Computers, Materials & Continua
DOI: 10.32604/cmc.2022.030778
Article

Tech Science Press

An Adaptive Genetic Algorithm-Based Load Balancing-Aware Task Scheduling Technique for Cloud Computing

Mohit Agarwal[1,*] and Shikha Gupta[2]

[1]Deptartment of Computer Science & Engineering, School of Engineering & Technology, Sharda University, Greater Noida, Uttar Pradesh, 201319, India
[2]Department of Information Technology, Maharaja Agrasen Institute of Technology, Delhi, 110086, India
*Corresponding Author: Mohit Agarwal. Email: rs.mohitag@gmail.com
Received: 01 April 2022; Accepted: 29 May 2022

Abstract: Task scheduling in highly elastic and dynamic processing environments such as cloud computing have become the most discussed problem among researchers. Task scheduling algorithms are responsible for the allocation of the tasks among the computing resources for their execution, and an inefficient task scheduling algorithm results in under-or over-utilization of the resources, which in turn leads to degradation of the services. Therefore, in the proposed work, load balancing is considered as an important criterion for task scheduling in a cloud computing environment as it can help in reducing the overhead in the critical decision-oriented process. In this paper, we propose an adaptive genetic algorithm-based load balancing (GALB)-aware task scheduling technique that not only results in better utilization of resources but also helps in optimizing the values of key performance indicators such as makespan, performance improvement ratio, and degree of imbalance. The concept of adaptive crossover and mutation is used in this work which results in better adaptation for the fittest individual of the current generation and prevents them from the elimination. CloudSim simulator has been used to carry out the simulations and obtained results establish that the proposed GALB algorithm performs better for all the key indicators and outperforms its peers which are taken into the consideration.

Keywords: Cloud computing; genetic algorithm (GA); load balancing; makespan; resource utilization; task scheduling

1 Introduction

Since its inception, cloud computing technology has witnessed phenomenal growth in its adoption in a very short period. Advancements in communication technologies and the exponential rise in Internet usage by people for carrying out their day-to-day computing-related activities is also one of the major reasons behind such growth. The prominent characteristics of the cloud computing model, which include on-demand self-service, rapid elasticity, broad network access, and resource pooling, also help in gaining the popularity of cloud-based computing models among users [1]. Cloud service

providers (CSPs) strive to optimally use the underlying computing resources in data centers to fulfill the computing-related demand raised by the variety of customers to carry out their activities [2–5].

The scheduling of tasks in a collaborative computing environment, such as cloud computing, is a challenging problem which lies in the nondeterministic polynomial (NP)-hard category. Lots of efforts have been made by the researchers to provide the solution for the task scheduling problem in cloud computing using the meta-heuristic algorithms still lots of space for improvement is there. In this work, we propose a genetic algorithm-based mechanism for task scheduling that primarily considers load balancing among virtual machines (VMs). To achieve the same two novel techniques for the selection of the parents in genetic algorithm, i.e., adaptive crossover and adaptive mutation has been used which helps in increasing the rate of convergence and reduction in loss of diversity two main problems from which standard genetic algorithm used to suffer. To the best of our knowledge, the proposed adaptive genetic algorithm-based load balancing (GALB) aware task scheduling strategy, which we consider in this work with many performance indicators in a cloud computing environment, has never been used.

The primary contributions of this paper are summarized as:

- The proposed GALB algorithm considers load balancing as an important criterion for task scheduling in a cloud computing environment, as very little work has been conducted in this area.
- The proposed algorithm uses the concept of adaptive rate for crossover and mutation, which helps in the better adaptation of the fittest individual of the current generation and helps in protecting them from elimination. The adaptive approach used for parent selection significantly improved the performance of the proposed algorithm.
- Performance evaluation of the proposed GALB algorithm using key performance indicators such as makespan, performance improvement rate (PIR %), degree of imbalance, and resource utilization.

The remainder of this paper is structured as follows: Section 2 describes the related work, and Section 3 presents the system model and problem formulation used in this work. The proposed GALB algorithm is described in Section 4. The performance evaluation of the proposed algorithm, along with the performance metrics, is presented in Section 5. Section 6 discusses the results and compares them. Finally, Section 7 presents the conclusions of the study.

2 Related Work

The field of distributed computing has witnessed significant development in the domain of scheduling since the 1980 s, as it involves the allocation of a limited number of available computing resources to the task or applications submitted by the users for their execution. The adoption of the Internet by the general masses to carry out their computing activities opens a new area of research for modern scheduling. In the last decade, researchers have proposed several algorithms to solve the most discussed problems of task scheduling [6–14]. In this section, References and its citations has been reordered to maintain sequencial order. Kindly verify. we present a brief overview of some of the popular task scheduling algorithms that may or may not involve load balancing factors but are important in the context of our proposed work.

Static task scheduling algorithm such as First Come First Serve (FCFS), Round Robin (RR), and Shortest Job First (SJF) are not suitable for varying load environments, such as cloud computing, as they require information regarding the tasks (such as task size/length, task count, any deadline associated with them, etc.) and available computing resources (such as bandwidth, storage capacity,

processing elements, power, etc.) in advance [15]. On the other hand, task scheduling mechanisms, which are based on dynamic and meta-heuristic algorithms, such as artificial bee colony (ABC), ant colony optimization (ACO) [16], cuckoo search (CS) [17], genetic algorithm (GA) [18], particle swarm optimization (PSO) [19,20], etc., prove their efficiency over static algorithms in cloud computing, as they do not require prior information about tasks and computing machines.

The problem associated with heuristic algorithms is that they are often used to trap into local optima, and because they fail to converge quickly and do not result in better solutions. On the other hand, meta-heuristic algorithms have proven their efficiency in solving NP-hard problems [21]. Some of the most popular metaheuristic techniques used to solve such NP-hard problems are the artificial bee colony [22], ant colony optimization [23], genetic algorithm [24], and particle swarm optimization [25]. The ability of metaheuristic techniques to find the near-optimal solution makes them the most appropriate techniques for solving the cloud computing scheduling problem.

The authors in [25] presented the ABC algorithm-based load-balancing technique, which results in the minimization of the makespan along with the balancing of workload across the VMs in a cloud computing environment. D. B. et al. [26] proposed a task scheduling mechanism based on PSO. Wang et al. [27] in their work presented a load-balancing mechanism based on a hyper-heuristic algorithm to provide a system that distributes tasks in a balanced manner. The authors in [28] proposed a dynamic resource allocation technique that helps in reducing the energy consumption by data centers in a cloud computing environment, whereas Agarwal and Srivastava [29] proposed a task scheduling mechanism in which the initial population of the PSO is generated by an opposition-based learning concept so that diversity in the population can be achieved and results in improvement for the set of performance indicators.

3 System Model and Problem Formulation

In this work, it is assumed that a cloud-based computing model comprises several data centers, which are responsible for the provisioning of resources and computing facilities as per the demand of the users. Data centers in cloud computing are mainly a collection of heterogeneous computing nodes known as virtual machines (VMs), which are connected through high–speed links. VMs are computing machines primarily responsible for the execution of different workloads with diverse computation requirements Tab. 6.

For more clarity regarding the problem of task scheduling in cloud computing, let us consider a cloud computing system that consists of ***m*** heterogeneous independent VMs that are represented as $VM = \{VM_1, VM_2, VM_3, \ldots VM_m\}$, and a set of ***n*** tasks that are represented by $T = \{T_1, T_2, T_3, \ldots T_n\}$. In this study, it is assumed that each machine will follow the first-come-first-serve policy to execute the submitted tasks or workloads that are mutually independent and can be executed on any virtual machine. Let Tij be the time required by virtual machine VM_j to execute an ith task, which is calculated using Eq. (1).

$$T_{ij} = \frac{TL_i}{V_j} \tag{1}$$

where $\boldsymbol{TL_i}$ is the size of task $\boldsymbol{T_i}$ in million instructions (MI), ***and*** $\boldsymbol{V_j}$ represents the computation speed of the ***jth*** virtual machine in million instructions per second (MIPS).

The finishing time of the jth virtual machine is denoted by $\boldsymbol{VFT_j}$ and computed using Eq. (2) as follows:

$$VFT_j = \sum_{i=1}^{n} ET_{ij} * A\,(i,j) \tag{2}$$

where i represents the task index or number whose value lies in [1 – n], j represents the index of VM whose value lies in [1 – m], and $\boldsymbol{ET_{ij}}$ is the time consumed while the execution of task $\boldsymbol{T_i}$ on $\boldsymbol{VM_j}$ and calculated as shown below:

$$ET_{ij} = T_{ij} + (TIS_i + TOS_i)\,/Bw_j \tag{3}$$

$\boldsymbol{ET_{ij}}$ consists of two parts: the time required for execution and data transfer time. $\boldsymbol{TIS_i}$ and $\boldsymbol{TOS_i}$ are used to represent the input size and output size of the ith task respectively, while $\mathrm{Bw_j}$ denotes the bandwidth of the jth node.

$A\,(i,j) = 1$ when task $\boldsymbol{T_i}$ is allocated to machine $\mathrm{VM_j}$ otherwise $A\,(i,j) = 0$.

If a cloud computing-based system starts the execution of the tasks at time 0; then ***makespan*** is the time when all VMs will complete the execution of the entire load on them.

$$makespan = \max\left\{VFT_j\right\} \tag{4}$$

Our objective was to choose an optimal solution that provides the minimum value for the makespan for the discussed scenario.

Load Balancing Mechanism in Cloud Computing

Load balancing in a dynamic and distributed environment, such as cloud computing, is an important aspect from a performance point of view, and it must be considered while forming any task scheduling strategy. A load-balancing-enabled task scheduling strategy can result in (i) balanced distribution of the load or tasks among the virtual machines for better utilization of the underlying VMs, and (ii) reduction in waiting time for the execution of tasks.

Let $\boldsymbol{V_i}$ represent the computation power of the ith VM and $\boldsymbol{V}$ be the processing capacity of all available VMs, which is calculated as shown in Eq. (5):

$$V = \sum_{i=0}^{m} V_i \tag{5}$$

After determining the maximum processing capacity, $\boldsymbol{V}$; the next step is to determine the number of underloaded, overloaded, and balanced virtual machines.

For this work, a VM is categorized as an underloaded virtual machine (UVM) if its utilization is less than 30% of its capacity and as an overloaded virtual machine (OVM) if its utilization is more than 75% of its capacity.

The load on a VM at time t can be defined as the total size of the tasks on a particular VM and can be denoted as $\boldsymbol{L_{VM_{i,t}}}$ and the load on all the VMs, $\boldsymbol{L}$ can be calculated as shown in Eq. (6):

$$L = \sum_{i=1}^{m} L_{VM_i} \tag{6}$$

The finishing time of a VM ($\boldsymbol{FT_{VM}}$) is the time required by the VM to complete the execution of tasks mapped to it. The finishing time of ith VM can be determined using Eq. (7) as follows:

$$FT_{VM,i} = \frac{L_{VM,i}}{V_i} \tag{7}$$

Finishing time of all the VMs, $\boldsymbol{FT_{VM}}$ can be calculated as:

$$FT_{VM} = \frac{L}{V} \tag{8}$$

In cloud computing, load balancing can be determined using the standard deviation ($\boldsymbol{\sigma}$) of the entire system load as shown in Eq. (9),

$$\sigma = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(FT_{vm,i} - FT_{vm}\right)^2} \tag{9}$$

A system is said to be unbalanced if the value obtained for the standard deviation (σ) is greater than the threshold value (***VThres***), which is used in the range [0,1] [26]; otherwise, the system is said to be in a balanced state.

Algorithm1: Load Balancing Process

Begin
Generate a set of tasks randomly; $T = [T_1, T_2, T_3, \ldots T_n]$.
Arrange the task in decreasing order of their task size.
Generate random set of VMs $V = [VM_1, VM_2, \ldots VM_m]$.
Map $T_i \rightarrow VM_j$ using the first-come-first-serve mechanism.
Calculate load of VMs;
$L_{VM_i} = (total\ task\ allocated * Task\ length) / (MIPS * Number\ of\ processing\ unit)$ Calculate processing capacity of VM; $V_i = (MIPS * Number\ of\ processing\ unit) + Bw_i$.
if $(L_{VM_i} < V_i)$
    Load Balancing is possible.
else
    Load Balancing is not possible.
end if
Determine the state of the Virtual Machine.
if $(Resource\_Utilization > 80\% * (V_i))$
    VM is said to be in an Overloaded (OL) state.
else if $(Resource_{Utilization} < 25\% * (V_i))$
    VM is said to be in Under Loaded (UL) state.
else
    VM is said to be in a Balance state.
end if
if $(OL\ \&\ UL\ state\ exist)$
    Shift task from overloaded machine to under loaded machine
    Until $V_i <= OL$ && $V_i >= UL$
    Repeat the process till any VM remain overloaded.
end if
End

4 Proposed Adaptive Genetic Algorithm Based Load Balancing Aware Task Scheduling Mechanism: GALB

In this section, the proposed GALB-aware task scheduling technique is elaborated. The genetic algorithm belongs to the evolutionary algorithm family, which is based on the principle of natural evolution. GA works iteratively and maintains a set of solutions, also known as the population. Broadly, each genetic algorithm comprises three main operators, namely selection, crossover, and mutation, to generate a new population from the old one.

Genetic algorithms are well known for solving large, nonlinear, and discrete problems. As the solution development follows the probabilistic approach, they do not guarantee the optimal solution but can produce near-optimal solutions for the concerned problems.

4.1 Representation of the Chromosomes

In this work, we used the structure of chromosomes shown in Fig. 2. Let there be 12 different tasks (T_1, T_2, T_3, ..., T_{11}, T_{12}) and five distinct virtual machines (VM_1, VM_2, ..., VM_5) which are allocated as shown in Fig. 1. As shown in Fig. 1., tasks T_1, T_5 and T_9 are allocated to VM_2; tasks T_2 & T_{11} are allocated to VM_5; T_3, T_{10} and T_{12} are allocated to VM_3; T_4, T_6 allocated to VM_1; T_7 and T_8 allocated to VM_4.

T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
VM_2	VM_5	VM_3	VM_1	VM_2	VM_1	VM_4	VM_4	VM_2	VM_3	VM_5	VM_3

Figure 1: Allocation of 12 tasks on 5 VMs

4.2 Initialization of Population

A population in the GA is the collection of individuals or chromosomes, which is a representation of the probable solution of the given problem. The number of chromosomes in a population is controlled by sPop, which represents the size of the population. For the combination of 12 tasks and 5 VMs, a sample population is shown in Fig. 2.

2	5	3	1	2	1	4	4	2	3	5	3	Chromosome (Ch_1)
1	3	2	3	5	2	4	1	5	2	4	1	Chromosome (Ch_2)
3	2	5	5	3	1	2	1	2	4	3	4	Chromosome (Ch_3)
4	3	1	5	4	2	5	1	3	2	2	1	Chromosome (Ch_4)
4	2	1	4	2	5	1	4	3	1	5	2	Chromosome (Ch_5)

Figure 2: Sample initial population

4.3 Fitness Calculation

To check the efficiency of the proposed algorithm, it is necessary to evaluate the performance of every probable solution by using a fitness function. The proposed GALB algorithm has been used in this study to present a solution for the task scheduling problem that can result in (i) a lower value of makespan, (ii) a lower degree of imbalance, and (iii) an increase in resource utilization. Many attempts have been made to present the trade-off between the two conflicting objectives, such as makespan and resource utilization; however, in this work, we establish the relationship between these objectives, as

shown in Eq. (10).

$$VM_{utilization,i} = \frac{FT_{vm,i}}{makespan} \tag{10}$$

4.4 Resource Utilization

This may be defined as the duration of the overall execution time required for tasks during which the virtual machine remains occupied for execution. Let $VM_{utilization,i}$ denote the utilization of the ith VM, which can be determined as shown in Eq. (10).

The average utilization of all available VMs can be determined by using the Eq. (11) as,

$$VM_{Average\ Utilization} = \frac{\sum_{i=1}^{m} VM_{utilization,i}}{m} \tag{11}$$

To facilitate the design of the load-balancing aware task scheduling mechanism, a genetic algorithm for the above-mentioned objectives has been incorporated into a single objective function according to which a solution is said to be better if it has a lower value for the fitness function, which is defined in Eq. (12) as follows:

$$Fitness = \frac{makespan}{VM_{Average\ Utilization}} \tag{12}$$

4.5 Selection

The primary objective of the selection operation in the genetic algorithm is to create an intermediate population or mating pool for reproduction by selecting the fittest individuals from the current population. A tournament selection operator was used in this study for the selection of individuals from the current population. In this method of selection, a set of individuals is selected, and a tournament is carried out between them as a result of which an individual with better fitness is selected for further reproduction in the mating pool.

4.6 Adaptive Crossover

In this work, a two-point crossover operator is used which involves the random selection of two points in selected chromosomes for the generation of two offspring. The crossover rate plays a significant role in the efficient operation of any genetic algorithm. In this study, an adaptive crossover rate was used to avoid the problems associated with classical or standard genetic algorithms. The adaptive crossover rate needs to be calculated for the current population of each generation, which results in an improved rate of convergence and helps in avoiding the local optima and converges to the global optimal solution.

The adaptive crossover rate (Pc) used in this study was calculated as shown in Eq. (13):

$$P_c = \begin{cases} r_2, & if\ \ Makespan_{current}\ is\ memory \\ r_1 * \frac{Makespan_{best} - Makespan_{current}}{Makespan_{best} - Makespan_{avg}}, & elseif\ Makespan_{current} \geq\ Makespan_{avg} \\ r_2, & Makespan_{current} \leq Makespan_{avg} \end{cases} \tag{13}$$

Here, r_1 and r_2 are the two real numbers that range in [0.8, 0.95] based on the simulation analysis and help in significantly improving the performance of the algorithm.

4.7 Mutation

Mutation in the genetic algorithm helps maintain the variety in the population. In this study, a random mutation operator is used, which involves the random swapping of two gene positions. The development of the future population is the result of the application of selection, crossover, and mutation operations to the initial population in a single generation. Similar to the crossover rate, the mutation rate also significantly affects the performance of the algorithm to a large extent.

In this work, the adaptive mutation rate is used, which is calculated as shown in Eq. (14):

$$P_m = \begin{cases} m_2, & if \quad Makespan_{current} \text{ is memory} \\ m_1 * \frac{Makespan_{best} - Makespan_{current}}{Makespan_{best} - Makespan_{avg}}, & elseif\ Makespan_{current} \geq Makespan_{avg} \\ m_2, & Makespan_{current} \leq Makespan_{avg} \end{cases} \tag{14}$$

Here, m_1 and m_2 are the two real numbers that lie between 0.01 and 0.001 based on the simulation analysis and help improve the performance of the genetic algorithm significantly.

The values of $\boldsymbol{P_c}$ and $\boldsymbol{P_m}$ computed above are inversely proportional to the difference between the best fitness and average fitness values, that is,$fitness_{best} - fitness_{avg}$; $\boldsymbol{P_c}$ and $\boldsymbol{P_m}$ adjust themselves according to the situation to avoid premature convergence. Hence, the proposed adaptive probability rate of crossover and mutation helps protect the fittest individual from elimination and enters the next generation without any changes.

4.8 Termination Condition

To determine the point at which the GA must stop further execution of the involved steps, it is necessary to decide on termination conditions. For this task scheduling problem, the GA will stop its execution if any of the two stopping criteria have been met, first when we reach an absolute number of iterations that are used to define before commencing the experiments and, second, when there is no improvement in the population for 25% of the absolute iterations.

5 Performance Evaluation

This section presents the experimental and simulation setups used to test and analyze the performance of the proposed GALB algorithm for task scheduling.

The results obtained for the key performance indicator are compared with other prominently used algorithms, such as FCFS, dynamic load balancing (DLB), cuckoo search, standard genetic algorithm (sGA), particle swarm optimization (PSO), and hyper-heuristic (HyperLoad), using the CloudSim platform [30]. This simulator provides a virtualized environment for the simulation and supports on-demand provisioning of resources. Existing packages of the CloudSim simulator have been extended for modeling and simulation purposes, as discussed by researchers in their work [18,20].

Various experiments with different combinations of tasks and VMs were performed using the following machine configuration: Core i5 processor, 16 GB RAM, and 64-bit Windows 10 operating system to judge the efficiency of the proposed GALB algorithm. A random generator was used to generate different tasks with varying sizes and computing machines with varying processing power. The values of the input parameters used in this study are listed in Tab. 1.

Table 1: Simulation parameters used for result analysis

Parameter	Values
Number of tasks	100—1000
Size of tasks (MI)	1000—100000
Number of virtual machines (VMs)	25—100
Processing power of VM (MIPS)	100—2500
Chromosome size, nPop	50
Maximum number of generations, maxGen	75
Initial crossover probability, p_c	.9

The performance of the proposed algorithm is determined with the help of the performance metrics which include makespan, performance improvement ratio (PIR %), degree of imbalance (DI), and resource utilization. To the best of our knowledge, no researchers have considered these parameters together in their work, and most of them consider only the makespan as the main evaluation criteria.

5.1 Makespan

The computation of the values of the makespan obtained for the variety of experiments in this work is computed using Eq. (4) and the definition discussed in Section 3, which is used to denote the time required by the virtual machine to complete the execution of the submitted tasks. This is one of the most extensively used parameters for determining the efficiency of any algorithm.

5.2 Performance Improvement Ratio-PIR %

This parameter helps in determining the extent to which the makespan value is reduced for the proposed algorithm in comparison to the base algorithm, which is used for comparison purposes. In this study, it was calculated using Eq. (15):

$$PIR\ \% = \frac{Makespan - \ Makespan_{GALB}}{Makespan} \times 100 \tag{15}$$

here $Makespan$ and $Makespan_{GALB}$ represents the makespan of base and proposed GALB algorithm respectively.

5.3 Degree of Imbalance (DI)

DI is used to present the level of imbalance in terms of load allocation among the available VMs, and is computed using Eq. (16), as follows:

$$DI = \frac{T_{max} - \ T_{min}}{T_{avg}} \tag{16}$$

here T_{avg}, T_{min}, T_{max} represents the average, minimum, and maximum time a VM requires to execute all the tasks allocated to it. Therefore, an algorithm that results in the minimum value of DI with an increase in the number of tasks is considered efficient.

5.4 Resource Utilization ($VM_{Average\ Utilization}$)

Clouds may be defined as the pool of resources, and efficient utilization of resources, especially virtual machines, is considered an integral part of any efficient task scheduling algorithm. In this study, the value of resource utilization was computed using Eq. (11), as described in Section 4.

6 Results and Comparison

This section discusses the performance and results obtained for the proposed task scheduling mechanism based on GALB in cloud computing with the help of a variety of environments, which is divided into two categories: category-1 and category-2. Here *Category-1* involves randomly generated tasks ranging from 100 to 1000 with varying task lengths, as mentioned in Tab. 1, whose execution is carried out on 50 VMs that are homogeneous and remain fixed in number. *Category-2* involves the execution of 1000 homogeneous tasks with the same task length on randomly generated heterogeneous VMs with numbers varying from 25 to 100.

6.1 Result Comparison Based on Makespan

The performance of the proposed GALB algorithm is presented in this subsection for different sets of tasks and VMs, in which the makespan is considered as the key performance indicator. Tab. 2 represents the makespan values of all seven algorithms, which are taken into consideration for category-1 in which ten combinations of tasks are executed with the help of 50 VMs, and the values of makespan obtained for the proposed GALB algorithm are minimum (better) in comparison to the other extensively used algorithms. It can also be easily observed from Fig. 3. that the proposed GALB performs better than other algorithms as the number of tasks also increases, which needs to be executed with a fixed number of VMs, as stated in category-1.

Table 2: Makespan values obtained for different tasks–Category-1

# Tasks	FCFS	DLB	CS	GALoad	PSOLoad	HyperLoad	GALB (proposed)
100	1993	1920	1848	1811	1842	1763	1710
200	2758	2617	2495	2540	2596	2471	2216
300	3396	3210	3138	3093	3082	2905	2791
400	4419	4241	4160	4071	4097	3824	3690
500	4990	4914	4889	5007	4793	4810	4581
600	6035	6009	5910	5868	5762	5529	5110
700	7398	7154	6817	6732	6581	6293	5778
800	8557	8245	7909	7797	7518	7280	6632
900	10052	9708	9582	9210	9239	8727	8005
1000	14291	12147	11025	10654	9928	9401	8997

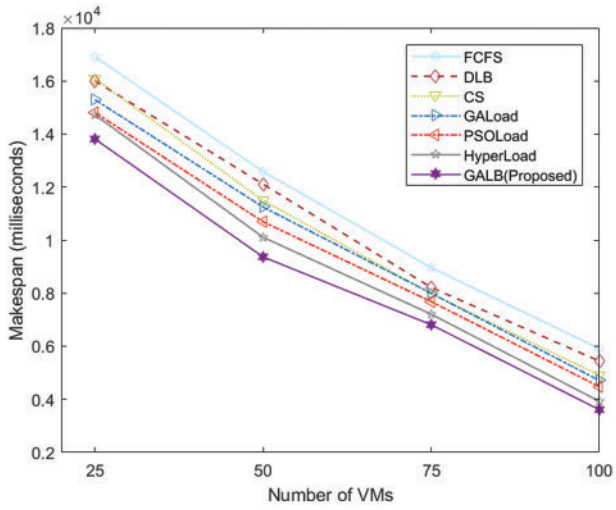

Figure 3: Comparison of makespan–Category-1

Tab. 3 presents the values of makespan obtained for the execution of 1000 homogeneous tasks over the combinations of VMs, as stated in category-2 and the proposed GALB algorithm outperforms the other algorithms, which are considered for the comparison. Fig. 4. presents a graphical representation of the obtained values of the makespan for category-2.

Table 3: Makespan values obtained for different VMs–Category-2

# VMs	FCFS	DLB	CS	GALoad	PSOLoad	HyperLoad	GALB (proposed)
25	16912	16007	16092	15297	14812	14727	13815
50	12587	12111	11498	11249	10695	10110	9368
75	8963	8218	7983	7998	7682	7209	6812
100	5914	5441	4907	4711	4476	3925	3609

6.2 Result Comparison Based on PIR%

In this subsection, the performance of the proposed GALB algorithm is measured in terms of the PIR%. Tab. 4 presents the value of PIR% obtained for different combinations of tasks, as stated in category-1. Values marked in bold represent the reduction in execution time in percentage by the proposed GALB algorithm in comparison to its competitor algorithm. For category-1, GALB resulted in a reduction in makespan by 10.31% to 37.04%. Tab. 5 presents the PIR% value for category-2. Again, the GALB algorithm outperforms its peers and can reduce the value of the makespan by 8.05% to 38.97% and establishes that the proposed algorithm is much better and able to reach the optimal solution.

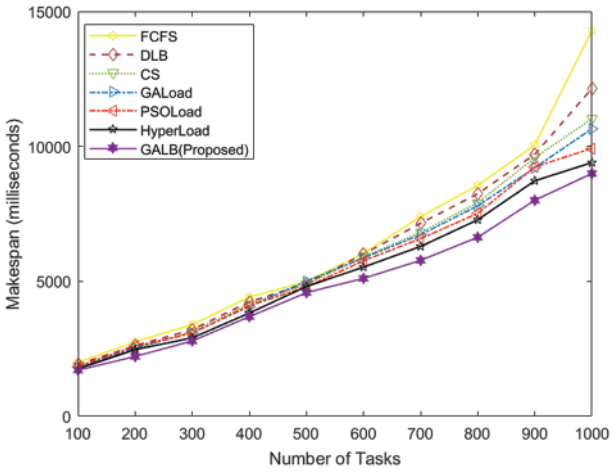

Figure 4: Comparison of makespan–Category-2

Table 4: PIR% for GALB (proposed) for different task combinations–Category-1

# Tasks	FCFS	DLB	CS	GALoad	PSOLoad	HyperLoad
100	14.19	10.9	7.46	5.57	7.05	3.0
200	19.65	15.32	11.18	12.75	14.63	10.31
300	17.81	13.05	11.05	9.76	9.44	3.92
400	16.49	12.99	11.29	9.35	9.93	3.5
500	8.19	6.77	6.29	8.5	4.42	4.76
600	15.32	14.96	13.53	12.91	11.31	7.57
700	21.89	19.23	15.24	14.17	12.20	8.18
800	22.49	19.56	16.14	14.94	11.78	8.90
900	20.36	17.54	16.45	13.08	13.35	8.27
1000	37.04	25.93	18.39	15.55	9.37	4.29

Table 5: PIR% for GALB (proposed) for different task combinations–Category-2

# VMs	FCFS	DLB	CS	GALoad	PSOLoad	HyperLoad
25	18.31	13.69	14.14	9.68	6.73	6.19
50	25.57	22.64	18.52	16.72	12.4	7.33
75	23.99	17.10	14.66	14.82	11.31	5.50
100	38.97	33.67	26.45	23.39	19.36	8.05

Table 6: Denote symbol and their meaning

Symbol	Meaning
m	Number of virtual machines (VMs)
n	Number of tasks
T	Set of task
T_{ij}	Execution time of i_{th} task on j_{th} VM
TL_i	length or size of $_{ith}$ task
V_j	Processing speed of j_{th} VM
MIPS	million instructions per second
VFT_j	finishing time of jth VM
MI	million instructions
ET_{ij}	Complete execution time
UVM	under loaded virtual machine
OVM	Over loaded virtual machine
L_{VM_i}	Load on i_{th} VM at time t.
L	Load on all VMs
$FT_{VM,i}$	Finishing time of i_{th} VM
FT_{VM}	Finishing time of all VM
σ	standard deviation
VM_{max}	maximum number of available virtual machines

6.3 Result Comparison Based on Degree of Imbalance (DI)

This subsection discusses the performance of the proposed GALB algorithm, based on the degree of imbalance.

Fig. 5. presents the performance of various task scheduling algorithms based on DI for category-1 and the proposed GALB algorithm can achieve a lower value of DI for the cases. Reduction in the value of DI when compared with its close competitor algorithm, HyperLoad by 32.64%, and PSOLoad, GALoad, CS, DLB, and FCFS by 38.30%, 39.58%, 71.54%, 58.36%, and 56.04%, respectively, which suggests that the load will be allocated more uniformly among the VMs in the case of the proposed GALB algorithm. Similarly, Fig. 6. presents the result for Category-2, our proposed GALB task scheduling algorithm results in a reduction of the DI for the different combinations of VMs by up to 41.40%, 46.27%, 52.87%, 61.72%, 60.70%, and 64.29% for HyperLoad, PSOLoad, GALoad, CS, DLB, and FCFS, respectively.

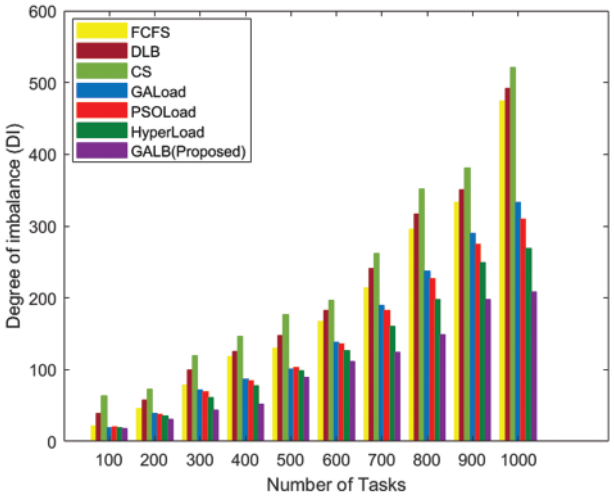

Figure 5: Comparison based on degree of imbalance (DI)–Category-1

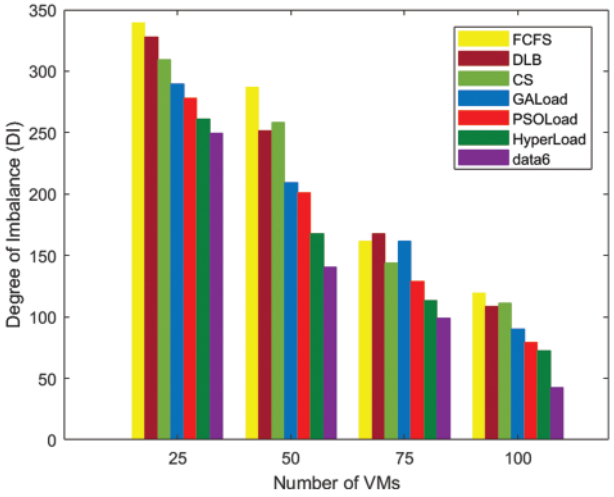

Figure 6: Comparison based on degree of imbalance (DI)–Category-2

6.4 Result Comparison Based on Resource Utilization

A performance analysis of the proposed GALB algorithm is presented in this subsection, in which resource utilization is considered as the performance evaluation parameter, as defined in Section 4. Different combinations of tasks and VMs were used, as shown in Figs. 7 and 8, respectively. Fig. 7. shows the performance of the algorithms that are taken into consideration, and it is easy to conclude that our proposed GALB-based task scheduling algorithm results in a higher rate of resource utilization for all combinations of the task ranging from 100 to 1000, whose execution has been done with the help of 50 VMs, as mentioned in category-1. Similarly, for category-2 which involves the execution of a fixed number of tasks on a varying number of VMSs, the proposed GALB outperforms the other prominently used algorithms reported in the literature, as presented in Fig. 8. Therefore, the GALB-based task scheduling algorithm again proves its superiority for both categories that we considered for the purpose of sensitivity analysis.

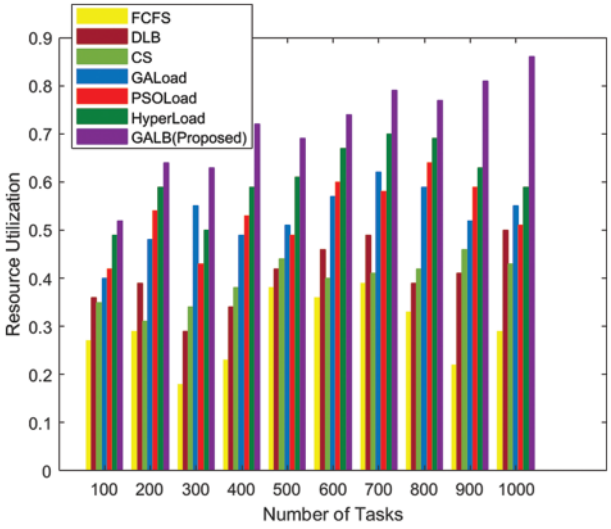

Figure 7: Comparison based on rate of resource utilization–Category-1

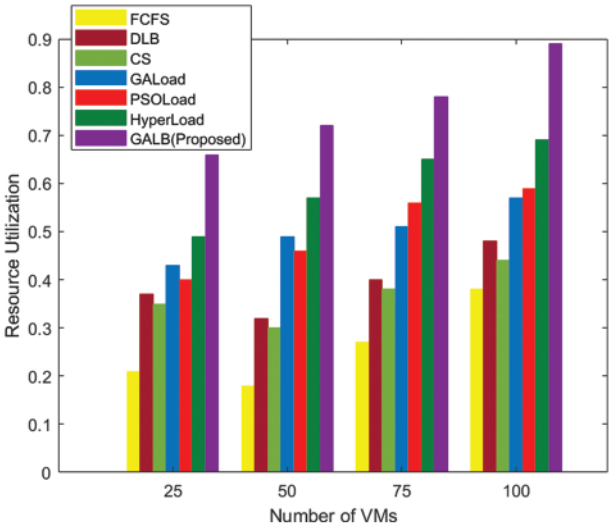

Figure 8: Comparison based on rate of resource utilization–Category-2

7 Results and Comparison

Load balancing in a high-performance computing environment, such as the cloud, proves to be a significant factor that helps both the cloud service user and provider economically and also plays a very important role in minimizing the overhead related to decision-making. The popularity and adoption of cloud computing, especially in the last decade, is possible because of its pay-per-usage mode of renting the required computing resources, which helps the service user meet their computing-based expectations. This work presents a novel meta-heuristic load-balancing aware task scheduling algorithm (GALB) to solve the most popular problem of task scheduling in cloud computing, which results in the allocation of tasks among the available virtual machines in a balanced manner and helps in the reduction of makespan, better utilization of resources, etc. The performance of GALB

is evaluated and compared with the optimal environment and state-of-the-art algorithms available in the literature. An extensive simulation study with sensitivity analysis was performed by introducing variations in several tasks and VMs to determine the efficiency of the proposed GALB algorithm. The proposed GALB algorithm results in better performance for the number of parameters discussed in the performance metrics section for both categories, and outperforms other meta-heuristic and traditional algorithms. Furthermore, in the future, the proposed GALB algorithm may be extended to include more features and can also be used for scientific workflow with other QoS parameters in a cloud computing environment.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly and S. M. Abdulhamid, "Recent advancements in resource allocation techniques for cloud computing environment: A systematic review," *Cluster Computing*, vol. 20, no. 3, pp. 2489–2533, 2016.

[2] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic, "Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility," *Future Generation Computer Systems*, vol. 25, no. 6, pp. 599–616, 2009.

[3] F. Durao, J. F. S. Carvalho, A. Fonseka and V. C. Garcia, "A systematic review on cloud computing," *the Journal of Supercomputing*, vol. 68, no. 3, pp. 1321–1346, 2014.

[4] B. P. Rimal and I. Lumb, "The rise of cloud computing in the era of emerging networked society," *Cloud Computing*, pp. 3–25, 2017.

[5] S. Singh, Y. -S. Jeong and J. H. Park, "A survey on cloud computing security: Issues, threats, and solutions," *Journal of Network and Computer Applications*, vol. 75, pp. 200–222, 2016.

[6] A. Jyoti, M. Shrimali, S. Tiwari and H. P. Singh, "Cloud computing using load balancing and service broker policy for IT service: A taxonomy and survey," *Journal of Ambient Intelligence and Humanized Computing*, vol. 11, no. 11, pp. 4785–4814, 2020.

[7] Q. Jiang, V. C. M. Leung, H. Tang and H. -S. Xi, "Adaptive scheduling of stochastic task sequence for energy-efficient mobile cloud computing," *IEEE Systems Journal*, vol. 13, no. 3, pp. 3022–3025, 2019.

[8] D. Lei, "Minimizing makespan for scheduling stochastic job shop with random breakdown," *Applied Mathematics and Computation*, vol. 218, no. 24, pp. 11851–11858, 2012.

[9] S. -S. Kim, J. -H. Byeon, H. Yu and H. Liu, "Biogeography-based optimization for optimal job scheduling in cloud computing," *Applied Mathematics and Computation*, vol. 247, pp. 266–280, 2014.

[10] C. -W. Tsai, W. -C. Huang, M. -H. Chiang, M. -C. Chiang and C. -S. Yang, "A Hyper-heuristic scheduling algorithm for cloud," *IEEE Transactions on Cloud Computing*, vol. 2, no. 2, pp. 236–250, 2014.

[11] K. Gu, N. Wu, B. Yin and W. Jia, "Secure data query framework for cloud and fog computing," *IEEE Transactions on Network and Service Management*, vol. 17, no. 1, pp. 332–345, 2020.

[12] M. Agarwal and G. M. Saran Srivastava, "A fuzzy enabled genetic algorithm for task scheduling problem in cloud computing," *International Journal of Sensors, Wireless Communications and Control*, vol. 10, no. 3, pp. 334–344, 2020.

[13] M. Agarwal and G. M. S. Srivastava, "Genetic algorithm-enabled particle swarm optimization (psoga)-based task scheduling in cloud computing environment," *International Journal of Information Technology & Decision Making*, vol. 17, no. 04, pp. 1237–1267, 2018.

[14] M. Abdullahi, M. A. Ngadi, S. I. Dishing and S. M. Abdulhamid, "An adaptive symbiotic organisms search for constrained task scheduling in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, 2022.

[15] Y. -K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Computing Surveys*, vol. 31, no. 4, pp. 406–471, 1999.
[16] M. Diallo, A. Quintero and S. Pierre, "An efficient approach based on ant colony optimization and tabu search for a resource embedding across multiple cloud providers," *IEEE Transactions on Cloud Computing*, vol. 9, no. 3, pp. 896–909, 2021.
[17] M. Agarwal and G. M. S. Srivastava, "A cuckoo search algorithm-based task scheduling in cloud computing," *Advances in Intelligent Systems and Computing*, pp. 293–299, 2017.
[18] M. Agarwal and G. M. S. Srivastava, "A genetic algorithm inspired task scheduling in cloud computing," in *2016 Int. Conf. on Computing, Communication and Automation (ICCCA)*, Greater Noida, India, vol. 554, 2016.
[19] N. Mansouri, B. Mohammad Hasani Zade and M. M. Javidi, "Hybrid task scheduling strategy for cloud computing by modified particle swarm optimization and fuzzy theory," *Computers & Industrial Engineering*, vol. 130, pp. 597–633, 2019.
[20] M. Agarwal and G. M. S. Srivastava, "A pso algorithm based task scheduling in cloud computing," *International Journal of Applied Metaheuristic Computing*, vol. 10, no. 4, pp. 1–17, 2019.
[21] C. -W. Tsai and J. J. P. C. Rodrigues, "Metaheuristic scheduling for cloud: A survey," *IEEE Systems Journal*, vol. 8, no. 1, pp. 279–291, 2014.
[22] A. Walker, J. Hallam and D. Willshaw, "Bee-havior in a mobile robot: The construction of a self-organized cognitive map and its use in robot navigation within a complex, natural environment," in *IEEE Int. Conf. on Neural Networks*, San Francisco, CA, USA, Mar. 1993.
[23] M. Dorigo and L. M. Gambardella, "Ant colony system: A cooperative learning approach to the traveling salesman problem," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 53–66, 1997.
[24] H. Holland, "Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and artificial intelligence," *MIT Press*, University of Michigan Press, Ann Arbor, MI, 1992.
[25] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. of ICNN'95-Int. Conf. on Neural Networks*, Perth, WA, Australia, Nov. 1995.
[26] L. D. D. B. and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing*, vol. 13, no. 5, pp. 2292–2303, 2013.
[27] X. Wang, C. S. Yeo, R. Buyya and J. Su, "Optimizing the makespan and reliability for workflow applications with reputation and a look-ahead genetic algorithm," *Future Generation Computer Systems*, vol. 27, no. 8, pp. 1124–1134, 2011.
[28] J. Praveenchandar and A. Tamilarasi, "Dynamic resource allocation with optimized task scheduling and improved power management in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 3, pp. 4147–4159, 2020.
[29] M. Agarwal and G. M. S. Srivastava, "Opposition-based learning inspired particle swarm optimization (OPSO) scheme for task scheduling problem in cloud computing," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 10, pp. 9855–9875, 2021.
[30] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. F. De Rose and R. Buyya, "CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2010.