness and randomization:

$$X_i(t+1) = X_i(t) + \beta_0 e^{-\gamma r_{ij}^2}(X_j(t) - X_i(t)) + \alpha(\text{rand} - 0.5) \tag{7}$$

Algorithm 1 The Proposed Firefly Algorithm for Selecting Data Replicas in a Cloud Environment

Input:
N: Number of fireflies
$maxIterations$: Maximum number of iterations
Data availability probabilities
Costs and time-based Decay function (TBDF)
Light absorption coefficient (γ), attractiveness (β_0), and randomness (α)
Output: Best replica configuration (firefly with optimal costs)
Begin: Initialize Fireflies with random replica configurations.
for each iteration from 1 to $maxIterations$ **do**
  for each firefly i **do**
    for each firefly j **do**
      if firefly j has a better (lower) cost than firefly i **then**
        Calculate distance between fireflies i and j
        Update firefly i's position based on firefly j
        $\beta = \beta_0 \cdot e^{-\gamma \cdot distance^2}$
        Update transfer_Cost, storage_Cost, and access_Time using the Firefly update rule
      else
        Apply random movement to firefly i
      end if
    end for
  end for
  Calculate the TBDF (Time-Based Decay Function)
  Calculate the replica factor
  Select the Best Firefly (with the lowest cost)
end for
Return Best Firefly (optimal replica selection)
end

The Replication Factor (RF) of a data file D_k and the System Replica Factor (RF_{sys}) is given by Equation(8) and Equation(9):

$$RF_k = \frac{\sum_{t_i=t_s}^{t_c}(nA_k(t_i, t_{i+1}) \times TBDF(t_i, t_c))}{nBR_k \times \sum_{i=1}^{n_k} BS_i} \tag{8}$$

$$RF_{\text{sys}} = \frac{\sum_{k=1}^{s}\left(\sum_{t_i=t_s}^{t_c}(nA_k(t_i, t_{i+1}) \times TBDF(t_i, t_c))\right)}{\sum_{k=1}^{s}\left(nBR_k \times \sum_{i=1}^{n_k} BS_i\right)} \tag{9}$$

where $nA_k(t_i, t_{i+1})$ refers to the number of accesses to a data file D_k in the time interval (t_i, t_{i+1}).

This function ensures that replication is dynamic, adjusting the number of replicas based on current demands and task requirements. The Time-Based Decay Function(TBDF) factor which is calculated by Equation(10)and Equation(11) helps prioritize which replicas to create or update based on their recent access patterns, optimizing

resource utilization and improving data availability. The TBDF function is crucial for assessing the priority of accessing different replicas based on their temporal availability and task demand. It is defined as:

$$TBDF(tc - ts) = e^{(tc-ts)k}, \quad k \in \{1, 2, 3, \ldots, n\} \tag{10}$$

Here:

- tc is the current time,
- ts is the start time of replica access,
- k is the step value,
- e is the exponential function.

The function assigns greater weight to replicas that have been accessed recently, allowing the system to prioritize replicas that are more frequently used.

An alternative form of TBDF includes exponential decay, which gives less importance to older data:

$$TBDF(tc - ts) = e^{-\Delta tk} \tag{11}$$

where $\Delta t = tc - ts$.

4.2. THE PROPOSED FFO-P FOR PLACEMENT

This section discusses the application of the proposed Firefly Optimization Placement (FFO-P)Algorithm for the large-scale implementation of data replica placement in CloudSim. The management of replicas is critical for determining optimal placements based on factors such as replication costs, data migration time, and the available space within DCs. Following the placement process, the data replicas become accessible to users. The FFO-P algorithm is utilized to optimally position replicas in DCs to fulfill user requests. In this approach, the objective function evaluates the light intensity of the fireflies, which serves as an indicator of the quality of the current placement configuration. The movement of fireflies toward those with greater light intensity corresponds to the search for improved solutions, focusing on minimizing costs and optimizing the utilization of DC resources. The cumulative effects of migration time, network bandwidth, and fetching resources are assessed through the following equations:

$$\tau(k)_{ij} = BS_k \times \left(\frac{1}{r_i} + \frac{1}{w_j} + \frac{1}{b_{ij}} \right) \tag{12}$$

The Equation(12) calculated the Total Transfer cost $(\tau(k)_{ij})$ of d_k from data center DC_i to DC_j, where BS_k is the block size of the data file, r_i is the fetching resource in DC_i, w_i represents the communication cost to DC_j, and b_ij reflects the bandwidth(Distance) between DC_i and DC_j. Since the data file d_k could be used by many tasks, The total migration time from dc_i, taking into account all tasks during execution is given by Equation(13) and Equation(14):

$$\tau_{ki} = \sum_{n=1}^{|T|} (f_{kn} \times \tau(k)_{i\alpha n}) = \sum_{n=1}^{|T|} f_{kn} \left(\frac{1}{r_i} + \frac{1}{w_{\alpha n}} + \frac{1}{b_{i\alpha n}} \right) \times BS_k \tag{13}$$

Where:

- αn is the index of the data centre where task t_n is assigned to.
- f_{kn} is defined as:

$$f_{kn} = \begin{cases} 1 & \text{if data file } d_k \text{ is used by task } t_n \\ 0 & \text{otherwise} \end{cases} \tag{14}$$

Thus, the total migration time considers the cumulative use of the dataset by multiple tasks. The firefly's movement toward better positions reflects minimizing the total replication and migration cost by selecting more

Algorithm 2 The Proposed Firefly Algorithm for Placement Data Replicas in a Cloud Environment

Input:
Population size (No. of fireflies)
No. of iterations
Minimum distance between DCs
Data replication costs and size
Output: Optimal Data Replica Placement
Begin:
Initialize parameters: firefly population, iterations, distances, replication costs, and probabilities.
repeat
  for each firefly i **do**
    Set initial firefly distribution in DCs.
    for each DC **do**
      Calculate attractiveness and movement probability.
      if rand() ¡ attractiveness **then**
        for each firefly j **do**
          if firefly j is brighter (lower cost) **then**
            Calculate distance between i and j.
            Update position:
            new_position = current_position + attractiveness $\times$ (brighter_position - current_position) + randomness $\times$ (rand() - 0.5)
            Update replica placement.
          else
            Calculate fitness function (distance, costs).
            Update local and global positions.
            Adjust attractiveness.
          end if
        end for
      end if
    end for
    if DC storage is small **then**
      Apply global update.
    else
      Delete small replicas.
    end if
  end for
until maximum iterations or an optimal solution found.
Return: Optimal Data Replica Placement
End.

optimal DCs. This iterative process continues until the algorithm converges to the best solution, ensuring efficient replica placement with minimal communication cost to DCs Can be calculated using the Equation(15)

$$W_i(\text{DCs}) = \sum_{x=1}^{y} (\text{cost}(\text{DC}_y) \times nBr_k(\text{DC}_y)) \tag{15}$$

, And the Bandwidth between DCs Can be calculated using Equation(16)and Equation(17):

$$B_{ij} = \min \sum_{i=1}^{x} \sum_{k=1}^{x} d_{ij}\, y_{ij} \tag{16}$$

$$\text{s.t.} \sum_{i=1}^{x} x_i\, y_{i \geq k}, \quad y_i \in \{0,1\}, \quad (1 \leq i \leq x) \tag{17}$$

4.3. Consistency and Fault Tolerance

To enhance the reliability and robustness of the proposed algorithms, future efforts will focus on addressing two critical aspects: consistency across replicas and fault tolerance in cloud environments.

- Consistency Across Replicas Maintaining data consistency across multiple replicas is crucial for ensuring the integrity and usability of replicated data. Future work will explore the integration of advanced consistency protocols, such as eventual consistency models, into the FFO-P algorithm. These protocols will ensure synchronization during data updates, allowing the system to maintain consistency without introducing significant delays or computational overhead. Additional experiments will evaluate the trade-offs between strict consistency and performance, identifying the optimal balance for dynamic cloud environments.
- Fault Tolerance Mechanisms Cloud systems are prone to failures, such as node outages and network partitions, which can disrupt data replication processes. To address these challenges, the proposed algorithms include mechanisms for replica redistribution and load balancing. These mechanisms ensure that the system can recover quickly from failures while maintaining data availability and minimizing downtime. Future experiments will simulate various failure scenarios to further assess the algorithms' ability to handle disruptions effectively, demonstrating their robustness under real-world conditions.

By integrating consistency protocols and enhancing fault tolerance mechanisms, the proposed algorithms aim to provide a resilient and efficient solution for data replication in dynamic and large-scale cloud environments.

5. Simulation Configurations

This section discussed the experimental results, focusing on the design of the replica selection and placement methods within the proposed cloud system. The system utilized the FFO-S algorithm for the optimal selection of data replicas and the FFO-P algorithm for the efficient placement of these replicas in data centers. Both algorithms were executed using CloudSim. The performance of these methods was assessed by comparing execution time, data access speed, cost efficiency, placement effectiveness, and replication availability against other existing algorithms. The results demonstrated the enhanced efficiency of the proposed FFO-S and FFO-P algorithms in managing replication processes within cloud environments.

5.1. Experiment Configuration Details

As illustrated in Figure 1, the cloud environment was designed to represent various types of (DCs) with different configurations, as outlined in Table 3. Each DC consisted of physical machines (hosts) and multiple virtual machines (VMs), which stored blocks of data replicas. For high-tier DCs, There is three distinct data placements were implemented. A total of 1,200 tasks were randomly selected for the data replication process. The proposed FFO-S (Firefly Optimization for Selection) and FFO-P (Firefly Optimization for Placement) algorithms were evaluated against several well-established methods.

Table 3. Simulation parameters of the configuration system

Entity Proposed System	**High-tier DataCenter**	**Mid-tier DataCenter**	**Low-tier DataCenter**
No. of DCs (31)	1	5	25
DCs Costs	600	400	200
No. of hosts per DC (180)	40	50	80
Processing element per host	12-16	4-8	1-4
Processing element MIPS	1000-2000	500-1000	100-500
Processing element bandwidth	5-10 GB	2-4 GB	1-2 GB
No. of VMs (640)	300	200	140
VM MIPS	800	400	200
VM Memory	2 GB	1 GB	512 MB
VM Bandwidth	10 GB	2 GB	1 GB
No. of processing elements	8-16	4-8	1-4
Task scheduler	Time and Space shared		
VM scheduler	–		
No. of tasks	1200		
Task length	1200-20000		
No. of Datasets (3)	A	B	C
Replication costs	600	400	200
No. of users	10-100		

5.2. Incorporating Real-World Constraints

To reflect real-world scenarios, additional factors such as network congestion, node failures, and dynamic task submissions were integrated into the simulation environment.

- Network Congestion: Bandwidth limitations and communication delays were introduced to simulate constrained network conditions.
- Node Failures: Random node outages were implemented to assess the algorithms' ability to maintain data replication integrity.
- Dynamic Task Submissions: Variable task arrival rates were simulated to replicate unpredictable workload patterns.

These enhancements allow a more accurate evaluation of the algorithm's robustness in addressing real-world challenges, demonstrating their ability to adapt to dynamic conditions while maintaining optimal replication performance.

6. Results and Discussion

The proposed system was simulated and evaluated using CloudSim, a Java-based simulation framework. CloudSim provides a variety of classes designed to model and simulate cloud environments, and it allows for the easy development of new classes or customization through inheritance from existing ones. For the simulation of the proposed system, custom classes were created to meet the specific requirements of the design.
To provide a comprehensive evaluation, we compared FFO-S and FFO-P with state-of-the-art methods, including reinforcement learning-based algorithms and hybrid optimization approaches. The results showed that the proposed algorithms achieved superior performance in terms of cost reduction, response time, and replication efficiency. The integration of firefly-inspired optimization techniques enables faster convergence and adaptability compared to the baseline methods.

6.1. Experiment for Optimal Replica Selection

Figure 2 illustrates the impact of utilizing the FFO-S, multi-objective particle swarm optimization(MO-PSO), Enhance Fast Spread(EFS), and Dynamic Cost-aware Re-replication and Re-balancing Strategy(DCR2S) algorithms on replication costs as the number of tasks assigned to users increases. Among these algorithms, the FFO-S algorithm demonstrates superior performance in reducing replication costs, surpassing the efficiencies of both MO-PSO and DCR2S. While DCR2S maintains replication costs close to the budget limit, EFS incurs significantly higher costs. The evaluation considered three primary constraints: maintaining constant costs, maximizing replica availability, and minimizing user-waiting time. The proposed FFO-S algorithm emerged as the optimal approach, achieving the lowest replication costs and increased availability. Furthermore, tasks were leveraged to enhance the effectiveness of the replica selection process, ensuring that FFO-S stands out as the most efficient solution compared to the other algorithms. The simulation results presented in Figure 3 illustrate the relationship between work time and the number of jobs utilizing the proposed FFO-S algorithm for optimizing data replica selection. The simulation demonstrates that FFO-S delivers the most efficient performance, achieving lower time costs and faster access to replicas. Compared to the other techniques, FFO-S consistently yields superior results in reducing overall time costs, establishing it as the optimal algorithm for data replica selection in cloud environments.

The simulation results illustrate the capability of the FFO-S algorithm to optimally select replicas, facilitating efficient access to the most relevant replicas. As depicted in Figure 4, FFO-S outperforms the MO-PSO and other algorithms regarding the execution time required to retrieve optimized replicas. The results clearly indicate that FFO-S is more efficient, providing faster access to optimal replicas and reducing overall selection time, thereby establishing it as the superior choice compared to other approaches.

6.2. Scalability Analysis

We conducted additional scalability tests to evaluate the performance of FFO-S and FFO-P in handling larger cloud environments. These experiments involved increasing the number of data centers to 100, with thousands of data files and tasks randomly distributed across the system. Results indicate that the proposed algorithms maintain efficiency and robustness, with replication costs and response times scaling linearly as the workload increases. These findings validate the adaptability of FFO-S and FFO-P in large-scale cloud environments.

6.3. Energy Consumption Analysis

Energy efficiency is a critical factor in cloud environments. We analyzed the energy consumption of FFO-S and FFO-P by evaluating their impact on data transfer and storage processes. Future iterations of the algorithms will

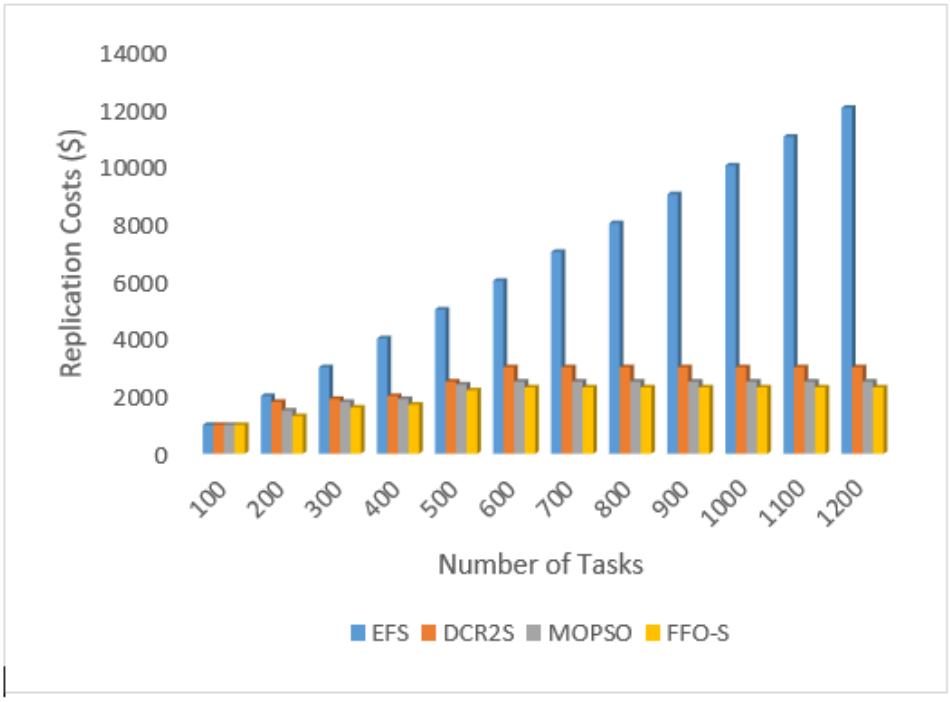

Figure 2. Replication costs with different number of tasks.

incorporate energy-aware optimization techniques to minimize power usage, particularly during replica placement. Strategies such as energy-efficient data centers and green cloud initiatives will be considered.

7. Performance and Evaluation

7.1. STORAGE CAPASITY USAGE

In Figure 5, the proposed scheme illustrates varying storage consumption rates, which reflect the relative size of data when transmitting and storing replicas across different datacenter (DC) sizes. The capacity consumption rate also serves as an indicator of both the costs and the time required for data transfers between DCs. This relationship is further elaborated in Equation(18)

$$SC = \frac{\text{File-SpaceAvailable}}{\text{space}} \tag{18}$$

7.2. REPLICATION FREQUENCY

In Figure 6, the replication frequency is tested by increasing the number of replicas to assess user access to data. As the number of replicas increases, a higher replication level is achieved, leading to reduced load congestion among datacenters(DCs), the network, and other resources. However, with the integration of the FFO-S algorithm, our strategy demonstrates superior performance by prioritizing access to the most frequently used data, with high replica placement percentages distributed across the DCs. Experimental results show that the proposed FFO-S approach outperforms the commonly used Prefetching-aware Data Replication(PDR) and Multi-Objective Ant-Colony Optimization(MOACO) algorithms, improving availability and reducing congestion in the DC network.

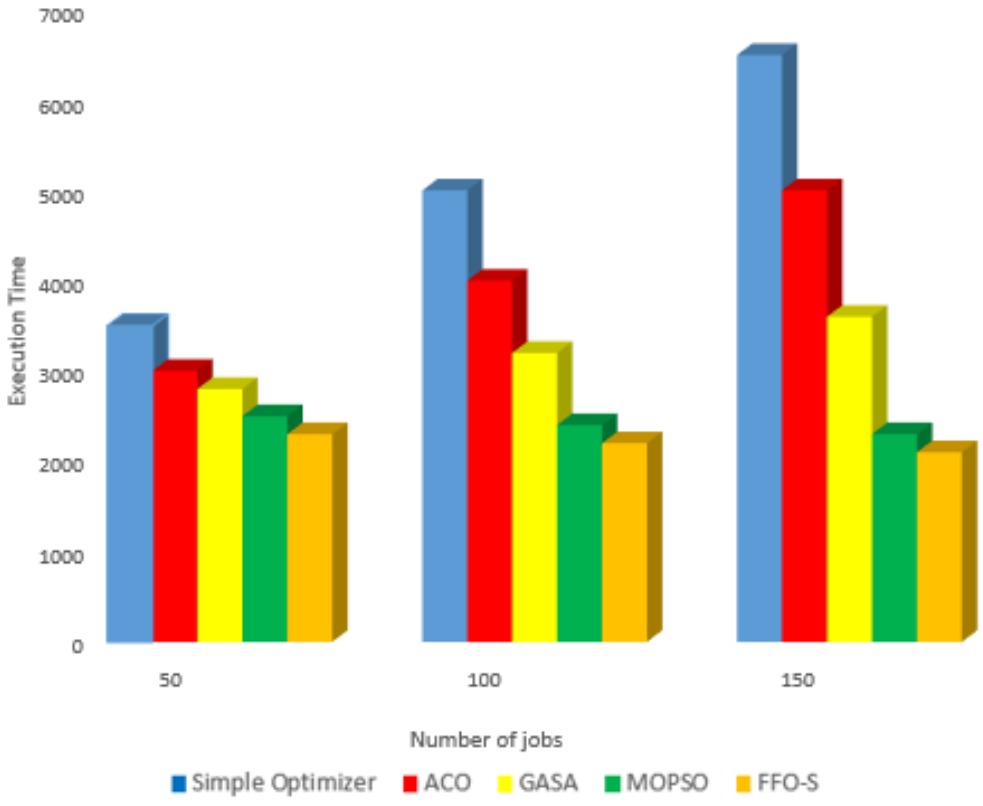

Figure 3. Simulation of Work time.

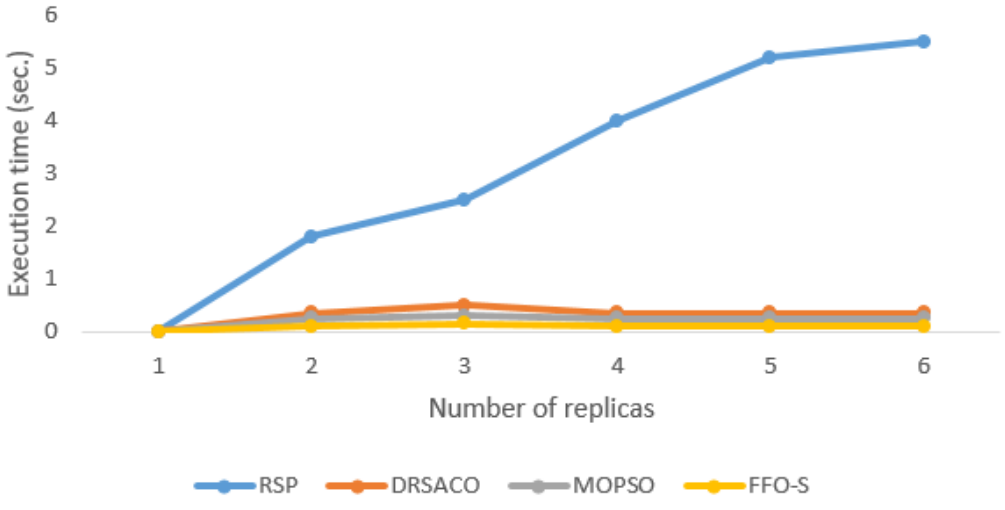

Figure 4. Execution time for selecting optimal replica.

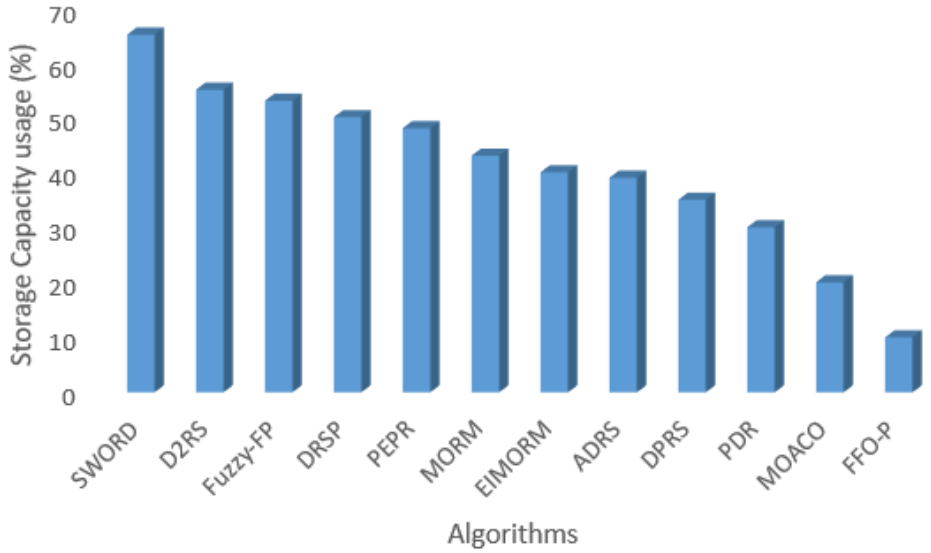

Figure 5. Storage capacity usage rates for different data replication algorithms.

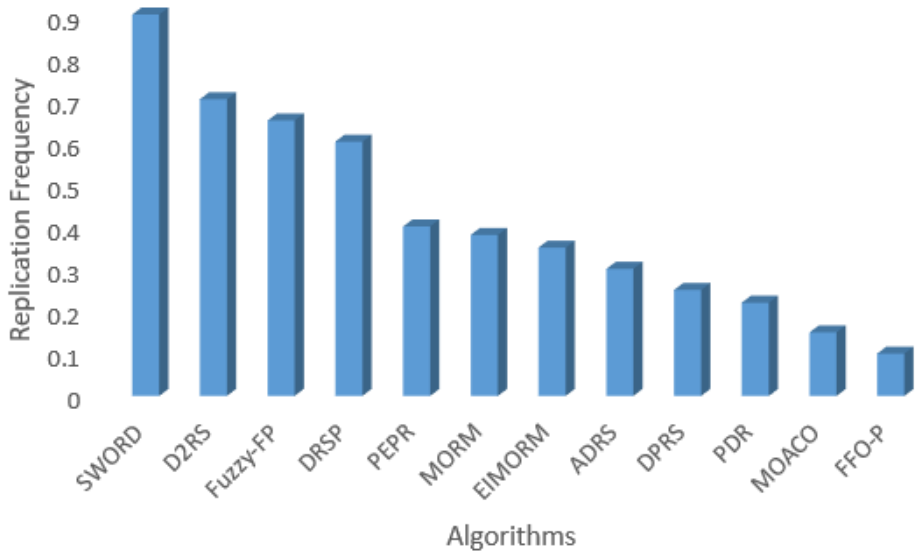

Figure 6. Replication frequencies for different data replication algorithms.

7.3. HIT RATIO

In Figure 7, the hit ratio is defined as the proportion of data available at varying distances from tasks, ranging from 1,000 to 3,000 units. Our method shows a higher hit ratio than PDR and MOACO. It efficiently adapts to user behavior for both nearby and distant replica access. This indicates a precise alignment with user actions concerning data access patterns. The hit ratio is calculated using Equation(19)

$$HR = \frac{\text{Number of local file accesses}}{\text{Number of replicas} + \text{Number of remote file accesses}} \tag{19}$$

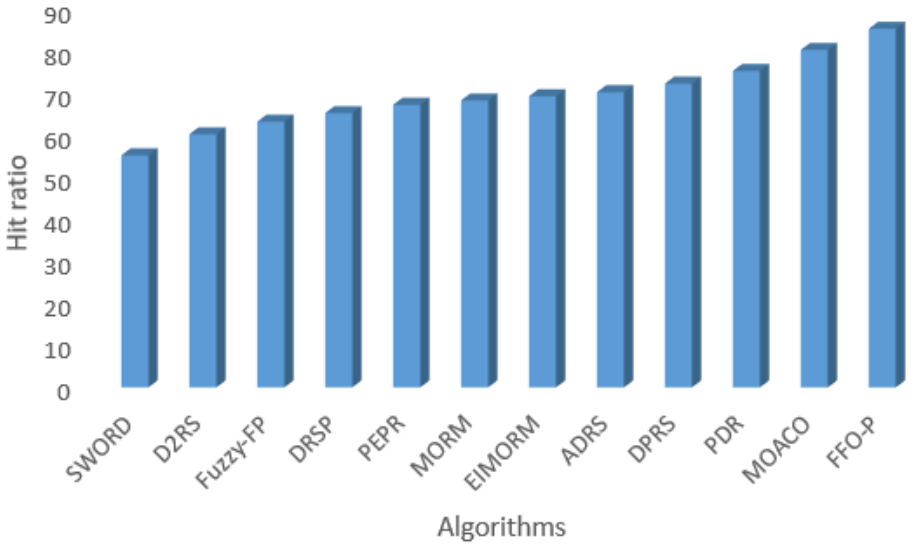

Figure 7. Hit Ratios for different data replication algorithms.

8. Conclusion

In cloud computing environments, achieving high availability, performance, and fault tolerance while minimizing replication costs is essential. This paper introduces two bio-inspired algorithms: FFO-S for selecting data replicas and FFO-P for the placement of data replicas. These algorithms aim to optimize the efficiency of cloud systems by addressing replication costs, availability, and user response times. The FFO-S algorithm dynamically selects the most frequently accessed data replicas, ensuring that only the most popular data is replicated to meet the growing demand in cloud systems. By incorporating selection criteria based on access intensity and cost, FFO-S guarantees that replicas are created at the optimal time to maximize availability while minimizing unnecessary resource consumption. Meanwhile, the FFO-P algorithm optimally places the selected data replicas in appropriate data centers, focusing on reducing access time, improving storage utilization, and maintaining replication costs within budget constraints. This approach not only increases system availability but also ensures efficient resource usage by strategically distributing replicas across data centers. If replication costs exceed the available budget, FFO-P employs a cost-optimization technique that repositions replicas in lower-cost data centers, thereby ensuring availability is maintained while keeping costs under control. The proposed algorithms, FFO-S and FFO-P, have been simulated and evaluated using CloudSim, demonstrating superior performance compared to existing techniques such as ADRS, D2RS, DRACO, EFS, MOPSO, and MOACO. The results indicate that FFO-S and FFO-P outperform these algorithms in terms of replication costs, availability, and response time. Our algorithms also achieved higher hit ratios, ensuring that data is efficiently accessed by users, even as the cloud ecosystem scales. Understanding user access patterns is crucial for optimizing data replication strategies. Future research will focus on analyzing factors such as access frequency and temporal trends to enhance the selection and placement of replicas. By dynamically adapting to changes in user demand, the proposed algorithms can improve data availability and system efficiency. Additionally, future work will extend the system to real cloud environments for validation and explore the integration of consistency protocols to ensure reliability. Block-level replication techniques will also be investigated to further optimize efficiency and address the evolving requirements of cloud-based systems.

REFERENCES

1. R. Salem, M. Abdul Salam, H. Abdelkader, and A. Awad Mohamed, "An artificial bee colony algorithm for data replication optimization in cloud environments," *IEEE Access*, vol. 8, pp. 51841–51852, 2020.
2. L. Bouhouch, C. Tadonki, and M. Zbakh, "Dynamic data replication and placement strategy in geographically distributed data centers," *Concurrency and Computation: Practice and Experience*, vol. 35, 02 2022.
3. F. Karamimirazizi, S. M. Jameii, and A. Rahmani, "Data replication methods in cloud, fog, and edge computing: A systematic literature review," *Wireless Personal Communications*, vol. 135, 04 2024.
4. A. Mohamed, L. Abualigah, A. Alburaikan, and H. Khalifa, "Aoeho: A new hybrid data replication method in fog computing for iot application," *Sensors*, vol. 23, p. 2189, 02 2023.
5. A. Mohamed, A. Diabat, and L. Abualigah, "Optimizing energy-efficient data replication for iot applications in fog computing," *International Journal of Communication Systems*, vol. 37, 07 2024.
6. A. Awad, R. Salem, H. Abd elkader, and M. Abdul Salam, "A swarm intelligence-based approach for dynamic data replication in a cloud environment," *International Journal of Intelligent Engineering and Systems*, vol. 14, pp. 271–284, 03 2021.
7. A. Javadpour, A. M. H. Abadi, S. Rezaei, M. Zomorodian, and F. Rostami, "Improving load balancing for data-duplication in big data cloud computing networks," Jun 2021.
8. A. Awad, D. Ashraf, A. Alburaikan, H. Khalifa, M. Elsayed Abd Elaziz, L. Abualigah, and A. M.AbdelMouty, "A novel hybrid arithmetic optimization algorithm and salp swarm algorithm for data placement in cloud computing," 11 2022.
9. J. Nayak, B. Naik, A. Jena, D. R. Barik, and H. Das, *Nature Inspired Optimizations in Cloud Computing: Applications and Challenges*, pp. 1–26. 02 2018.
10. W. K. Awad and E. T. Mahdi, "Tasks scheduling techniques in cloud computing," in *2022 3rd Information Technology To Enhance e-learning and Other Application (IT-ELA)*, pp. 94–98, 2022.
11. A. Awad, R. Salem, H. Abdelkader, and M. A. Salam, "A novel intelligent approach for dynamic data replication in cloud environment," *IEEE Access*, vol. 9, pp. 40240–40254, 2021.
12. R. Dugyani and D. Govardhan, "Data replication and scheduling in the cloud with optimization assisted work flow management," *Multimedia Tools and Applications*, vol. 83, pp. 1–23, 01 2024.
13. D. Govardhan and R. Dugyani, "Survey on data replication in cloud systems," *Web Intelligence*, vol. 22, pp. 1–27, 01 2024.
14. S. Simaiya, D. Kumar Lilhore, D. Y. Sharma, K. B. V. Brahma Rao, M. R. V V R, A. Baliyan, A. Bijalwan, and R. Alroobaea, "A hybrid cloud load balancing and host utilization prediction method using deep learning and optimization techniques," *Scientific Reports*, vol. 14, 01 2024.
15. A. Awad, R. Salem, H. Abdelkader, and M. A. Salam, "A novel intelligent approach for dynamic data replication in cloud environment," *IEEE Access*, vol. 9, pp. 40240–40254, 2021.
16. L. Bouhouch, C. Tadonki, and M. Zbakh, "Dynamic data replication and placement strategy in geographically distributed data centers," *Concurrency and Computation: Practice and Experience*, vol. 35, 02 2022.
17. X. Sun, G. Wang, L. Xu, and H. Yuan, "Data replication techniques in the internet of things: a systematic literature review," *Library Hi Tech*, vol. ahead-of-print, 06 2021.
18. M. Ashawa, O. Douglas, J. Osamor, and R. Jackie, "Improving cloud efficiency through optimized resource allocation technique for load balancing using lstm machine learning algorithm," *Journal of Cloud Computing*, vol. 11, 12 2022.
19. N. Mansouri, M. Javidi, and B. Mohammad Hasani Zade, "Hierarchical data replication strategy to improve performance in cloud computing," *Frontiers of Computer Science*, vol. 15, 04 2021.
20. D. Rambabu and A. Govardhan, "Task scheduling and data replication in cloud with improved correlation strategy," *International Journal of Computers and Applications*, vol. 45, no. 11, pp. 697–708, 2023.
21. H. Yahia, S. Zeebaree, N. O. M.Salim, S. Kak, A. Al-zebari, A. Salih, and H. Hussein, "Comprehensive survey for cloud computing based nature-inspired algorithms optimization scheduling," *Asian Journal of Computer Science and Information Technology*, vol. 8, pp. 1–16, 05 2021.
22. H. Yu, "Evaluation of cloud computing resource scheduling based on improved optimization algorithm," *Complex & Intelligent Systems*, vol. 7, 09 2020.
23. P. Pirozmand, H. Jalalinejad, A. A. Rahmani Hosseinabadi, S. Mirkamali, and Y. Li, "An improved particle swarm optimization algorithm for task scheduling in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, pp. 1–15, 02 2023.
24. Y. Ebadi and N. Navimipour, "An energy-aware method for data replication in the cloud environments using a tabu search and particle swarm optimization algorithm," *Concurrency and Computation: Practice and Experience*, vol. 31, p. e4757, 08 2018.
25. N. Mansouri, M. Kuchaki Rafsanjani, and M. Javidi, "Dprs: A dynamic popularity aware replication strategy with parallel download scheme in cloud environments," *Simulation Modelling Practice and Theory*, vol. 77, pp. 177–196, 09 2017.
26. M. Zheng, X. Du, Z. Lu, and Q. Duan, "A balanced and reliable data replica placement scheme based on reinforcement learning in edge–cloud environments," *Future Generation Computer Systems*, vol. 155, pp. 132–145, 2024.
27. T.-Y. Hsu, A. Kshemkalyani, and M. Shen, "Causal consistency algorithms for partially replicated and fully replicated systems," *Future Generation Computer Systems*, vol. 86, pp. 1118–1133, 2018.
28. F. Prity, K. Uddin, and N. Nath, "Exploring swarm intelligence optimization techniques for task scheduling in cloud computing: algorithms, performance analysis, and future prospects," *Iran Journal of Computer Science*, vol. 7, pp. 1–22, 11 2023.
29. N. Mansouri, B. Mohammad Hasani Zade, and M. Javidi, "A multi-objective optimized replication using fuzzy based self-defense algorithm for cloud computing," *Journal of Network and Computer Applications*, vol. 171, p. 102811, 08 2020.