Arab J Sci Eng
DOI 10.1007/s13369-017-2766-x

RESEARCH ARTICLE - SPECIAL ISSUE - COMPUTER ENGINEERING AND COMPUTER SCIENCE

GA-Based Customer-Conscious Resource Allocation and Task Scheduling in Multi-cloud Computing

Tamanna Jena[1] · **J. R. Mohanty**[2]

Received: 5 April 2017 / Accepted: 27 July 2017

Abstract Resource allocation in multi-cloud computing is a complicated chore; there are many constraints and configuration in accordance with cloud providers as well as cloud customers. Mapping the incoming job request to available virtual machines (VMs) is a non-polynomial complete problem as the nature of traffic quite arbitrary. Customer requirements and capacity of applications change frequently. To bridge the gap between frequently changing customer requirement and available infrastructure for the services, we propose Genetic Algorithm-based Customer-Conscious Resource Allocation and Task Scheduling in multi-cloud computing. The algorithm is basically divided into two phases, namely genetic algorithm-based resource allocation and shortest task first scheduling. The objective is to map the tasks to VMs of the multi-cloud federation in order to have minimum makespan time and maximum customer satisfaction. Rigorous experiments were done on synthetic data and compared the simulation results with the existing scheduling algorithm. Results of simulation illustrate that the proposed algorithm outrun the existing ones as per concerned metrics.

Keywords Resource allocation · Genetic algorithm · Heterogeneous multi-cloud computing · Virtual machine · Task scheduling

✉ Tamanna Jena
tamannasinghdeo@gmail.com

J. R. Mohanty
jnyana1@gmail.com

1 School of Computer Engineering, KIIT University, Bhubaneswar, India

2 School of Computer Application, KIIT University, Bhubaneswar, India

1 Introduction

Multi-cloud computing is a technical paradigm, which provides resources of various capacities to customers. Each job request receives unique job id after registration in the multi-cloud federation. For day-to-day enterprise operation, health, military, IT industry operations, etc., cloud computing is inevitable. The varied features of multi-cloud computing like on-demand services, pay-as-you-go pricing model, dynamic scaling, virtualisation and no vendor lock-in make cloud platform appealing for industries of all capacities and research society. The cloud services are provisioned in the form of storage datacentre. However, Infrastructure as a Service (IaaS) cloud uses a scheduling policy to allocate VMs to the customer requests. Amazon Web Services (AWS) use First In First Out, scheduling using batch processing. The success of cloud computing relies on load balancing, efficient scheduling and importantly collaborating among peer cloud service providers to form a reliable federation to provide complex problem-solving techniques handling day-to-day business, scientific and engineering applications. No datacentre provides unlimited resources to accommodate dynamic scaling. Customer having high demanding applications reserves higher instances to get the timely deployment. The multi-cloud computing has its own share of challenges. In multi-cloud computing platform optimising numerous resources which are distributed in a different geographical region is challenging. Basically, a centralised management is established to achieve resource allocation followed by task scheduling. Centralised cloud broker monitors the fitness, status of VMs and the scheduling procedures. Assigning available tasks to VMs is one of the concerned problem which has taken attention from academia, business and research, assigning VMs to include two-phase processes, namely mapping and scheduling. Mapping includes allocating the incoming tasks to processing units termed VMS, followed by arranging the order of the allocated tasks. In this paper, we propose GA-based Customer-Conscious Resource Allocation and Task Scheduling (GACCRATS). The algorithm is divided into two crucial phases, namely resource allocation and task scheduling. The experimental outcomes prove that the proposed model outperforms the existing algorithms in terms of makespan time and customer satisfaction rate in the multi-cloud platform.

Outline of the paper is as follows: (1) development of the GA-based resource allocation algorithm in the multi-cloud environment. (2) Scheduling of the allocated tasks to resources using shortest task first algorithm. Simulation of proposed algorithm on synthetic data. We compared the simulation results with the preexisting algorithms, i.e. COTS algorithm and TLBO algorithm. The remaining paper is organised as follows: Sect. 2 shows the related work along with application in a multi-cloud environment. Section 3 describes the proposed cloud model along with the scheduling policy in detail. Section 4 contains the proposed algorithm, and results will be discussed in Sect. 5 along with performance metrics. The conclusion is in Sect. 6.

2 Related Work

Resource allocation is assigning tasks to resources having certain objectives like load balancing, minimum execution time, minimum cost and much more [1]. The objective of the paper is to allocate resources to incoming tasks in a balanced manner such that minimisation of makespan time and maximisation of throughput and customer satisfaction rate can be attained. Load balancing is one of the important criteria in resource allocation; the objective is to assign the tasks to the resources such that the processing units are neither overloaded nor idle. Scenarios used in this paper are the multi-cloud heterogeneous federation system. Our model consists of numerous physical machines which further splits into multiple virtual machines. We tried to simulate the multi-cloud, consisting of many physical machines owned by different cloud providers which are under one federation. Our model consists of three vital components: the multi-cloud federation, cloud provider and cloud user. The multi-cloud consists of multiple physical machines (mainly owned by different cloud provider). Cloud user is interested in deploying applications using the multi-cloud resources. The simulation, an incoming workload, is generated using Poisson's distribution function, where the range of rate of arrival of incoming application (lambda) varies from four to hundred. Applications are of various sizes. Each application is divided into multiple tasks. Tasks are assigned to virtual machines (VMs). Task scheduling is of an immense importance which relates to the efficiency of the entire multi-cloud computing environment. It is basically deciding the sequence of execution of the tasks by virtual machines. So both load balancing and scheduling are two approaches on two separate levels of abstraction. Resource allocation is somewhat even more abstract than load balancer and scheduler. Resource allocation involves the assignment of available tasks to VMs in an optimum manner, minimising the makespan time. It is found that multiple tasks are allocated to single VM, therefore after optimum resource allocation of an efficient task scheduling enhances the performance of the system. In major cases executing the prioritised job requests/ tasks is crucial for the system behaviour.

Scheduling of the cloud-task pair is one of the most challenging problems for distributed computing [1–9]. Customers requirement is dynamic in nature. Many existing algorithms become futile, as the requirement of users changes as well as working environments. Task scheduling algorithm based on changing customer requirement is extensively studied by many researchers. Haizea, a lease management architecture that enables resource consumers to negotiate the two kinds of leases, is described [10–14]. Tasks are broadly divided into two types, Advance Reservation (AR) and Best Effort (BE) tasks. Tasks which can reserve resource in advance are AR type; it is non-preemptive in nature. The requirement of resource needs to be mentioned clearly during the time of registration like the type of the application, processing power required for the execution of the application, duration of resource required for complete execution. Detail of requirement needs to be communicated between the cloud provider and cloud user in SLA. As applications are divided into two categories AR and BE applications, AR application further divided into numerous AR task and BE application is further divided into numerous BE tasks. AR tasks are non-preemptive in nature, whereas BE tasks are preemptive in nature.

Armstrong et al. [2] tried to allocate resources to the task and predict runtime of the incoming workload when the rate of arrival is uncertain, proposed Opportunistic Load Balancing (OLB) and Limited Best Assignment (LBA) for determining the performance of heterogeneous mapping algorithm. They suggested that machine learning algorithm (basically genetic algorithm) will be better suited in finding resource allocation to a task. Dasgupta et al. [15] used Cloud Analyst simulator for implementation and GA to evenly distribute the load and allocate tasks to VMs. Criteria they used for simulation are a homogeneous physical machine, linear nature of incoming of job request and the same priority for all jobs. The drawbacks are the same priority of task which may lead to improper usage of resources. Important applications may suffer long waiting, whereas applications which are not very important may get executed first. It is improper usage of resources. Incoming job request is not linear in a real-life scenario. Fang et al. [16] proposed two types of mapping in cloud computing, where the objective is load balancing. First phase involves mapping of the tasks to the VM, and the second phase involves mapping of VM to host. They considered response time and demand for resource to schedule task. They proposed the formula to calculate load balancing from the predicted load. Jang et al. [17] used GA for task allocation and task scheduling. They divided cloud computing into four categories (budgetary strategy of cloud computing), i.e. very high priority, high priority, mid-priority and low priority. Attributes on which these categories are divided and how they are handled differently are unclear. Javanmardi et al. [18] used hybrid model using GA and fuzzy theory for resource allocation in the cloud environment. They generate two chromosomes for each application, first one based on job length, CPU speed, and the second chromosome is created on the job length and bandwidth of resources. Authors used fuzzy theory on these chromosomes to produce offspring instead of traditional crossover operator. However, how they reduce execution time and cost using fuzzy theory is unclear. Chandrasekaran et al. [19] used GA for load balancing of virtual machine resources. Here every possible solution is represented as a tree. The scheduling and managing node of the system on the first level is the root node, while all of the N nodes on the second level stand for physical machines and the M nodes on the third level stand for the VMs on certain physical machines. The drawback of the algorithm is incoming load is assumed to be linear, and load amount to be stable. Dam et al. [20] used GA and GEL (gravitational emulation) load balancing problem among VMs. GA has global nature towards the problem space where GEL searches locally. Basically, the proposed algorithm tries to minimise the makespan as well as reduce the number of VMs who are going to miss their deadlines. The drawback is GEL algorithm uses some random elements that do not move always in the same way and does not stop always with the best possible solution. Panda et al. [12] presented three task scheduling algorithms, namely MCC, MEMAX and CMMN, for heterogeneous multi-cloud systems. The MCC is a single-phase scheduling, while other two algorithms are two-phase scheduling. The objective on which they focussed is reducing makespan and enhancing average cloud utilisation. Not much is done regarding customer satisfaction rate.

Researchers have used improved differential evolution algorithm to optimise task scheduling followed by resource allocation on the basis of cost and time, Taguchi method is used to populate potential offspring, and model used includes processing time, waiting time, receiving time, processing cost and receiving cost [21]. Some researcher enforced the multi-objective approach to calculating Pareto optimal of total makespan and cost [22,23]. Particle swarm optimisation (PSO) is used to schedule application to cloud resources considered two types of cost (transmission cost and computation cost). All available computing tasks are gathered and ranked in the task pool; resources are allocated in accordance with task weightage [24–26]. A quality of service (QoS)-constrained resource allocation problem is proposed using game theory. They showed that Nash equilibrium always exists whenever resource allocation game has feasible solutions [25]. ‘Skewness’ is termed as the measure of unevenness in the multi-dimensional resource utilisation of server [26]. Lesser the skewness, higher the utilisation of server in cloud computing will be. Some researchers used map reduce involving three phases: part, comp and group. Key of each phase is saved in hash key table [27]. Part phase initiates mapping of tasks to respective resources, comp phase compares between parts, and finally, group phase wraps the similar entities using task reduce. Numerous mapped task can read and execute entries running concurrently, causing reduce tasks to be overloaded. Many researchers propose one load balancer in between the map phase and reduce phase to address the overloading [28–30]. Some researcher used GA-based mapping [15] in cloud computing, not in federated multi-cloud computing. They did not suggest scheduling after mapping. Some used historical data and current states compute in advance; they used GA-based scheduling strategy. The solution which has least influence on the system and the least migration cost is termed as the best solution [31,32].

Many researchers have used GA-based task scheduling in cloud computing [33–36]. However to the best of our knowledge GA-based resource allocation and categorization of the applications into AR and BE types in multi-cloud computing domain is not attempted by any researchers yet. We proposed GA-based resource allocation algorithm to allocate resources to tasks and shortest job first scheduling algorithm to schedule the allocated tasks to the resources. Some researchers used ant behaviour to collect information about the status of cloud node before assigning any task to VMs. Algorithms propose phases, on the arrival of the request pheromone is initiated, and ant starts searching for the forward path from the start node [37,38]. Load balancing mechanism using ant colony and complex network theory in open cloud computing federation (OCCF) is also proposed by researchers [38]. Forward movement is meant by one overloaded node to next node in order to validate whether it is overloaded or not. If the next node is found to be overloaded, then it will detour to the previous node. Basically, ant behaviour is practiced to identify the lightly loaded node to allocate incoming tasks in order to achieve load balance. Some used weighted least connection (WLC) algorithm; the available tasks are allocated to the virtual machines of the clouds having least number of connections (assigned loads) [39]. All proposed methods have its share of pros and cons; few are discussed above.

Our model comprises of GA-based resource allocation and shortest job first task scheduling. The objective is to efficiently allocate resources such that makespan time will be minimised and customer satisfaction can be enhanced. Researchers [40] have established a relationship between makespan time and customer satisfaction rate but ignored that waiting time also impacts customer satisfaction rate. Nature of task also contributes towards the makespan time. If a number of AR tasks will be there, then provisioning will be better, as these tasks are registered prior to its execution time and time of computability is communicated. In the case of BE task, provisioning is a more on-demand type. So cloud federation does not get much time for auto-scaling. In our model, a relationship between a type of task, waiting time, execution time and makespan time with customer satisfaction is established. Many researchers have used Haizea concept. Haizea is an open source resource lease manager in cloud computing [1,14,30,32]. In accordance to Haizea, all incoming applications are divided into two categories, i.e. AR (advance reservation) and BE (best effort). We have also considered Haizea concept in our model. We categorised all the incoming applications into two groups AR and BE. The applications are queued in their respective queues. The applications in AR queue are prioritised over BE applications.

3 Multi-cloud Model and its Description

Multiple datacentres are connected to provide services to multiple customers. Datacentres having a variant processing capabilities from different cloud service providers, connected by voluntary federation, termed multi-cloud computing. Cloud user can send the request to the centralised cloud broker. Then, the centralised mapping (resource allocation to tasks) is done in accordance with the availability of resources. Load balancing is practiced across all involved datacentres. Mainly a number of resources are limited, so computational capability of resources is limited, whereas incoming workload is arbitrary in nature.

The objective of multi-cloud computing is to deploy maximum applications made by cloud user, trying to minimise makespan time and to minimise waiting time. A number of servers in multi-cloud computing are numerous compared to a single cloud provider. Each application is subdivided into multiple independent tasks. These tasks are allocated to VMs. In cloud computing, allocation of resources to tasks is similar to travel salesman problem, NP-complete problem. Allocating VMs to a large number of tasks is challenging. Scheduling huge number of tasks having a different configuration, following different SLA is quite challenging. Figure 1 includes diagrammatic representation of multi-cloud computing, its vital components and flow of control among its units. Multi-cloud computing composed of multiple heterogeneous cloud providers having the same or different geographical distribution. Cloud broker is responsible for assigning unique job ID to all incoming applications or workloads after SLA agreement is established between the cloud provider and cloud user.

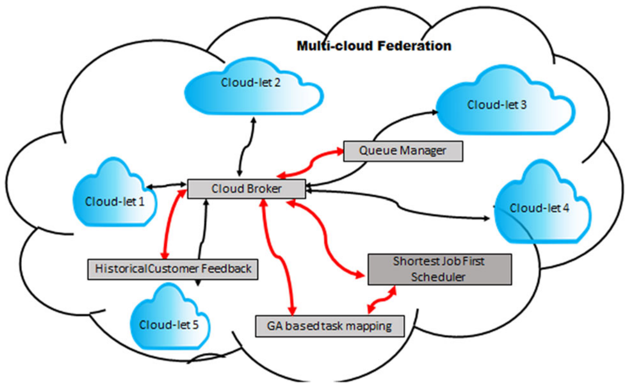

Fig. 1 Diagrammatic representation of multi-cloud federation

Incoming applications are divided into multiple tasks. Similarly, physical machines are divided into multiple VMs. Genetic algorithm (GA) is used to find the best task-VM pair, for all tasks to be executed. GA is a better-suited heuristic algorithm to balance load evenly. After allocation of a task to resources, it is found that many instances multiple tasks are assigned to single VM. Some authors suggested First In First Out (FIFO) task scheduling. We implemented FIFO and traditional shortest job first (SJF) algorithm. When multiple tasks are allocated to single VM, then task having shortest CPU burst is executed first.

By using GA-based resource allocation algorithm, it is found that load is balanced, resource utilisation is better, and by implementing traditional shortest job first (SJF) makespan time is minimised. Tasks in a batch constitute chromosome. The length of the chromosome depends upon the number of tasks in a batch that needs to be executed.

Figure 1 shows the architecture of the model supporting GA-based task mapping to balance workload evenly among available virtual machines followed by shortest job first scheduling. The main components are as follows:

1. *Users* User can register their application for deployment from anywhere in the world to the cloud broker in a multi-cloud environment.
2. *Cloud Broker* Broker acts as an interface between a group of cloud providers and users. In order to deploy an application, it requires resources from all the multiple cloud provider. During the registration stage, SLA needed to be signed and job request type needs to defined like Advance Reservation (AR) or Best Effort (BE).
3. *GA-based resource allocation* Each application is divided into multiple tasks. Each task is categorised into two categories (i) Advance Reservation (AR) and (ii) Best Effort (BE). Depending upon the available resources in the multi-cloud environment and expected makespan time, the task is allocated to resources using GA operators.

4. *Shortest Job First Scheduler* Since tasks are numerous and of varying capacities, when tasks are allocated to resources, then deployment of the tasks is scheduled on the basis of the shortest job first instead of traditional FIFO.

When a new cloud user wants to use cloud computing for deploying its application, as so many cloud providers are there in the market, new cloud user relies on historical performances of cloud providers to verify its trend and dependability. We proposed the relationship between customer satisfaction with expected time of completion, waiting time and depending upon the task type. Cloud provider with better customer satisfaction can have higher sustainability in the market for a long run.

The performance of constituting cloud providers, physical machines even VMs is recorded in historical customer feedback database. All the components of multi-cloud computing are connected to cloud broker so that QoS can be maintained as per SLA.

3.1 Case Study

Consider a set of C cloud provider connected to form a multi-cloud computing, where $C = \{C_1, C_2, \ldots, C_i\}$. There are set of Q cloud applications, where $P = \{P_1, P_2, \ldots, P_j\}$. Each cloud user can raise as many as job requests. Each job application (or job request) is divided into multiple independent tasks, such that $P_{ij} = \{P_{11}, P_{12}, \ldots, P_{q1}, P_{q2}, \ldots, P_{qi}\}$ is the set of tasks and $C_{ij} = \{C_{11}, C_{12}, \ldots, C_{p1}, C_{p2}, \ldots, C_{pi}\}$ is the set of VMs. Mapping function $M : P_{ij} \rightarrow C_{ij}$. The main objective is to map the customer task to VMs such that maximum customer tasks can be deployed in minimum time and high customer satisfaction. Here job request is primarily categorised into two types: Advance Reservation (AR) and Best Effort (BE). BE tasks are preemptive in nature, whereas AR tasks are non-preemptive in nature. Priority of AR tasks is higher than BE tasks, and service charge of AR task is higher than BE task.

We assume there are four roles in our model, namely the application owner, the application users, the cloud application and the multi-cloud federated environment. The application owner deploys her application in a cloud provider or multi-cloud environment and tries offering services to the application users. The application users submit jobs request with detailed performance requirements (in the form of computing capacity, duration of resource, type of task, deadlines) to the application. The application owner rents resources (e.g. VMs and storage) from multi-cloud provider to execute the applications and service incoming job requests. The objective is to automate the process of resource provisioning in the multi-cloud federated environment and execute all job requests using the minimal amount of physical resources as well as VM through the auto-scaling mechanism. We make the following assumptions:

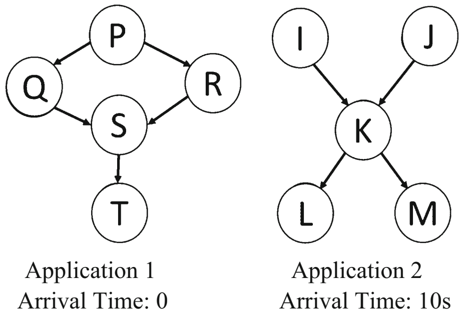

Fig. 2 The DAG of two applications

Definition 1 (*Cloud Application*) A cloud application consists of a set of service units following certain protocols and knowledge based on its domain. A service unit is an abstraction of a processing module/component in an application, such as the order submission and data persistence steps for an online shopping site, ticket booking or service-oriented applications using the Internet. Each application is divided into multiple independent tasks. In our model, we are considering two types of applications only, i.e. AR applications and BE applications. The AR applications further break down to AR tasks, and BE applications break down to BE tasks.

Definition 2 (*Job Class*) In Fig. 2, application 1 is AR-type application and application 2 is BE-type application. An application is divided into multiple tasks, represented in the form of directed acyclic graph (DAG) showing its precedence order with a strict deadline. For example application 1 is divided into 5 tasks of type AR, having deadline 45 min. A job request is represented as *{{(P), {(Q, R)}, (S), (T)},* AR, *45 min*}. Application 2 is divided into 5 tasks of type BE, having deadline 90 min. It is represented *as {{(I)}, {(J)}, (K), (L)}, (M)},* BE, *90}.*

Definition 3 (*Cloud VM*) The cloud federation may offer different types of VM instances, suitable for different types of workloads. The VMs have different processing powers (the number of cores, memory, disc, etc.) and service prices. We assume that each VM is capable of running all tasks (whether the task is computationally intensive or input–output intensive). We assume time zone to be one at all times.

Definition 4 (*Workload or job request*) We assume the application owner is unaware of incoming requests in advance. Therefore, the workload is defined as all the incoming job requests that have been submitted with multi-cloud to be executed using application forum.

Definition 5 (*Resource Allocation*) The auto-scaling mechanism needs to balance two decisions. The first decision is to find which VM (of which physical machine) is to be allocated for execution of the tasks. GA operators are used to finding the best VM-task pair having least makespan time and most throughput.

Definition 6 (*Task Scheduling Plan*) The second decision, which we termed task scheduling, is to decide the sequence of tasks to be executed when more than one task is allocated to a single VM. We used traditional shortest job first scheduling algorithm to minimise the makespan time and maximise the throughput.

Definition 7 (*Customer Satisfaction Rate*) The objective is to find a resource allocation and task scheduling plan to enhance the customer satisfaction rate. In the market, where multiple cloud providers are available, cloud user relies on historical customer satisfaction rate while deciding the cloud provider. So to attain higher customer satisfaction rate, factors which impact are throughput, makespan time and waiting time.

Supposedly if AR tasks are readily available and waiting for a resource where BE task is under execution using the same resource, then BE tasks are preempted and concerned resource is given to AR tasks. Partially executed BE task is saved and stored in temporary queue till the complete execution of AR task. In real-world service charge for AR, tasks are comparatively higher than BE tasks. A matrix generated to represent the expected executed time is as follows:

$$\mathrm{ETC} = \begin{array}{c|cccc} & C_1 & C_1 & \cdots & C_n \\ T_1 & \mathrm{ETC}_{11} & \mathrm{ETC}_{12} & \cdots & \mathrm{ETC}_{1m} \\ T_2 & \mathrm{ETC}_{21} & \mathrm{ETC}_{22} & \cdots & \mathrm{ETC}_{2m} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ T_n & \mathrm{ETC}_{n1} & \mathrm{ETC}_{n2} & \cdots & \mathrm{ETC}_{nm} \end{array},$$

ETC_{ij} represents the expected time taken to execute the ith task in the jth cloud. Any task having a valid job request id can be executed in any cloud, and at the same time, any cloud can execute multiple tasks concurrently as per the priority. Many cloud providers use chronological order. Different cloud has different executing capacities, basically categorised into high, medium and small capacities. Similarly, storage is also categorised into high, medium and small capacities. Basically, user categorisation is based on customer's choice and capacity of the resource. Mapping algorithm is based on GA followed by shortest job first scheduling. In real-life scenario, it is found that multiple tasks are allocated to single VMs for execution. A customer can decide nature of the task to be AR or BE (instead of booking high reserved instances with cloud provider). A customer can consciously decide (either categorise task as AR or BE) about its nature of the task and accordingly pay per usage.

4 Proposed Algorithm

The proposed algorithm is used to map p independent tasks to q virtual machines of r clouds such that each task is executed having minimum makespan time and maximum customer satisfaction rate. Efficient resource allocation (using GA-based task mapping) involves mapping of tasks to the virtual machines, followed by tasks scheduling using shortest job first. By doing as above, the balance of workload is achieved and server usage of each resource is maximised. In our simulation, we used GA-based resource allocation algorithm called GACCRATS and compared with TLBO [41] and COTS algorithm [40].

The predominance of our algorithm is as follows:

- A GA-based Customer-Conscious Resource Allocation followed by shortest task first scheduling algorithm is developed for a heterogeneous multi-cloud environment, considering both AR and BE tasks.
- TLBO algorithm is used to allocate resources in multi-cloud computing. It is a recent optimisation algorithm which is easy to implement having few constraints.
- Assessment of the proposed GACCRATS algorithm and TLBO algorithm is compared with COTS algorithm [40]. Simulation of the proposed algorithm is done using MATLAB. Numerous cloud providers are generated which are further divided into multiple virtual machines. Workload arriving is in accordance with Poisson's distribution.
- Two performance metrics, i.e. customer satisfaction rate and makespan time, are used to validate the simulation using Genetic Algorithm-based Customer-Conscious Resource Allocation and Task Scheduling (GACCRATS) and TLBO [41]. GA-based mapping is followed by the scheduling of tasks, as mainly resources received multiple tasks to be executed. So the sequence of allocated tasks needs to be done. In this paper, we are using shortest job first scheduling. Each task is allocated to only one VM of any cloud. One physical machine of any cloud provider (which is further divided into multiple VMs) can simultaneously execute multiple tasks initiated from multiple customers.

4.1 GA-Based Resource Allocation in GACCRATS

With the rise in an incoming application, the complexity of resource allocation and scheduling tasks increases. In the multi-cloud environment, the number of resources is numerous and capacity of processing power is different. A number of incoming applications to be executed are arbitrary. Therefore, a strong and robust optimisation algorithm is required for the scheduling of applications. To satisfy QoS, minimise makespan time and better resource usage GA is better suited as heuristic algorithm [2].

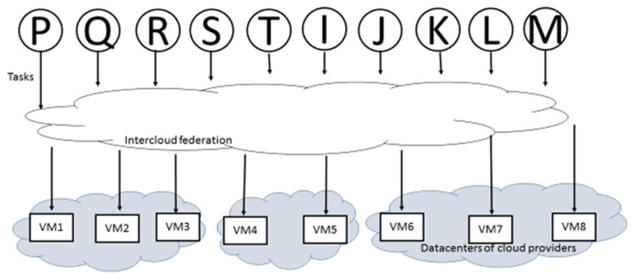

Fig. 3 Resource allocation principle

Figure 3 illustrates the resource allocation process where each user introduces his application's tasks, and the cloud federation termed multi-cloud uses the appropriate approaches to allocating resources to these tasks by considering optimisation parameters, such as minimum makespan, resources utilisation and maximum customer satisfaction. Therefore, the optimisation problem can be solved using heuristic algorithms such as genetic algorithm (GA), particle swarm optimisation (PSO) and ant colony optimisation (ACO). In this paper, the proposed resource allocation algorithm in the multi-cloud environment is based on GA with some modifications. According to modifications, the parents will be considered in each population besides the offspring produced after the crossover process. Also, the elitism selection is used to select the best chromosomes to overcome the limitation of premature convergence. Genetic algorithm (GA) is based on the biological concept of generating the best fit solution using Darwin's theory of survival strategy in evolution. GA is considered a population-based heuristic algorithm in growing area of artificial intelligence. According to the genetic algorithm, 'survival of the fittest' is used as the method of resource allocation, in which the tasks are assigned to resources according to the value of fitness function for each parameter. The main principles of the GA are described as follows [17]:

1. *Initial Population* The initial population consists of a set of all possible solution that is used in GA to find out the best solution. The initial population is populated by random allocation of tasks to VMs. Each individual solution of a population is called chromosome. So initial population consists of numerous chromosomes. These chromosomes are making the set of all individuals that are used in the GA to find out the optimal solution. The number of chromosomes in the initial population is equal to the population size. Every individual is represented as a chromosome for making it suitable for the genetic operations. From the initial population, the individual's chromosomes are selected, and genetic operators are applied on them to form the chromosomes for next generation. The mating chromosomes are selected randomly.
2. *Fitness Function* The productivity of any individual chromosome depends on the fitness value. It is the measure of the superiority of an individual chromosome in the population. The fitness value shows the performance of an individual in the population. Therefore, the individual chromosome will survive or die out according to the fitness value. Hence, the fitness function is the deciding factor in the GA.
3. *Selection* The selection mechanism is used to select an intermediate solution for the next generation based on Darwin's theory of survival. This operation is the guiding channel for the GA based on the performance. There are various selection strategies to select the best chromosomes such as roulette wheel, Boltzmann strategy, tournament selection and selection based on rank. We used elitism selection process in which the best fit chromosome is moved to the next generation as it is so that the chromosome having highest fitness value is not lost in the process of GA.
4. *Crossover* The crossover operation is done by selecting two parent chromosomes from the population and then creating a new child chromosome by alternating and re-joining the parts of those parents chromosome. This hybridisation process is a guiding process in the GA, and it boosts the searching mechanism.
5. *Mutation* After crossover, mutation process takes place. It is the important genetic operator that boosts genetic diversity in the population pool. The mutation is the vital operation which addresses premature convergence to a large extent. It takes place whenever the chromosome population tends to become homogeneous due to an iterative process of reproduction and crossover operators. Mutation probability is usually set to a considerable low value. Mutation alters (changes 0–1 or vice versa in the case of binary chromosomes) one or more gene (constituting components of chromosomes are called genes) values in the chromosome from its initial state. This results in the entirely new chromosome values being added to the population pool. With these new gene values, the genetic algorithm may be able to produce a better solution [33].

Mapping is one of the critical process used in the resource allocation phase. It involves identification of the cloud (VM)-task pairs such a way that makespan time is minimised, high customer satisfaction rate is attained, and the load is evenly balanced across all resources. Since the search space is huge, GA is used to identify the best task-cloud pair having minimum makespan time and maximum customer satisfaction algorithm for the proposed GACCRATS algorithm, notations referred in Table 1. GACCRATS algorithm uses a queue to store all incoming job requests. Initially, the queue was null when valid job requests start coming; then, each application further subdivided into multiple tasks. This process selects a task to its appropriate VM using algorithm based on GA [15]. For this, it finds the unassigned tasks along with its require-

Table 1 Notations and their definitions

C—Set of cloud provider

C_{ij}—Set of virtual machines present in multi-cloud computing, where i number of cloud providers and each physical machine is divided into j number of VMs

P_{ij}—Set of tasks, where i number of applications and each application is further divided into j number of task

Q_t—Temporary queue for readily available tasks.

Q_{AR}—Queue for available Advanced Reservation task

Q_{BE}—Makespan time of the task (sum of execution time and waiting time)

MS (*temtask*, *tempcioud*)—Makespan time of the task (sum of execution time and waiting time)

EST (temptask, tempcioud)—EST is the waiting time for the resource. Time period at which the task can start execution, provided its interdependent predecessor tasks are all executed prior

q—Physical resources which are further subdivided into multiple VMs, $|C_{ij}|$ *is* q

p—Application is further divided into multiple independent tasks, which are to be deployed, $|P_{ij}|$ *is* p

ETC (temptask, tempcioud)—expected time of completion *ith* task of *jth* VM. *X(temptask, tempcioud)*—Execution status of task

ments. Then, it compares the completion time/execution time of other available tasks to find the most suitable task-VM pair having least makespan time. The above procedure is repeated till all unassigned tasks are allocated to respective VMs. Load balancing problem is formulated as p number of tasks submitted by cloud user having q number of VMs in the multi-cloud computing environment. Makespan time depends on waiting time of resource and processing capacity of the VM.

$$F(x) = \min(\text{MS}) + (1/\max(\text{CSR}) \tag{1}$$

$$\text{MS} = f(\text{MIPS}_{\text{temptask}}, \text{EST}) \tag{2}$$

$$\text{MS} = w_1{}^*(\text{NIC/MIPS}) + w_2{}^*\text{EST} \tag{3}$$

$$\text{CSR} = f(\text{EST}_{\text{temptask}}, \text{ETC}_{\text{temptask}}) \tag{4}$$

Computability metric called makespan time (MS) of each processing unit indicates the rate of utilisation of resource expressed in Eqs. 2 and 3, whereas Eq. 4 indicates the relationship between customer satisfaction rates, waiting time for a resource, expected a time of completion. Where MS is makespan time of the task/job request/application, CSR is customer satisfaction rate, NIC is a number of million instruction present in the job, MIPS is a number of million instructions that can be executed by that machine, and $w_{1,} w_2$ are predefined weights. It is challenging to decide the value of the weights as it varies from organisation to organisation. Weights w_1 and w_2 range between 0.1 and 0.9 such that their summation is 1.

Algorithm 1 for GACCRATS

```
Algorithm1: GACCRATS(GA based customer conscious resource allocation and
task scheduling in Intercloud computing
Input:
  A) A set of customer job-requests following Poisson's distribution.
  B) A set of independent tasks (each job request is sub-divided into multiple
independent tasks).
  C) A set of cloud providers involved in the federation.
  D) A set of virtual machines (multiple cloud providers are further divided into
numerous VMs).
Output:
  A) Makespan time
  B) Customer satisfaction rate

1) While $Q_T \neq NULL$
2)   Set makespan= 0
3)   Breakup job-application into multiple tasks
4)   Call GA_MAPPING( ETC, EST, p, q)
5)   Call TASK_SCHEDULING( ETC, EST, p, q, MS)
6) endwhile
```

Algorithm 2 for Resource Allocation

```
Algorithm2: GACCRATS (GA based Customer-Conscious Resource
Allocation and Task Scheduling)

1. While $Q_T \neq null$ do
2.    If $Q_{AR} \neq null$ (if task ready available is Advance Reservation then)
3.       If $Q_{BE} \neq 0$ (if task ready available is Best Effort task then)
4.         For tempcloud= {1, 2, 3,...., q}
5.            For temptask= {1, 2, 3,...., p}
6.             temptask← Task($Q_{AR}$)
7.             Find EST (temptask, tempcloud)
8.             MS (temptask, tempcloud) = ETC(temptask, tempcloud) + EST(
temptask, tempcloud)
9              Call GA_task_cld_pair($p_i$, $q_i$) that gives min (MS(temptask,
tempcloud))
10.            Call BE_PREMPT_TASK(EST(temptask), MS(temptask,
tempcloud), tempcloud)
11.            endfor
12.          endfor
13.        else
14.        temptask← Task($Q_{BE}$)
15.        Call UPDATE $Q_T$
16.        Call SCHEDULE_AR_TASKS_MMS(ETC_AR, temptask$_i$)
17.        Call SCHEDULE_BE_TASKS_MMS(ETC_BE, temptask$_i$)
18.        MS (temptask, k) = ETC(temptask, k) + EST (temptask)
19.      endif
20.   endif
21. endwhile
```

Algorithm 3 for GA based resource allocation and task scheduling

```
Algorithm3: Procedure 1: GA_task_cld_pair ($p_i$, $q_i$)
 1. Set parameters← Population(Population$_{size}$, crossover rate($p_c$) and
    activation function)
 2. Evaluate Population
 3. Fitness Evaluation: Use Eqn 1 to evaluate fitness value of chromosomes
 4. Selection: next generation ←Best_fit_chromosome(elitism)
 5. Best_Solution(Population)
 6. parents← select parents(Population, Population$_{size}$ )
 7. for each parent$_1$ , parent$_2$ ϵ Parent(Population)
 8. child$_1$, child$_1$←Crossover(parent$_1$ , parent$_2$, P$_{Crossover}$)
 9. children← Mutate( child$_i$ , P$_{Mutation}$)
 10.end
 11.Evaluate_Population(Children)
 12.Best_Solution←Get (Best_Solution(Children))
 13.New_Population←Replace(Population, Children)
 14.End
 15.Return Best_Solution
 16.Call SJF algorithm
 17.Endif
```

Algorithm 4 for BE_PREMPT_TASK

```
Algorithm4: Procedure2: BE_PREEMPT_TASKS(EST(temptask), MS(temptask, tempcloud), tempcloud)
    1. If $t_i = 1$                 // resource tempcloud is executing temtask of type BE is
    2.   Temptask= $t_{BE}$          // Tempcloud→ $t_{BE}$, resource is allocated to $t_{BE}$
    3.     While $t_{i+1} = 2$        // resource tempcloud prempt BE task and resource is given to AR task
    4.       $Q_t = t_{BE}$           // Preempted BE task is stored in temporary queue $Q_t$
    5.       Temptask= $t_{AR}$       // temptask is of type AR, Tempcloud→ $t_{AR}$, resource is allocated to $t_{AR}$
    6.     end while
    7.   $t_{i+1} = 1$               // resource tempcloud is executing temptask of type BE
    8. Temptask= $t_{BE}$            // preempted $t_{BE}$ is moved from $Q_t$ and execution of $t_{BE}$ is resumed.
    9. Return
```

Algorithm 5 for Scheduling AR_tasks

```
Algorithm5: Procedure 3: SCHEDULE_AR_TASKS_MMS (ETC_AR, temptask_i)
1. $Q_{AR} \leftarrow Q_T$ (if task is Advance Reservation task)
2.   For tempcloud=1, 2, 3… $p_i$
3.     For temptask= 1, 2, 3… $q_i$
4.       Find sort (temptask ($Q_{AR}$))
5.       min (temptask) = min (temptask ($Q_{AR}$))
6.       MS (temptask, q) = ETC (temptask, q) + EST (temptask)
7.     Endfor
8.   Endfor
9. Execute min (temptask) till $Q_{AR}$ = null.
```

Algorithm 6 for Scheduling BE_tasks

```
Algorithm6: Procedure 4: SCHEDULE_BE_TASKS_MMS (ETC_BE, temptask_i)
1. $Q_{BE} \leftarrow Q_T$ (if task is Best Effort task)
2.   For tempcloud=1, 2, 3… $p_i$
3.     For temptask= 1, 2, 3, $q_i$
4.       Find sort (temptask ($Q_{BE}$))
5.       min (temptask) = min (temptask ($Q_{BE}$))
6.       MS (temptask, q) = ETC (temptask, q) + EST (temptask)
7.     Endfor
8.   Endfor
9. Execute min(temptask) till $Q_{BE}$ = null.
```

Algorithm 7 for Shortest Job First Algorithm

```
Algorithm7: Procedure 5: SJF algorithm (n number of tasks are allocated to resource m)
1. Sort (temptask, m) as per CPU burst time.
2. While temptask_min ← min (temptask)
3.     execute (temptask, tempcloud) where temptask=temptask_min
4.     temptask =temptask_min +1
5.     Excute temptask =0.
6. endwhile
```

Algorithm 8 for Update Queue

```
Algorithm8: Procedure 6: UPDATE $Q_T$ (UPDATE SUCCESSOR)
1. Add the next ready task available as successor in $Q_T$ and calculate I $Q_T$ I
2.  if I $Q_T$ I < $Q_{th}$
3.    For q= 1, 2, ..., I $Q_t$ I
4.      temp ←Delete ($Q_t$)
5.      If all the ready task predecessor to temptask is executed
6.           Then add next batch of ready tasks to $Q_t$
7.         Endif
8.    Endfor
9.  endif
10. Return
```

Scheduling is a process in which allocated task-cloud pair is scheduled sequentially so that makespan time is minimum and customer satisfaction rate is higher. In this process load is evenly distributed, resulting in no server overloading. The algorithm for GA-based Customer-Conscious Resource Allocation and Task Scheduling is described in GACCRATS.

The algorithm 1 involves initialisation of temporary queues QT. By using Poisson's distribution many applications are created having a different capacity (in MIPS, million instructions per second). The applications split into many independent tasks. Physical machines involved split into multiple VMs in step 3. GA-based resource allocation function is called in step 4. Step 5 includes scheduling of the multiple tasks allocated to single VM. The output of the algorithm is best task-VM pair having least makespan time and high customer satisfaction rate.

Algorithm 2 involves the step-wise description of our proposed algorithm, i.e. GACCRATS. The tasks of the applications are stored in QT. Depending upon the nature of the application, tasks get stored in QAR or QBE. The VMs involved are stored in the set tempcloud and all the tasks to be executed are stored in the set temptask. Estimated time of execution is calculated in step 7. Step 8 reveals that makespan time is the sum of estimated completion time and waiting time. In step 9, GA-based resource allocation procedure is called. The number of tasks to be executed in a batch is equal to the size of the chromosome in initialisation stage of GA. In the first generation, tasks are randomly assigned to VMs which are capable of executing the tasks. The fitness function is minimisation of makespan time and maximisation of customer satisfaction rate. Steps of GA-based resource allocation are expressed in procedure 1 as follows. Step 17 represents that the best fit chromosome is obtained as the convergence criteria are met. Then shortest job first scheduling is used to address the case when multiple tasks are allocated to a single VM.

Algorithm 3 illustrates steps of the genetic algorithm. Initialisation of chromosomes is done randomly. The size of the chromosome is equal to the number of tasks considered in a batch. Many parameters of GA need to be initialised like a number of chromosomes in population, encoding of the chromosome, initialisation of genetic operator. Fitness value of each chromosome is calculated in step 3. We have used elitism selection. Genetic operators like crossover and mutation are executed on selected chromosomes. The best task-VM pair is obtained in step 16. Then shortest job first is invoked in step 18.

We have used two types applications AR and BE. AR tasks are given more priority over BE task, if a BE task is executing in time x. However, in time (x+1), the same resource is reserved for an AR task, then BE task is preempted and stored in the temporary queue. The resource is given to the AR task, and on complete execution of AR task, execution task of the BE task is resumed. These steps are carried out in the procedure 2 of the algorithm 4. Algorithm 5 involves procedure 3 where AR tasks are sorted as per the burst time and makespan time of each task is calculated when executed in different VMs. Algorithm 6 involves procedure 4 where

Table 2 An ETC matrix

Task/cloud	C_1	C_2	C_3	C_4	C_5
T_1	30	50	80	40	66
T_2	50	40	70	85	65
T_3	155	110	90	170	100
T_4	140	170	130	190	100
T_5	120	160	130	70	50
T_6	145	30	170	110	90
T_7	90	140	150	190	120

Table 3 Mapping sequence for GACCRATS

Task/cloud	C_1	C_2	C_3	C_4	C_5
T_1	30				
T_2		40			
T_3			90		
T_4					100
T_5				70	

Table 4 Scheduling sequence for GACCRATS algorithm

Task/cloud	C_1	C_2	C_3	C_4	C_5	Time of completion
T_1	30					30
T_6		30				30
T_2		40				70
T_5				70		70
T_3			90			90
T_7	90					120
T_4					100	100
Ready time	120	70	90	70	100	

Table 5 Gantt chart for GACCRATS

Task/cloud	C_1	C_2	C_3	C_4	C_5
0–30	T_1	T_6	T_3	T_5	T_4
30–70	T_7	T_2			
70–90		*		*	
90–100		*	*	*	
100–120		*	*	*	*

BE tasks are sorted as per the burst time, and makespan time of each task is calculated when executed in different VMs.

The tasks are scheduled as per the priority. Advanced reservation tasks are computed prior to best effort tasks. By practicing the above steps higher customer satisfaction rate can be achieved. Balancing is a process to allocate the remaining unassigned tasks to the idle slots of all the clouds such that the overall makespan remains intact. The process of balancing has two advantages. (1) It improves the customer satisfaction. (2) It reduces the number of iterations required to assign the tasks.

4.1.1 Illustration of GACCRATS

The detail illustration of GACCRATS algorithm is described as follows: Let us assume there are seven different tasks (T_1 to T_7) mapped to five clouds (C_1 to C_5). An ETC matrix is populated, as per the expected execution time to reveal the mapping process in Table 2. The GA-based mapping is shown in Table 3. Scheduling sequence followed by Gantt chart for GACCRATS is shown in Tables 4, 5 in which '*' represents the idle time.

In the mapping process, task T_1 is suitably mapped to $C_{1,}$ task T_2 to $C_{2,}$ task T_3 to C_3, T_4 to C_5, T_5 to C_4, T_6 to C_2 and T_7 to C_1.

Now the tasks T_1 and T_7 are allocated to C_1, tasks T_2 and T_6 to C_2, task T_3 to C_3, task T_4 to C_5, task T_5 to C_4, task T_6 to C_2 and task T_7 to C_1. After allocation of tasks to the cloud, efficient scheduling is done. Here we have used shortest deadline first scheduling, so that makespan time will be less. The Gantt chart for the GACCRATS shown in Table 5 is as follows:

4.2 Resource Allocation Using TLBO

The TLBO algorithm is a population-based teaching-learning inspired optimisation algorithm. It is based on the effect of the influence of a teacher on the teaching output of learners in a class. The algorithm describes two basic modes (i) learning through the teacher in teachers phase, (ii) learning through interaction with the other learners in learners phase. In this optimisation algorithm, the population consists of a group of learners which make population size. Different subjects offered for learning are considered as different design variables for the optimisation problem. The learners' result is corresponding to the 'fitness' value of the optimisation problem. The solution having the highest learners' value in the entire population is considered as the teacher. The design variables are actually the parameters involved in the objective function of the given optimisation problem. TLBO algorithm requires parameters like population size and a number of generation. It does not include any algorithm-specific control operators. TLBO is basically divided into two phases.

Teacher phase: First phase of the algorithm includes learning through the teacher called teacher phase. The teacher tries to enhance the mean output of the class in the concerned subject, taught by the teacher, depending on his or her capability. At any iteration, assume there are 'm' number of subjects, 'n' number of learners (where population size= 1, 2, … n) and 'i' number of iteration, 'j' number of subjects. $M_{j,}$ be the mean result of the learners in the particular subject 'j'. The best overall result $\mathrm{Xtotal}-\mathrm{kbest}_i$ considering all the subjects together obtained in the entire population of learners can be considered as the result of best learner kbest. However, the teacher is considered as one of the highly learned individuals in the population who teaches learners so that they can gain knowledge and can attain the better result. The best learner is identified as the teacher in the algorithm. The difference between the current mean result of each concerned subject and the corresponding result of the teacher for each corresponding subject is given by the following equation,

$$\mathit{Difference\_Mean}_{j,k,i} = r_i \left(X_{j,\mathrm{kbest},i} - T_F M_{j,i}\right)$$

where $X_{j,\mathrm{kbest},i}$ is the result of the best learner in subject j. Teaching factor T_F is included which decides the value of mean to be updated, and r_i is the random number in the range [0, 1]. The range of value of T_F can be either 1 or 2. The value of T_F is decided randomly with equal probability as,

$$T_F = \mathrm{round}\,[1 + \mathrm{rand}\,(0, 1)\{2, 1\}]$$
$$X'_{j,k,i} = X_{j,k,i} + \mathit{Difference\_Mean}_{j,k,i}$$

where $X'_{j,k,i}$ is the updated value of $X_{j,k,i}$. $X'_{j,k,i}$ is accepted if it gives better function value. All the accepted function values at the end of the teacher phase are maintained, and these values become the input to the learner phase.

Learner phase: It is the second phase of the TLBO algorithm where learners enhance their knowledge by interacting among peer learners. A learner interacts with other learners randomly in order to boost knowledge. Learner objective is to acquire new things, each time learner value is higher than the existing one, and learner value is updated. Let us consider a population size of 'n', randomly pick two learners P and Q such that $X'_{\mathrm{total}-P,i} \neq X'_{\mathrm{total}-Q,i}$, where $X'_{\mathrm{total}-P,i}$ is updated value of $X_{\mathrm{total}-P,i}$ and $X'_{\mathrm{total}-Q,i}$ is corresponding updated value of $X_{\mathrm{total}-Q,i}$ of learner P and Q using the following equation

$$X''_{j,P,i} = X'_{j,P,i} + r_i(X'_{j,P,i} - X'_{j,Q,i}),\ \text{If } X'_{\mathrm{total}-P,i} < X'_{\mathrm{total}-Q,i}$$
$$X''_{j,P,i} = X'_{j,P,i} + r_i(X'_{j,Q,i} - X'_{j,P,i}),\ \text{If } X'_{\mathrm{total}-Q,I} < X'_{\mathrm{total}-P,i}$$

$X''_{j,P,I}$ is accepted if it gives a better function value. The above two equations are used for minimisation problem. Following two equations are used for maximisation problem.

$$X''_{j,P,i} = X'_{j,P,i} + r_i(X'_{j,P,i} - X'_{j,Q,i}),\ \text{If } X'_{\mathrm{total}-Q,i} < X'_{\mathrm{total}-P,i}$$
$$X''_{j,P,i} = X'_{j,P,i} + r_i(X'_{j,Q,i} - X'_{j,P,i}),\ \text{If } X'_{\mathrm{total}-P,i} < X'_{\mathrm{total}-Q,i}$$

5 Performance Metrics

The performance of the preexisting algorithm [40] is compared with proposed algorithm using performance metrics. The performance metrics are customer satisfaction rate and makespan time.

5.1 Customer Satisfaction Rate (CSR)

The customer satisfaction is the response given by a customer for the service provider, regarding how smoothly application is deployed within real-time and pricing strategy. In today's scenario 'customise' is the approach for higher customer satisfaction rate. The user is impatient than ever, they want faster services, lesser price and customised pricing plan [42,43]. Let us assume that there are two algorithms P and Q to process 5 tasks, which are T_1, T_2, T_3, T_4 and T_5. The expected finish time of these tasks in algorithms P is 20, 40, 10, 50 and 90, whereas using algorithm Q is 30, 20, 50, 70 and 60. Therefore, algorithm$P$ has customer satisfaction rate *CSR* (*P*) =3, as P takes the earliest time for task T_1, T_3 *and* T_4, whereas algorithm Q has *CSR (Q)* =2, since it takes the earliest time for task T_2 and T_5. The mathematical formula for calculating customer satisfaction rate is expressed in Eqs. 5, 6 and 7 as follows:

$$\text{Customer Satisfaction Rate (CSR }(A)) = \sum_{i=0}^{n} (\mathrm{ETC}i) + \sum_{i=0}^{n} (G) + w_2{}^{*} \sum_{i=0}^{n} (\mathrm{EST}i) \tag{5}$$

Customer satisfaction rate is dependent on followings:

1. Expected time to completion
2. G (customised pricing policy)
3. Waiting time for resource

Table 6 Calculation of customer satisfaction rate

Algorithm	CSR
P	3
Q	2

Table 7 Parameter and its instances

Parameter	Value of parameter
Number of clouds	5, 10, 15, 30, 50
Number of tasks	16, 32, 64, 256,512
Structure of the datasets	(Number of tasks) * (number of virtual machines)
Reserved instances	i1, i2, i3, i4

Assuming that customer satisfaction is high when high capacity of AR application completes execution [10]. The probability of a task to be completed in real time is higher in AR mode, as they are non-preemptive in nature, whereas BE tasks being preemptive in nature are prone to miss complete execution before the deadline. In case the application is really important for timely execution, its type of task should be changed to AR task from BE task.

Waiting time is inversely proportional to customer satisfaction rate. Higher the waiting time, lower is the customer satisfaction rate. Research reveals that the cloud users are impatient than ever. W_2 is a user-defined constant which varies from 0.01 to 0.05 (cloud user decides the value of weight while registration with the cloud). Many cloud users rely on the historical performance of cloud provider and customer satisfaction rate obtained from cloud users while finalising a reliable cloud provider for their enterprise.

Let us assume that there are two algorithms P and Q to process 5 tasks, which are T_1, $T_{2,}T_{3,}T_4$ and T_5. The expected finish time of these tasks in algorithms P is 20, 40, 10, 50 and 90, whereas using algorithm Q is 30, 20, 50, 70 and 60. Therefore algorithm P has customer satisfaction rate $CSR\ (P) = 3$, as P takes the earliest time for task $T_{1,}T_{3,}$ and T_4, whereas algorithm Q has $CSR\ (Q) = 2$, since it takes the earliest time for tasks T_2 and T_5. Here algorithm P be GACCRATS and Q be COTS.

$\mathrm{ETC}_i(T_i)$ measures the expected completion time; it is a Boolean variable which is defined as follows:

$$\mathrm{ETC}_i(T_i) = \begin{cases} 1, \text{ if } \mathrm{ETC}p\,(T_i) < \mathrm{ETC}q\,(T_i) \\ 0, \text{ otherwise} \end{cases} \tag{6}$$

Here, $\mathrm{ETC}_P(T_i)$ and $\mathrm{ETC}_Q(T_i)$ show the expected completion time of task T_i in algorithm P and algorithm Q, respectively. Here earliest completion time of task T_1 is 20 (using algorithm GACCRATS), whereas 30 (using COTS), earliest completion time of task T_2 is 40 (using algorithm GACCRATS) and 20 (using COTS), earliest completion time of task T_3 is 50 (using GACCRATS) and 50 (using COTS), for task T_4 is 50 (using GACCRATS) and 70 (using COTS), for task T_5 is 90 (using GACCRATS) and 60 (using COTS).

Table 8 Comparison of customer satisfaction rate in COTS, GACCRATS and TLBO algorithm is as follows

Datasets	Observations	Simulation 1	Simulation 2	Simulation 3
32 * 10	COTS	3.170e+02	3.112e+02	2.301e+02
	TLBO	3.679e+02	4.121e+02	3.219e+02
	GACCRATS	4.121e+02	4.761e+02	3.813e+02
64 * 20	COTS	2.183e+02	3.345e+02	3.271e+02
	TLBO	3.014e+02	2.997e+02	3.104e+02
	GACCRATS	3.234e+02	3.333e+02	3.715e+02
128 * 30	COTS	4.891e+02	3.914e+02	3.998e+02
	TLBO	4.504e+ 02	4.469e+02	3.568e+02
	GACCRATS	6.341e+02	5.341e+02	4.125e+02
256 * 40	COTS	2.917e+02	4.161e+02	3.714e+02
	TLBO	2.507e+02	4.018e+02	3.985e+02
	GACCRATS	3.348e+02	4.997e+02	4.983e+02

Table 9 Comparison of makespan time in COTS, GACCRATS and TLBO algorithm is as follows:

Datasets	Observations	Simulation 1	Simulation 2	Simulation 3
32 * 10	COTS	3.127e+02	4.123e+02	5.231e+02
	TLBO	3.153e+02	4.008e+02	5.004e+02
	GACCRATS	2.713e+02	3.987e+02	3.378e+02
64 * 20	COTS	5.128e+02	4.224e+02	5.036e+02
	TLBO	5.234e+02	4.354e+02	5.126e+02
	GACCRATS	4.916e+02	3.819e+02	4.912e+02
128 * 30	COTS	4.765e+02	3.451e+02	4.458e+02
	TLBO	4.890e+02	3.523e+02	4.652e+02
	GACCRATS	4.329e+02	3.923e +02	3.652e+02
256 * 40	COTS	5.392e+02	7.294e+02	5.721e+02
	TLBO	5.506e+02	7.479e+02	5.592e+02
	GACCRATS	4.671e+02	6.092e+02	4.295e+02

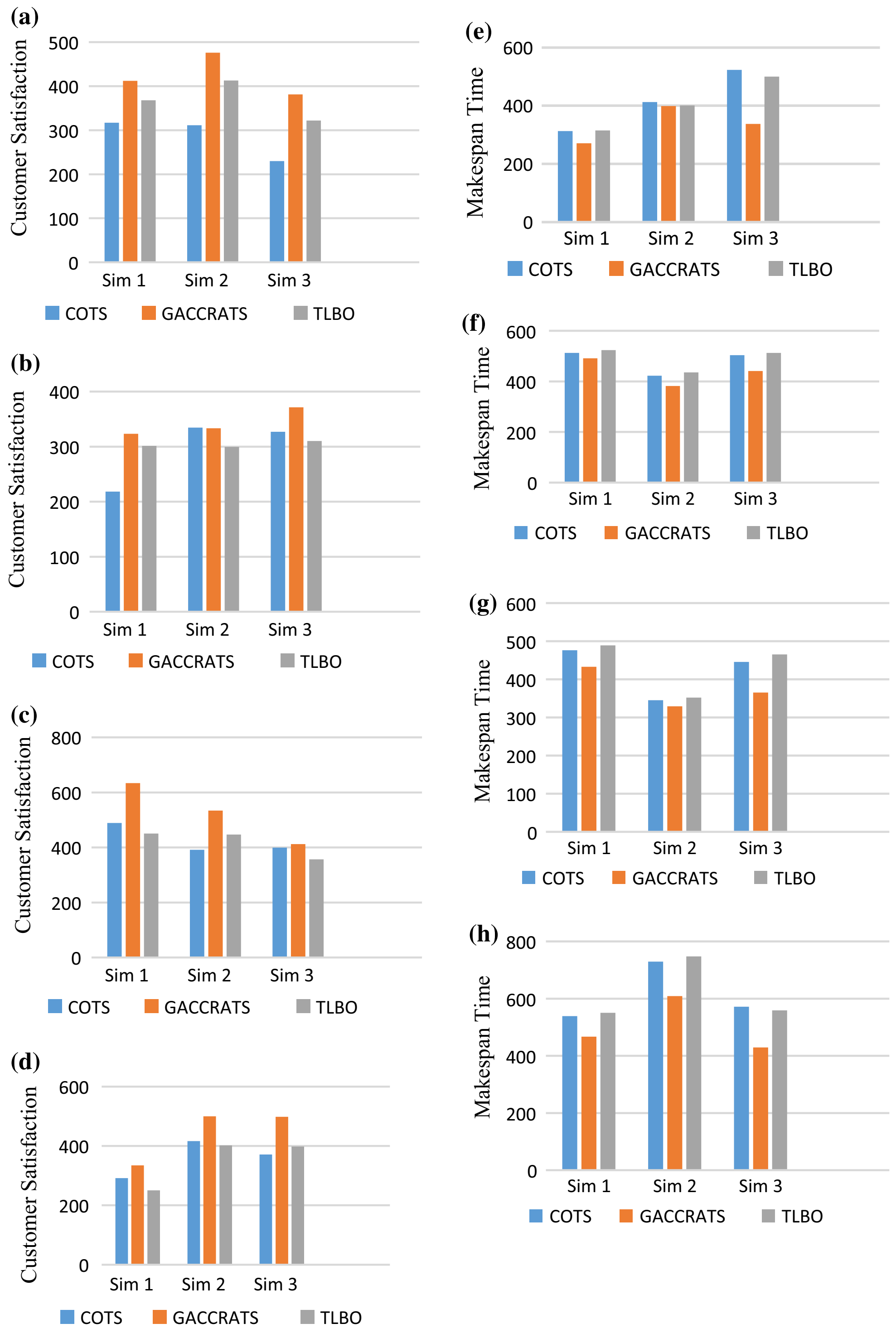

Fig. 4 **a** Instance (i) of dataset (32 * 10). **b** Instance (ii) of dataset (64 * 20). **c** Instance (iii) of dataset (128 * 30). **d** Instance (iv) of dataset (256 * 40). **e** Instance (v) of dataset (32 * 10). **f** Instance (vi) of dataset (64 * 20). **g** Instance (vii) of dataset (128 * 30). **h** Instance (viii) of dataset (256 * 40)

$G_i(T_i)$ measures the pricing policy,

$$G_i = \begin{cases} 1, & \text{if task is AR} \\ 0, & \text{if task is BE} \end{cases} \tag{7}$$

Pricing rate of *AR* is higher than *BE* task [10]. *BE* tasks are preemptive in nature, whereas *AR* tasks are not preemptive in nature. Customer satisfaction rate is dependent on least finish time, and *G* is customised pricing policy.

EST is waiting time for resource, and w_2 is a weight which varies from 0.01 to 0.05 (depending on the cloud provider tolerance, which varies organisationwise). Many new stakeholders rely on the historical customer satisfaction rate while finalising a reliable cloud provider for their enterprise. Using Eqs. 5, 6 and 7 it is found that customer satisfaction of algorithm P is 3 and Q is 2, illustrated in Table 6.

5.2 Simulations of the Proposed Algorithm

For simulation, we used MATLAB R2014a version 8.3.0.532. We simulate using Intel ® Core(TM)i3-5005U CPU@ 2GHz 2 GHz processor, 64-bit processing system and 8GB RAM running on Windows 10. The above environment conducts simulation by running the proposed algorithm on synthetic datasets. The arrival of job requests uses Poisson's distribution. The parameters of the data used in the simulation and its values are expressed in Table 7.

In the beginning of the program, we take inputs as a number of tasks and number of VMs. We have considered the task-cloud pair, which is further concreted as *TASK-VM* pair. In ETC matrix the task-cloud pair is represented, tasks along rows and clouds along columns. Each task has different granularities like capacity, memory and processing speed.

5.3 Generate Inputs of Proposed Algorithm

We populate dataset using MATLAB random function *randii()*, its a type of discrete uniform distribution. The rate of arrival is a user-defined value spanned 4–100. Arrival of job request is generated using MATLAB function Poisson's function as *poissrnd (JobQueueSize-UserSize)/UserSize, 1, UserSize)*. Tasks are basically divided into two types: Advance Reservation task (*AR* task) and Best Effort task (*BE* task). The user who reserves instances in advance for off-peak time is termed as Advance Reservation User, and respective tasks are called *AR* tasks, whereas user who does not opt for off-peak time reserved instance is termed as Best Effort Users, and respective tasks are called Best Effort tasks. Advanced Reserve tasks are non-preemptive in nature, whereas Best Effort task is preemptive in nature. In a scenario, when the resource is executing BE task, at the same time *AR* task is scheduled; then, the resource is taken away from *BE* task and given to *AR* task. The partially executed *BE* task is kept in temporary queue till the complete execution of *AR* task. Execution of *BE* task is resumed later. In the simulation, we run and execute both the algorithms (GACCRATS and TLBO and compared the performance of customer-oriented task scheduling (COTS) algorithm [40], TLBO and proposed algorithm GACCRATS).

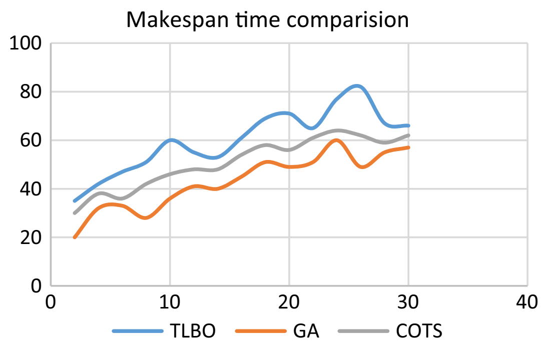

Fig. 5 Comparison regarding makespan time

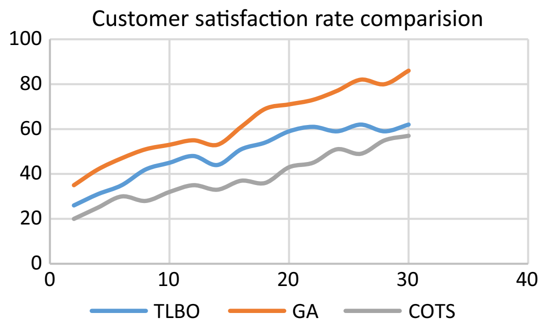

Fig. 6 Comparison regarding customer satisfaction rate

5.4 Result Analysis

The performance metrics, makespan time and customer satisfaction rate are used to compare the three algorithms: GACCRATS, COTS and TLBO. The makespan time and customer satisfaction of the proposed algorithm GACCRATS are determined for multiple instances and compared with COTS [40] algorithm and TLBO algorithm in Tables 8, 9, respectively. The graphical representation of the comparison of Tables 8, 9 is backed up in Fig. 4 instances a–h. The result shows GACCRATS has higher customer satisfaction rate in all datasets (32 * 10, 64 * 20, 128 * 30 and 256 * 40 in Fig. 4a–d in comparison with TLBO and COTS algorithm). The performance of the proposed algorithm as per the makespan time is represented in Fig. 4e–h, showing lesser makespan time in GACCRATS than COTS and TLBO algorithm. Implementation of TLBO is easy; however, its solution having highest learners value is found to have lesser fitness value than the best fit chromosome obtained by GACCRATS in all cases. The performance of COTS algorithm lies in between TLBO and GACCRATS algorithm in case of makespan time and customer satisfaction rate. Figure 5 depicts comparison regarding makespan time among COTS, GACCRATS and TLBO algorithms. Figure 6 represents a comparison of customer satisfaction rate among COTS, GACCRATS and TLBO algorithms.

Then figure shows a graphical comparison of makespan time among COTS and GACCRATS algorithm. Makespan time is found to be less using GACCRATS algorithm in comparison with COTS.

We generate the simulation results of proposed algorithms using two performance metrics, namely makespan time and customer satisfaction rate. The customer satisfaction of the proposed algorithm GACCRATS algorithm is calculated for twenty instances on different datasets, and compared with proposed COTS algorithm and TLBO algorithm, few instances are shown in Table 8. For the sake of better eminence, the graphical comparison of customer makespan time is shown in Table 9. The results clearly indicate that the proposed algorithm outperforms COTS and TLBO algorithm for all the twenty instances.

From the above simulation results, it is clear that the proposed GACCRATS algorithm performs better in comparison with COTS algorithm and TLBO algorithm. The TLBO algorithm is a learning-based heuristic algorithm, which does not have any optimisation operator. It is easy to implement; only population size and a number of iterations are to be defined by the user. The value of teaching factor ranges between 1 and 2. We compared the makespan time and customer satisfaction rate of output obtained by COTS algorithm, GACCRATS algorithm and TLBO algorithm. It is found that in all instances GACCRATS outperforms COTS algorithm and TLBO algorithm.

6 Conclusion

To accommodate sporadic workload in the multi-cloud environment, where the capacity of job request may change dynamically, machine learning provides better resource allocation and task scheduling. A novel task scheduling algorithm termed as Genetic Algorithm-based Customer-Conscious Resource Allocation and Task Scheduling (GACCRATS) is proposed for the heterogeneous multi-cloud environment. We compared the algorithm performance of COTS, TLBO and GACCRATS. COTS aims to maximise the customer satisfaction and surplus customer expectation, TLBO is used to find the task-VM pair having minimum makespan time, whereas GACCRATS objective is the minimisation of makespan time of tasks and maximisation of customer satisfaction rate. The algorithm mainly has two phases, mapping and scheduling the mapped tasks. Multiple instances are carried out with different compositions of datasets in MATLAB. The results show that the performance of the proposed algorithm has outperformed existing algorithms. Scalability of the simulated multi-cloud environment is considerably high. Data locality cost, latency arbitration, energy consumption and running cost of multi-cloud environment are out of the scope of the simulated scenario.

References

1. Assunção, D.; Dias, M.; Di Costanzo, A.; Buyya, R.: Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. In: Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing, pp. 115–131. ACM (2009)
2. Armstrong, R.; Hensgen, D.; Kidd, T.: The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions. In: Proceedings of Seventh Heterogeneous Computing Workshop, pp. 79–87 (1998)
3. Pandey, S.; Wu, L.; Guru, S. M.; Buyya, R.: A particle swarm optimisation-based heuristic for scheduling workflow applications in cloud Computing environments. In: 24th IEEE International Conference on Advanced Information Networking and Applications, pp. 400–407 (2010)
4. Cao, Q.; Wei, Z. B.; Gong, W. M.: An optimized algorithm for task scheduling based on activity based costing in cloud Computing. In: 3rd IEEE International Conference on Bioinformatics and Biomedical Engineering, pp. 1–3 (2009)
5. Li, J.F.; Peng, J.: Task scheduling algorithm based on improved genetic algorithm in cloud computing environment. J. Comput. Appl. **31**(1), 184–186 (2011)
6. Jena, T.; Mohanty, J.R.; Sahoo, R.: Paradigm shift to green cloud computing. J. Theor. Appl. Inf. Technol. **77**(3), 394–402 (2015)
7. Jena, T.; Mohanty, J.R.: Disaster recovery services in intercloud using genetic algorithm load balancer. Int. J. Electri. Comput. Eng. **6**(4), 1828–1838 (2016)
8. Jena, T; Mohanty, J.R.: Cloud security and jurisdiction: need of the hour. In: Proceedings of the 5th International Conference on Frontiers in Intelligent Computing: Theory and Application, Advances in Intelligent Systems and Computing, vol. 515, pp. 425–433. Springer (2017)
9. Wu, L.; Garg, S.K.; Buyya, R.: Sla-based resource allocation for software as a service provider (saas) in cloud computing environments. In: Cluster, Cloud and Grid Computing (CCGrid), pp. 195–204 (2011)
10. Sotomayor, B.: Haizea. Computation Institute, University of Chicago. http://haizea.cs.uchicago.edu/whatis.html. Accessed 9 Jan 2014 (2014)
11. Panda, S.K.; Gupta, I.; Jana, P.K.: Allocation-aware task scheduling for heterogeneous multi-cloud systems. Proc. Comput. Sci. **50**, 176–184 (2015)
12. Panda, S.K.; Jana, P.K.: Efficient task scheduling algorithms for heterogeneous multi-cloud environment. J. Supercomput. **71**(4), 1505–1533 (2015)
13. Sotomayor, B.; Keahey, K.; Foster, I.: Combining batch execution and leasing using virtual machines. In: Proceedings of the 17th International Symposium on High Performance Distributed Computing, pp. 87–96 (2008)
14. Sotomayor, B.; Montero, R.S.; Llorente, I.M.; Foster, I.: Virtual infrastructure management in private and hybrid clouds. IEEE Internet Comput. **13**(5), 14–22 (2009)
15. Dasgupta, K.; Mandal, B.; Dutta, P.; Mandal, J.K.; Dam, S.: A genetic algorithm (ga) based load balancing strategy for cloud Computing. Proc. Technol. **10**, 340–347 (2013)
16. Fang, Y.; Wang, F.; Ge, J.: A task scheduling algorithm based on load balancing in cloud Computing. In: International Conference on Web Information Systems and Mining, pp. 271–277. Springer, Berlin (2010)
17. 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. **5**, 157–162 (2012)
18. Javanmardi, S.; Shojafar, M.; Amendola, D.; Cordeschi, N.; Liu, H.; Abraham, A.: Hybrid job scheduling algorithm for cloud

computing environment. In Proceedings of the Fifth International Conference on Innovations in Bio-Inspired Computing and Applications IBICA, pp. 43–52. Springer International Publishing (2014)

19. Chandrasekaran, K.; Divakarla, U.: Load Balancing of Virtual Machine Resources in Cloud Using Genetic Algorithm, pp. 156–168. ICCN (2013)
20. Dam, S.; Mandal, G.; Dasgupta, K.; Dutta, P.: Genetic algorithm and gravitational emulation based hybrid load balancing strategy in cloud computing. In: Third International Conference on Computer, Communication, Control and Information Technology (C3IT), pp. 1–7 (2015)
21. Tsai, J.T.; Fang, J.C.; Chou, J.H.: Optimized task scheduling and resource allocation on cloud computing environment using improved differential evolution algorithm. Comput. Oper. Res. **40**(12), 3045–3055 (2013)
22. Chand, P.; Mohanty, J.R.: Multi objective genetic approach for solving vehicle routing problem with time window. Trends in Computer Science, pp. 336–343. Engineering and Information TechnologySpringer, Berlin (2011)
23. Chand, P.; Mohanty, J.R.: Environmental multi objective uncertain transport trail model using variant of predator prey evolutionary strategy. Int. J. Appl. Decis. Sci. **8**(1), 21–51 (2014)
24. Gouda, K.C.; Radhika, T.V.; Akshatha, M.: Priority based resource allocation model for cloud computing. Int. J. Sci. Eng. Technol. Res. **2**(1), 215–219 (2013)
25. Wei, G.; Vasilakos, A.V.; Zheng, Y.; Xiong, N.: A game-theoretic method of fair resource allocation for cloud computing services. J. Supercomput. **54**(2), 252–269 (2010)
26. Xiao, Z.; Song, W.; Chen, Q.: Dynamic resource allocation using virtual machines for cloud computing environment. IEEE Trans. Parallel Distrib. Syst. **24**(6), 1107–1117 (2013)
27. Kolb, L.; Thor, A.; Rahm, E.: Load balancing for mapreduce-based entity resolution. In: 28th International Conference on Data Engineering, pp. 618–629 (2012)
28. Al Nuaimi, K.; Mohamed, N.; Al Nuaimi, M.; Al-Jaroodi, J.: A survey of load balancing in cloud computing: challenges and algorithms. In: Second Symposium on Network Cloud Computing and Applications, pp. 137–142 (2012)
29. Sudeepa, R.; Guruprasad, H.S.: Resource allocation in cloud computing. Int. J. Mod. Commun. Technol. Res. **2**(4), 19–21 (2014)
30. Gunarathne, T.; Wu, T.L.; Qiu, J.; Fox, G.: MapReduce in the clouds for science. In: Second International Conference on Cloud Computing Technology and Science, pp. 565–572 (2010)
31. Hu, J.; Gu, J.; Sun, G.; Zhao, T.: A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In: Third International Symposium on Parallel Architectures, Algorithms and Programming, pp. 89–96 (2010)
32. Liu, H.; Xu, C.-Z.; Jin, H.; Gong, J.; Liao, X.: Performance and energy modeling for live migration of virtual machines. In: Proceedings of the 20th International Symposium on High Performance Distributed Computing, pp. 171–182 (2011)
33. Tejaswi, T.T.; Azharuddin, M.; Jana, P.K.: A GA based approach for task scheduling in multi-cloud environment. arXiv:1511.08707 (2015)
34. Zheng, Z.; Wang, R.; Zhong, H.; Zhang, X.: An approach for cloud resource scheduling based on parallel genetic algorithm. Int. Symp. High Perf. Distr. Comput. Comput. Res. Dev. **3**(2), 444–447 (2011)
35. Gayathiri, N.R.: Dynamic Demes Parallel Genetic Algorithm for Scheduling the Resources in Cloud (2011)
36. Sawant, S.: A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment (2011)
37. Nishant, K.; Sharma, P.; Krishna, V.; Gupta, C.; Singh, K.P., Rastogi, R.: Load balancing of nodes in cloud using ant colony optimisation. In: 14th International Conference on Computer Modelling and Simulation, pp. 3–8 (2012)
38. Zhang, Z.; Zhang, X.: A load balancing mechanism based on ant colony and complex network theory in open cloud computing federation. In: 2nd International Conference on Industrial Mechatronics and Automation, vol. 2, pp. 240–243, (2010)
39. Ren, X.; Lin, R.; Zou, H.: A dynamic load balancing strategy for cloud computing platform based on exponential smoothing forecast. In: International Conference on Cloud Computing and Intelligent Systems, pp. 220–224 (2011)
40. Pande, S.K.; Panda, S.K.; Das, S.: A customer-oriented task scheduling for heterogeneous multi-cloud environment. Int. J. Cloud Appl. Comput. **6**(4), 1–17 (2016)
41. Rao, R.V.; Kalyankar, V.D.: Parameter optimisation of modern machining processes using teaching-learning-based optimisation algorithm. Eng. Appl. Artif. Intell. **26**(1), 524–531 (2013)
42. Ding, S.; Yang, S.; Zhang, Y.; Liang, C.; Xia, C.: Combining QoS prediction and customer satisfaction estimation to solve cloud service trustworthiness evaluation problems. Knowl. Based Syst. **56**, 216–225 (2014)
43. Mehdi, N.; Mamat, A.; Ibrahim, H.; Symban, S.: Virtual machines cooperation for impatient jobs under cloud Paradigm. Int. J. Inf. Commun. Eng. **7**(1), 13–9 (2011)