Cluster Computing (2019) 22:1241–1251
https://doi.org/10.1007/s10586-019-02909-1

Efficient task allocation approach using genetic algorithm for cloud environment

P. M. Rekha[1,2] · M. Dakshayini[1]

Received: 21 February 2018 / Revised: 6 November 2018 / Accepted: 8 January 2019 / Published online: 23 January 2019

Abstract
As the number of cloud applications is rising exponentially, efficient allocation of these tasks among multiple computing machines ensuring the quality of service and better profit to the cloud service providers is a challenge. Effective task allocation approach needs to be developed considering a number of objectives while making allocation decisions, such as less energy consumption and quick response, in order to make the best resource allocation satisfying the cloud user requirements and improving the overall performance of the cloud computing environment. Hence, in this paper, Genetic Algorithm based efficient task allocation approach has been proposed for achieving the reduced task completion time by making wise allocation decisions. This proposed algorithm has been simulated using cloudsim toolkit and the performance is evaluated by comparing with greedy and simple allocation methods on a set of parameters like makespan and throughput for task scheduling. The evaluation results have shown the better throughput with the proposed approach.

Keywords Cloud computing · Genetic Algorithm · Task scheduling · Cloudlets · Makespan · Minimum Finishing time

1 Introduction

In cloud computing system, large amount of computation tasks must be placed in a specified data centre and cannot be moved. A computation may process the data sets from different data centres, because of the enormous size of data and inadequate network bandwidth; task scheduling has seriously affected the effective use of system resources, limiting the rapid development of cloud computing technology [1].Cloud computing is a Distributed computing and web based utility computing that provides diverse processing resources and serves as a model for enabling on-demand access to a common pool of configurable computing resources [2]. Cloud computing has emerged as an accepted computing model in order to support the processing of large volume of data by using clusters of commodity computers [3]. The significant growth of cloud computing has increased the number of clients and additionally increasing the demand for the resources. With the advancements in cloud computing, innovative possibilities for internet-based applications are emerging [4]. As a consequence, this leads to heavy workload on the servers, which causes degradation to the overall performance of cloud system. In a distributed cloud computing system, the data-intensive computing may have to deal with huge amount of information and information must be available in the data centre and they cannot be moved. The data scheduling becomes a problem in cloud computing as huge bulk of data is available in a short network bandwidth.

The load balancing mechanism is the concept of dividing the load over a separate system for achieving the overall improvement in performance [5]. Without load balancing, it is very much difficult to manage the data access in cloud computing. The appropriate load balancing in cloud system provides the accurate allocation of the tasks with respect to the capacity of each system. In the distributed system, the task allocation may be either static or dynamic. The task allocation algorithm requires

✉ P. M. Rekha
rekhapm12@gmail.com

M. Dakshayini
dakshayini.ise@bmsce.ac.in

[1] Department of Information Science & Engineering, BMSCE, Bangalore 560019, Karnataka, India

[2] Department of Information Science & Engineering, JSS Academy of Technical Education, Bangalore, Karnataka, India

complete information related to workloads such as service time and the admission rates of the appropriate documents. The dynamic work allocation algorithm generates an online file disk allocation scheme to get adjust to a changing workload pattern without having a past information of the document to be allocated in future. The dynamic work allocation is done when the size of the task is comparatively small such as the case in web proxy caching [6]. Also, in cloud computing environment, a logical network which consists of a set of virtual machines must be deployed on to the physical network. In cloud computing system, several virtual machines are allocated with only a certain amount of data when the huge amount of data enters cloud [7]. The priority and extended priority-based Round-Robin service broker algorithms allocate the requests on the Position of data centres and give improved performance than the conventional Random selection algorithm [8]. Assigning of tasks among various virtual machines is done randomly based on the system capacity. The total response time for a particular request act as a major performance evaluation parameter in cloud data centre [9]. The Genetic Algorithm (GA) is computational intelligent algorithm which provides the most optimal solution for extensive multi-objective optimization problems. Therefore, in this research, Genetic Algorithm based Efficient Task Allocation technique is proposed (ETA-GA) for efficient allocation of tasks among the virtual machines in accordance with the data size in the cloud environment. This technique aids the system in making precise decisions for allocating the tasks among cloud nodes. This genetic algorithm based task allocation technique ensures the optimum distribution of user tasks. By deploying genetic algorithm for task allocation, the scheduler obtain a shorter makespan for the jobs allocated compared to the previously available scheduling policies and simultaneously achieves a better-balanced load scheduling system across all the connected nodes in the cloud network.

The task allocation algorithm is a complex process since it must schedule the huge number of tasks to fit in the available resources. While designing an effective task allocation algorithm, many parameters with respect to user and also the provider need to be considered. From the user perspective task completion time, cost, and response time and from the provider perspective resource utilization, fault tolerance, and power consumption need to be taken into account. The genetic algorithm based task allocation technique proposed in this paper makes the precise decisions considering all these parameters in assigning tasks to appropriate resources. A simulation results have shown the reduced makespan, execution time and also balanced load over the VMs. The proposed approach is evaluated in terms of these parameters to prove the improved performance in comparison with other decision strategies. The remainder of this paper is organized as follows: Second Section describes the detailed related work done. The third Section describes the proposed methodology. The fourth Section presents the system model, Fifth Section describes the evaluation methods and results obtained with the proposed system. The sixth section summarizes the conclusions.

2 Related work

In this section, different heuristic, meta-heuristic, and hybrid algorithms for task scheduling in cloud computing system are reviewed.

Xu et al. investigated the resource scheduling algorithms for virtual machine load balancing in cloud computing environment. Presented classifications based on a comprehensive study on existing Virtual Machine load balancing algorithms. The existing load balancing algorithms were analysed and classified for the purpose of providing an overview of a characteristic of the related algorithms. Detailed discussions of various algorithms were provided and also aimed to offer a complete understanding of the existing algorithms as well as added awareness into the field's future scope [10]. Radhakrishnan et al. proposed the methodologies in IaaS model to reduce the migration of the virtual machines namely VMMDA and RDFA. Based on resource demand from user timely decision of migration of virtual machine is monitored in VMMDA, in turn, the RDFA supported to the VMMDA for finding the appropriate destination of migrated Virtual Machines. Based on the resource demand further, the new destination is selected. Both the methodologies utilized genetically weight optimized Artificial Neural Network in order to perform their task effectively. The Artificial Neural Network is used to guess the upcoming workload of the computing hosts in cloud data centre based on previous workloads. The Cloud Analyst simulator has been used to evaluate the performance of the proposed methodology against the existing methodologies. The proposed methodology showed that Virtual Machine management minimizes the data centre processing time and response time of customer applications when compared with the existing methodologies [11]. Balagoni et al. proposed a Locality Load Prediction Aware Multi objective Task Scheduling algorithm for the dynamic cloud environment. It is an optimal task scheduling algorithm which provides minimum task transfer time, task waiting time, task execution time, and task completion time than the existing algorithms. The experimental results showed that the proposed Locality-Load-Prediction Aware Multi-Objective Task Scheduling algorithm outperformed the existing deadline aware scheduling, load-aware scheduling, and energy-aware scheduling algorithms in

terms of total completion of the task [12]. Yang et al. studied the joint optimization of service placement and load dispatching in mobile cloud systems. An efficient heuristic algorithm is designed to Basic Service Placement Problem and a set of competitive benchmark algorithms. From the results, the heuristic algorithm outperformed the benchmark algorithms in terms of access latency and algorithm run time. Based on the study of the BSPP the problem was extended to a more practical model known as Cost-aware Service Placement Problem. Authors also developed an outline algorithm to the problem that can be deployed in practical systems. The results showed that the superior performance in access latency and cost of the service providers can be achieved with outline algorithm [13]. Zhan et al. presented taxonomy for managing and scheduling cloud resource in the application layer, virtualization layer and in the deployment layer. The landscape of the cloud resource scheduling problem and its state of art solutions has been briefly reviewed. A comprehensive survey has been offered analytically according to the two-levelled taxonomy. The existing challenges and the future research directions have been discussed that included real-time scheduling, adaptive dynamic scheduling, large-scale scheduling, multi-objective scheduling and distributed and parallel scheduling [14]. Piraghaj et al. proposed Policies namely RRA, URA, FqRA, AvgRA, MeRA and ThqRA policies for estimating task populations residing in each VM type. In RRA policy the tasks were assigned to VMs based on their average requested resource. This policy was the baseline for the future comparison of the result since it was solely based on the requested resources submitted to the data center. Resource allocation in URA policy was based on an average resource utilization of task clusters obtained from historical data. In other four policies, the assignment was based on the four estimates extracted from the virtual machines usage logs from the URA policy. The extracted estimates were average, median, first and third quantile of the number of tasks that could be accommodated in a virtual machine without causing any rejections. RRA, URA, FqRA, AvgRA, MeRA and ThqRA policies are compared and showed improvement in the total energy consumption of the data centre [15]. Xu et al. proposed a mathematical model of scheduling data among the data centres in cloud computing. Authors implemented roulette-wheel selection to select the suitable individuals with high fitness value and low fitness value individuals are removed. Location of datasets is changed with the crossover and mutation operations. Datasets were increased, exhaustive search algorithm became infeasible because of the computation complexity. Then the data scheduling between data centres of approximate optimal solutions searched by the genetic algorithm with the results searched by Monte Carlo algorithm when the number of datasets was large, the optimization time of each algorithm was also compared [16]. Lin et al. proposed a task prioritizing algorithm based on priority to rank the order of tasks also proposed a Scalable-Heterogeneous-Earliest-Finish-Time algorithm to schedule workflows for the elastically varying compute resources. The experiments proved in optimizing workflow execution time, scale resources elastically during workflow execution for a Cloud computing environment [17]. Kumar et al. developed a method for generating initial population using the Min–Min and Max Genetic algorithm to get better initial population and further enhanced standard Genetic Algorithm. The initial population is produced randomly, resulting in different schedules with less fit, and fewer chances of producing the better child. The makespan of the Improved Genetic Algorithm was less compared to Standard Genetic Algorithm which proved in reducing the overall execution time of the tasks and in proper utilization of resources [18].

Authors presented task scheduler model using Genetic algorithm scheduling function for each task scheduling cycle. Based on availability of VMs and user demand, the function is created to set of task schedules and the quality of each task schedule is evaluated. This function iterates genetic operations to get an optimum task schedule. The developed model proved the effectiveness and efficiency in comparison with present task scheduling models, namely the round-robin task scheduling model, the load index-based task scheduling model, and the ABC based task scheduling model. Performance parameter comparisons were based on throughput, response time, virtual machine utilization, processing cost, and user satisfaction [19]. Kaleeswaran et al. presented Dynamic scheduling of data using genetic algorithm in cloud computing using Ubuntu Enterprise Cloud. The tasks were scheduled based on the computation and memory requirement. The scheduling of tasks was done by first sorting all the tasks and then the first task was chosen from the queue to allocate the resource that will best fit using the Genetic Algorithm. Once scheduling is done finally the data is stored in the cloud [20]. Mamat et al. proposed a genetic algorithm by using condition to speed up the mapping process and guarantee the task deadlines with real datasets collected as a cloud benchmark. Batch mapping with throughput as a fitness function is used to map jobs to cloud resources Mapping time and makespan are the performance metrics used to evaluate the proposed system. Results proved to be better compared to MCT algorithm [21].

Initially, VM resources are not chosen, based on the computed probability, the algorithm arbitrarily chooses the physical machine which is free and then starts scheduling. Thereafter when VM resources increase, the algorithm computes load and variance of every physical machine based on historical information and the current state. The

genetic algorithm provides the best solution to meet predefined constraints and chooses one with least cost [22]. Kaur et al. developed new approach by combining Shortest Cloudlet to Fastest Processor, Longest Cloudlet to Fastest Processor. Selected meta-heuristic Genetic algorithm optimization method for task scheduling. The initial population was modified by using the stochastic operators of genetic algorithm which lead to achieving better efficiency. Single user jobs were considered to achieve the time minimization. They claim that this algorithm could be implemented on both task and resource scheduling [23]. Dakshayini et al. proposed scheduling policy based on priority and admission control to satisfy the user requests by providing QoS. User requests were scheduled based on the deadline of the service-request by allowing the cloud to accept the service-requests only if the cloud can offer the service to meet desired QoS [24].

Ge et al. proposed Genetic Algorithm to optimize the tasks scheduling in the job queue using a centralized scheduler to allocate the waiting tasks to the different available resources based on the resources status messages. Results showed that the proposed schedule was improved than the First-In-First-Out and the delay scheduling method [25]. Lin et al. proposed a scheduling algorithm for big data workflows in the multi-cloud environment. Algorithm aimed at minimizing the cost of workloads by considering partial critical paths. Parameters such as the charge per time interval, instance types from various cloud providers, homogeneous and heterogeneous inter bandwidth were compared for different workflows and proved better [26]. Task scheduling is a significant part in deciding the performance of data centre which affect the Quality of Service of cloud [27]. Most of the above-discussed approaches have mainly focused on different techniques for efficient application execution in cloud environment, namely resource allocation methods, application partition, tasks migration and replacement approaches. But none of these proposed solutions have considered the better decision-making strategy for improving the task allocation efficiency and attaining the minimized task completion time in cloud environment. Therefore, a Genetic algorithm based decision making technique for efficient task allocation approach is presented here to solve the resource optimization problem by making precise decisions in assigning tasks among the processing nodes in the Cloud.

3 Genetic algorithm based efficient task allocation approach

3.1 Design of decision making model

In the cloud computing environment, user applications get executed in a distributed manner. In such environment, service providers would have to guarantee the quality of service to the users, as users expect their task to be processed with minimum time and cost. Each user application is divided into multiple tasks and resources need to be allocated for smooth processing of each of these tasks which are distributed among the virtual machines in the data centre. Each virtual machine (VM) may take different time to complete the processing of the task based on the number and type of tasks it is processing. Hence before execution, tasks need to be allocated and reallocated to required-resource-rich computing nodes in the cloud environment. So, a precise decision must be made in allocating resources for effective processing of tasks based on the availability of resources to satisfy the user requirements. Therefore, the optimum and right decision making technique for efficient allocation of tasks among the virtual machines is essential for accomplishing execution efficiency and better quality of service.

3.2 Task scheduling approach

User requests arrive in a Poisson distribution pattern and are placed into a Queue of Tasks and these tasks are considered as cloudlets (CLs). The task scheduler allocates resources to these cloudlets using the scheduler module to meet the requirements of the users. The spot-on decision is made to achieve the best allocation VM with maximum processing efficiency using genetic process so that the task is comparatively finished with the minimum time consumption in the cloud computing system. Let (VM_1, VM_2, VM_3, VM_4, VM_5…. VM_n) be the set of virtual machines available at the data centre to process the set of tasks or cloudlets (CL_1, CL_2, CL_3, CL_4, CL_5, CL_6,… CL_m) as shown in Fig. 1. Assume that all these VMs are running in parallel and are connected to each other. These VMs run with their own resources that are pre-allocated and shared with other VMs on hosts in the data centre. During execution, if any task experiences the insufficiency of resources, it brings the same to the notice of the scheduler which dynamically reallocates the resources among the VMs and aids in completing the task execution.

The quantitative analysis is made with the following assumptions:

(1) Quantitative analysis is within the range of code instruction length.
(2) The arrival of tasks is the Poisson distribution.
(3) Each node executes one task at a time.

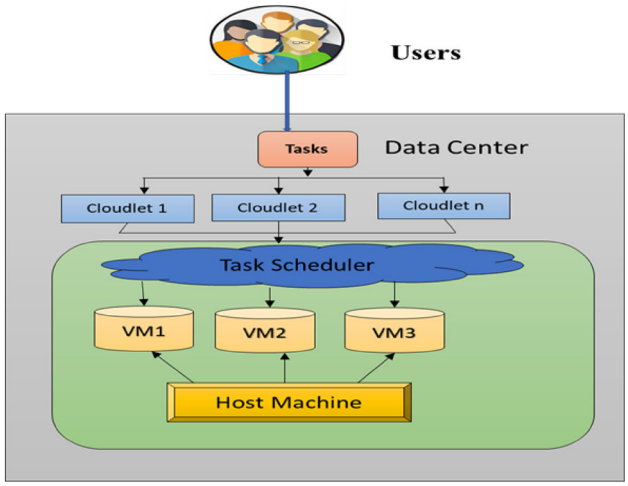

Fig. 1 GA based efficient task allocation model

Two scheduling targets are designed to:

(1) Reduce the task finishing time
(2) Improve the Resource Utilization.

3.2.1 Calculation of finish time

In ubiquitous cloud environment, all the computing nodes run in parallel that is in Cloudsim environment all the VMs start at same time. If more than one cloudlet is assigned to one VM, then all the cloudlets will be executed one after the other. A Two dimensional (2D) time array is built by calculating the fitness value (FV) or FT of each CL on every VM using fitness function based on the computing capability of virtual machines. Then for each VM, the finish time(FT) is calculated by considering the sum of completion time of all CLs allocated to that VM. In other words, it is the finish time of the last CL allocated for that VM. The total time T consumed by cloudlet j (CL_j) at ith VM (T^i_{CLj}) is computed by considering the sum of its execution time ET^i_{CLj}, resource reallocation time that is based on the input size $\boldsymbol{CL^{isz}_j}$ and output size $\boldsymbol{CL^{osz}_j}$ of cloudlet j and the network Bandwidth Bw_{ij}.

Considering n number of VMs and m number of Cloudlets

Algorithm for finding Minimum Finishing Time

// Building 2D time array of FV or FT of each CL on every VM
For j = 0 to m // number of cloudlets to be scheduled
  For i= 0 to n //number of virtual machines
    Construct the 2D time array with FT^i_{CLj}
  End for
End for
// calculating the finish time of the last CL allocated on every VM
For i = 0 to n // number of virtual machines
  For j= 0 to m // number of cloudlets to be scheduled
    $FT_i = \sum_{j=0}^{m} CL_j * E(i,j)$ //finish time of the last CL allocated in the VMi
    $T^i_{CLj} = ET^i_{CLj} + \frac{\left(CL^{isz}_j + CL^{osz}_j\right)}{BW^i_j}$
  End for
End for
//Allocation of CL to the VM that finishes its processing at the earliest
While (all cloudlets are assigned to suitable VM)
For each unscheduled cloudlet
For j = 0 to m // number of cloudlets to be scheduled
  For i= 0 to n //number of virtual machines
    Find the VM_i^{MinFT} (CLj) // VM that offers min FT for CLj
    $(i, j) = 1$ // when CLj is assigned to VM_i
  End for
End for

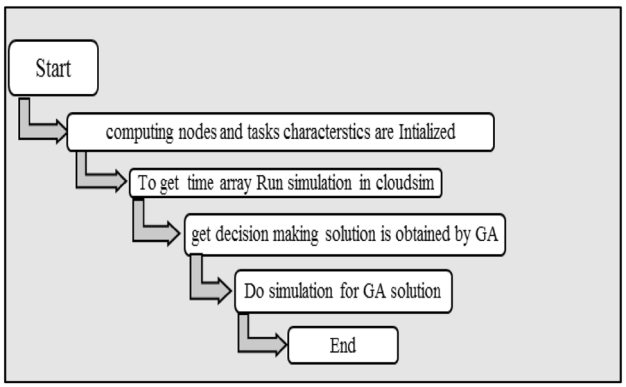

Fig. 2 Flow diagram for GA based decision making

4 Efficient task allocation genetic algorithm

Genetic algorithm is used to obtain optimized solution from many candidate solutions. Solution is decided based on the fitness value which is calculated using fitness function called an individual or a chromosome. A chromosome relates to a unique solution in the solution space. Further GAs operates with a collection of chromosomes, called a population, and uses two operators namely crossover and mutation to generate new solutions from existing ones.

Then simulations are carried out to gather tasks finishing time run on all VMs. With this time information GAs gives optimized solutions for task allocation. The entire task allocation design process, and the working flow of GAs-based decision making is depicted in Fig. 2.

4.1 Fitness function

The fitness function is the function that assesses the dominance of individual chromosome and then the evolution of the next generations are decided. Each individual chromosome is denoted by fitness and individual with high fitness has a better chance to survive. For each generation, the fitness value of each individual in the population is estimated, higher fitness value individuals are nominated from the current population, then crossover and mutation operator are used to form a new generation. The new generation of solutions is then used in the succeeding iteration of the algorithm.

The overall Fitness value of the chromosome is computed using Eq. (1) considering the total time taken to complete the schedule (α) and the failure probability(β).

$$Fitness\_chromosome_i = \alpha(Total\_time_i) + \beta(FP_i) \quad (1)$$

where $Total\_time = \sum_{i=i-n} \frac{T\_Len}{VM\_MIPSi}$, $\alpha + \beta = 1$, VM_MIPS_i is defined as millions of instructions per second for each processor of VM_j, FP_i is the network delay between nodes

4.2 Encoding method

In the encoding method, vectors of VMs are considered as chromosomes that is $V = [V_1, V_2, \ldots V_i \ldots, V_n]$ (where n is the number of decision variables) to represent a solution. i is a natural number, acts as a pointer to the ith VM VM_i to which the jth cloudlet is assigned in the sequence of VMs. Each VM in the chromosome denotes the gene with which the CL is running. Total number of possible allocation schemes available are depending on the number of VMs and CLs being used. If there are 25 CLs and 12 VMs then there are 12^{25} allocation schemes are available. The proposed approach ETA-GA selects the best allocation scheme based on the finishing time of the CL to achieve the maximize efficiency.

For the hypothetical scenario considered with 12 VMs and 25 CLs, initially the 2D time array is constructed by calculating the fitness value (FV) or FT of each candidate (CL) on each VM using fitness function based on the computing capability of virtual machines. ETA-GA forms the chromosome by selecting the fittest VM from each row using roulette wheel selection method. Each row of this array represents the length of available allocation schemes for each candidate (CL) shown in Table 1. From which, the

Table 1 Two dimensional time array

VM CL	0	1	2	3	4	5	6	7	8	9	10	11
0	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV
1	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV
2	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV
3	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV
4	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV
5	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV
6	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV
7	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV
8	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV	FV

following vector of 12 VMs is selected by ETA-GA as an optimal chromosome for efficient task allocation shown in Table 2.

V = [11, 8, 4, 5, 4, 0, 2, 1, 7, 11, 5, 2, 3, 5, 4, 4, 6, 11, 10, 3, 2, 4, 4, 5, 6]

That is, ETA-GA selects the twelfth VM whose sequence number is 11 to execute Cloudlet 0 and selects the ninth VM with sequence number 8 to execute Cloudlet 1 and so on, the seventh VM is selected as processing node of the last CL.

be changed by changing the value in create cloudlet () function. The algorithm is tested by setting the parameters as shown in Table 3 and considering two hosts of different capacities of resources as given in Table 4.

Simulation has been carried out with two different cases, considering varied sets of CLs and VMs for each case. For all this, initially a 2D time array is constructed by calculating the fitness value of each candidate (CL) on each VM of every combination using fitness function shown in Table 5. The decision has to be made to identify which

The Efficient Task Allocation Genetic Algorithm [ETA-GA] procedure is described below:

Input: Population, Fitness function Generations, Crossover percent, Mutation percent, Gene length

Output: Tasks allocation, Minimum Finishing time, Maximum resource utilization

1. [Start] Initialize random population of n chromosomes, Set POP = pop size, the algorithm begins task allocation by choosing the most suitable, free, appropriate VM

2.[Fitness]calculate the fitness chromosome value of every chromosome in the given population

3. [New population] produce the new population by repeating the subsequent steps till the creation of new population is done.

   3.1. [Selection] select two parent individuals from the population, calculate chromosome with lowest fitness and eliminate the chromosome with highest fitness.

   3.2. [Crossover] A new fittest chromosome is generated using multi-point crossover by interchanging the set of schedules between two chromosomes by using the crossover probability, the new offspring by reforming the parents is generated. The two fittest chromosomes are selected

   3.3. [Mutation] bits are changed from 0 to 1 or 1 to 0 with a mutation probability Pm. Mutate a tour using swap mutation. With the probability of mutation, mutate the new child at some positions.

   3.4. [Accept] Place new off spring as new population and use this population for next round of iteration

4. [Exchange] use the new generation as the existing generation.

5. [Test] if the exit condition is satisfied then end the algorithm and return the individual to the chromosome with least fitness value is selected for schedule

6. [Loop] go to step 2.

5 Simulation and evaluation

This section illustrates the simulation setup and the results obtained with the proposed ETA-GA approach. The Proposed ETA-GA is evaluated in Java Cloudsim. The main file is executed in default package which takes the input as the number of tasks, network delay and bandwidth. The number of tasks refers to the number of cloudlets that can cloudlet can be executed in the VM and how long it takes to finish. Time consumption may vary for different cloudlets running on the same VM and also same tasks may take different time to finish its execution in different VMs. Faster the VM lesser the task execution time. Bigger the

Table 2 Allocation of the task in the form of chromosome

CL_0	CL_1	CL_2	CL_3	CL_4	CL_5	…	CL_{24}
VM_{11}	VM_8	VM_4	VM_5	VM_4	VM_0	…	VM_6

Table 3 Parameters for GA

Population size	50
Max evaluations	500
Cross-over operator	Single point
Crossover probability	Pc
Mutation operator	Bitflip
Mutation rate	0.15

Table 4 Hosts characteristics

Host characteristics		
HostId	0	1
Datacenter	Hostdc	Hostdc
RAM	32,768	2048
PesN	1	2
MIPS	1,000,000	100,000
Storage	100,000	10,000
BW	10,000	1000

tasks more the time it consumes and with good network bandwidth.

Simple allocation method is a sequential allocation policy, where all tasks are assigned to group of virtual machines in sequential order say the first task to first VM, second task to second VM and so on, when all VMs has been assigned with one task, next round will assign next task to first VM and so on. In greedy policy, matrix time [i][j] gives execution time of $task_i$ to VM_j. Tasks are sorted in descending order with respect to length and VMs are sorted in ascending order with respect to processing capacity. After sorting, the first task is assigned to last VM, task execution is completed with minimum time. The results are compared with simple allocation and greedy method shown in Tables 6, 7, and 8.

From the above three tables the ETA-GA approach takes 1262.96-time units to finish all of 12 tasks while the other two methods take 1670.74 and 1280.96-time units. Hence, the proposed ETA-GA approach can do the best decision and, user tasks can be finished as fast as possible.

Table 6 Simple allocation method

Cloudlet ID	VM ID	TIME	Start time	End time
0	0	1.2	1649.14	1650.34
1	1	8	1650.34	1658.34
2	2	1.2	1658.34	1659.54
3	3	1.2	1659.54	1660.74
4	4	1.2	1660.74	1661.94
5	5	1.2	1661.94	1663.14
6	6	1.2	1663.14	1664.34
7	7	0.8	1664.34	1665.14
8	8	0.8	1665.14	1665.94
9	9	0.8	1665.94	1666.74
10	10	0.8	1666.74	1667.54
11	11	0.8	1667.54	1668.34
12	0	0.8	1668.34	1669.14
13	1	0.8	1669.14	1669.94
14	2	0.8	1669.94	1670.74

Apart from the above-mentioned scenario discussed, implementation is further carried out for two different cases by varying CLs and VMs and are described below.

Case 1: In 1st case keeping the number of VMs constant as 12 and varying the number of CLs as 15, 20 and 40 for each iteration, total FT or makespan has been calculated for each combination of VMs and CLs [(12,15), (12,20) and (12,40)] and compared with the existing

Table 5 Cloudlet to VM execution time (12VMs, 14CLs)

Time	VMID 1											
CLID	0	1	2	3	4	5	6	7	8	9	10	11
0	4.4	4.2	4.1	1.9	1.05	1.2	1.2	0.5	0.5	0.5	0.4	0.4
1	4.6	4.3	4.1	4.1	1.15	1.15	1.4	1.4	0.6	0.7	0.7	0.7
2	4.8	3.8	5.8	4.8	5.8	1.8	3.8	2.8	1.8	1.2	1.7	0.8
3	6.2	6.2	6.2	7.2	3.2	6.2	6.2	6.2	6.2	3.2	6.2	1.2
4	5.5	5.5	5.5	1.5	1.5	5.5	5.5	4.5	3.5	5.5	5.5	2.2
5	6.4	6.4	4.4	5.4	4.4	6.4	6.4	5.4	4.4	6.4	6.4	1.4
6	7.2	7.2	3.2	4.2	6.2	7.2	7.2	7.2	6.2	5.2	7.2	4.2
7	10.5	8.5	6.5	7.5	8.5	10.5	10.5	8.5	7.5	7.5	10.5	5.5
8	12.5	12.5	11.5	9.5	9.5	12.5	12.5	7.5	10.5	11.5	12.5	6.5
9	12.8	20.5	12.23	10.25	10.25	10.25	12.56	8.5	11.25	12.23	13.25	13.25
10	15.5	45.23	12.25	12.25	50.26	55.25	54.20	55.23	54.21	25.23	28.35	24.25
11	20.5	58.25	42.23	20.25	100.25	58.26	55.28	58.63	53.62	58.65	56.23	58.25
12	50.25	100.25	85.25	45.25	100.52	110.25	100.23	102.23	110.25	135.20	132.4	138.25
13	100.5	110.23	110.25	112.25	110.23	102.53	100.25	108.45	110.23	115.23	112.23	112.23
14	110.25	110.23	110.25	100.63	110.25	115.25	114.23	112.23	110.54	110.25	110.25	110.1

Table 7 Greedy method

Cloudlet ID	VM ID	TIME	Start time	End time
14	3	5.71	1265.81	1271.53
13	5	2.86	1261.1	1263.96
12	11	5.71	1269.1	1274.81
11	10	5.71	1269.1	1274.81
10	10	0.43	1274.81	1275.24
9	9	5.71	1270.81	1276.53
8	8	5.71	1261.1	1266.81
7	7	0.43	1276.53	1276.96
6	5	5.71	1271.53	1277.24
5	4	0.43	1266.81	1267.24
4	3	5.71	1263.96	1269.67
3	4	2.86	1276.96	1279.81
2	2	0.43	1279.81	1280.24
1	0	5.71	1274.81	1280.53
0	1	5.71	1275.24	1280.96

Table 8 Enhanced task allocation genetic algorithm method

Cloudlet ID	VM ID	TIME	Start time	End time
3	3	0.43	1251.1	1251.53
5	9	0.43	1251.1	1251.53
6	11	0.43	1251.1	1251.53
9	8	0.43	1251.53	1251.96
15	10	5.71	1251.1	1256.81
13	7	5.71	1251.53	1257.24
8	6	5.71	1251.53	1257.24
12	5	5.71	1251.96	1257.67
11	3	0.43	1257.24	1257.67
7	2	0.43	1257.67	1258.1
2	2	0.43	1258.1	1258.53
14	0	2.86	1256.81	1259.67
1	1	2.86	1257.24	1260.1
4	4	2.86	1259.67	1262.53
3	11	0.43	1262.53	1262.96

approaches(Greedy and Simple allocation). The makespan obtained for the first combination of 12 VMs and 15 CLs using ETA-GA and comparison study made are shown in Table 9.

Case 2: In 2nd case, the number of VMs and the CLs are both varied, total FT or makespan has been calculated for each combination of VMs and CLs [(8,24), (12,12) and (10,15)] and compared with the existing approaches (Greedy and Simple allocation). The makespan obtained with all 3 combinations of VMs and CLs using ETA-GA and the comparison study made with Greedy and Simple allocation methods are shown in Table 10.

Table 9 Makespan comparisons for three sets of cloudlets for 12 VMs

Cloudlets	Simple allocation method	Greedy method	ETA-GA
15	1670.74	1280.96	1262.96
20	1870.82	1420.96	1300.86
40	2600.52	2300.85	2100.96

Table 10 Comparisons of makespan with different policies for different sets of cloudlets and vms

Configurations	ETA-GA	Simple allocation method	Greedy method
8 VMs, 24 CL	140.5	141.5	200.14
12 VMs, 12 CL	100.11	100.2	1200.14
10VMs,15 CL	65.15	64.15	110.73

The obtained results for two different cases by varying VMs and CLs shows that ETA-GA decision can attain better task allocation scheme. This indicates that choosing genetic algorithm as decision making algorithm for task allocation is efficient compared to other methods.

5.1 Performance metrics

The performance metrics used for comparative analysis of ETA-GA are based on makespan and throughput. The performance metrics are discussed below:

Makespan is used to evaluate the minimum completion time using Eq. (2), by estimating the finishing time of the latest task when all tasks are scheduled. The demand will not be completed on time if the makespan of specific cloudlet or task is not minimized.

$$Makespan = MAX_{taski}(FT) \tag{2}$$

Throughput: Tasks completion in a certain time period as in Eq. (3), minimum throughput is required for task scheduling.

$$T = \sum_{task}^{i} (ExeTime) \tag{3}$$

where *ExeTime* shows the execution time of an ith task.

In Fig. 3 the comparison of makespan of simple allocation, greedy approach, and ETA-GA with x-axis as number of cloudlets and the y-axis as Makespan. When the

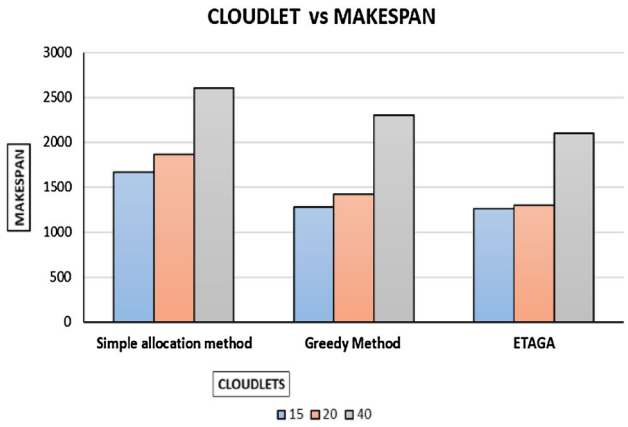

Fig. 3 Comparisons of three allocation methods

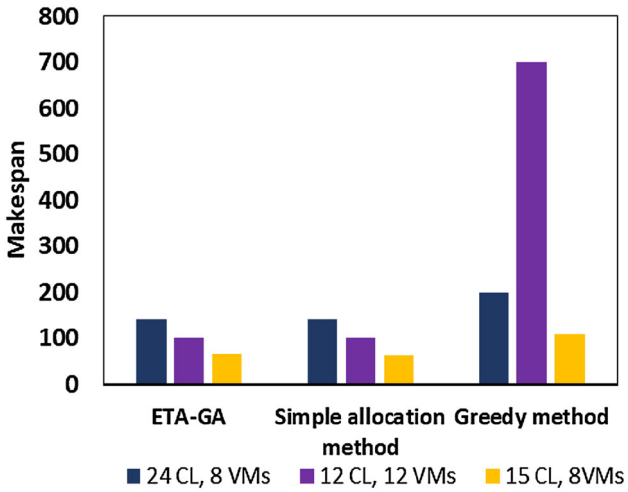

Fig. 4 Comparison of makespan for three sets of cloudlets and VMs

numbers of cloudlets are less, then ETA-GA algorithm give better Makespan.

Comparisons with different policies for different sets of cloudlets and VMs are shown in Fig. 4. As the number of cloudlets increases, proposed ETA-GA produces still improved Makespan time hence the quality of performance is enhanced. If the makespan of specific cloudlet or task is not minimized then the demand will not be completed on time.

Figure 5 depicts the comparison of throughput achieved with simple allocation, greedy approach, and ETA-GA. The x-axis represents the number of cloudlets and Y-axis represents the throughput. The simulation results evidently prove that the ETA-GA approach attained the better throughput when compared with Greedy and simple allocation algorithms.

After assessing the performances of heuristic approaches, ETA-GA is most optimum method in achieving the

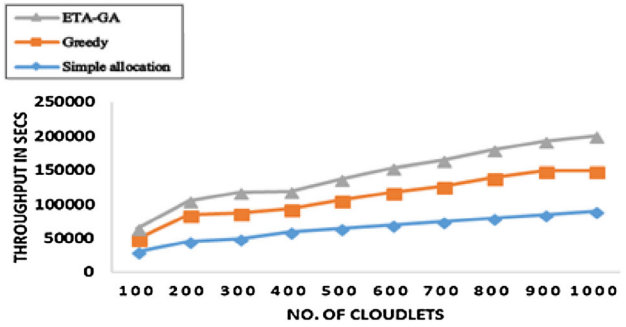

Fig. 5 Comparison of throughput

makespan and throughput for attaining the optimal task scheduling in cloud computing.

6 Conclusion

When the huge amount of data enters the cloud, the workload of the cloud system increases as the huge amount of data needs to be stored and processed. Decision making for task allocation is one of the important issue in cloud computing. In this paper, the problem has been addressed by implementing ETA-GA approach for obtaining optimized task allocation. The proposed approach has improved the performance of cloud system comparing with the existing studies where the task allocation was carried out using heuristics and random search by cloud server. The results obtained by the proposed approach is feasible with acceptable performance compared to other approaches. Further work could be considered with parameter's such as energy consumption and resources requirement in decision making process.

References

1. Ge, J., He, Q., Fang, Y.: Cloud computing task scheduling strategy based on improved differential evolution algorithm. In: International Conference on Computer-Aided Design, Manufacturing, Modeling and Simulation. AIP, Melville (2017)
2. Aggarwal, M., Kumar, N., Kaushik, A.: Review of research issues in cloud computing. Int. J. Appl. Eng. Res. **9**(21), 9479–9488 (2014)
3. Prasad, R.B., Eunm, C., Lumb, I.: A taxonomy and survey of cloud computing systems. NCM 2009: 5th International Joint Conference on INC, IMS, and IDC, pp. 44–51 (2009)
4. Wickremasinghe B., Calheiros, R. N., Buyya, R.: Cloud analyst: a cloudsim-based visual modeller for analysing cloud computing environments and applications. In: Advanced Information Networking and Applications, pp. 446–452 (2010)
5. Manvi, S.S., Shyam, G.K.: Resource management for Infrastructure as a service (IaaS) in cloud computing: a survey. J. Netw. Comput. Appl. **41**, 424–440 (2014)

6. Hameed, A., Khoshkbarforoushha, A., Ranjan, R., Jayaraman, P.P., Kolodziej, J., Balaji, P., Khan, S.U.: A survey and taxonomy on energy-efficient resource allocation techniques for cloud computing systems. Computing **98**(7), 751–774 (2016)
7. Beloglazov, A., Buyya, R., Lee, Y.C., Zomaya, A.: A taxonomy and survey of energy-efficient data centers and cloud computing systems. Adv. Comput. **82**(2), 47–111 (2011)
8. Mishra, R.K., Bhukya, S.N.: Service broker algorithm for cloud-analyst. Int. J. Comput. Sci. Inf. Technol. **5**(3), 3957–3962 (2014)
9. Ge, Y., Wei, G.: GA-based task scheduler for the cloud computing systems. Web Inf. Syst. Min. **2**, 181–186 (2010)
10. Xu, M., Tian, W., Buyya, R.: A survey on load balancing algorithms for VM placement in cloud computing. Concur. Comput. **29**(12), e4123 (2017)
11. Radhakrishnan, A., Kavitha, V.: Energy conservation in cloud data centres by minimizing virtual machines migration through artificial neural network. Computing **98**(11), 1185–1202 (2016)
12. Balagoni, Y., Rao, R.R.: Locality-load-prediction aware multi-objective task scheduling in the heterogeneous cloud environment. Indian J. Sci. Technol. **10**(9), 1–9 (2017)
13. Yang, L., Cao, J., Liang, G., Han, X.: Cost-aware service placement and load dispatching in mobile cloud systems. IEEE Trans. Comput. **65**(5), 1440–1452 (2016)
14. Zhan, Z.H., Liu, X.F., Gong, Y.J., Zhang, J., Chung, H.S.H., Li, Y.: Cloud computing resource scheduling and a survey of its evolutionary approaches. ACM Comput. **47**(4), 63 (2015)
15. Piraghaj, S.F., Calheiros, R.N., Chan, J., Dastjerdi, A.V., Buyya, R.: Virtual machine customization and task mapping architecture for efficient allocation of cloud data centre resources. Comput. J. **59**(2), 208–224 (2016)
16. Xu, Q., Xu, Z., Wang, T.: A data-placement strategy based on genetic algorithm in cloud computing. Int. J. Intell. Sci. **5**(03), 145 (2015)
17. Lin, C., Lu, S.: Scheduling scientific workflows elastically for cloud computing. In: Proceedings of the IEEE 4th International Conference on Cloud Computing, Washington, DC, USA (2011)
18. Kumar, P., Verma, A.: Independent task scheduling in cloud computing by improved genetic algorithm. Int. J. Adv. Res. Comput. Sci. Softw. Eng. **2**(5), 111–114 (2012)
19. Jang, S.H., Kim, T.Y., Kim, J.K., Lee, J.S.: The study of genetic algorithm based task scheduling for cloud computing. Int. J. Control Autom. **4**(5), 157–162 (2012)
20. Kaleeswaran, A., Ramasamy, V., Vivekananda, P.: Dynamic scheduling of data using genetic algorithm in cloud computing. Int. J. Adv. Eng. Technol. **5**(2), 327–334 (2013)
21. Mehdi, N.A., Mamat, A., Ibrahim, H., Subramaniam, H.K.: Inpatient task mapping in elastic cloud using genetic algorithm. J. Comput. Sci. **7**(6), 877–883 (2011)
22. Gu, J., Hu, J., Zhao, T., Sun, G.: A new resource scheduling strategy based on genetic algorithm in cloud computing environment. J. Comput. **7**(1), 42–52 (2012)
23. Kaur, S., Verma, A.: An efficient approach to genetic algorithm for task scheduling in cloud computing environment. Int. J. Inf. Technol. Comput. Sci. **10**, 74–79 (2012)
24. Dakshayini, M., Guruprasad, H.S.: An optimal model for priority-based service scheduling policy for cloud computing environment. Int. J. Comput. Appl. **32**(9), 23–29 (2011)
25. Ge, Y., Wei, G.: GA-based task scheduler for the cloud computing systems. Proc. Int. Conf. Web Inf. Syst. Min. **2**, 181–186 (2010)
26. Lin, B., Guo, W., Xiong, N., Chen, G., Vasilakos, A., Zhang, H.: A pre-treatment workflow scheduling approach for big data applications in multi-cloud environments. IEEE Trans. Netw. Serv. Manage. **13**(1), 1–12 (2016)
27. Kumar, N., Aggarwal, M., Kumar, R.: A comparative analysis of scheduling algorithms affecting QoS in cloud environment. Int. J. Comput. Sci. Netw. **4**(1), 142–147 (2015)

P. M. Rekha Research Scholar, BMSCE, Working as Assistant Professor at Department of Information Science & Engineering, JSS Academy of Technical Education, Bangalore, Karnataka, India. Area of Interest include Cloud computing, Internet of things, Microcontroller, Embedded sytems. Published Number of journals, more than a decade experience in teaching field.

M. Dakshayini has two decades experience in teaching field. She has published many papers. Currently working as Professor and Head in the Department of Information Science and Engineering at BMS College of Engineering, Bangalore, India. Area of interest include Networks & Communications, Internet of Things, Cloud Computing, Data Analytics, Algorithms.