International Journal of Computational and Experimental Science and ENgineering (IJCESEN)

Vol. 10-No.4 (2024) pp. 1601-1609
http://www.ijcesen.com

ISSN: 2149-9144

Copyright © IJCESEN

Research Article

Biased Random Sampling with Firefly Optimization (BRS-FO) based on Load Balancing for Virtual Machine Migration in Cloud Computing.

A. Siva Sankari[1]*, S. Vimalanand[2]

[1*]Research Scholar, Department of Computer Science, Periyar University, Salem Tamil Nadu, India.
* **Corresponding Author Email:** sankarisiva001@gmail.com - **ORCID:** 0009-0006-8312-4069

[2]Research Supervisor, Principal, Achariya college of arts and science, Puducherry, India
Email: aaschead@achariya.org - **ORCID:** 0009-0007-2479-1790

Article Info:

DOI: 10.22399/ijcesen.753
Received : 12 June 2024
Accepted : 14 December 2024

Keywords :

Biased random sampling,
Firefly optimization,
Cloud computing,
Virtual machine,
Load balancing,
Resource utilization.

Abstract:

A concept known as "Cloud Computing" aims to simplify the on-demand delivery of software, hardware, as well as data as services and give end users adaptable, scalable, and accessible services through the Internet. The major goal of the suggested technique is to create a healthy balance of load across all the Cloud resources servers while maximizing resource usage. Every resource will first have a load model created based on a number of variables, including memory use, processing time, and access rate. Several meta heuristics optimization algorithm are presented in literature for VM migration with load balancing in Cloud Computing (CC). However, in the paper Biased Random Sampling with Firefly Optimization (BRS-FO) was combined. The Load balancing is performed by biased random sampling and Firefly Optimization by maintaining the virtual server availability. In this method the performance of proposed algorithm was compared with PSO, GA and Honey Bee Optimization (HBO). The parameters taken for analysis are Makespan, Response time and energy consumption. From this experimental results, the proposed BRS-FO achieved the makespan of 5s, response time of 1s and energy consumption of 5J and proved this method is efficient than other system.

1. Introduction

The Cloud Computing is an arising innovation and recent fad for figuring in light of virtualization of assets [1]. In Cloud climate the actual machines run numerous Virtual Machines (VM) which are introduced to the clients as the figuring assets. The engineering of a VM depends on an actual PC with comparative usefulness [2]. As a matter of fact, VM is a visitor program with programming assets working like an actual PC. Asset portion strategy is a significant cycle to assign assets in light of client's application requests to accomplish an ideal number of servers being used [3]. This cycle is done powerfully with the end goal of burden adjusting of non-preplanned errands. Load adjusting is a NP-hard streamlining issue in Cloud Computing. This procedure endeavors to adjust the responsibility across VMs, which expects to limit reaction time to keep commitments and nature of administration as per Service Level Agreements (SLA) between the clients and the supplier. Moreover, this interaction must be completed routinely because of the time-variation nature of the heaps of Application Environments (AE). As a matter of fact, Cloud's clients are intrigued to have their positions finished in the most brief conceivable time and at the base expense [4].

Then again, the Cloud suppliers are intrigued to boost the utilization of their assets with a lower by and large expense to build their benefit. Clearly these two goals are in struggle and frequently they are not happy with the customary strategies for asset distribution and burden adjusting methods [5]. The old style strategies are exceptionally tedious [6]. Conventional rough strategies are accounted for uncertain and mistaken and frequently caught in neighborhood ideal [7].

In Cloud climate the actual machines run numerous Virtual Machines (VM) which are introduced to the clients as the figuring assets. The engineering of a VM depends on an actual PC with comparative usefulness. As a matter of fact, VM is a visitor program with programming assets working like an

actual PC. Asset portion strategy is a significant cycle to assign assets in light of client's application requests to accomplish an ideal number of servers being used. This cycle is done powerfully with the end goal of burden adjusting of non-preplanned errands. Load adjusting is a NP-hard streamlining issue in Cloud Computing. This procedure endeavors to adjust the responsibility across VMs, which expects to limit reaction time to keep commitments and nature of administration as per Service Level Agreements (SLA) between the clients and the supplier. Moreover, this interaction must be completed routinely because of the time-variation nature of the heaps of Application Environments (AE). As a matter of fact, Cloud's clients are intrigued to have their positions finished in the most brief conceivable time and at the base expense [8-10].

In the one-sided arbitrary testing calculation, for each cluster of hundred positions, little changes in the scope of seconds were acquired. The change which happened is a direct result of the postponement being caused because of inaccessibility of assets. An irregular hub is chosen just at first and the walk is taken in view of the probabilistic capability and subsequently the walk just goes up to the worth of jump count for each portion of work. The time taken for load adjusting of each and every work changes by seconds in light of the fact that the calculation searches for a server with most extreme assets accessible and afterward attempts to distribute the work. In this way, overheads are acquired as a result of finding the way for each occupation followed by contrasting it; despite the fact that main at long last the work size is checked [11,12].

The powerful improvement method rather than hereditary calculation [13] can prompt better burden adjusting since it is a conventional and old calculation. In like manner, I have wanted to use a new enhancement calculation, called firefly calculation [14] to do the heap adjusting activity in our proposed work. Load record will be registered in light of the recently determined formulae. In view of burden

file, load adjusting activity will be done utilizing firefly calculation.

Besides, the proposed calculation well adjusts the heap and actually considers load adjusting in light of Make range, Response investment utilization that prompts negligible measure of asset use. The commitment of the work is introduced as follows:

- To plan the BRS-FO load adjusting framework for VM relocation in Cloud Computing for better asset usage.
- To think about the exhibition of BRS-FO calculation with existing GA, HBO, PSO calculation.

The association of the paper is introduced as follows: Section 2 depicts the connected works. In area 3, proposed technique is introduced. Results and conversation is examined in area 4. At long last, end is introduced in area 5.

2. Related Works

A powerful added balance strategy is utilized in [15] to take care of this issue. Cloud load adjusting (CLB) thinks about both server handling power and PC stacking, in this way making it doubtful that a server will not be able to deal with unreasonable computational prerequisites. At long last, two calculations in CLB are likewise addressed with trials to demonstrate this approach is imaginative

A clever dynamical burden adjusted planning (DLBS) approach was introduced in [16] for boosting the organization throughput while adjusting responsibility powerfully. DLBS issue, and afterward foster a bunch of proficient heuristic booking calculations for the two run of the mill OpenFlow network models, which balance information streams time allotment by time allotment. Trial results show that DLBS approach altogether outflanks other delegate load-adjusted planning calculations Round Robin and LOBUS.

Asset Intensity Aware Load adjusting technique (RIAL) was introduced in [17]. For every PM, RIAL progressively allots various loads to various assets as per their use power The time taken for load adjusting of each and every work changes by seconds in light of the fact that the calculation searches for a server with most extreme assets accessible and afterward attempts to distribute the work. In this way, overheads are acquired as a result of finding the way for each occupation followed by contrasting it; despite the fact that main at long last the work size is checked. It has a stricter relocation setting off calculation to keep away from pointless movements while as yet fulfilling Service Level Objects (SLOs). A broad follow driven recreation results and certifiable exploratory outcomes show the better presentation of RIAL looked at than other burden adjusting strategies. Another worldview for virtual machine movement, in view of the requests of the was introduced in [18,19]. VMM Approach Based on Distance and Traffic is created. The decrease in full circle time and keeping up with the full circle time absent a lot of vacillation even on account of a disappointment of one of the actual machine assists with working on the exhibition by offering quicker types of assistance to the clients.

The time taken for load adjusting of each and every work changes by seconds in light of the fact that the calculation searches for a server with most extreme assets accessible and afterward attempts to distribute the work. In this way, overheads are acquired as a result of finding the way for each occupation followed by contrasting it; despite the fact that main at long last the work size is checked. The decrease in full circle time and keeping up with the full circle time absent a lot of variance even on account of a disappointment of one of the actual machine assists with working on the presentation by offering quicker types of assistance to the clients.

An original cross breed calculation in view of the Fuzzy rationale and insect province enhancement (ACO) ideas to further develop the heap adjusting in the Cloud climate was introduced in [20]. The accomplished recreations through Cloud Analyst stage exhibit the viability of the consolidated Fuzzy-ACO calculation in examination with other burden adjusting calculations. A half breed metaheuristics method which consolidates the osmotic way of behaving with bio-enlivened load adjusting calculation was introduced in [21]. The osmotic way of behaving empowers the programmed arrangement of virtual machines (VMs) that are moved through Cloud foundations. Since the half breed fake honey bee province and subterranean insect state enhancement demonstrated its proficiency in the powerful climate in Cloud Computing. It upgrades the nature of administrations (QoSs) which is estimated by administration level understanding infringement (SLAV) and execution debasement because of relocations (PDMs).

The issue of middle hub determination in Scatter-Gather relocation was introduced in [22] and demonstrate that it is NP-finished. The issue is numerically demonstrated as a number programming issue in light of two optimality models: limiting removal time and limiting energy. Two heuristic calculations: greatest decline in-expulsion time and least-expansion in-energy are utilized to take care of the issue and their presentation is broke down as for three boundaries removal time, energy and absolute relocation.

An answer for take care of the issues of inertness on HEC servers brought about by their restricted assets was introduced by [23]. The expansion in the rush hour gridlock rate makes a long line on these servers, i.e., a raise in the handling time (delay) for demands. The strategy called HEC-Clustering Balance was utilized. HEC-Clustering Balance is more proficient than pattern grouping and burden adjusting procedures. Consequently, contrasted with the HEC design, e handling time was diminished on the HEC servers to 19% and 73% separately on two trial situations.

A powerful strategy for adjusting of burden among the virtual machines utilizing hybridization of changed Particle swarm optimization (MPSO) was introduced in [24]. The osmotic way of behaving empowers the programmed arrangement of virtual machines (VMs) that are moved through Cloud foundations. Since the half breed fake honey bee province and subterranean insect state enhancement demonstrated its proficiency in the powerful climate in Cloud Computing. It upgrades the nature of administrations (QoSs) which is estimated by administration level understanding infringement (SLAV) and execution debasement because of relocations (PDMs).

The changed bumble beeinspired calculation was introduced in [25] for better designation of help with a heap adjusting plan. This paper presents a productive calculation in view of exploratory execution examination of burden adjusting of undertakings utilizing bumble bee motivated for asset assignment in Cloud climate. The calculation introduced in [26], the Cloud suppliers are intrigued to boost the utilization of their assets with a lower by and large expense to build their benefit. Clearly these two goals are in struggle and frequently they are not happy with the customary strategies for asset distribution and burden adjusting methods. The old style strategies are exceptionally tedious. Conventional rough strategies are accounted for uncertain and mistaken and frequently caught in neighborhood ideal.

Prescient Priority-based Modified Heterogeneous Earliest Finish Time calculation was introduced in [27] An original cross breed calculation in view of the Fuzzy rationale and insect province enhancement (ACO) ideas to further develop the heap adjusting in the Cloud climate was introduced in [20]. The accomplished recreations through Cloud Analyst stage exhibit the viability of the consolidated Fuzzy-ACO calculation in examination with other burden adjusting calculations.

Two hereditary based techniques are coordinated and introduced in [28]. The exhibition models of VMs are removed from their making boundaries and comparing execution estimated in a Cloud Computing climate. Quality articulation programming is applied for producing emblematic relapse models that depict the exhibition of VMs and are utilized for anticipating heaps of VMHs after load-balance. Exploratory outcomes show that this technique outflanks past strategies, like heuristics and measurements relapse. In the majority of the current framework, difficulty happens with the relocation of virtual machines, this makes network awkwardness and leads wasteful asset use. For appropriate use of the asset, reaction time and

makespanwill be decreased and this is diminished by this proposed BRS-FO technique.

3. Proposed Methodology

The proposed technique depends on holding the calculations of BRS and FO. These two calculations are at present utilized as estimation calculation for laying out load adjusting in view of time and cost among assets and productivity. With such hybridization it is pointed toward accelerating the cycle while keeping up with the improvement of neighborhood advancement and expanding the exactness. The BRS and FO calculations are subsequently acquainted as the essential arrangements with the depicted issue.

In this calculation every one of the servers are treated as hubs. Here network is addressed as a virtual diagram. The in-degree addresses the free assets accessible to the hub. Based on in-degree the heap balancer allocates the assignments to the hub. At the point when an errand is doled out then the in-degree is decremented and it is augmented when the occupation gets executed. Figure 1 represents the engineering of BRS-FO framework

Asset virtualization is the main closeness between block figuring and Grid. In this way, the one-sided irregular examining calculation will be appropriate to Cloud Computing moreover.

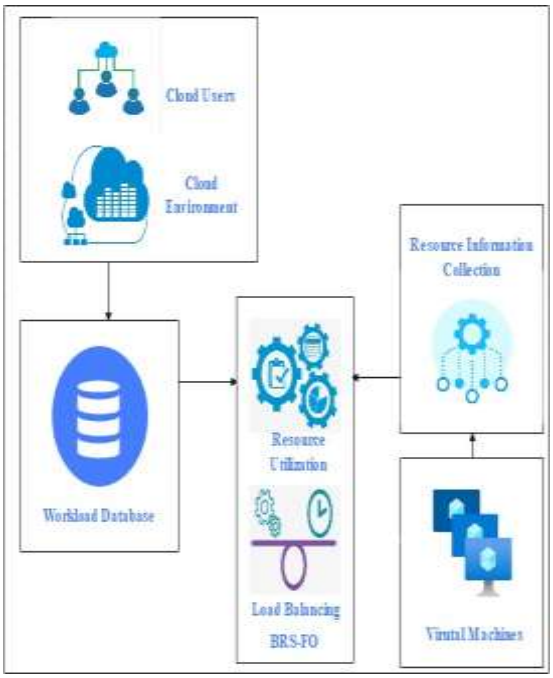

Figure 1. *Architecture of BRS-FO system.*

Possibility of a VM being chosen in this biassed random sampling is computed by dividing its relative importance by the total nominal attributes of all contenders in the selection set. The priorities are changed previously into their reciprocals in the situation of ext = min, where the VMs with the smallest priority should be given the highest selection probability. In eq (1), the following formula is presented in its entirety as follows:

$$v'(i) \leftarrow \begin{cases} 1/v(i) \; if \; ext = min \\ v(i) if \; ext = max \end{cases}$$

$$v'(i) \leftarrow \begin{cases} 1/v(i) \; if \; ext = min \\ v(i) if \; ext = max \end{cases} \; (j \in D_n) \in D_n) \quad (1)$$

Where, The random sampling system assigns a probability value p (i) to every virtual machine (VM) in cloud computing using a mapping p: Dn —> [0,1]. Every candidate's priority value is essentially converted into a probability value using this mapping. All probability, of course, add up to one. Afterwards, the selection probabilities are derived according toeq (2) presented below

$$p(i) \leftarrow \frac{v'(i)}{\sum_{i' \in D_n} v'(i')} p(i) \leftarrow \frac{v'(i)}{\sum_{i' \in D_n} v'(i')}$$

$$\in D_n; v(i) > 0) \in D_n; v(i) > 0) \quad (2)$$

In the situation of ext = min, this scheme (BRS) is modified. Different methods are used to modify the priority values by

$$v'(i) \leftarrow \begin{cases} M - v(i) \; if \; ext = min \\ v(i) \; if \; ext = max \end{cases}$$

$$v'(i) \leftarrow \begin{cases} M - v(i) \; if \; ext = min \\ v(i) \; if \; ext = max \end{cases} \; (i \in D_n$$

$$\in D_n) \quad (3)$$

M must be a sufficient size to ensure that all updated priority values are nonnegative. It is obvious that this technique is still valid if certain priority values are zero. The selection probabilities are then calculated from these using eq (1).

By skewing the sampling process in favour of a totally random one, values of M that are quite big compared to the priorities actually lessen the influence the priority rules have.

This priorities v(j) are altered when extr = min, i.e..

$$v'(i) \leftarrow \begin{cases} \max v(D_n) - v(i) + \min V(D_n) \; if \; xt = min \\ v(i) \; if \; ext = max \end{cases}$$

$$v'(i) \leftarrow \begin{cases} \max v(D_n) - v(i) + \min V(D_n) \; if \; xt = min \\ v(i) \; if \; ext = max \end{cases}$$

$$(i \in D_n)(i \in D_n) \quad (4)$$

Where $V(D_n)V(D_n)$ denotes the allpositive transformed priorities of VM

$$V(D_n) \leftarrow \{v'(i) | \, i \in D_n \wedge v'(i) > 0 \}$$
$$V(D_n) \leftarrow \{v'(i) | \, i \in D_n \wedge v'(i) > 0 \}$$
(5)

Next, the priorities are adjusted so that every virtual machine has a selection probability higher than zero.

$$v"(i) \leftarrow \begin{cases} (v'(i) + \mathcal{E})^{\alpha} \; if \; i \in D_n v'(i) = 0 \\ (v'(i))^{\alpha} \; otherwise \end{cases}$$
$$v"(i) \leftarrow \begin{cases} (v'(i) + \mathcal{E})^{\alpha} \; if \; i \in D_n v'(i) = 0 \\ (v'(i))^{\alpha} \; otherwise \end{cases}$$
(6)

Here, £ is determined from

$$\boldsymbol{\varepsilon} \leftarrow \begin{cases} minV'(D_n)/10 \; if \; i \in D_n v'(i) > 0 \\ 1 \; otherwise \end{cases}$$
$$\boldsymbol{\varepsilon} \leftarrow \begin{cases} minV'(D_n)/10 \; if \; i \in D_n v'(i) > 0 \\ 1 \; otherwise \end{cases}$$
(7)

The selection probabilities are obtained from the so-transformed priorities as shown in (4). is used sparingly; yet, the plan guarantees that no virtual machine is left out of the selection process. At times when certain transformed priorities are zero and others are not, its initial branch in (7) tends to maintain a minimal impact. The selection probabilities are obtained from the so-transformed priorities as shown in (4). is used sparingly; yet, the plan guarantees that no virtual machine is left out of the selection process. In (7), its first branch tends to maintain a little effect whenever certain modified priorities are zero and others are not. Only when all of the candidates in the decision set have zero priority, which would leave them all undefined, does the second branch apply; in that case, adding an arbitrary constant will give each contender an equal chance of being chosen. The fact that (4) attempts to normalize the altered priorities v"(i) is referenced in the scheme's name. the candidates with the highest and lowest probability are assigned to the very same interval as the initial priority v(i), irrespectively of whether ext is max or min. The best schemes are RBS; the former should indeed be employed if all plausible rules are to be used, and the latter if simply the best algorithm is.

It is obvious that the differences between various parallel processing should be lower than those among different serial ones as they explore the shorter Solution space, especially when using more iterations. BRS is one of the finest methods for resource scheduling in parallel. The greatest schedules of all the parallel algorithms analysed were produced by RBRS for 4 of the 7 promising rules. In other words, just the best algorithm should be used, which is RBRS.

Firefly Optimization

The following three idealised processes are used by the Firefly Optimization (FFO) method, which is based on the flashing characteristics of fireflies: Since all fireflies are unisex, they are all attracted to one another regardless of their gender. Attraction is inversely correlated with distance, therefore for every 2flashing firefly, the less attractive one will move closer to the more attractive one. A firefly moves randomly and its brightness is controlled by the topography of the optimization problem that has to be improved if no dragonfly is brighter than a certain firefly.

The appealing qualities and light intensity of the firefly optimization technique are its distinguishing features. In accordance with the inverse-square law, the intensity of the light I decreases as the distance d grows. The distance d has an exponential relationship with the fluctuation in light intensity I(p).

$$I(d) = I_0 e^{-\gamma d^2} I(d) = I_0 e^{-\gamma d^2}$$
(8)

Where,$\gamma\gamma$ = the light absorption rate and $I_0 I_0$ = the initial light intensity

Thus, the attractiveness of Firefly is defined by eq. (9)

$$\beta(d) = \beta_0 e^{-\gamma d^2} \beta(d) = \beta_0 e^{-\gamma d^2}$$
(9)

Where, d = the distance among the fireflies $\beta_0 \beta_0$ = the attractiveness at the distance d = 0. $\gamma\gamma$ = the attractiveness variation.

$$x_j(v+1) = x_i + \beta_0 e^{-\gamma d^2} \left(x_j - x_i\right) + \alpha(ran - \tfrac{1}{2})$$
$$x_j(v+1) = x_i + \beta_0 e^{-\gamma d^2} \left(x_j - x_i\right) + \alpha(ran - \tfrac{1}{2})$$
(10)

Where, $\alpha \, \alpha$ = the randomization parameter, d = the random number which is distributed uniformly.

P(i) = {p1, p2, ….pg) is a symbol for the actual machine in the cloud data centre. Users of the Cloud provide the jobs to the Cloud data centre. Depending on a BRS algorithm, the datacenter broker decides how to distribute the satisfied workloads across virtual machines. The detection of a virtual machine that is overloaded is the first goal of this operation. A list with the recognised virtual machines is organised in decreasing order. The jobs are deleted from the chosen virtual machine after recognisingthe excessively overcrowded virtual machines. The tasks that are eliminated must be distributed to the

best available, low-loaded VMs. The eliminated jobs will be considered for migration using the underutilised resources. The virtual machine's load is determined.

$$\text{VM Load} = \frac{\text{allocated jobs}}{VM_i} \quad \frac{\text{allocated jobs}}{VM_i} \quad (11)$$

$$\text{Makespan VM} = \max(\text{Completion\_Time(jobs(i))}) \quad (12)$$

In a VM Makespanis frequently used to refer to the total time required to complete the jobs. The suggested method considerably speeds the project completion. The firefly's attraction is used by VM as a criterion for choosing which job to assign to the one that was eliminated. The BRS-FO, such as low, high, determines how desirable it is to select the optimal virtual machine. The BRS may also be used to determine how effective Firefly is at choosing the right virtual machine.

Based on the BRS variables, the triangle membership functions are created. As for firefly to select a suitable virtual machine should allocate the deleted jobs, the following guidelines are taken into account.

If $\beta_{ij}\beta_{ij}$ is low and VMLoad is very very low then efficacy of selecting virtual machine is very very low

If $\beta_{ij}\beta_{ij}$is medium and VMLoad is very low then efficacy of selecting virtual machine is very low

If $\beta_{ij}\beta_{ij}$is high and VMLoad is low then efficacy of selecting virtual machine is low

If $\beta_{ij}\beta_{ij}$is low and VMLoad is medium then efficacy of selecting virtual machine is low

If $\beta_{ij}\beta_{ij}$is medium and VMLoad is medium then efficacy of selecting virtual machine is medium

If $\beta_{ij}\beta_{ij}$is high and VMLoad is medium then efficacy of selecting virtual machine is high

If $\beta_{ij}\beta_{ij}$is low and VMLoad is high then efficacy of selecting virtual machine is high

If $\beta_{ij}\beta_{ij}$is medium and VMLoad is high then efficacy of selecting virtual machine is very high

If $\beta_{ij}\beta_{ij}$is high and VMLoadis high then efficacy of selecting VM is very very high

Thus the resource scheduling is effectively

Pseudo code:

Step 1: Start BRS-FO
Step 2: Compute Energy and response time
Step 3: Create SortList() for n;
Step 4: Compute Distance() between n;
for (n)
{
Initialize Vm;
calculateVm with v
calculate R1, R2...Rn;
create D;
}
Step 5: Set D = Rn;
Step 6: Stop the process.

4. Results and Discussion

The Load balancing is performed by biased random sampling and Firefly Optimization by maintaining the virtual server availability. In this method the performance of proposed algorithm was compared with Particle Swarm Optimization, Genetic Algorithm and HBO. The parameters taken for analysis are Makespan, Response time and energy consumption.

Makespan: Makespan is often referred to as the overall completion time of the tasks. Table 1represents the comparison of makespan with the proposed BRS-FO with GA, HBO and PSO Figure 2 describes the comparison of Makespan with the proposed BRS-FO, PSO, HBO, GA. In this

Table 1. *Comparison of makespan with the proposed BRS-FO with GA, HBO and PSO.*

Number of task	Makespan (s)			
	GA [27]	HBO [25]	PSO [24]	Proposed (BRS-FO)
20	15	7	10	5
40	28	15	20	10
60	30	18	22	12
80	58	32	36	25
100	60	45	50	43

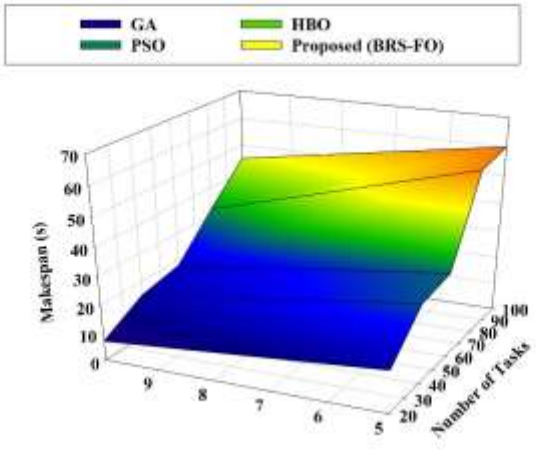

Figure 2. *Comparison of Makespan.*

figure X-axis represents the number of tasks and makespan in secondsis represented in Y-axis. Blue color represents GA, light green color represents HBO, Dark green color represents PSO and the proposed BRS-FO is indicated by yellow color.

Response time: Response time, in the context of load balancing in Cloud Computing, which is the elapsed time between the task. Table 2 represents the comparison of response time with the proposed BRS-FO with GA, HBO and PSO

Table 2. *Comparison of response time with the proposed BRS-FO with GA, HBO and PSO.*

Number of task	Response Time(s)			
	GA [27]	HBO [25]	PSO [24]	Proposed (BRS-FO)
20	5	2	3	1
40	42	20	30	10
60	52	25	43	21
80	63	46	58	35
100	70	52	60	46

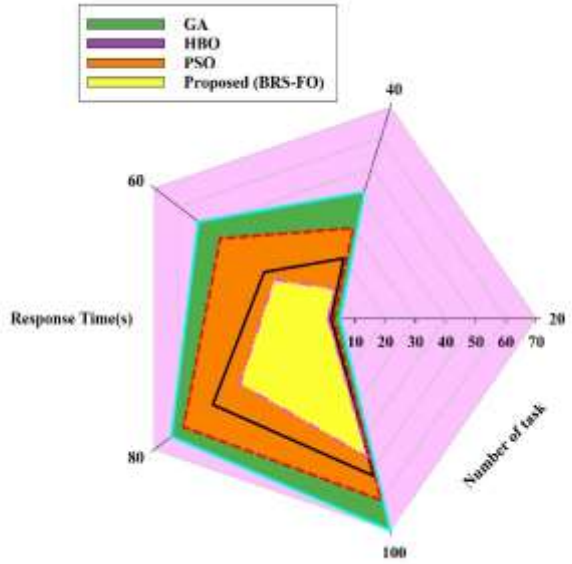

Figure 3. *Comparison of response time.*

Figure 3 describes the comparison of response time with the proposed BRS-FO, PSO, HBO, GA . In this figure X-axis represents the number of tasks and response time in seconds is represented in Y-axis. Green color represents GA, purple color represents HBO, orange color represents PSO and the proposed BRS-FO is indicated by yellow color.

Energy Consumption: Energy Consumption calculation is equal to the total operating hours of the total power supplied to complete a job. Table 3 presents the comparison of energy consumption with the proposed BRS-FO with GA, HBO and PSO Figure 4 describes the comparison of energy consumption with the proposed BRS-FO, PSO, HBO, GA. In this figure X-axis represents the number of tasks and energy consumption in joules is represented in Y-axis. Grey color represents GA, red color represents HBO, green color represents PSO and the proposed BRS-FO is indicated by yellow color.

Table 3. *Comparison of energy consumption with the proposed BRS-FO with GA, HBO and PSO.*

Number oftask	Energy Consumption(J)			
	GA [27]	HBO [25]	PSO [24]	Proposed (BRS-FO)
20	2	5	6	4
40	4	6	7	5
60	3	7	8	5
80	5	9	9	9
100	9	12	10	8

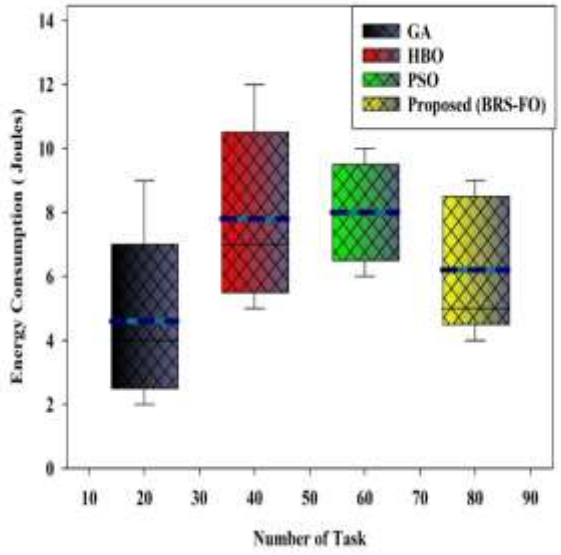

Figure 4*. Comparison of Energy Consumption*

5. Conclusion

Biased Random Sampling and Firefly Optimization (BRS-FO) were coupled in this study. Biased random sampling and Firefly Optimization are used to balance the load while preserving the availability of the virtual servers. The effectiveness of the suggested approach was evaluated using this method in comparison to PSO, Genetic Algorithm and Honey Bee Optimization (HBO). Makespan, Response time, and energy usage are the factors considered for examination. The suggested BRS-FO demonstrated that this approach is more effective than previous systems by achieving a number of iterations of 5s, response time of 1s, and energy usage of 5J. It would be worthwhile to do studies on the effectiveness and application of various metaheuristic algorithms for various balance objectives. These will be included in our upcoming work. Interesting similar works reported in the literature [29-34].

Author Statements:

- **Ethical approval:** The conducted research is not related to either human or animal use.
- **Conflict of interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper
- **Acknowledgement:** The authors declare that they have nobody or no-company to acknowledge.
- **Author contributions:** The authors declare that they have equal right on this paper.
- **Funding information:** The authors declare that there is no funding to be acknowledged.
- **Data availability statement:** The data that support the findings of this study are available on request from the corresponding author. The data are not publicly available due to privacy or ethical restrictions.

References

[1] Ramachandra, G., Iftikhar, M., & Khan, F. A. (2017). A comprehensive survey on security in Cloud Computing. *Procedia Computer Science*, 110, 465-472.

[2] Basu, S., Bardhan, A., Gupta, K., Saha, P., Pal, M., Bose, M., ...& Sarkar, P. (2018, January). Cloud Computing security challenges & solutions-A survey. In *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)* (pp. 347-356).

[3] Zhou, Y., Zhang, D., &Xiong, N. (2017). Post-Cloud Computing paradigms: a survey and comparison. *Tsinghua Science and Technology*, 22(6), 714-732.

[4] Bokhari, M. U., Makki, Q., &Tamandani, Y. K. (2018). A survey on Cloud Computing. *In Big Data Analytics* (pp. 149-164).

[5] Ghomi, E. J., Rahmani, A. M., &Qader, N. N. (2017). Load-balancing algorithms in Cloud Computing: A survey. *Journal of Network and Computer Applications*, 88, 50-71. https://doi.org/10.1016/j.jnca.2017.04.007

[6] Noshy, M., Ibrahim, A., & Ali, H. A. (2018). Optimization of live virtual machine migration in Cloud Computing: A survey and future directions. *Journal of Network and Computer Applications*, 110, 1-10. DOI:10.1016/j.jnca.2018.03.002

[7] Pradhan, A., Bisoy, S. K., &Mallick, P. K. (2020). Load balancing in Cloud Computing: Survey. *Innovation in Electrical Power Engineering, Communication, and Computing Technology* (pp. 99-111).

[8] Jyoti, A., Shrimali, M., & Mishra, R. (2019, January). Cloud Computing and load balancing in Cloud Computing-survey. In 2019 *9th International Conference on Cloud Computing, Data Science & Engineering* (Confluence) (pp. 51-55).

[9] Shafiq, D. A., Jhanjhi, N. Z., & Abdullah, A. (2019, December). Proposing a load balancing algorithm for the optimization of Cloud Computing applications. *2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS)* (pp. 1-6).

[10] Garg, D., & Kumar, P. (2018, July). A Survey on Metaheuristic approaches and its evaluation for load balancing in Cloud Computing. *International Conference on Advanced Informatics for Computing Research* (pp. 585-599).

[11] Sajjan, R. S., &Yashwantrao, B. R. (2017). Load balancing and its algorithms in Cloud Computing: A survey. *International Journal of Computer Sciences and Engineering*, 5(1), 95-100.

[12] Cabrera, G., Gonzalez-Martin, S., Juan, A. A., Marquès, J. M., &Grasman, S. E. (2014, December). Combining biased random sampling with metaheuristics for the facility location problem in distributed computer systems. In *Proceedings of the Winter Simulation Conference* 2014 (pp. 3000-3011).

[13] Xu, M., Tian, W., &Buyya, R. (2017). A survey on load balancing algorithms for virtual machines placement in Cloud Computing. *Concurrency and Computation: Practice and Experience*, 29(12), e4123. https://doi.org/10.48550/arXiv.1607.06269

[14] Kansal, N. J., & Chana, I. (2016). Energy-aware virtual machine migration for Cloud Computing-a firefly optimization approach. *Journal of Grid Computing*, 14(2), 327-345. https://doi.org/10.1007/s10723-016-9364-0

[15] Chen, S. L., Chen, Y. Y., &Kuo, S. H. (2017). CLB: A novel load balancing architecture and algorithm for Cloud services. *Computers & Electrical Engineering*, 58, 154-160. https://doi.org/10.1016/j.compeleceng.2016.01.029

[16] Tang, F., Yang, L. T., Tang, C., Li, J., &Guo, M. (2016). A dynamical and load-balanced flow scheduling approach for big data centers in Cloud s. *IEEE Transactions on Cloud Computing*, 6(4), 915-928.

[17] Shen, H., & Chen, L. (2017). A resource usage intensity aware load balancing method for virtual machine migration in Cloud datacenters. *IEEE Transactions on Cloud Computing*, 8(1), 17-31.

[18] Kumar, M., & Sharma, S. C. (2017). Dynamic load balancing algorithm for balancing the workload among virtual machine in Cloud Computing. *Procedia computer science*, 115, 322-329.

[19] Shahapure, N. H., &Jayarekha, P. (2018). Distance and traffic based virtual machine migration for scalability in Cloud Computing. *Procedia computer science*, 132, 728-737.

[20] Ragmani, A., Elomri, A., Abghour, N., Moussaid, K., &Rida, M. (2019). An improved hybrid fuzzy-ant colony algorithm applied to load balancing in Cloud

Computing environment. *Procedia Computer Science*, 151, 519-526.

[21] Gamal, M., Rizk, R., Mahdi, H., &Elnaghi, B. E. (2019). Osmotic bio-inspired load balancing algorithm in Cloud Computing. *IEEE Access*, 7, 42735-42744. DOI:10.1109/ACCESS.2019.2907615

[22] Annadanam, C. S., Chapram, S., & Ramesh, T. (2020). Intermediate node selection for Scatter-Gather VM migration in Cloud data center. *Engineering Science and Technology, an International Journal*, 23(5), 989-997. https://doi.org/10.1016/j.jestch.2020.01.008

[23] Babou, C. S. M., Fall, D., Kashihara, S., Taenaka, Y., Bhuyan, M. H., Niang, I., &Kadobayashi, Y. (2020). Hierarchical load balancing and clustering technique for home edge computing. *IEEE Access*, 8, 127593-127607. doi: 10.1109/ACCESS.2020.3007944.

[24] Jena, U. K., Das, P. K., &Kabat, M. R. (2022). Hybridization of meta-heuristic algorithm for load balancing in Cloud Computing environment. *Journal of King Saud University-Computer and Information Sciences.* 34(6)Part A;2332-2342 https://doi.org/10.1016/j.jksuci.2020.01.012

[25] Sharma, A. K., Upreti, K., &Vargis, B. (2020). Experimental performance analysis of load balancing of tasks using honey bee inspired algorithm for resource allocation in Cloud environment. *Materials Today Proceedings* DOI:10.1016/j.matpr.2020.09.359

[26] Shafiq, D. A., Jhanjhi, N. Z., Abdullah, A., &Alzain, M. A. (2021). A Load Balancing Algorithm for the Data Centres to Optimize Cloud Computing Applications. *IEEE Access*, 9, 41731-41744.

[27] Sohani, M., & Jain, S. C. (2021). A Predictive Priority-Based Dynamic Resource Provisioning Scheme With Load Balancing in Heterogeneous Cloud Computing. *IEEE Access*, 9, 62653-62664. doi: 10.1109/ACCESS.2021.3074833

[28] Hung, L. H., Wu, C. H., Tsai, C. H., & Huang, H. C. (2021). Migration-based load balance of virtual machine servers in Cloud Computing by load prediction using genetic-based methods. *IEEE Access*, 9, 49760-49773

[29] M, P., B, J., B, B., G, S., & S, P. (2024). Energy-efficient and location-aware IoT and WSN-based precision agricultural frameworks. *International Journal of Computational and Experimental Science and Engineering*, 10(4);585-591. https://doi.org/10.22399/ijcesen.480

[30] Pattanaik, B. C., Sahoo, B. kumar, Pati, B., & Pradhan, A. (2024). Enhancing Fault Tolerance in Cloud Computing using Modified Deep Q-Network (M-DQN) for Optimal Load Balancing. *International Journal of Computational and Experimental Science and Engineering*, 10(4);1094-1100. https://doi.org/10.22399/ijcesen.601

[31] S. Praseetha, & S. Sasipriya. (2024). Adaptive Dual-Layer Resource Allocation for Maximizing Spectral Efficiency in 5G Using Hybrid NOMA-RSMA Techniques. *International Journal of Computational and Experimental Science and Engineering*, 10(4);1130-1139. https://doi.org/10.22399/ijcesen.665

[32] Naresh Kumar Bhagavatham, Bandi Rambabu, Jaibir Singh, Dileep P, T. Aditya Sai Srinivas, M. Bhavsingh, & P. Hussain Basha. (2024). Autonomic Resilience in Cybersecurity: Designing the Self-Healing Network Protocol for Next-Generation Software-Defined Networking. *International Journal of Computational and Experimental Science and Engineering*, 10(4);1187-1203. https://doi.org/10.22399/ijcesen.640

[33] guven, mesut. (2024). Dynamic Malware Analysis Using a Sandbox Environment, Network Traffic Logs, and Artificial Intelligence. *International Journal of Computational and Experimental Science and Engineering*, 10(3);480-490. https://doi.org/10.22399/ijcesen.460

[34] S.P. Lalitha, & A. Murugan. (2024). Performance Analysis of Priority Generation System for Multimedia Video using ANFIS Classifier. *International Journal of Computational and Experimental Science and Engineering*, 10(4);1320-1328. DOI: https://doi.org/10.22399/ijcesen.707