*Journal of Information & Optimization Sciences*
ISSN 0252-2667 (Print), ISSN 2169-0103 (Online)
Vol. 47 (2026), No. 3, pp. 1145–1162
DOI : 10.47974/JIOS-2113

# Advancing cloud load balancing : An energy-aware model using a hybrid genetic and nature-inspired algorithm

Yashika Sharma *
Sachin Lakra †
*Department of Computer Science and Technology*
*School of Engineering*
*Manav Rachna University*
*Faridabad 121003*
*Haryana*
*India*

**Abstract**

Cloud computing is necessary for the current global computing demand since it is becoming difficult to maintain disk space and power requirements for individual users. The dependency on computing is increasing day by day, thus increasing the pressure on the infrastructure required to meet those needs. Cloud computing is the solution to cater to this problem. When the load on a cloud is increased, it becomes imperative to manage the load or distribute the load judiciously to make job scheduling on the cloud as fair as possible. No server should ideally either be underloaded or overloaded. This equilibrium is to be maintained for the smooth functioning of the cloud environment. This research work aims to develop an energy efficient algorithm meant for the cloud setup that distributes the load evenly and has a substantial enhancing impact on the throughput of the entire system as well. These two subgoals when achieved, will permit the load distribution to move towards the main goal of achieving higher energy efficiency also and translating the entire system towards a greener system. To realize this, this paper proposes a hybrid optimization algorithm with a unification of biogeography-based optimization algorithm and genetic algorithm. The results of the new algorithm are presented in the paper and were found to lead to an increase in throughput and a reduction in scheduling cost while the number of tasks submitted is increased.

***Subject Classification:*** *Primary 68Q07, Secondary 49N30, 90C90.*

***Keywords:*** *Cloud computing, Genetic algorithm, Biogeography-based optimization, Ant colony optimization, Energy efficiency.*

* *ORCID-ID:* **0000-0002-6367-5852**
† *ORCID-ID:* **0000-0002-6977-4974**

* *E-mail:* **yashika.sharma85@gmail.com (Corresponding Author)**
† *E-mail:* **sachin@mru.edu.in**

## Introduction

Technology has re-engineered every facet of the life of human beings. Today, it is a crucial ingredient in every activity of one's life – be it shopping, cooking, studying, or working. It has touched every vertical of human existence. During the times of the Covid-19 pandemic, the world saw a much greater dependency on technology as compared to any other time in history. When people were not allowed to go out of their houses and the people of the entire world were confined inside their households, it was technology that kept the world connected. Countries could keep going with their economies and manage to cope with such a grave situation only with advancements in technology. Corporate employees were working, students were studying and teachers were teaching – all from their homes – and this became feasible essentially due to the information technology set-up that was developed over many years of research. Presently, technology is the driving force behind the ever-faster daily life of humans [1].

With the increasing use of technology, however, carbon footprints are also increasing exponentially. The carbon footprint of an entity is the quantity of carbon emissions or greenhouse gases that are released into the environment by the entity, either directly or indirectly. The climatic change or the carbon footprint is driven by the heat produced by computer systems and also by the energy consumed by these systems to achieve the goal of processing a large number of processing requests. The energy efficiency of such a system requires to be optimized to mitigate the environmental impact of this technology. Minute steps towards saving energy can make a huge transition towards a green IT infrastructure. Higher processing time results in a high energy requirement and also higher heat emission from the system. Therefore, the energy requirement of such computing systems requires to be minimized or optimized to reduce the negative effects of computing on the environment [2].

With the use of technology comes a huge amount of data that needs to be stored as well as processed. This, in turn, has led to increased resource requirements which have to be catered to, while ensuring that users' activities are not disrupted. With the sudden surge in demand for computing resources whether it is in terms of disk space or computing capacity, virtualization received its due share of importance. It was only the development of cloud computing infrastructure, that made the seamless movement from the routine lifestyle to the all-new virtual lifestyle, possible [3].

Users at large became unaware of the amount of disk space their data was using on the cloud, and all their resources began to be accessed through the cloud. However, due to the massive volume of requests, an efficient scheduling algorithm is crucial to balance the workload and optimize the waste energy dissipation to the environment. To attain this, various nature-inspired metaheuristic procedures have been projected, which tend to provide quick results and resolve issues more efficiently than other techniques. In this research paper, a hybrid algorithm called GA-BBO was proposed by combining two nature-driven algorithms, the biogeography-based optimization (BBO)

algorithm and the genetic algorithm (GA). This approach combines the optimal features of both algorithms while integrating crossover and mutation operations of GAs with the BBO algorithm [4]. As the modern cloud workload contains sensitive information, the model can further be extended by taking care of the confidentiality of the transmitted data [13] and adding randomness to achieve keystream unpredictability making the transmission more secure [14]. This can be achieved by using nonlinear and random mathematical modelling techniques leading to a secure and energy efficient cloud load balancing model.

Energy consumption in the cloud is dependent on numerous factors, together with throughput, scheduling, and bandwidth utilization as depicted in Fig. 1.

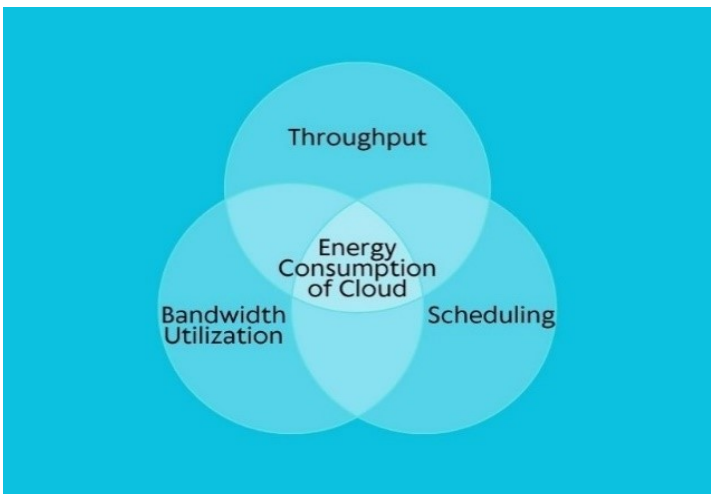


**Fig. 1**

**Factors affecting energy consumption of the cloud**

Throughput refers to the transmission rate of data through a network. Higher throughput implies higher energy consumption. Inefficient data transmission can lead to network congestion and increased energy consumption. Therefore, optimizing network throughput can reduce energy consumption in the cloud. Scheduling is another key aspect that influences energy consumption in the cloud. The scheduling algorithm used determines how computing resources are allocated to different tasks. If the scheduling algorithm is inefficient, it could impact the overutilization or underutilization of resources, which can steer higher energy consumption. Therefore, developing efficient scheduling algorithms is critical for cutting down energy usage in the cloud.

Optimizing the utilization of bandwidth is also crucial for reducing the energy usage of a cloud infrastructure since network congestion during transmission can lead to an increase in energy consumption. Thus, bandwidth utilization also must be optimized to reduce the wastage of energy in the cloud.

**Energy Consumption of a Cloud**

Energy consumption of cloud systems might be affected by user requests, request prioritization, scheduling, and request serving. Cloud systems are composed of various components such as servers, storage devices, and network infrastructure, among others. These components consume energy to perform their respective tasks. A cloud system needs to allocate resources to handle any request submitted by a user, which can result in increased energy consumption.

In terms of request prioritization, scheduling, and serving, these activities can also impact the power usage of a cloud system. For instance, if the system is designed to prioritize certain types of requests or requests from certain users, it may allocate more resources to those requests, possibly resulting in increased energy consumption. GA's can be used to optimize a variety of problems, including job prioritization in clouds. Here is how GA's can be used for better prioritization of jobs in clouds [5]:

1. **Chromosome Representation:** In a GA, each solution is represented as a chromosome. For job prioritization in clouds, the chromosome can represent the order in which jobs are processed.
2. **Fitness Function:** The fitness of the chromosomes in a GA is checked by the fitness function. A fitness function can assess how a system is performing in terms of complying with Service Level Agreements (SLAs), and the usage of resources.
3. **Crossover and Mutation:** Two characteristic operations of a GA, namely, crossover and mutation, are applied to produce new enhanced offsprings. Crossover can be applied on two chromosomes to combine them together and result in a new chromosome, while the mutation operation introduces changes in genes selected randomly in a single chromosome.
4. **Selection:** The step of selection is applied by a GA on the current generation of chromosomes to select the fittest ones, based on their fitness function values, for the next generation.
5. **Iteration:** The preceding steps are applied by a GA iteratively on newer and newer generations until the current generation gives chromosomes that all fit. This iterative process can be used to reach the most optimum job schedule while maximizing energy efficiency.

Overall, a GA can be a strong mechanism for optimizing job scheduling in the cloud. Similarly, scheduling requests at specific times or in specific patterns could also impact the energy usage of the system.

Cloud job scheduling (CJS) is essentially the process of assigning computing resources to jobs in a cloud environment, depending on the availability of the resources. When jobs are assigned to the most appropriate resources, CJS can lead to an overall reduction in the computing resources required, which, in turn, results in higher energy efficiency and a reduction in

usage cost. This occurs as an outcome of the timely completion of jobs with maximum utilization of resources in the minimum amount of time [6].

The energy usage of a cloud and the scheduling of jobs on the cloud are linked to each other since allocating resources efficiently to jobs can result in an increase in energy optimization of the cloud. This is usually achieved through algorithms that considers factors such as resource availability, job priority, and estimated execution time.

Data centers and related infrastructure on the cloud use the maximum amount of energy for their day-to-day operations, producing a significant carbon footprint. Cloud providers have expressed a concern regarding the influence of the infrastructure on the atmosphere and therefore, are interested in optimizing the usage of the infrastructure. The traditional approach has been to apply methods that manage power dynamically, by reallocating the demand for power according to user requirements. For example, the power being consumed by a particular server currently not in use, can be reduced by putting it in sleep mode [7].

In cloud computing, serving of requests is closely linked to energy consumption. A cloud service provider (CSP) receives requests from users requesting for information to be sent back or for operation. These activities require energy which also incurs a monetary cost to the provider.

CSPs apply different techniques including virtualization, load balancing, and dynamic resource allocation to reduce energy consumption. Virtualization permits a single physical machine to run multiple virtual machines, leading to an overall reduction in energy consumption. Load balancing consists of distributing user requests to many servers with the intent of avoiding overloading any one server, which can result in a decrease in energy consumption. Dynamic resource allocation consists of providing resources to each request in a dynamic manner depending on the current workload, allowing the CSP to only use the required resources thus reducing energy consumption. Besides these techniques, some CSPs also use sources of renewable energy for their data centers, leading to a reduction in both their carbon footprint along energy costs. A combination of these procedures, when utilized for increasing energy efficiency, is known as green cloud computing. Thus, providing services to user requests must be carried out in an energy efficient along cost-efficient manner, ensuring a minimal influence on the environment [7].

Summing together the energy consumed by all the different components of a cloud system the total energy consumed by the system can be calculated. Using the value of the total energy consumed by the system, the durations in which high amounts of energy are consumed can be identified and energy consumption can be optimized using optimization algorithms. Further, using this information, the components that are consuming the maximum amount of energy might also be identified and alternative energy-efficient components can be utilized to replace them. This will lead to a reduced carbon footprint and lower costs.

**Proposed Energy-Efficient Cloud Load Balancing Model**

A cloud-based environment employs a technique called load balancing, which refers to the process of distributing work in the form of jobs and user requests to many servers or resources so as to enhance the usage of resources and reduce downtime.

Solutions based on hardware or software could be utilized to implement load balancing, at the stage of either the application, the network, or the server. CSPs include load balancing as a service in their platforms, aiding users and organizations to carry out load balancing in the cloud.

Load balancing in the cloud is essential for managing the balance between the energy used by the cloud and its optimum output. A proposed energy-efficient cloud load balancing model can be implemented using the below-given steps, as shown in Fig. 2:

1. **Selection of Virtual Machines (VM):** The appropriate VM needs to be selected initially based on the resources required by the job(s) which is(are) to be processed, and also whether the required resources are available [10].
2. **Consolidating jobs onto VMs:** This step consists of combining jobs onto fewer physical servers. It could be attained by selecting those VMs that are using fewer resources and allocating the jobs to these VMs.
3. **Dynamically Allocating Resources to VMs:** Having consolidated jobs, the next step is to allocate resources to VMs in a dynamic manner. For this to be accomplished, each VM needs to be monitored for identifying the resources being used by it and then adjusting the allocation of resources in a dynamic manner.
4. **Load Balancing:** This step involves redistributing the jobs to many VMs without overloading any single VM. It might be accomplished with the help of a suitable algorithm that will consider the availability of each VM and keep the count of the jobs to be allocated [11].

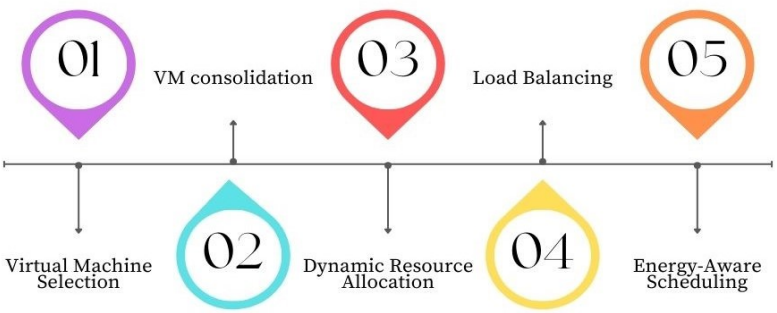


**Fig. 2**
**The Load Balancing Model**

5. **Scheduling Jobs with Energy-Awareness:** The last step consists of applying an energy-aware scheduling algorithm that considers the energy consumption of each VM, and then selects the VM with minimum consumption of energy.

The inclusion of an energy-aware algorithm in the process of scheduling jobs on the cloud will ensure the optimum consumption of energy, thereby reducing the carbon footprint.

The proposed load-balancing model prioritizes the incoming jobs using a GA. The redundant jobs are filtered out and then the jobs are assigned to the dispatcher. An input matrix is created by calculating the Habitat Suitability Index and the fitness value of each job using the GA-BBO algorithm. The fittest population is then allocated to VM's for execution. Virtual machines are allocated to the jobs according to their requirement that result in optimized throughput, least waiting time and maximum bandwidth utilization.

**Mathematical Model of the Cloud Energy Consumption**

The energy consumption in clouds can be modeled using various mathematical equations depending on the cloud infrastructure and the resources available for use. The following equations may be used for calculating the energy consumption[8][9]:

1. The power consumption of a server is given by Equation (1)

$$P = U \times I \tag{1}$$

where P is the power consumption in watts (W), U is the voltage in volts (V), and I is the current in amperes (A).

2. The total energy consumption of a server over time is described by Equation (2):

$$E = P \times t \tag{2}$$

where E is the energy consumption in watt-hours (Wh), P is the power consumption in watts (W), and t is the time in hours.

3. Equation (3) gives the Energy consumption of a virtual machine:

$$EC = P \times t \times \left(\frac{CPUutil}{100}\right) \tag{3}$$

Where EC is the energy consumption in watt-hours (Wh), P is the power consumption in watts (W), t is the time in hours, and $CPU_{util}$ is the percentage of CPU usage.

4. The total energy consumption of a cloud service is depicted in Equation (4) :

$$EC_i = \sum \left(P_i \times t_i \times \left(\frac{CPUutil_i}{100}\right)\right) \tag{4}$$

Where $EC_i$ is the total energy consumption in watt-hours (Wh), $P_i$ is the power consumption of the $i_{th}$ server in watts (W), it is the time the $i_{th}$ server

was active in hours, and $CPU_{util_i}$ is the percentage of CPU usage of the $i_{th}$ server. The summation is taken over all servers used by the cloud service.

5. Energy Efficiency (EE): Energy Efficiency is measured as the useful work done (throughput) per unit of energy consumed as given in Equation (5)

$$EE = \frac{Throughput}{Energy\ Utilization} \tag{5}$$

**Load Balancing using the Ant Colony Optimization (ACO) Algorithm**

The Load Balancing Model for a cloud can also be implemented using the ACO algorithm although the results of implementing the ACO have been presented in [12]. Ref. [12] also presents a comparative analysis of the results of ACO, BBO, GA-BBO, Grey Wolf Optimization (GWO) and Whale Optimization Algorithm (WOA). The steps of the ACO algorithm are given below.

**Algorithm 1. Load Balancing in a Cloud using ACO**

1. Initialize necessary parameters and pheromone trails;
   Initialize initial pheromone value $\tau_0$, global pheromone decay rate $\rho_g$, local pheromone decay rate $\rho_l$, pheromone sensitivity $\alpha$, visibility sensitivity $\beta$ [15].
2. Set initial pheromone value $\tau$ for each Resource Utilization Matrix (RTM) $M_k$ to $\tau_0$
3. Repeat
   a. Generate RTM population;
   b. Calculate global pheromone decay for each RTM

$$\tau = \left(1 - \rho_g\right) * \tau \tag{6}$$

The overall objectives of implementing the ACO are minimizing the cost of load balancing C and to minimize the energy consumption E. When applying ACO on each job the objectives are maximizing throughput as well as resource utilization, while minimizing response time.

Waiting time $w_J$ of job J is given by

$$w_J = t_{allocJ} - t_{arrivalJ} + L \tag{7}$$

where,

$t_{allocJ}$ = Amount of time consumed in allocating resources to the job J

$t_{arrivalJ}$ = Time at which job arrived

$L$ = Latency

Throughput $T_J$ of Job J is given by

$$T_J = \frac{n_{IJ}}{R_J}, where\ R_J < Max\ SLA\ limit \tag{8}$$

where,

$n_{IJ}$ = number of instructions I in job J
$R_J$ = response time required for completing job J
The expression used to calculate the Resource Utilization $U_J$ of a single job J, is

$$U_J = \frac{t_{usedJ}}{t_{availJ}} \times 100\% \quad (9)$$

where,
$t_{usedJ}$ = time for which the job uses the VM
$t_{availJ}$ = time for availability of VM for being used
The formula for the fitness function $fit_{ACO}$ to check for the fitness of an individual job J, as used in this algorithm for implementing energy-aware VM load balancing, is as follows:

$$fit_{ACO}\ for\ job\ J = \frac{T_J \times U_J}{w_J} \quad (10)$$

To provide for the impact of each of the four parameters determining the fitness function, namely, throughput, resource utilization, and waiting time, for minimizing load balancing cost, a weight w is allocated to each parameter.
Thus, the fitness function is modified to:

$$fit_{ACO} = \frac{(T_J \times w_T) \times (U_J \times w_U)}{(w_J \times w_w)} \quad (11)$$

where, $w_T$ = throughput weight
$w_U$ = resource utilization weight
$w_w$ = waiting time weight
Based on the above, now the fitness of each RTM is found using the expression for $fit_{ACO}$ in Eq. 11.

c. Find the optimal solution based on fitness value
d. Update pheromone trail;
   i. Use the probabilities to generate new solutions
   ii. Calculate initial probabilities for selecting each RTM based on pheromone amounts $\tau_i$, i=1, 2, 3, ...

$$P_i = \tau_i^{\alpha}, i = 1,2,3 \ldots \quad (12)$$

$$P_i = \frac{P_i}{\sum_{i=1}^{n} P_i} \quad (13)$$

   iii. Calculate the maximum probability $P_{max}$ out of all the RTMs and select the RTM $M_{max}$ with maximum probability.

$$M_{max} = M_i \mid \max(P_i), i = 1,2, \ldots, n \quad (14)$$

   iv. Update local pheromone of RTM $M_i$ by the equation

$$\tau_i = (1 - \rho_l) \times \tau_i + \rho_l \times \tau_0 \quad (15)$$

e. Go to Step 3 and repeat until the stopping criterion for the ACO is reached. If the stopping criterion is reached, return the RTM $M_{best}$ as the optimal solution.

The most elite individual, that is, $M_{best}$, will be the one that has the maximum resource requirement out of all $M_i$. This will be that matrix whose resource requirement $Q_{req}^{M_i}$ is above the threshold resource requirement $Q_T$, that is,

$$M_{best} = \max_i Q_{req}^{M_i} \mid Q_{req}^{M_i} > Q_T, \tag{16}$$

where, $i = 1, \ldots, r$

The process flow diagram of the stages of Load Balancing using the ACO algorithm is given in Fig. 3.

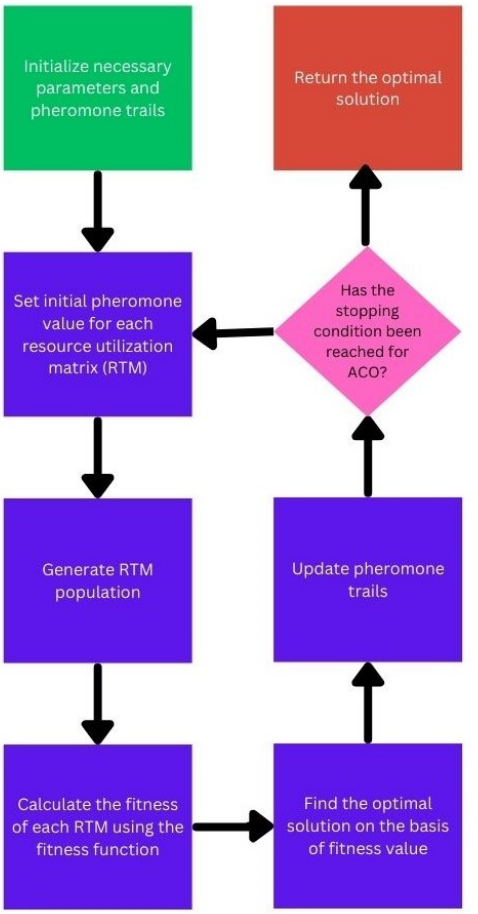


**Fig. 3**
**The Process Flow diagram of Load Balancing by ACO algorithm**

**Proposed GA-BBO Algorithm for Load Balancing in a Cloud**

Based on the above discussion, the following are the steps of the proposed GA-BBO algorithm:

**Algorithm 2: GA-BBO algorithm for Load Balancing in a Cloud**

1. **Population Initialization:** Create an initial population of q possible resource configurations (habitats), each represented as a Resource Utilization Matrix (RTM) $M_i$, where i=1,…..q.
2. **Habitat Suitability Index (HSI):** Find the fitness of each $M_i$ through calculation of its HSI.

   The objectives which have to be achieved by the collection of RTMs are: Load balancing cost C is to be minimized and energy consumption EC is also to be minimized.

   Each job J has to satisfy the following Suitability Index Variables (SIVs): Throughput, Response time, Resource utilization. $D_J = (T, U, w)$ is the matrix representing the SIVs for each individual job J in each M.

   Each individual RTM has to satisfy the following objectives: throughput and resource utilization have to be maximized, whereas response time has to be minimized.

   A set of RTM's M is given by

   $$M = (M_1, M_2, \dots\dots, M_q) \tag{17}$$

   Each $M_i$ is represented by a composition matrix $D_i$ for its SIVs given by

   $$D_i = (T_i, U_i, w_i) \tag{18}$$

   and

   $$H_{iT} = L_{iT} + random(P_{iT} - L_{iT}) \tag{19}$$

   $$H_{iU} = L_{iU} + random(P_{iU} - L_{iU}) \tag{20}$$

   $$H_{iw} = L_{iw} + random(P_{iw} - L_{iw}) \tag{21}$$

   where, $L_{iT}, L_{iU}, L_{iw}$ = each of the SIVs, that is, $(T_i, U_i, w_i)$ of each $M_i$, has these lower bounds,

   $P_{iT}, P_{iU}, P_{iw}$ = each of the SIVs, that is, $(T_i, U_i, w_i)$ of each $M_i$, has these upper bounds,

   $H_{iT}, H_{iU}, H_{iw}$ = the variables storing the values of the 3 SIVs of each $M_i$.

   Each individual M has its own value of the HSI and is given by:

   $$HSI_{GABBO}^{M} = (H_{iT}, H_{iU}, H_{iw}) \tag{22}$$

3. Repeat
   a. Using the immigration and emigration rates (BBO) apply and perform migration of various $M_i$.

      Calculate Migration Rates: Firstly, use the HSI value for each $M_i$ to find the immigration and emigration rates.

Each $M_i$ has an upper limit on the number of jobs it can perform. This number for a species is given by $C_{max}$, that is, the maximum species count. The number q of $M_i$'s is assigned to $C_{max}$.

$M_i$'s are then ordered according to their decreasing HSI values.

To identify the number of species in each $M_i$ the following expression is used:

$$C_i = C_{max} - i \tag{23}$$

To find the immigration rate $\lambda_i$ for each $M_i$ the following expression is applied:

$$\lambda_i = I(1 - \frac{C_i}{C_{max}}) \tag{24}$$

To find the emigration rate $\mu_i$ for each $M_i$ the following expression is applied:

$$\mu_i = D(\frac{C_i}{C_{max}}) \tag{25}$$

where, $I$ = maximum $\lambda_i$ into each $M_i$ and $D$ is the maximum $\mu_i$ from each $M_i$.

To immigrate from the source matrix $M_j$ the destination matrix $M_j$ is identified on the basis of probabilities. Once the values of $HSI_{GABBO}^{M}$ of each $M_i$ is replaced by the corresponding value of each $M_j$, the HSI values of each $M_i$ are updated.

b. Apply Roulette Wheel Selection (GA) to identify parents using the values of their $HSI_{GABBO}^{M}$.

c. Apply single-point crossover to generate offspring $M_i^{off}$ (GA).

d. Following this, perform mutation on the offspring (BBO).

Finding the Mutation Rate: Next, the mutation rate is found for each $M_i^{off}$ (habitat).

The value of 0.005 is assigned to the starting mutation probability $Pr_{mutate}^{init}$.

Then find the probability $Prob$ of the species count of each $M_i$ by following these steps:

Calculate the value of $Pr_{max}$, the maximum probability out of all values of $Prob$ for each $M_i$.

$$Pr_{max} = \max(Prob) \tag{26}$$

Apply the following expression for finding the the mutation rate of each species count:

$$R_{mutate} = Pr_{mutate}^{init} \times \left(1 - \frac{Prob}{Pr_{max}}\right) \tag{27}$$

All the $M_i$ are sorted in decreasing order with the best at the top.

Mutation Process: To create variations of the set of $M_i$ perform mutations on the $M_i$ while applying a certain probability. This operation is performed only on the worst half of this set.

To identify whether to apply mutation or not, it is checked whether $R_{mutate} > rnd$, where $rnd$ is a random value, by applying the following formula:

$$SIV_i^M = \left\lfloor SIV_{i_{min}}^M + \left(SIV_{i_{max}}^M - SIV_{i_{min}}^M + 1\right) \times rnd \right\rfloor \quad (28)$$

where i=1,2,3,…

e. Find the value of $HSI_{GABBO}^{M_i^{off}}$ of each $M_i^{off}$ to check their fitness.

f. Identify and keep the best or elite individuals in each $M_i^{off}$ while replacing the remaining individuals in the population (GA and BBO).

**HSI Update:** Apply equation 17 to 22 on the old HSI, namely, $HSI_{GABBO}^{M}$, to find the new HSI, that is, $HSI_{GABBO}^{M_{new}}$ for each $M_i$.

**Elitism:** Keep the most elite $M_i$ (those having the best HSI values) and replace the remaining with those with the updated HSI. This is done by checking if $HSI_{GABBO}^{M} < HSI_{GABBO}^{M_{new}}$ and using the more elite $M_i$, which have a higher HSI to replace the older $M_i$.

The number of most elite $M_i$, represented by the Elitist Retention Count maintained at 2.

4. Return the most elite individual $M_i$ as the optimal solution.

The most elite individual, that is, $M_{best}$, will be the one having the highest resource requirement out of all $M_i$. This will be that matrix whose resource requirement $Q_{req}^{M_i} > Q_T$, the threshold resource requirement, that is,

$$M_{best} = \max_i Q_{req}^{M_i} \mid Q_{req}^{M_i} > Q_T, \quad (29)$$

where, i=1, …, r

Fig. 4 presents the flow diagram of the GA-BBO Algorithm for Load Balancing.

**Research contribution of the proposed work**

The proposed GA-BBO algorithm will help to boost the energy-efficiency of the system of job scheduling on the cloud. The algorithm will consider the multiple factors of high throughput, minimum response time, and maximum resource utilization while aiming for the dual objectives of minimizing the cost of load balancing and minimizing the energy usage of the cloud infrastructure. Thus, the GA-BBO algorithm will be able to improve the task of job scheduling in terms of VM assignment and allocation of resources, which will in turn improve energy efficiency.

**Description of dataset**

The data set has been taken from Hadoop Cluster logs of Carnegie Mellon University Parallel Data Lab. The data set includes 51975 successful jobs, 4614 failed jobs and 1762 killed jobs from 78 users [16]. The various fields are job configuration, job history, task history, task attempts, counters, and splits. The data set is formatted as a .csv file and organized month-wise. The data logs were analyzed using MATLAB.

**Results and Discussion**

The graph in Fig. 5 clearly shows that the GA-BBO algorithm significantly improves the throughput of tasks submitted to the cloud environment scheduling system. As the number of submitted jobs increases the throughput improves considerably. The box plot in Fig. 6 depicts a low median value (red line within the box) of the scheduling cost with very few outliers. The graph in Fig. 7 illustrates the trend in scheduling costs as the number of submitted tasks rises. Initially, when fewer tasks were submitted, scheduling costs were higher but decreased significantly as the task volume increased. Therefore, scheduling costs are substantially reduced when the system operates at full capacity.

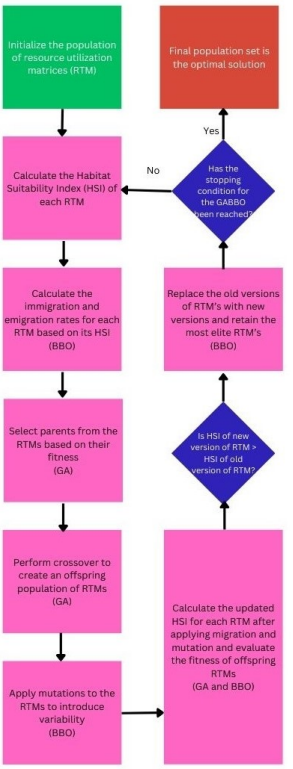


**Fig. 4**
**The Flow Diagram of the GA-BBO Algorithm for Load Balancing in a Cloud**

## Conclusions

Energy efficiency in cloud job scheduling helps to optimize the overall performance of a cloud infrastructure. Towards this goal, nature-inspired algorithms are effective since they give simple solutions to complex job scheduling tasks in a cloud infrastructure.

This research paper proposes to enhance energy efficiency and improve the task of load balancing by applying the GA-BBO algorithm, which is a combination of a GA and the nature-inspired BBO algorithm.

The GA-BBO algorithm combines the steps of both algorithms and thereby attempts to solve the problem of finding an optimal energy-efficient solution for the task of job scheduling. The GA-BBO algorithm gave good results in terms of the cost of scheduling and energy consumption for the set of jobs in the dataset.

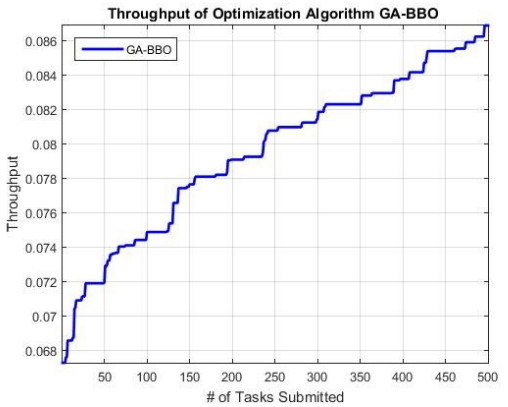


**Fig. 5**
**Throughput of GA-BBO Algorithm**

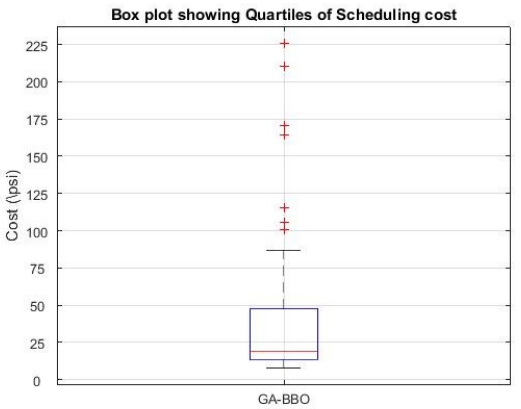


**Fig. 6**
**Box Plot Showing Quartiles of Scheduling Cost**

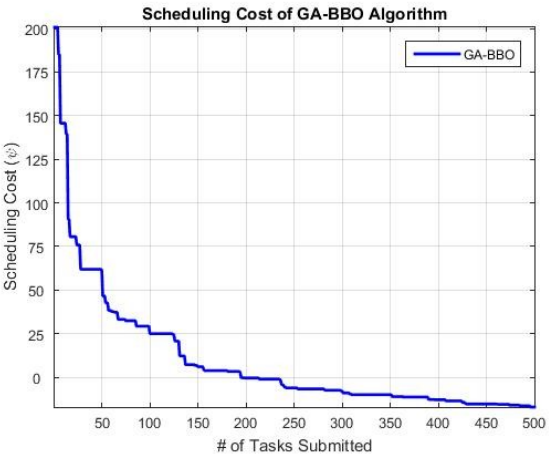


**Fig. 7**
**Scheduling Cost of GA-BBO Algorithm**

Equation (4) describes that energy consumption of a cloud environment depends on various factors – CPU utilization being the major contributor. Equation (5), on the other hand, gives a tradeoff between energy efficiency, throughput, and energy consumption. Though increasing throughput generally requires more energy; however, energy consumed per unit of throughput decreases as the system operates at a higher load. In a cloud computing environment, servers continue to consume energy even when they are idle. As the tasks are processed and throughput increases, energy efficiency is improved since energy cost per task decreases. Graphs in Fig. 5 and Fig. 7 show the trends of the substantial increase in throughput and the steep decline in scheduling cost, respectively, using the GA-BBO optimization algorithm as the number of tasks submitted is increased and the cloud operates at a high capacity, thus making the overall system more energy efficient.

## Future Scope

With the growing impact of computing devices and cloud infrastructure on the environment, the need for energy efficiency in their operation is also becoming imminent. In this direction, there is a clear need for improving the energy efficiency of these infrastructures to reduce the impact on the environment.

To reduce the carbon footprint of computing and cloud infrastructure, the future holds the use of renewable and clean energy sources for powering them. The algorithm can also be combined with edge computing to manage the distance that transmitted data needs to cover between servers, thereby further improving energy efficiency. These improvements will not only help improve the efficiency of managing load balancing but also reduce the energy consumption of the cloud infrastructure.

**References**

[1] S. K. Panda and P. K. Jana, "An energy-efficient task scheduling algorithm for heterogeneous cloud computing systems," *Cluster Comput.*, vol. 22, no. 2, pp. 509–527 (2019).

[2] D. Jiang, Y. Wang, Z. Lv, W. Wang, and H. Wang, "An energy-efficient networking approach in cloud services for IIoT networks," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 5, pp. 928–941 (2020).

[3] A. Ali, M. M. Iqbal, H. Jamil, F. Qayyum, S. Jabbar, O. Cheikhrouhou, and F. Jamil, "An efficient dynamic-decision-based task scheduler for task offloading optimization and energy management in mobile cloud computing," *Sensors,* vol. 21, no. 13, p. 4527 (2021).

[4] M. Ghobaei-Arani, "A workload clustering-based resource provisioning mechanism using biogeography-based optimization technique in cloud-based systems," *Soft Comput.,* vol. 25, no. 5, pp. 3813–3830 (2021).

[5] Z. Ali, L. Jiao, T. Baker, G. Abbas, Z. H. Abbas, and S. Khaf, "A deep learning approach for energy-efficient computational offloading in mobile edge computing," *IEEE Access*, vol. 7, pp. 149623–149633 (2019).

[6] R. Garg, M. Mittal, and L. H. Son, "Reliability- and energy-efficient workflow scheduling in cloud environment," *Cluster Comput.*, vol. 22, no. 4, pp. 1283–1297 (2019).

[7] D. Ding, X. Fan, Y. Zhao, K. Kang, Q. Yin, and J. Zeng, "Q-learning-based dynamic task scheduling for energy-efficient cloud computing," *Future Gener. Comput. Syst.*, vol. 108, pp. 361–371 (2020).

[8] Y. Sharma and S. Lakra, "Green cloud job scheduling and load balancing using hybrid biogeography-based optimization and genetic algorithm," in Micro-Electronics and Telecommunication Engineering, Proc. 3rd ICMETE, Singapore: Springer, pp. 185–195 (2019).

[9] R. Medara and R. S. Singh, "Energy-efficient and reliability-aware workflow task scheduling in cloud environment," *Wireless Pers. Commun.*, vol. 119, no. 2, pp. 1301–1320 (2021).

[10] J. Praveenchandar and A. Tamilarasi, "Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing," *J. Ambient Intell. Humaniz. Comput.,* vol. 12, pp. 4147–4159 (2021).

[11] A. Marahatta, S. Pirbhulal, F. Zhang, R. M. Parizi, K.-K. R. Choo, and Z. Liu, "Classification-based and energy-efficient dynamic task scheduling scheme for virtualized cloud data centers," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1376–1390 (2019).

[12] Y. Sharma and S. Lakra, "A comparative study of cloud job scheduling algorithms with a hybrid genetic and biogeography-based optimization algorithm," *communicated* (2025).

[13] H. A. Younis, I. M. Hayder, I. S. Seger, and H. A. K. Younis, "Design and implementation of a system that preserves the confidentiality of stream cipher in non-linear flow coding," *J. Discrete Math. Sci. Cryptogr.*, vol. 23, no. 4, pp. 1409–1419 (2020), doi: 10.1080/09720529.2020.1714890.

[14] I. Cherkaoui and F. Zinoun, "On the use of Egyptian fractions for stream ciphers," *J. Discrete Math. Sci. Cryptogr.*, vol. 26, no. 1, pp. 139–152 (2023), doi: 10.1080/09720529.2021.1923921.

[15] Y. L. Liu and F. Gomide, "A participatory search algorithm," *Evol. Intell.*, vol. 10, no. 1, pp. 23–43 (2017).

[16] OpenCloud, "Hadoop cluster trace: Format and schema." [Online]. Available: https://ftp.pdl.cmu.edu/pub/datasets/hla/dataset.html

*Received March, 2025*