BULGARIAN ACADEMY OF SCIENCES

CYBERNETICS AND INFORMATION TECHNOLOGIES • Volume **23**, No 1
Sofia • 2023 Print ISSN: 1311-9702; Online ISSN: 1314-4081
DOI: 10.2478/cait-2023-0009

An Energy-Aware QoS Load Balance Scheduling Using Hybrid GAACO Algorithm for Cloud

Arivumathi Ilankumaran, Swathi Jamjala Narayanan

School of Computer Science and Engineering, Vellore Institute of Technology, Vellore, Tamilnadu, India
E-mails: arivumathi1202@gmail.com jnswathi@vit.ac.in

***Abstract*:** *In recent days, resource allocation is considered to be a complex task in cloud systems. The heuristics models will allocate the resources efficiently in different machines. Then, the fitness function estimation plays a vital role in cloud load balancing, which is mainly used to minimize power consumption. The optimization technique is one of the most suitable options for solving load-balancing problems. This work mainly focuses on analyzing the impacts of using the Genetic Algorithm and Ant Colony Optimization* (*GAACO*) *technique for obtaining the optimal solution to efficiently balance the loads across the cloud systems. In addition to that, the GA and ACO are the kinds of object heuristic algorithms being proposed in the work to increase the number of servers that are operated with better energy efficiency. In this work, the main contribution of the GAACO algorithm is to reduce energy consumption, makespan time, response time, and degree of imbalance.*

***Keywords*:** *Load balancing; energy consumption; Genetic Algorithm; Ant Colony Optimization Algorithm; cloud computing.*

1. Introduction

Due to the recent development of computer networks, devices, and high-speed internet, cloud services are becoming more popular in many businesses and other applications. In this system, the process of resource sharing has been improved with the integration of different services, and scheduling techniques along with energy minimization[1-3], which are mainly used to improve the efficiency of a data center. Due to these facets, the growing popularity of cloud services has drastically increased, where the users are highly required to have advanced systems and fast networks for connecting with cloud data centers across the globe. In this domain, the utilization of energy is one of the most essential factors that need to be concentrated on for improving the overall efficiency and performance of the cloud system. This objective can be accomplished by using cloud server power management techniques. Recently, many research works [4] have been accomplished with some alternate algorithms for reducing cost consumption, energy consumption, and greenhouse gas emission. According to recent survey reports, it is analyzed various heuristic models

are developed for improving the process of resource allocation with minimized software/hardware requirements. Among the other technologies, the virtualization concept has been widely used in many cloud computing applications for obtaining minimal energy consumption [5, 6]. When compared to the other techniques, the metaheuristic models [7, 8] are considered one of the most suitable options for minimizing energy usage. Due to its simplicity and ease of implementation, it is considered one of the successful mechanisms for solving cloud load balancing problems. Beyond the advantages of service consumers, cloud service providers, and economic organizations also have an increased option to utilize the metaheuristic models, since it reduces or eliminates infrastructure maintenance expenses.

Based on the requirements of customers [9, 10], cloud providers could deliver reliable services with the help of cloud-based applications for commercial operations, which may be sensitive to the users' data. In many cases, the service level agreement provides an ensured quality of services to the customers according to their demands. Here, virtualization is mainly used to increase the performance of cloud systems and is more popular in the last few decades [11, 12]. It allows customers' applications to run more efficiently by moving virtual machines between different hosts without disrupting the services [13]. As the customers' needs change, it provides various services to them based on their demands. Because, only the customers pay for what they use, and cloud computing allows users to access their data from anywhere globally [14] by transferring it to the cloud (the Internet) from their personal computers. Moreover, the consumers are not required to have high-capacity processing and storage systems, since all cloud computing activities are handled by the well-equipped and advanced servers in the cloud service provider sector [15].

Cloud accounting [16] is one of the extensively used terms in today's business world. Because most organizations and business sectors choose service plans according to their requirements and conditions. Also, the cloud offers unlimited resource capacity at a reasonable cost, which has the following unique factors [17, 18]: CPUs, memory, input/output network, and disc. Then, the organizations can provide the opportunity to host their apps on the cloud, which allows for the eventual removal of structural resources, such as prior ultra-structural resources, that lasted for many months. As for cloud service providers, it is essential to assure whether the clients' expectations are fulfilled or not [19-21].

The main objective of this work is to store the bits in a cloud environment by constructing the virtual network infrastructure, which helps to even out network pressure precisely as in the phase map and radius produced in big data. For this purpose, the virtualization and load balancing methods have been utilized in this work that could significantly improve the performance of the entire cloud system with the efficient allocation of resources [22]. Although, resource scheduling and request management are utilized in conventional works for load balancing in the cloud, but are limited to the key problem of increased energy consumption. Typically, energy optimization is one of the important goals that need to be addressed in the cloud system, hence this paper intends to provide a new solution for solving the load balancing problem with the help of a hybrid GAACO-based heuristic model. Here, the energy-performance tradeoff [23, 24] can be evaluated for a physical machine

with computational capability at its peak. When compared to the other conventional techniques, the proposed hybrid GAACO technique can efficiently balance the loads across the machines in the cloud with reduced energy consumption, makespan time, response time, and degree of imbalance.

2. Literature review

This section examines the working principles and characteristics of the existing heuristic and meta-heuristic models used for virtualization and load balancing in cloud systems. In [25] a modified ACO method is utilized for improving the job scheduling process in the cloud and it is a kind of random method in which all ants appear on as processors. Then, the ants produce pheromones for determining whether or not they are allowed to migrate. In addition to that, the market-based resource management scheme is utilized in many fields for managing resources across various systems. In [26] the multi-tiered web application issue is intended to be solved by using virtualized heterogeneous systems. Generally, if the number of nodes is limited, it is quite difficult to find a suitable way for optimizing the controller, which renders it inappropriate for large-scale issues in the real world [27].

The researchers have explained that the resource allocation is based on priorities while virtualizing several applications on a cluster. Authors have suggested a simple method to find a solution for solving this problem. Also, the bin packing problems are resolved by using the heuristic models with variable costs and sizes. The energy-efficient use of virtual machines in High-Density Cluster (HDC) settings has been described in [9]. Generally, the VM has implemented max, min, and shares parameters to indicate the maximum and minimum CPU allocation rate per resource. This technique is the most suitable option in the private cloud or corporate setting. A team [8], has analyzed the possibility of linking several VMs for enhancing the communication efficiency between them. However, they have found that the primary goal has not been to save energy.

The genetic algorithm process that shapes all living things may be harnessed by algorithms, based on the natural world's genetic algorithm. In order to use a genetic algorithm, you will need a specific solution domain (chromosome) to be represented, along with a fitness function to judge that solution domain. Solutions with optimized genetic attributes may be discovered via selective breeding and evolutionary algorithms (called individuals). The scheduling issue in a workshop may be solved using genetic algorithms, which use variations of natural selection. They are used as extensions when used to describe heterogeneous systems, grid computing, and cloud computing. The estimates in this research often presume that each job will take a specific duration (homogeneous system).

An experimental study has been done on the task scheduler in a distributed heterogeneous computing environment to see whether it could execute tasks across different sets of heterogeneous computing resources. Researchers have looked at heterogeneously distributed systems to see how efficient a high-performance computing system with load-balancing and a central server is. A central scheduler has been suggested to make load-balancing decisions using a genetic algorithm to

distribute the load. Using genetic algorithms and task network representation has increased the dependability of distributed computing systems.

Changing the frequency and voltage of a server's CPU is not adequate because it only consumes around a third of the total energy. Physical Machines should utilize energy in proportion to the amount of work they accomplish, as shown in Table 1.

Table 1. Energy consumption of various CPUs with resource utilization [31]

Performance			Power	Performance to power ratio
Target load	Actual load	ssj_ops	Average power (W)	
100%	99.8%	190,234	119	1601
90%	90.7%	172,967	116	1494
80%	80.8%	154,130	112	1380
70%	69.7%	132,811	106	1251
60%	60.8%	115,866	99.8	1161
50%	49.6%	94,582	90.9	1041
40%	39.7%	75,792	82.5	919
30%	29.8%	56,857	74.4	764
20%	19.9%	37,980	68.2	557
10%	10.2%	19,410	60.8	309
Active Idle		0	56.7	0
∑ssj_ops / ∑power				1064

A prolonged 100% CPU or memory use may result in substantial degradation of the system performance. Most servers can handle 70-80% server load or memory without deteriorating performance, whereas host servers also handle 90% for the whole computer, host energy consumption changes with CPU usage. As a result, for improved energy efficiency, the CPU use rate should be raised.

If we use the cloud computing infrastructure, it is often possible to use and deploy service-oriented applications. Server businesses or data centers make cloud computing services accessible to their customers by providing their servers. Setting up your cloud computing involves provisioning highly efficient computers and large quantities of storage since cloud computing services are in constant demand for computations and colossal data. Power in data centers is primarily derived from the resources, often paired with air conditioning and cooling systems. Data centers use more electricity than all of Europe. Green data center design requires efficient technology since substantial energy usage in data centers consumes a lot of energy. Cloud data centers may help reduce total energy use by helping to lower both idle and active server energy usage using virtualization, where resources may be consolidated and shared on a single server. Each physical machine adds up to the total processing power of a Cloud data center. Data centers that host cloud services utilize virtualization technologies to allocate resources as needed. SaaS, PaaS, and IaaS are the three degrees of cloud access available to consumers. From client to client, the job that the customer has set out might be vastly different [28-30].

Cloud entities are independent and self-interested to attain individual and communal goals, yet they are prepared to share their resources. Due to the decentralized structure of the system, the scheduling choice is difficult in such an open setting. There are particular criteria and goals for each organization. By consolidating servers on a single physical server, data centers can reduce their energy

consumption. As a result of the virtual machine concept, you may operate many servers on one physical server [31]. Some people refer to the task consolidation issue as the server/workload consolidation issue. In order to save energy, all computer resources are fully used, and virtual machines are dispersed [32].

Professorial and business debates on energy use in cloud data centers are raging right now. As a cloud service provider, it is also essential that you fulfill your clients' Quality-of-Service criteria (QoS). When scheduling tasks, maintaining a high quality of service (QoS) remains a challenge [36].

It is studied in this review that the existing load balancing models in the cloud are highly concentrated on scheduling, the resources across various VMs in the server based on the factors of priority, execution time, the current status of machines, CPU usage, memory consumption, and response time. However, it fails to analyze the energy level of VMs that are executing the tasks in a cloud server, because the increased level of energy utilization creates the problems of high delay in the process, increased response time for the given tasks, slow process, and inefficient task completion. These factors could degrade the performance and efficiency of the entire cloud load-balancing system. In order to solve these issues, this research work objects to develop an efficient and intelligent hybrid heuristic optimization methodology for perfectly allocating the resources across the VMs on the cloud system by finding the best optimal solution. For this purpose, the most extensively used heuristic optimization techniques such as GA and ACO are incorporated together for improving the overall load-balancing process in the cloud. Moreover, the hybrid GAACO identifies the best fitness function for optimally allocating the resources among various VMs based on the following parameters: response time, degree of imbalance, makespan time, and minimizing energy level. These features are mainly considered in the proposed load-balancing system for improving the process of scheduling with ensured energy efficiency.

3. Proposed method

This section presents a detailed description of the proposed methodology with its appropriate mathematical illustrations and flow chart representations. The main contribution of this paper is to develop a hybrid heuristic methodology named, Genetic Algorithm integrated with Ant Colony Optimization (GAACO) for efficiently allocating the resources across various machines in cloud server in order to execute the given task. When compared to the other heuristic models, the GA and ACO are extensively used in many multi-objective optimization systems for providing a suitable solution to solve the problem, hence, this research work intends to incorporate these two technologies for attaining these benefits. Here, the novel contribution of using the GAACO technique is that it provides the optimal solution for balancing the loads across the cloud systems by estimating the best fitness function. Moreover, energy is one of the most essential parameters that need to be considered for developing an efficient load-balancing system. Hence, energy-efficient scheduling and optimal load balancing are mainly concentrated in this work, which is accomplished by the use of the hybrid GAACO technique.

In this environment, both the VMs and PMs in the cloud server are partitioned according to the demands on the resources, where the factors such as CPU utilization rate, memory usage, and time have been considered before scheduling the tasks to the cloud machines. Then, the average response time of each machine that is executing the tasks is calculated for analyzing its energy level. In addition to that, this work considers the QoS parameters for ensuring increased energy efficiency and better scheduling of loads. After estimating these parameters, the hybrid GAACO model is deployed for scheduling the tasks to the appropriate resources based on the best fitness value, which improves the entire performance of cloud load balancing with the ensured energy efficiency of resources. Both GA and ACO find the better solution even working separately. In our proposed hybrid algorithm, GAACO they work simultaneously and get the best optimal solutions, while GA finds the fitness function values, tasks may get dropped sometimes due to overload, task failure, etc., the dropped tasks will run simultaneously on the ACO algorithm, and a better solution is found. So the proposed GAACO provides the following benefits: increased energy efficiency, optimal resource allocation, requires a minimum of iterations to identify the best fitness value for job scheduling, and better performance outcomes.

3.1. System model

Normally, the cloud data centers comprise many physical machines, which are partitioned into several VMs across the cloud. Then, this partitioning is done based on the measures of resource demand, memory, CPU, and time. This multi-objective function ensures that the essential resources are effectively used in cloud environments. This is done by moving the virtual machine of the overcrowded PM to another PM that is not overloaded while analyzing the QoS and energy consumption [33].

3.2. Energy model

The main focus of this research work is to obtain an increased efficiency of resources. The goal of the energy model is to maintain a high level of energy efficiency to meet society's needs. A more critical aspect of energy modeling is using the least energy while maximizing resource utilization [34]. Below is a power-based energy model for the system:

(1) $$P = aCv^2f.$$

Voltage *v*, capacitance load *C*, clock frequency *a*, and activity factor *f*, which shows the number of switches each clock cycle, are the essential variables for power usage. Because of this, power reduction may be influenced by a decrease in supply voltage, as shown in the equation above. *vf* and *f*, which are directly linked to frequency (*vf*) mean that power is calculated as $P = aCv^3$ and,

(2) $$E = P \times T,$$

where *E* is the total Energy. The average response time taken by the tasks is indicated as *T*. The main thing of the proposed load balancing strategy is to assign the *p* number of processors to *n* jobs. As a result of the suggested schedule, it is hoped that both the makespan and energy consumption *E* would be decreased [35]:

(3) $$E = aCv^3 \sum_{i=1}^{n} \overline{W}\ (T_i) + \sum_{i=1}^{n} \bar{C}\ (T_i),$$

where $\bar{W}\ (T_i)$ is the Waiting time of Tasks T_i, and $\bar{C}\ (T_i)$ is the Communication time of T_i.

3.3. QoS model

System components and symbols utilized in the QoS model are defined as follows: let consider *N* number of the Virtual Machines (VMs) existing in a cloud data center to calculate the QoS response time QoS_RT_i and QoS throughput QoS_TP_i of all tasks in *N* number of VMs.

Then, the QoS Response Time RT is calculated as follows:

$$\text{QoS}_{\text{RT}_i} = \frac{\text{RT}_i}{\text{RespTime}}, \tag{4}$$

where RT_i is the response time, and RespTime is the Average response time. The average response time could be calculated as

$$\text{RespTime} = \sum_{i=1}^{N} \frac{\text{RT}_i}{N}, \tag{5}$$

where *N* is the Number of Virtual Machines. QoS throughput may be denoted like the formula:

$$\text{Qos}_{\text{TP}_i} = \frac{\text{Throughput}}{\text{TP}_i}, \tag{6}$$

where Throughput indicates the average Throughput and TP_i defines the throughput. Average Throughput could be calculated as

$$\text{Throughput} = \sum_{i=1}^{N} \frac{\text{TP}_i}{N}. \tag{7}$$

Then calculate the objective function of QoS model Q_i using the formula as follows:

$$Q_i = \lambda_1 \times \text{Qos}_{\text{RT}_i} + \lambda_2 \times \text{Qos}_{\text{TP}_i} + \beta \times E, \tag{8}$$

where λ_1, λ_2, β are weight factors (*w*) of QoS model, *w* is from 0 to 1.

Hence, it is stated that the QoS model will efficiently improve resource utilization by minimizing the response time and throughput while allocating the tasks to *N* number of VMs.

3.4. Load balance model

Let consider, the following example: the case where the resources have a dimension of *d* and the resources of each provider *i* may be represented using the following formula. In collecting applications arriving at a specific time slot, P_j is the value of VM_j. The vector $\overrightarrow{r_{ij}}$ representing the resources utilized by *j* while executing on provider *i* [37, 38]. In the real world, technology will almost always only be developed for one provider. Therefore, assume that every application cannot be further divided. The cloud model captures the value of an application as soon as a successful application is performed on a provider's platform. While each service provider's resource capacity limits the scheduling goal, the overall goal is to maximize the cloud model profits. Accordingly, the scheduling problem, P_j can be formulated in the following manner. The Fitness Function Value (FFV) of GA is given below:

$$\text{Maximize } \sum_{j=1}^{n} (P_j \sum_{i=1}^{m} x_{ij} + Q_i), \tag{9}$$

subject to

$$\sum_{j=1}^{n} \overrightarrow{r_{ij}} x_{ij} + Q_i \leq \overrightarrow{c_i}, \ i = 1, 2, \ldots, m,$$

$$\sum_{j=1}^{n} x_{ij} + Q_i \leq 1, \qquad j = 1, 2, \ldots, m,$$
$$x_{ij} + Q_i \in \{0, 1\}, \qquad j = 1, 2, \ldots, n,$$

where Q_i is the Quality of service, and $\sum_{i=1}^{m} x_{ij}$ is the capacity of each task i in VM. It means that Equation (9) is NP-hard, multidimensional knapsack problem, where the applications often utilize the VMs, r_{ij}^k for gaining the successful QoS [39, 40]. It also guarantees that all VMs utilize the same energy, preventing instances when specific VM with large loads run out of power and leave the system. Load of the k-th dimensional resource of a given provider is defined as follows:

$$L_{ik} = \frac{\sum_{j=1}^{n} (r_{ij}^k (x_{ij+Q_i}))}{c_i^k}, \tag{10}$$

where c_i^k is the Average capacity of tasks. Load L_i is defined as the mean value of all its d-dimensional task's loads; that is,

$$L_i = \frac{\sum_{k=1}^{d} L_{ik}}{d}. \tag{11}$$

3.5. Genetic Algorithm for task scheduling

A natural resource allocation technique assigns the work to available VMs across hosts. When working with the load balancing issue, we can use heuristic techniques, which rely on genetic algorithms to explore the exponential solution space [41]. It uses an objective function (genetic) to choose a single solution from the population [42].

A random beginning population of POP-SIZE (POPulation SIZE) people has been produced, and their fitness values have been calculated to create a new population from scratch to the destination [43, 44].

It verifies whether the termination condition is satisfied in looping sections. Upon starting, the program generates a random solution and calculates its fitness score. The looping, cross-over, and mutation algorithm three techniques are assumed to be employed after the first method.

3.6. Encoding and decoding

Each chromosome should represent a different scheduling strategy. This paper examines the topic using an indirect encoding approach. Each resource-consuming task is encoded so that it may be tracked and identified. For the number of sub-tasks, multiply chromosome length by 1. The value of Gene-bit reflects the amount of utilized resource in Equation (9), and every bit location in Gene-bit shows how many gene sub-tasks are presently being executed.

- **Fitness function.** Using Equation (9), determines the time of a particular job in fitness function,

$$\text{Maximize } \sum_{j=1}^{n} (P_j \sum_{i=1}^{m} x_{ij} + Q_i).$$

- **Initial population.** A random population of individuals is produced using the method. Each individual is said to be a chromosome. The chromosomes will give the best solution to the problems.

- **Crossover.** In this model, the adaptive crossover techniques are also utilized to possibly prevent the early occurrence by increasing the crossover chance. The algorithm's last phase reduces the crossover probability with increased speed-up convergence, and simple to implement excellent individuals and generate new good individuals.
- **Mutation.** From 1 to 0, and from 0 to 1 are examples of single point mutation used in this work to alter individual bits in groups with reduced likelihood. The fitness value of the function is smaller than the average after multiple recursive cycles. The chromosome randomly chooses a gene and inverts its value based on the mutation operation. As a result of its removal, ACOs are granted based on the group's best solution.

3.7. Combined GA and ACO or GAACO algorithm

In this model, the chromosomal population is estimated according to the evolutionary rates of consecutive dates, and five generations. Here, the persons are elected according to the population-based fitness function values, then 10% of them are chosen as an optimization solution and then convert into beginning pheromones after the genetic algorithm is complete. Once the genetic algorithm is finished, it may be terminated $T_i^G(t)$, and the ACO can be entered,

(12) $$T_i^G(t) = \rho S_n,$$

where ρ means self-set constant and S_n are the genetic algorithm's optimization solutions. We can determine the distribution of pheromones by using a genetic algorithm.

1. Pheromone updating. The resource pheromone's initial value is set in (13). Assume that the pheromone value on application i at time t is (t); then at the next update time t', the value is updated to $\tau_i(t')$:

(13) $$\tau_i(t') = \delta\ (1-p)\tau_i(t) + \Delta\tau_i(t,t') + T_i^G(t)1,$$

δ is the termination condition of ACO. When the cycle counter N reaches the maximum number of iteration's i, range$(1-p)$, the current value is the optimal scheduling scheme, and then the ACO terminates. Where $0 < \rho \leq 1$ is a coefficient which represents pheromone evaporation and $\Delta_i(t,t')$ is the pheromone value increment.

2. **Pheromone increment.** The pheromone value increment, $\Delta\tau_i^j(t,t')$ obtained from all the ants' partial solutions; that is,

(14) $$\Delta\tau_i^j(t,t') = \sum_{j=1}^{q} \Delta\tau_i^j(t,t'),$$

where q is the number of ants and $\Delta\tau_i^j(t,t')$ is the pheromone value laid on task i and VM$_j$ with ant's partial solution at the time $(t,\ t')$ and is defined as

(15) $$\Delta\tau_i^j(t,t') = \begin{cases} G\left(f\left(\widetilde{S}_j(t')\right)\right) & \text{if } j\text{-th ant incorporates application } i, \\ 0 & \text{otherwise,} \end{cases}$$

where $\widetilde{S}_j(t')$ is the partial solution of ant j at time t' and $f\left(\widetilde{S}_j(t')\right)$ is the value of the evaluation function of this solution. To maximize the profit p_k, the total value of the VM should belong to$\left(\widetilde{S}_j(t')\right)$. The function G is defined

$G\left(f\left(\widetilde{S}_j(t')\right)\right) = Q_i\left(f\left(\widetilde{S}_j(t')\right)\right)$, in which Q_i is a parameter of the method. The evaluation is defined as

(16) $$f\left(\widetilde{S}_j(t')\right) = \sum_{k\in\widetilde{S}_j(t')} p_k.$$

3. Routing rule. After obtaining the pheromone value, select the scheduled VM_j according to the equation. The best optimal solution of ant h can be calculated by,

(17) $$P_h^j(t) = \begin{cases} \dfrac{[\tau_h(t)^\alpha]\left[\left(S_j(t)\right)\right]^\beta}{\sum_{k\epsilon \text{allowed}_j(t)}[\tau_k(t)]^\alpha\left[\left(S_j(t)\right)\right]^\beta} & \text{if } h \in \text{allowed}_j(t), \\ 0 \quad \text{otherwise,} \end{cases}$$

where allowed$(t) \subseteq S - \widetilde{S}_j(t')$ is the set of the remaining schedulable VMs. The above equation shows that the more pheromone value $\tau_h(t)$ as a VM, the higher probability of α, β will be scheduled. A VM can be selected, but more than one provider $\tau_k(t)$ has the resources to do it. Be aware that in most hybrid methods, the possibility of VM scheduling is only taken into account.

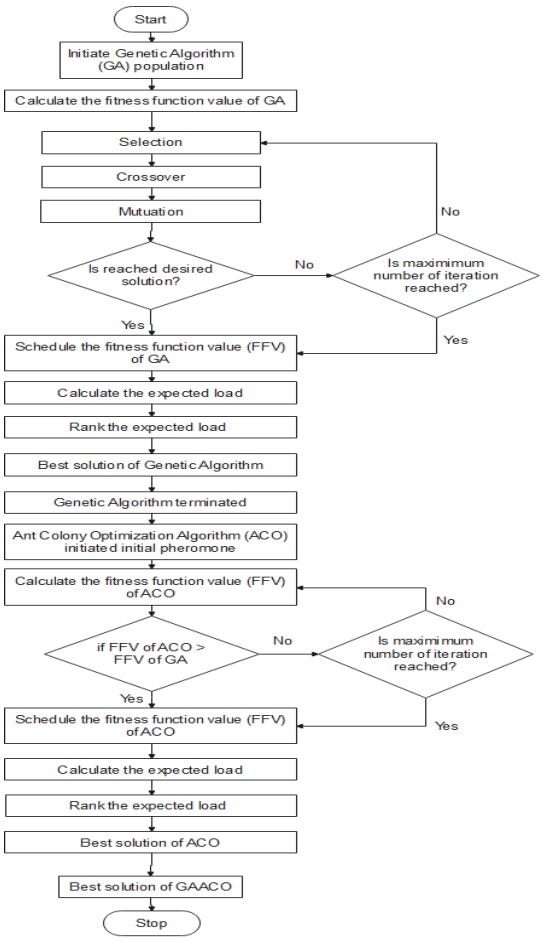

Fig. 1. The GAACO algorithm of flow chart

Assume the dimension of the resources is d and each provider's capacity can be expressed c_i^k is the k-th dimensional resource that the provider i has. The resources utilized by VM_j when executed on provider i is denoted as r_{ij}^k. For some particularly feasible provider i, the load of its k-th dimensional resource is L_{ik} as defined in the Equation (10). After adding VM_j, the expected load of its k-th dimensional resource is

$$L'_{ik} = L_{ik} + \left(\frac{r_{ij}^k}{c_i^k}\right). \tag{18}$$

The expected load of provider i is defined as

$$L'_i = \frac{\sum_{k=1}^{d} L'_{ik}}{d}. \tag{19}$$

This means that if j is executed on provider i, i's expected load will be L'_i. To balance the load of all the providers, the provider with the lowest expected load will be selected to execute VM_j.

This paper develops a hybrid GAACO algorithm for scheduling tasks by solving complex combinatorial optimization issues. The section above suggests a method, including the genetic algorithms Fitness Function Value (FFV) and Ant Colony Optimization algorithms fitness function value [25, 45].

GAACO Algorithm

Input: List of Tasks and List of VMs

Output: The Best Solution for Tasks Allocation on VMs

Step 1. Initialize QoS weight factor $\sum_{i=1}^{n} w_i = 1$, $q = n$, number of cycles C

Step 2. Define QoS model objective function based on equation (8)

Step 3. Define load balance model using equation (11)

Step 4. Define GA fitness function values according to the equation (9)

Step 5. for (t = 1: t <= C; t++)

Step 6. for (j = 1; j < q; j++)

Step 7. while task size < VM size

Step 8. Select randomly the first task

Step 9. Selection, crossover, and mutation

Step 10. End while

Step 11. GA Optimal solution group is generated

Step 12. Initialize ACO

Step 13. while allowed j (t) ≠ 0

Step 14. if random first == 0

Step 15. Select the first scheduled application randomly

Step **16.** random first = 1

Step 17. else

Step 18. Select the scheduled task according to the pheromone value of ant

Step 19. End if

Step 20. Calculate the expected loads L of all feasible VMs

Step 21. Rank feasible VMs into increasing order based on expected load L

Step 22. The VM which has a low expected load L is selected for pheromone value

Step 23. Selected pheromone value added to the ant list

Step 24. End while
Step 25. Calculate objective function load L, for the generated solution of ant
Step 26. if L > best solution
Step 27. best solution = L
Step 28. Save ant solution in VM
Step 29. End if
Step 30. End for
Step 31. Calculate the incremental pheromone on each task
Step 32. Clear the ant list for each ant
Step 33. End for

Fig. 1 shows that the GAACO algorithms minimizes the energy consumption and also maximizes the resource utilization in cloud load balancing. The best solution of GA is used as an initial pheromone value of the ACO algorithm. The best solution of ACO algorithm is the final best solution of GAACO algorithm.

4. Results and discussion

This section discusses the performance and results of both existing and proposed load balancing methods concerning the parameters of the degree of imbalance, makespan time, response time, and energy consumption. For execution and results evaluation, the NS3 simulator has been utilized to test each method. In order to prove the efficacy of the proposed GAACO model, it is compared with some other conventional load balancing techniques. In the proposed work, the process of energy-efficient resource scheduling is accomplished with the help of the GAACO model. In which, the GA technique first sorts the chromosomes based on their level of importance, because gene execution is prioritized during crossover and mutation processes, the order in which they are executed is never disturbed. Then, the best solution of GA will be considered as the initial value of ACO and, finally, the best optimal fitness value is calculated by using the ACO algorithm. The proposed GAACO algorithm could efficiently reduce the makespan time, response time and increase the energy efficiency with better utilization of resources by solving the degree of imbalance issues, where the Degree of Imbalance (DoI) is calculated as follows:

$$\text{DoI} = \frac{\max t(i) - \min t(i)}{\text{avg } t(i)}, \tag{20}$$

where max $t(i)$ means maximum tasks ($t(i)$) of all VMs, and min $t(i)$ means minimum tasks ($t(i)$) of all VMs, and avg $t(i)$ is the average of task $t(i)$.

Fig. 2 shows the DoI analysis of the existing EFOA and proposed GAACO techniques concerning a varying number of tasks ranging from 100 up to 1000, which are allocated to 100 VMs [32]. Based on this analysis, it is evident that the proposed GAACO technique outperforms the other technique, reducing the DoI measure.

Similar to that, the makespan time of existing EFOA and proposed GAACO models are calculated, concerning the varying number of tasks ranging from 100 up to 1000, which are allocated to 100 VMs as shown in Fig. 3. This result also proves that the proposed GAACO technique outperforms the other technique with reduced makespan time. As in the proposed time, the CPU usage, response time, and

execution time have been estimated before allocating the jobs to the resources, this helps to reduce the makespan time.

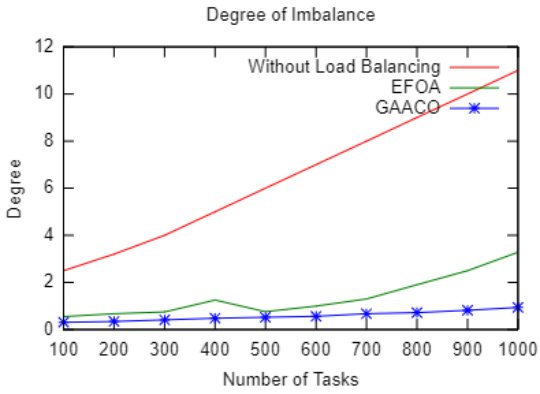

Fig. 2. Degree of Imbalance for 1000 tasks with 100 VMs

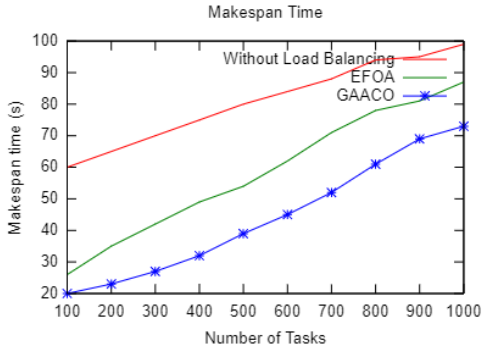

Fig. 3. Makespan time required to allocate 1000 tasks with 100 VMs

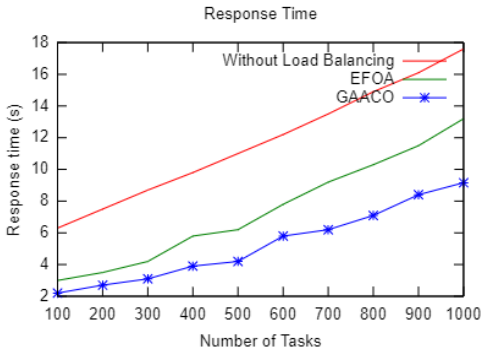

Fig. 4. Response time for allocating 1000 tasks with 100 VMs

Consequently, Fig. 4 estimates the response time of existing EFOA and proposed GAACO techniques under the varying number of tasks ranging from 100 up to 1000, which are allocated to 100 VMs. Normally, the response time is one of the essential measures that need to be considered while scheduling the jobs to the resources for execution. Moreover, the entire performance of the load balancing model highly depends on the energy level and response time of machines in the cloud

system. When compared to the EFOA model, the hybrid GAACO-based task allocation scheme requires a reduced response time for executing the given tasks.

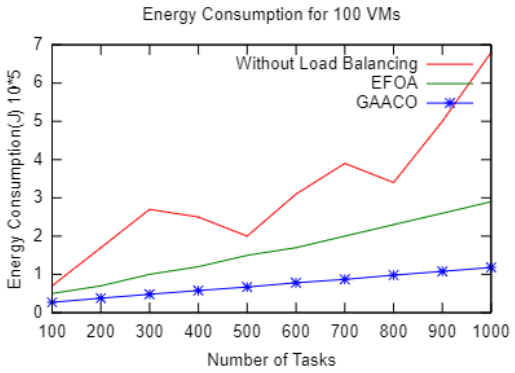

Fig. 5. Energy consumption for allocating 1000 tasks with 100 VMs

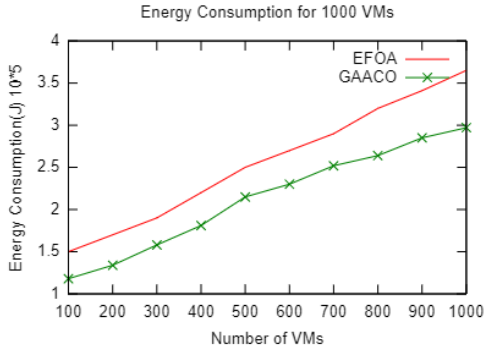

Fig. 6. Energy consumption for allocating 1000 tasks with 1000 VMs

Fig. 5 shows that the energy consumption is minimized while allocating the 100 up to 1000 tasks to 100 VMs as compared to EFOA method. And in Fig. 6, the energy consumption of conventional EFOA and GAACO techniques is estimated under a varying number of VMs ranging from 100 up to 1000. For validating the energy efficiency of the proposed GAACO model, the increased number of tasks (i.e., 1000 tasks) and VMs (i.e., 100 up to 1000 VMs) has been taken for this analysis. From these evaluations, it is identified that the proposed GAACO technique outperforms the other technique with ensured efficiency and results.

Based on this analysis, Figs 5 and 6 is evident that the load has been efficiently balanced and the energy consumption is minimized in the proposed model, and we have shown the optimal energy efficiency by comparing the results of less loaded VMs and more loaded VMs.

5. Conclusion

An ACO and GA hybridization with a multi-objective function was explored to enhance the global optimization solution. The energy problem has received minimal attention in most cloud system scheduling approaches, and in some systems that have

been tuned for energy use, the makespan has increased. In our proposed GAACO algorithm, the energy efficiency is improved while maintaining a high level of services by combining GA and ACO algorithm-based QoS load scheduling approaches. The proposed GAACO algorithm shows that energy consumption is minimized even user schedules a large number of tasks to less number of VMs without any task failure occurring. In this research study, we have minimized energy consumption, makespan time, and response time and also reduced the imbalance of load balancing.

6. Future works

In order to fulfill customer demand, Cloud computing platforms offer more applications and services via the Internet. It is becoming more and more common to host applications on the cloud. To serve these applications via virtualization, a host's scalability is becoming more critical. There are always new challenges to overcome in terms of resource allocation and scalability. The hosts' scalability and fault tolerance must be evaluated. There may be scope for utilizing evolutionary algorithms in cloud computing, such as Particle Swarm Algorithm abbreviated as PSO and Simulated Annealing. In the cloud computing context, effective and reliable services become a problem. The goal of autonomous computer systems is for them to monitor, repair, and optimize themselves. To fulfill the SLA for user needs, an autonomic resource allocation method may be designed.

References

1. Beloglazov, A., J. Abawajy, R. Buyya. Energy-Aware Resource Allocation Heuristics for Efficient Management of Data Centers for Cloud Computing. – Future Generation Computer Systems, Vol. **28**, 2012, No 3, pp. 755-768.
2. Priya, V., C. S. Kumar, R. Kannan. Resource Scheduling Algorithm with Load Balancing for Cloud Service Provisioning. – Applied Soft Computing, Vol. **76**, 2019, pp. 416-424.
3. Kunwar, V., N. Agarwal, A. Rana, J. Pandey. Load Balancing in Cloud – A Systematic Review. – Big Data Analytics, 2018, pp. 583-593.
4. Rekha, P., M. Dakshayini. Dynamic Cost-Load Aware Service Broker Load Balancing in Virtualization Environment. – Procedia Computer Science, Vol. **132**, 2018, pp. 744-751.
5. Braun, T. D., H. J. Siegel, N. Beck, L. L. Böloni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. – Journal of Parallel and Distributed Computing, Vol. **61**, 2001, No 6, pp. 810-837.
6. Duan, J., Y. Yang. A Load Balancing and Multi-Tenancy Oriented Data Center Virtualization Framework. – IEEE Transactions on Parallel and Distributed Systems, Vol. **28**, 2017, No 8, pp. 2131-2144.
7. Buyya, R., A. Beloglazov, J. Abawajy. Energy-Efficient Management of Data Center Resources for Cloud Computing: A Vision, Architectural Elements, and Open Challenges. ArXiv preprint arXiv: 1006.0308, 2010.
8. Calheiros, R. N., R. Buyya, C. A. De Rose. A Heuristic for Mapping Virtual Machines and Links in Emulation Testbeds. – In: Proc. of IEEE International Conference on Parallel Processing, 2019, pp. 518-525.

9. C a r d o s a, M., M. R. K o r u p o l u, A. S i n g h. Shares and Utilities Based Power Consolidation in Virtualized Server Environments. – In: Proc. of IEEE International Symposium on Integrated Network Management, 2009, pp. 327-334.
10. C h e d i d, W., C. Y u, B. L e e. Power Analysis and Optimization Techniques for Energy Efficient Computer Systems. – Advances in Computers, Vol. **63**, 2005, pp. 129-164.
11. K e p h a r t, J. O., D. M. C h e s s. The Vision of Autonomic Computing. – Computer, Vol. **36**, 2003, No 1, pp. 41-50.
12. K u m a r, D. Energy Efficient Resource Allocation for Cloud Computing. 2014.
13. A h m a d, M. O., R. Z. K h a n. Load Balancing Tools and Techniques in Cloud Computing: A Systematic Review. – Advances in Computer and Computational Sciences, 2018, pp. 181-195.
14. J i n g, S. Y., S. A l i, K. S h e, Y. Z h o n g. State-of-the-Art Research Study for Green Cloud Computing. – The Journal of Supercomputing, Vol. **65**, 2013, No 1, pp. 445-468.
15. K a n g, Q. M., H. H e, H. M. S o n g, R. D e n g. Task Allocation for Maximizing Reliability of Distributed Computing Systems Using Honeybee Mating Optimization. – Journal of Systems and Software, Vol. **83**, 2010, No 11, pp. 2165-2174.
16. C h e n, S. L., Y. Y. C h e n, S. H. K u o. CLB: A Novel Load Balancing Architecture and Algorithm for Cloud Services. – Computers & Electrical Engineering, Vol. **58**, 2017, pp. 154-160.
17. S h a h, J. M., K. K o t e c h a, S. P a n d y a, D. C h o k s i, N. J o s h i. Load Balancing in Cloud Computing: Methodological Survey on Different Types of Algorithm. – In: Proc. of International Conference on Trends in Electronics and Informatics, 2017, pp. 100-107.
18. M i s h r a, S. K., M. A. K h a n, B. S a h o o, D. P u t h a l, M. S. O b a i d a t, K. F. H s i a o. Time Efficient Dynamic Threshold-Based Load Balancing Technique for Cloud Computing. – In: Proc. of International Conference on Computer, Information and Telecommunication Systems, 2017, pp. 161-165.
19. L e e, Y. C., A. Y. Z o m a y a. Energy Efficient Utilization of Resources in Cloud Computing Systems. – Journal of Supercomputing, Vol. **60**, 2012, No 2, pp. 268-280.
20. L i u, L., H. W a n g, X. L i u, X. J i n, W. B. H e, Q. B. W a n g, Y. C h e n. GreenCloud: A New Architecture for Green Data Center. – In: Proc. of 6th International Conference Industry Session on Autonomic Computing and Communications Industry Session, 2017, pp. 29-38.
21. L o r p u n m a n e e, S., M. N. S a p, A. H. A b d u l l a h, C. C h o m p o o i n w a i. An Ant Colony Optimization for Dynamic Job Scheduling in Grid Environment. – International Journal of Computer and Information Science and Engineering, Vol. **1**, 2007, No 4, pp. 207-214.
22. K u s i c, D., J. O. K e p h a r t, J. E. H a n s o n, N. K a n d a s a m y, G. J i a n g. Power and Performance Management of Virtualized Computing Environments via Lookahead Control. – Cluster Computing, Vol. **12**, 2009, No 11, pp. 1-15.
23. P r a d h a n, A., S. K. B i s o y, P. K. M a l l i c k. Load Balancing in Cloud Computing: Survey. – In: Innovation in Electrical Power Engineering, Communication, and Computing Technology, 2020, pp. 99-111.
24. A l-J o b o u r y, I. M., E. H. A l-H e m i a r y. Virtualized Fog Network with Load Balancing for IoT Based Fog-to-Cloud. – JOIV: International Journal on Informatics Visualization, Vol. **4**, 2020, No 3, pp. 123-126.
25. M a d n i, S. H. H., M. S. A b d L a t i f f, S. I. M. A b d u l h a m i d, J. A l i. Hybrid Gradient Descent Cuckoo Search (HGDCS) Algorithm for Resource Scheduling in IaaS Cloud Computing Environment. – Cluster Computing, Vol. **22**, 2019, No 1, pp. 301-334.
26. S r i k a n t a i a h, S., A. K a n s a l, F. Z h a o. Energy Aware Consolidation for Cloud Computing. – In: Proc. of Workshop on Power Aware Computing and Systems at OSDI, USENIX HotPower'08, 2008.
27. L i u, F., J. T o n g, J. M a o, R. B o h n, J. M e s s i n a, L. B a d g e r, D. L e a f. NIST Cloud Computing Reference Architecture. – NIST Special Publication, Vol. **500**, 2011, No 2011, pp. 1-28.
28. G a m a l, M., R. R i z k, H. M a h d i, B. E l h a d y. Bio-Inspired Based Task Scheduling in Cloud Computing. – In: Machine Learning Paradigms: Theory and Application, Springer, 2019, pp. 289-308.

29. G e o r g e A m a l a r e t h i n a m, D., S. K a v i t h a. Rescheduling Enhanced Min-Min (REMM) Algorithm for Meta-Task Scheduling in Cloud Computing. – In: Proc. of International Conference on Intelligent Data Communication Technologies and Internet of Things, 2018, pp. 895-902.
30. A l w o r a f i, M. A., S. M a l l a p p a. A Collaboration of Deadline and Budget Constraints for Task Scheduling in Cloud Computing. – Cluster Computing, Vol. **23**, 2020, No 2, pp. 1073-1083.
31. G r a y, L., A. K u m a r, H. L i. SPECpower Committee. Power and Performance Benchmark Methodology V2. – In: Standard Performance Evaluation Corporation (SPEC), 2014.
32. L a w a n y a S h r i, M., S. S u b h a, B. B a l u s a m y. Energy-Aware Fruitfly Optimisation Algorithm for Load Balancing in Cloud Computing Environments. – International Journal of Intelligent Engineering and Systems, Vol. **10**, 2017, No 1, pp. 75-85.
33. S h o j a f a r, M., M. K a r d g a r, A. A. R. H o s s e i n a b a d i, S. S h a m s h i r b a n d, A. A b r a h a m. TETS: A Genetic-Based Scheduler in Cloud Computing to Decrease Energy and Makespan. – In: Proc. of International Conference on Hybrid Intelligent Systems, 2016, pp. 103-115.
34. P o l e p a l l y, V., K. S h a h u C h a t r a p a t i. Dragonfly Optimization and Constraint Measure-Based Load Balancing in Cloud Computing. – Cluster Computing, Vol. **22**, 2019, No 1, pp. 1099-1111.
35. S a n g a i a h, A. K., A. A. R. H o s s e i n a b a d i, M. B. S h a r e h, S. Y. B o z o r g i R a d, A. Z o l f a g h a r i a n, N. C h i l a m k u r t i. IoT Resource Allocation and Optimization Based on Heuristic Algorithm. – Sensors, Vol. **20**, 2020, No 2, p. 539.
36. X u e, S., Y. Z h a n g, X. X u, G. X i n g, H. X i a n g, S. J i. $$\varvec {Q} ET $$ QET: A QoS-Based Energy-Aware Task Scheduling Method in Cloud Environment. – Cluster Computing, Vol. **20**, 2017, No 4, pp. 3199-3212.
37. F a r a h a b a d i, A. B., A. H o s s e i n a b a d i. Present a New Hybrid Algorithm Scheduling Flexible Manufacturing System Consideration Cost Maintenance. – International Journal of Scientific & Engineering Research, Vol. **4**, 2013, No 9, pp. 1870-1875.
38. H o m e P r a s a n n a R a j u, Y., N. D e v a r a k o n d a. Makespan Efficient Task Scheduling in Cloud Computing. – In: Emerging Technologies in Data Mining and Information Security, Springer, 2019, pp. 283-298.
39. W e i, X., J. F a n, Z. L u, K. D i n g, R. L i, G. Z h a n g. Bio-Inspired Application Scheduling Algorithm for Mobile Cloud Computing. – In: Proc. of 4th International Conference on Emerging Intelligent Data and Web Technologies, 2013, pp. 690-695.
40. T o p c u o g l u, H., S. H a r i r i, M. Y. W u. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. – IEEE Transactions on Parallel and Distributed Systems, Vol. **13**, 2002, No 3, pp. 260-274.
41. Z h u, X., M. H u s s a i n, X. L i. Energy-Efficient Independent Task Scheduling in Cloud Computing. – In: Proc. of International Conference on Human Centered Computing, 2018, pp. 428-439.
42. P r a s a n n a K u m a r, K., K. K o u s a l y a. Amelioration of Task Scheduling in Cloud Computing Using Crow Search Algorithm. – Neural Computing and Applications, Vol. **32**, 2020, No 10, pp. 5901-5907.
43. S r i c h a n d a n, S., T. A. K u m a r, S. B i b h u d a t t a. Task Scheduling for Cloud Computing Using Multi-Objective Hybrid Bacteria Foraging Algorithm. – Future Computing and Informatics Journal, Vol. **3**, 2018, No 2, pp. 210-230.
44. B a s u, S., M. K a r u p p i a h, K. S e l v a k u m a r, K. C. L i, S. H. I s l a m, M. M. H a s s a n, M. Z. A. B h u i y a n. An Intelligent/Cognitive Model of Task Scheduling for IoT Applications in Cloud Computing Environment. – Future Generation Computer Systems, Vol. **88**, 2018, pp. 254-261.
45. K a s h i k o l a e i, S. M. G., A. A. R. H o s s e i n a b a d i, B. S a e m i, M. B. S h a r e h, A. K. S a n g a i a h, G. B. B i a n. An Enhancement of Task Scheduling in Cloud Computing Based on Imperialist Competitive Algorithm and Firefly Algorithm. – Journal of Supercomputing, Vol. **76**, 2020, No 8, pp. 6302-6329.

Received: 17.10.2022; Accepted: 11.03.2023