International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

Enhancing Efficiency in Cloud Computing Entails Optimizing Resource Apportionment Through the Utilization of the Shuffled Frog-Leaping Algorithm (SFLA) and Firefly Algorithm

Namrata H. Patadiya*[1], Dr. Nirav V. Bhatt[2]

Submitted: 03/02/2024 **Revised**: 11/03/2024 **Accepted**: 17/03/2024

Abstract: The imperative role of 'cloud computing' in modern technology brings attention to Resource apportionment as a pivotal facet. This paper introduces a Hybridized Optimization algorithm that combines the effectiveness of the 'Shuffled Frog Leaping Algorithm' (SFLA) and the 'Firefly Algorithm.' This innovative approach overcomes limitations seen in current works like the HABCCS algorithm, GTS algorithm task, and the krill herd algorithm, while amalgamating the unique features of both SFLA and the Firefly Algorithm. Within this methodology, the SFLA section oversees initial steps, encompassing the initialization of request size, request generation, estimation of SFLA's fitness value, sorting, division, and evaluation of user requests. SFLA is recognized for its rapid convergence and straightforward implementation, boasting the capability for global optimization and widespread utilization across diverse domains. Concurrently, the Firefly Algorithm takes on pivotal operations such as initialization, request generation, fitness function evaluation, modification, and the assessment of new solutions. The Firefly Algorithm is characterized by its ease of evaluation and suitability for complex situations, providing a notable advantage. In this system, the evaluation of request speed and sizes plays a critical role in Resource apportionment on the server side, contributing to reduced computation times. Experimental results substantiate the efficacy of this hybrid approach, illustrating its superior performance in comparison to additional similar technique.

Keywords: *Cloud Computing, Shuffled frog leaping Algorithm, Firefly Algorithm, Resource apportionment*

1. Introduction

Cloud Computing (CC) is a revolutionary computing approach centered on the Internet, capitalizing on the data repository capabilities of cloud servers and drawing in multitudes of users. [1]. This model thrives on the shared utilization of hardware resources, offering a compelling computing paradigm that caters to the dynamic allotment of resources as needed [2] . Information delivery is tailored to the specific requirements of computers and other devices, reflecting a notable surge in the usage, services, and delivery models associated with the Internet. The three primary service models include 'Infrastructure as a Service' (IaaS), 'Platform as a Service' (PaaS), and 'Software as a Service' (SaaS). In the realm of Cloud Computing, auctions emerge as a strategic method for selling cloud assets on a user network through intense competition. Typically, cloud users actively participate in these auctions by placing bids for the specific resources they require [3].

In the dynamic landscape of cloud computing, two pivotal roles are played by cloud users and cloud providers. The providers, armed with an array of computing resources housed in expansive data centers, embrace a 'pay-per-use' model, leasing these resources to users.

This not only serves as a revenue catalyst but also optimizes the utilization of resources. Simultaneously, cloud users, navigating applications with varying workloads, strategically lease resources from providers, activating their applications with minimal expenditure. Users frequently seek multiple resources for specific tasks or cloudlets, aiming to enhance performance and ensure timely completion [4].

The adoption of this computing paradigm has proven highly advantageous for organizations, providing comprehensive solutions that translate into tangible business benefits, including heightened flexibility, adaptability, cost reduction, and increased operational efficiencies. Its seamless integration within organizational frameworks has consistently contributed to annual revenue growth. Organizations effectively employ this computing model, capitalizing on its comprehensive solutions that deliver advantages such as increased flexibility, scalability, agility, cost savings, and heightened operational efficiencies [5]. Its integration within organizational frameworks not only facilitates operational improvements but also plays a pivotal role in annual revenue augmentation.

The management of resources encompasses diverse stages, ranging from submitting the task list for final execution in reference to cloud, this management process involves two pivotal stages: (1) resource scheduling and (2) resource provisioning [6]. Resource apportionment (RA) emerges as a noteworthy element in both grid and distributed

[1]*Ph.D Scholar, School of Engineering, RK University, Rajkot, India*
[2] *Professor, School of Engineering, RK University, Rajkot, India*
* *Corresponding Author Email: namrata1124@gmail.com*

computing paradigms Implementing a utility-centered technique in Resource apportionment (RA) is essential across various levels of grid computing structured around the concept of utility. The prevalence of utility-centric computing is on the rise, seamlessly integrating into not just user-facing scenarios but also seamlessly weaving through diverse business environments. A cloud incorporating utility features delivers services tailored to the diverse resource requirements of users. (Fig. 1). The allocation of services in the cloud depends on models centered around bidding, combinations, and various financial frameworks—all crafted to optimize the revenue for the owner of the cloud. These financial paradigms stimulate client requests and boost earnings through assets allocation. [7].

Numerous recent research efforts have suggested methods for Resource apportionment in pervasive computing that are customized to prioritize energy efficiency. While numerous studies on allotment of resources in Cloud Computing are available, they fail to specifically focus on the fundamental challenges associated with energy saving allocation of assets in cloud environments. Previous studies have not adequately addressed the crucial aspect of energy saving assets management when viewed from the standpoint of software engineering.

Cloud users benefit from cost-effective, high-quality services provided by their service providers. The efficiency and excellence of these services depend on the specific assets allocation process within the service realms. Providers optimize the allocation of resources to clients using a variety of Resource apportionment (RA) models in the cloud setting. These models employ distinct algorithms and methods, as elucidated in the survey of prior works on RA models within the cloud environment, with a particular emphasis on RA methodologies.

Consequently, there is a proposed evaluation of RA application in the realm of Cloud Computing (CC). This approach is well-suited for proficient RA based on user requests, leading to reduced processing time and energy consumption. The subsequent sections introduce an innovative technique designed to elevate system performance through a hybrid algorithm, addressing observed performance deterioration in existing metho

The paper's structure unfolds as follows: Section 2 scrutinizes pertinent works with respect to the method that has been proposed. Section 3 highlights a concise illustration of the proposed methodology, while Section 4 delves into the exploratory outcomes. Finally, Section 5 draws conclusions derived from the paper.

2. Literature Survey

Jitendra and Narander [8] proposed Spider Monkey Optimization (SMO) for optimized Resource apportionment in cloud computing, considering key parameters like application time, migration time, and resource utilization. It addresses energy consumption through the Green Cloud Scheduling Model (GCSM), prioritizing energy-efficient nodes for deadline-constrained tasks. Evaluation is done using a cloud simulator, with energy consumption as the primary metric. The proposed approach outperforms existing strategy in aspects of response time, makespan, energy depletion, and resource utility.

Seyed Hasan Hosseini, Javad Vahidi, Seyed Reza Kamel Tabbakh, Ali Asghar Shojaei [9] proposed Whale optimization strategy to mitigate the assets distribution in the cloud environment. Cloud computing's "pay-per-use" model is popular, but efficient Resource apportionment is challenging due to high demand. This document presented an innovative whale optimization-based algorithm for optimizing Resource apportionment in the cloud. The algorithm employs a discrete whale representation, defining a new distance function. Spiral and search-prey functions aid movement. Results indicate the algorithm effectively addresses Resource apportionment problems, offering efficient solutions in cloud environments. Proper adjustment of the transition from exploration to exploitation phases is crucial for algorithm success.

R.Vadivel, Sudalai Muthu T[10] proposed an innovative approach, merging Hybrid Particle Swarm Optimization (HPSO) and a modified Genetic Algorithm for dynamic Resource apportionment among Virtual Machines (VMs). Users initiate the process by collecting data online, including details on system resources. The algorithm optimizes Resource apportionment by sorting features, efforts directed towards reducing processing time and elevating overall performance. The focus is on user-driven efficiency in data collection and extraction steps.

Javad Vahidi, Maral Rahmati [11] paper addresses the Resource apportionment challenges in cloud computing caused by the popularity of Pay-Per-Use models. It introduces the Grasshopper Optimization Algorithm (GOA) as an innovative solution, demonstrating its superior performance through MATLAB simulations and comparisons with Genetic Algorithm (GA) and SEIRA. Results strongly support GOA's effectiveness in optimizing Resource apportionment, emphasizing its capability to explore problem spaces and provide viable solutions.

In their study, Sharma and Guddeti [12] introduced a Euclidean distance-centered Resource apportionment model for VMs and migration policies in data centers. Using the HGAPSO hybrid approach (Genetic Algorithm and Particle Swarm Optimization), they efficiently distributed VMs across Physical Machines (PMs). This, coupled with virtual machine migration, reduced energy utilization, minimized resource wastage, and prevented SLA violations in cloud data centers. Experimental comparisons in heterogeneous and homogeneous data centers demonstrated that HGAPSO

and VM migration outperformed a bound and branch-based exact algorithm in achieving optimal resource utilization, energy efficiency, and SLA compliance.

Kayalvili and Selvam [13] proposed a Cloud Computing (CC) Resource apportionment (RA) method using virtualization technology and Virtual Machines (VMs). VMs were strategically employed for optimal outcomes by adjusting placement and layout. Addressing the NP-Hard problem of distributing cloud resources based on user requests, heuristic methodologies for RA optimization were utilized. The hybrid SFLA-GA (Shuffled Frog Leaping Algorithm - Genetic Algorithm) approach demonstrated efficiency in achieving optimal resource distribution in the CC environment.

Mireslami et al. [14] proposed an economically advantageous solution that optimizes performance algorithm that reduced deployment costs while meeting Quality of Service (QoS) requirements. The algorithm optimally selected web application usage in the cloud from the customer's perspective, instantly decreasing costs and enhancing QoS performance. Validated through experiments on various workloads in two different cloud service providers, the results highlighted the algorithm's ability to determine an optimal combination of cloud resources, striking a balanced equilibrium between performance standards and implementation expenses in a relatively short time.

2.1. Hybridization of Shuffled Frog Leap Algorithm and Firefly

The hybrid algorithm blends SFLA's collaborative exploration and Firefly Algorithm's adaptability for efficient exploration and exploitation in cloud computing. By concurrently utilizing their parallel search capabilities, the algorithm swiftly explores multiple solutions, adjusting the population based on attractiveness values. This hybridization leverages SFLA's diversity preservation with the Firefly Algorithm's intensification, balancing diversity while focusing on promising solutions. Through iterative cycles, the algorithm dynamically adapts to changing cloud conditions, converging towards optimal or near-optimal solutions that adhere to knapsack constraints in Resource apportionment.

2.2. Hybrid Shuffled Frog-Leaping Algorithm and Firefly Algorithm (Hybrid SFLA-FA)

The human face is a varying object. With the passing of time, everything changes, including a person's appearance, which has an impact on the facial recognition system. To improve the chances of recognizing a person with aging effects by using a large dataset for face images, which included images of the same person; taken at various times throughout his/ her life.

Begin
1. Initialize the population size, number of memplex m, total frogs P = Mn
2. Generate required population (X_{i}) i = 1 to m by random generation.
3. Evaluate the fitness values for each X
4. Determine the best frog with the best robotness value.
5. **while** ≤ max generation do
   for i = 1, 2, ... , m do
   Perform local search using SFLA on the memplex X_{i}
   Evaluate the fitness value F_{i} = f(X_{i})
   Move fireflies towards brighter solutions using Firefly Algorithm
   Update positions based on attractiveness and randomization
   Evaluate the fitness value F_{j} for the attracted firefly X_{j}
   if (F_{j} > F_{i}) then
   Replace frog X_{i} with firefly X_{j}
   End if
   Perform global search using SFLA on the entire population
   Abandon a fraction P * of the worst frogs to create new solutions
   Keep the best frog with the highest quality solution.
   Rank the frogs and find the best one
   End while
End

1. Initialization:

- Initialize the population of frogs and fireflies with random solutions.
- Calculate the fitness values for each solution in both populations.

2. Shuffled Frog-Leaping Phase:

- Divide the frog population into subgroups and shuffle the frogs within each subgroup.
- Apply local search to each frog in its subgroup and update their solutions.
- Select the best frogs from each subgroup to form a new set of frogs.
- Determine the global mean and move the frogs towards the global mean.

3. Firefly Algorithm Phase:

- Calculate the attractiveness between fireflies based on their brightness and distance.
- Update the position of each firefly by moving towards brighter fireflies while introducing randomness.
- Update the light intensity of each firefly based on its new position and objective function value.

4. Hybridization Strategy:

- Combine the frog population and the firefly population to create a hybrid population.
- Apply a strategy to select solutions from the hybrid population, such as alternating between SFLA and FA steps, or choosing solutions based on their fitness values.

5. Exploration and Termination:

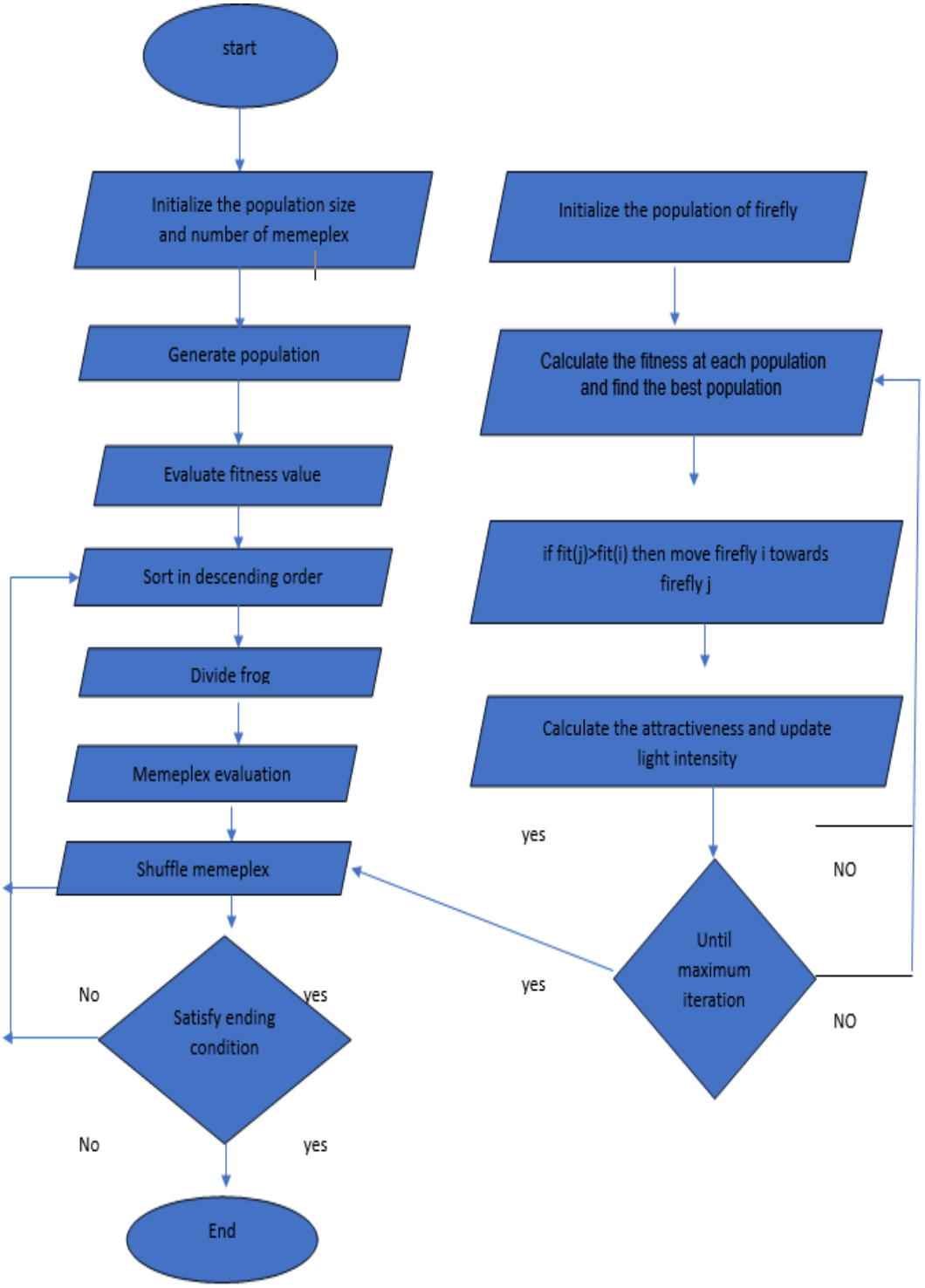
- Repeat the hybrid steps for a certain number of iterations or until a termination criterion is met.

6. Solution Extraction:

- After the hybrid iterations, select the best solution obtained from both algorithms as the final solution to the optimization problem.

2.3. Flowchart of Shuffled frog leap and firefly algorithm

Algorithmic representation for the hybrid optimization SFLA-FA algorithm



3. Findings and Analysis

The presented approach aims to alleviate challenges associated with knapsack problems in resource allocation. This not only reduces computation duration and competence constraints but is also executed in java programming language leveraging cloudSim. The data repository serves as a benchmark for addressing fundamental scheduling issues.

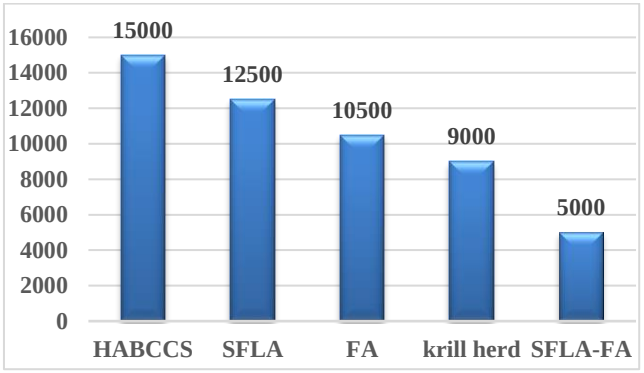


Fig.1. Comparative evaluations of suggested framework with current systems for resource elapsed time. face

3.1. Computational time:

In comparison to other current systems, the suggested system has a short execution time. The SFLA-FA outperforms the HABCCS algorithm, krill herd algorithm, Shuffled Frog Leap Algorithm, Firefly and SFLA-FA in terms of time required for execution. The execution time is calculated as

$$PT = E(t)- S(t)$$

where PT is the processing time, E(t) denotes the end time of the process, and S(t) is the start time of the process.

Figure 1 depicts a juxtaposition of the executing times of the present and new methods. In Fig. 1, the names of several strategies for optimization are written along the x- axis, and the execution time is written along the y-axis.

3.2. Conveyance rate:

Conveyance rate refers to the volume of data transmitted between locations within a defined time period. It is also used to test the efficiency of storage drives, memory, Internet connectivity. The suggested system ought to have a higher conveyance rate than the current one. The conveyance rate estimated as:

$$C_t = I_t /t$$

where It is the amount of data transmitted, and t is the duration. Figure 2 compares and displays the conveyance rate of several algorithms. In Fig. 2, the x-axis has the names of the different algorithms, while the y-axis contains the quantity.

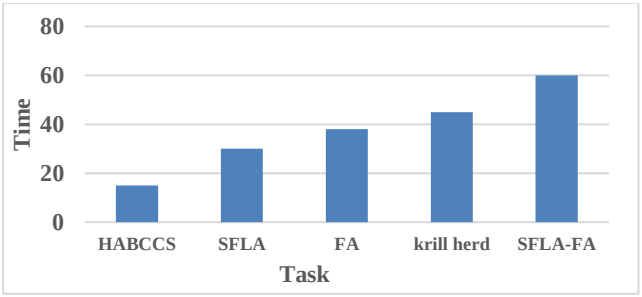


Fig. 2. Evaluation of the conveyance rate of resources in the existing system against the newly introduced system.

3.3. Job Turnaround:

Job Turnaround time in computing is the entire amount of time that passes between sending a procedure for execution and receiving the final outcome back from the client or user. It could differ for different programming languages based on who created the program or software. Turnaround time may be defined as the whole amount of time after a program is begun that it takes to give the user the desired result.

$$J(t) = E(t)- O(t)$$

J(t) = Mean completion time

E(t)= End time of the operation

O(t)= Onset time

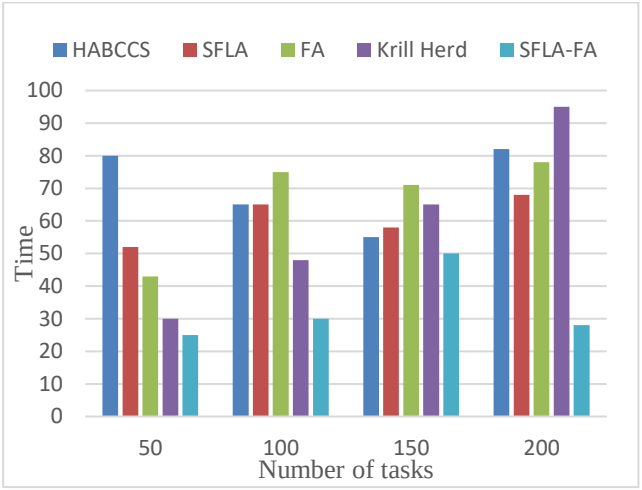


Fig. 3. Comparative evaluations of suggested framework with current systems for job turnaround time

3.4. Latency:

Latency is the amount of time that must pass between making a request and when the requested action or service is actually provided. It can also mean the entire amount of time a process waits in a ready queue in order to get to the CPU. The difference between a process's turnaround and cycle time is known as the waiting time. Another way to think of waiting time is the interval of time that passes between finishing one task and beginning another.

$$W(a)Avg = U(a)Avg – T(a)$$

M(a)Avg = depicts Average latency time

U(a)Avg = depicts Average turnaround time

T(a) = Task Duration

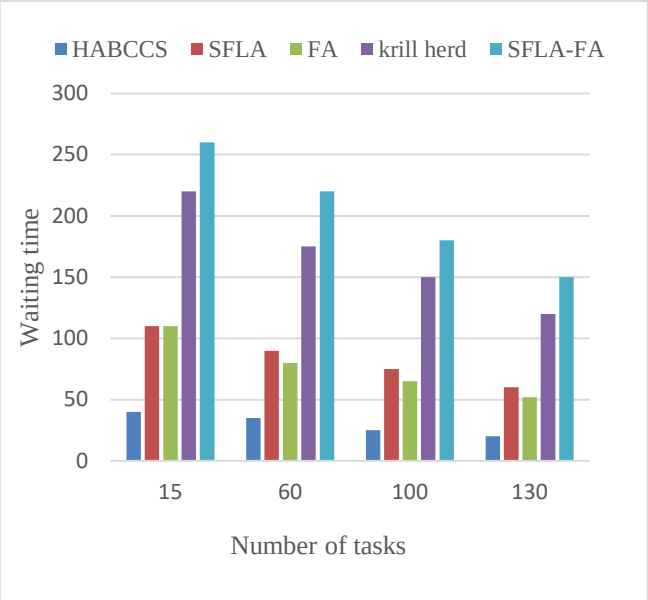


Fig. 4. Comparative evaluations of suggested framework with current systems for latency

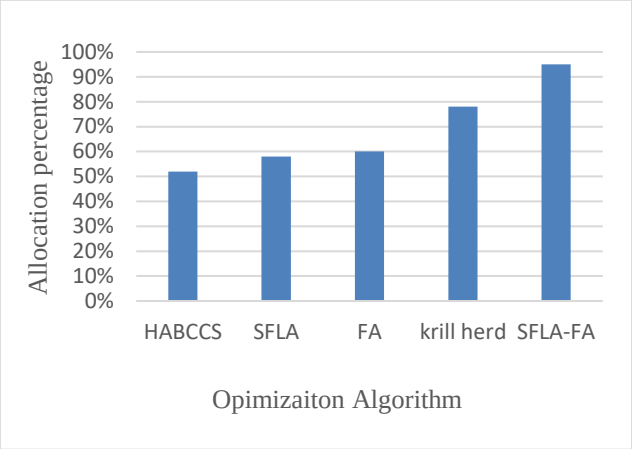


Fig. 5. Comparative evaluations of suggested framework with current systems for allocation mechanisms

Here is a comparison between the suggested system and the current system using this Mechanism. The hybrid SFLA-FA optimization method performs better overall when it comes to Resource apportionment in cloud environments. Fig. 5 shows a comparison of this method. In Figure 5, the assignment proportion is calculated along the vertical axis and the optimization procedure is taken along the horizontal axi

4. Conclusion and Future Work

In this scholarly work, the SFLA-FA strategy is developed to curtail allocation Knapsack Dilemma for the granting of assets in a Cloud Computing context. The optimization problem is tackled in the suggested system by utilizing hybrid SFLA and Firefly algorithm optimization techniques. The suggested RA is implemented using the JAVA working platform and Cloud Sim. The suggested work and the current framework are contrasted. The suggested work's time to execution was found to be 6000 ms, the conveyance rate was 60 s, the time needed for job fulfillment was short, the 'time frame' was short, and the apportionment proportion was 92. The previous systems, such as HABCSS, SFLA, FA, and krill herd, demonstrated extended run time, conveyance rate, time frame, and proportion of Resource apportionment, which may result in a significant reduction in total system efficiency, as seen by the experimental findings. The study has shown that the suggested works outperformed the existing approaches pertaining of waiting time and assignment procedures. This work might be improved by increasing the efficacy of the suggested work.

References

[1] Xiaoying, T., Dan, H., Yuchun, G., Changjia, C.: Dynamic Resource apportionment in cloud download service. J. China Univ. Posts Telecommun. 24(5), 53–59 (2017)

https://doi.org/10.1016/S1005-8885(17)60233-4

[2] Pradhan, P., Prafulla, B.K., Ray, B.N.B.: Modified round robin algorithm for Resource apportionment in cloud computing. Procedia Comput. Sci. 85, 878–890 (2016) https://doi.org/10.1016/j.procs.2016.05.278

[3] Madni, S.H.H., Latiff, M.S.A., Coulibaly, Y.: Recent advancements in Resource apportionment techniques for cloud computing environment: a systematic review. Clust. Comput. 20(3), 2489–2533 (2017)

https://doi.org/10.1007/s10586-016-0684-4

[4] Kumar, N., Saxena, S.: A preference-based Resource apportionment in cloud computing systems. Procedia Comput. Sci. 57, 104–111 (2015)

https://doi.org/10.1016/j.procs.2015.07.375

[5] Xue, C.T.S., Xin, F.T.W.: benefits and challenges of the adoption of cloud computing in business. Int. J. Cloud Comput. Serv. Arch. (IJCCSA) 6(6), 1–15 (2016)

https://doi.org/10.5121/ijccsa.2016.6601

[6] Singh, S., Chana, I.: A survey on resource scheduling in cloud computing: issues and challenges. J. Grid Comput. 14(2), 217–264 (2016)

https://doi.org/10.1007/s10723-015-9359-2

[7] Kolhar, M., Abd El-atty, S.M., Rahmath, M.: Storage allocation scheme for virtual instances of cloud computing. Neural Comput. Appl. 28(6), 1397–1404 (2017)

https://doi.org/10.1007/s00521-015-2173-8

[8] Jitendra Kumar Samriya and Narander Kumar: Spider Monkey Optimization based Energy-Efficient Resource apportionment in Cloud Environment. Trends in science 2022 19(1): 1710 https://doi.org/10.48048/tis.2022.1710

[9] Seyed Hasan Hosseini, Javad Vahidi, Seyed Reza Kamel Tabbakh, Ali Asghar Shojaei: Resource apportionment optimization in cloud computing using the whale optimization algorithm. Int. J. Nonlinear Anal. Appl. Volume 12, Special Issue, Winter and Spring 2021, 343-360 http://dx.doi.org/10.22075/ijnaa.2021.5188

[10] R.Vadivel, Sudalai Muthu T: An effective HPSO-MGA Optimization Algorithm for Demand based Resource apportionment in Cloud Environment. 2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)

https://dx.doi.org/10.1109/icaccs48705.2020.9074442

[11] Javad Vahidi, Maral Rahmati: Optimization of Resource apportionment in Cloud Computing by Grasshopper Optimization Algorithm. 5th Conference on Knowledge-Based Engineering and Innovation, Iran University of Science and Technology, Tehran, Iran https://dx.doi.org/ 10.1109/KBEI.2019.8735098

[12] Sharma, N., Guddeti, R.M.: Multi-objective energy efficient virtual machines allocation at the cloud data center. IEEE Trans. Serv. Comput. (2016). https://doi.org/10.1186/s13677-017-0086-z

[13] Kayalvili, S., Selvam, M.: Hybrid SFLA-GA algorithm for an optimal Resource apportionment in cloud. Clust. Comput. (2018). https://doi.org/10.1007/s10586-018-2011-8

[14] Mireslami, S., Rakai, L., Far, B.H., Wang, M.: Simultaneous cost and QoS optimization for cloud Resource apportionment. IEEE Trans. Netw. Serv. Manag. 14(3), 676–689 (2017) https://doi.org/10.1109/TNSM.2017.2738026