J Netw Syst Manage (2018) 26:463–485
https://doi.org/10.1007/s10922-017-9425-0

Multi-objective Task Scheduling to Minimize Energy Consumption and Makespan of Cloud Computing Using NSGA-II

A. Sathya Sofia[1] · P. GaneshKumar[2]

Received: 24 February 2016 / Accepted: 29 August 2017 / Published online: 2 September 2017

Abstract The utilization of cloud services has significantly increased due to the easiness in accessibility, better performance, and decrease in the high initial cost. In general, cloud users anticipate completing their tasks without any delay, whereas cloud providers yearn for reducing the energy cost, which is one of the major costs in the cloud service environment. However, reducing energy consumption increases the makespan and leads to customer dissatisfaction. So, it is essential to obtain a set of non-domination solutions for these multiple and conflicting objectives (makespan and energy consumption). In order to control the energy consumption efficaciously, the Dynamic Voltage Frequency Scaling system is incorporated in the optimization procedure and a set of non-domination solutions are obtained using Non-dominated Sorting Genetic Algorithm (NSGA-II). Further, the Artificial Neural Network (ANN), which is one of the most successful machine learning algorithms, is used to predict the virtual machines based on the characteristics of tasks and features of the resources. The optimum solutions obtained using the optimization process with the support of ANN and without the support of ANN are presented and discussed.

Keywords Dynamic voltage frequency scaling · Genetic algorithm · Artificial neural network · Multi-task scheduling

✉ A. Sathya Sofia
sathyasofia@gmail.com

P. GaneshKumar
drpganeshkumar@gmail.com

[1] Department of Computer Science and Engineering, PSNA College of Engineering and Technology, Dindigul 624622, India

[2] Department of Information Technology, PSNA College of Engineering and Technology, Dindigul 624622, India

1 Introduction

Cloud computing is an Internet centric model and based on dynamic provisioning of hardware, software, or services that can be utilized by a pay-as-you-go method [1]. As cloud computing eradicates the initial investment cost, the number of cloud users is increases tremendously. The cloud providers and users can simultaneously receive the economic benefits by efficiently scheduling the resources. Hence, workflow scheduling is essential to minimize makespan and energy consumption for the benefit of the cloud users and the cloud providers, respectively. The cloud model utilizes the concept of Virtual Machines (VMs) that act as the computational units of the system. Depending upon the computational needs of the tasks to be executed, new VMs can be dynamically leased and released later. The need for a number of resources and their characteristics are unpredictable in the cloud environment. If the resources are properly allocated to all the tasks submitted by users, the execution time will be reduced. However, increasing the number of resources critically increases the energy consumption. So resource management and scheduling the tasks are important in the cloud in order to reduce energy consumption and execution time.

Due to the tremendous increase in energy costs, reducing energy consumption has turned into a key concern in the field of cloud computing. Several factors, including resource management should be addressed in order to decrease the energy consumption in cloud data centers. The dynamic voltage frequency scaling (DVFS) technique, which enables the VMs to run at different blends of frequencies and voltages, is used to reduce the energy consumption [2, 3]. As the energy consumption of a VM is approximately proportional to the square of its frequency, the energy consumption can be significantly reduced by optimally adjusting the working frequency. The different heterogeneous servers may be employed in cloud computing and each server is equipped with a lot of hardware (processing unit, memory unit, and network devices). The energy consumption of these servers can be controlled by adjusting the frequency in relation to workloads. For an example, if the maximum and minimum frequencies of a virtual machine (VM) are 50 and 750 MHz, respectively and a task requires 350 MHz, then the VM can be permitted to use only 350 MHz frequency. Hence, the other tasks, which require less than 400 MHz frequency, can also run simultaneously on the same VM to reduce the energy consumption effectively.

2 Related Work

Processing the jobs using clouds are getting more popular due to the cost effectiveness offered for businesses and individuals. However, the data centers consume huge amounts of energy for maintaining the clouds. Hence, many researchers aim to reduce the energy consumption and/or makespan in data centers. Jin et al. [4] proposed an energy-efficient solution to the problem of task scheduling on restricted parallel processors. They also used a speed scaling method to save the energy subjected to the execution time constraint. Piatek et al. [5] proposed a model that can be effectively used to estimate system performance, energy consumption, and energy-efficiency metrics. Ding et al. [6] developed a new VM scheduler to reduce energy costs. They concluded that an optimal frequency exists for a physical machine to process certain VMs. A taxonomy of resource management techniques was presented by Mustafa et al. [7] by considering energy efficiency, SLA-awareness, network load minimization, load balancing, and profit maximization as the main factors.

A smart green energy-efficient scheduling strategy was suggested by Lei et al. [8] to increase the utilization of renewable energy as well as the task satisfaction rate and reduce the system's running cost in a data center. Pedram [9] analyzed the resource provisioning and thermal management problems in data centers. Beloglazov et al. [10] proposed the taxonomy of energy-efficient design of computing systems like hardware, operating system, virtualization and data center levels. Quan et al. [11] dealt with reducing the energy consumption of the traditional data center. Rescheduling of resource allocation, workload consolidation and frequency adjustment have been carried out to save energy. Castane et al. [12] used power meter devices to measure the aggregated power use of a machine to calculate the energy consumption. They also used another method, which involves directly instrumenting the motherboard with multimeters to obtain each power connector's voltage and current in order to calculate the real-time power consumption. An optimal energy-aware load dispatching model was proposed by Zheng and Cai [13] to minimize the electricity and network costs of Online Service Providers (OSPs). This model can greatly reduce the costs for OSPs by considering the volatility of the electricity market and by applying energy efficient strategies in each data center.

Wang et al. [14] determined the energy efficiency using a time-aware model and solved the problem of assigning virtual machines to the servers for reducing the amount of traffic and generating favorable conditions to the traffic engineering. Their proposed method also reduces the number of active switches and balanced traffic flows. Kim et al. [15] estimated the energy consumption of a virtual machine that was based on in-processor events generated by the virtual machine. The Energy-Credit Scheduler (ECS) was proposed in order to schedule VMs according to their energy budgets instead of processor time. However, it only focuses on the processor energy consumption but not the energy consumption of other components. A simulation-driven methodology was proposed by Luo et al. [16]. to accurately model the energy consumption. They also introduced a new resource scheduling algorithm called the Best-Fit-Decreasing-Power (BFDP) to improve the energy efficiency without degrading the QoS of the system.

Hammadi and Mhamdi [17] categorized the existing Data Center Network (DCN) architectures into switch-centric and server-centric topologies and also described various techniques for minimizing the energy consumption. An energy-aware online provisioning approach was suggested by Rodero et al. [18] for HPC applications on consolidated and virtualized computing platforms. An Energy-aware Multistart Local Search algorithm (EMLS-ONC) was proposed by Kessaci et al. [19] to optimize the energy consumption of an Open Nebula-based cloud. A general power consumption optimization algorithm was suggested by Luo and Zhou [20] for the cloud workflow scheduling based on a SLA (Service Level Agreement). It reduces the power consumption while meeting the performance-based constraints of time and cost. Guoning et al. [21] carried out task scheduling using a genetic simulated annealing algorithm to minimize the execution time. Priyanto and Adiwijaya [22] used Ant Colony Optimization (ACO) to find the best schedule having minimum fluctuation. A new ant colony optimization algorithm was proposed by Preve [23] to minimize the makespan and to balance the entire system load in the grid environment. Banerjee et al. [24] proposed a modified ant colony optimization approach for the service allocation and scheduling mechanism to minimize scheduling throughput to service all requests according to different resource allocators. Feller et al. [25] modeled the workload placement problem as an instance of the multi-dimensional bin-packing (MDBP) problem to compute the placement dynamically according to the current load. A particle swarm optimization (PSO) based scheduling heuristic was presented by Pandey et al. [26] for data intensive applications that take into account both computation cost and data transmission cost. Tayal [27] suggested a task scheduling mechanism to get high resource utilization.

A machine learning (ML) algorithm runs on computer systems in order to learn complex relationships and to make the right decisions. Machine learning algorithms are used in cloud computing to predict the resources based on its future load and security. Ajila and Bankole [28] evaluated linear regression (LR), a Support Vector Machine (SVM), and an Artificial Neural Network (ANN) to predict the future resource demands in VM resource provisioning in cloud computing. They developed a cloud client prediction model to optimize the response time and throughput, but not the energy consumption. Bala and Chana [29] tested several machine learning algorithms such as ANN, SVM, and Random Forecasting (RF) to predict host overutilization and underutilization. Kumar and Patel [30] used an ANN-PSO model to minimize the cost and makespan. They used ANN to map the resources with requests. Islam et al. [31] stated that the accuracy of ANN in predicting the resource utilization in a cloud environment is better than other machine learning algorithms.

Even though many researchers aimed to minimize either the execution time or the energy consumption of the cloud data center, they did not concentrate on reducing both execution time and energy consumption simultaneously. In general, making an effort to minimize makespan leads to high energy consumption and vice-versa. Hence, it is necessary to obtain the non-domination solutions for these multiple and conflicting objectives by using a multi-objective optimization method. Suresh et al. [32] carried out a multi-objective optimization problem using Particle Swarm Optimization (PSO), which is a population based optimization technique and is stimulated by social behavior of bird flocking. Each solution in the search space is called as a particle and it adjusts its position in the search space according to its own flying experience and the flying experience of other particles. Omkar et al. [33, 34] applied Quantum behaved Particle Swarm Optimization (QPSO) and Vector Evaluated Particle Swarm Optimization (VEPSO) for the multi-objective optimization problems. The suitable weights have been added to the objective values and the multi-objective optimization problems were solved like a single objective optimization problem. The main disadvantage of PSO is that the global convergence cannot be guaranteed and it has been rectified in QPSO by representing the particle's state with wave function instead of position and velocity. The VEPSO method employs a separate swarm for each objective function, and information migration among the swarms ensures an optimal solution with respect to all the objectives. However, finding the exact weight factors of each objective function is practically impossible. Hence, the authors of this paper are fascinated by a multi-objective optimization approach, known as the Non-domination Sorting Genetic Algorithm (NSGA-II). NSGA II has been proposed by Deb [35] and successfully used by many researchers [36, 37] to find the non-domination solutions for the conflicting multi-objectives.

The main contributions of this paper are as follows:

- The mathematical models are formulated to calculate the energy consumption and makespan of multi-task scheduling problems.
- The Dynamic Voltage Frequency Scaling (DVFS) is included in the optimization procedure to cut down the energy consumption.
- The single objective optimization is performed using the Simple Genetic Algorithm (SGA) to minimize the energy consumption and makespan individually.
- The multi-objective optimization is performed using NSGA-II to minimize both energy consumption and makespan simultaneously.
- The Artificial Neural Network (ANN) is used in the genetic algorithm based optimization procedure to predict the available resources based on their features and the characteristics of the tasks.

The paper is organized as follows. The mathematical models are formulated in Sect. 3 for calculating energy consumption and makespan for the multi-task scheduling problem. The single and multi-objective optimization techniques are described in Sect. 4. The numerical examples are solved using the proposed concept and the main findings are discussed in Sect. 5. The conclusion of this work is presented in Sect. 6.

3 Mathematical Models

The formulation of mathematical models to calculate the energy consumption and makespan of cloud service that consists of j number of tasks and m number of resources is discussed in this section.

3.1 Model for Energy Consumption

As the DVFS system is employed in this work to minimize the energy consumption, each resource can run at a different voltage and frequency range. Let V is assumed as a set of virtual machines $\{V_1, V_2, V_3, \ldots, V_m\}$ that can run in a voltage range of v^i_{min} and v^i_{max}, T is a set of tasks $\{T_1, T_2, T_3, \ldots, T_j\}$ to be executed using V and ST is a set of subtasks $\{ST_1, ST_2, ST_3, \ldots, ST_s\}$ for each task T_j. If the required working frequency of a subtask is taken as f^s_{ST}, then the power consumption of the virtual machine to run the subtask can be calculated using Eq. (1).

$$P = \lambda v^2 f \tag{1}$$

where λ is a constant known as the switching activity factor and can be obtained by multiplying the flip frequency (the number of switches per clock cycle) with the load capacitance, v is the voltage required for the resource i to run the subtask of task j, and f is the working frequency. The value of the switching activity factor usually varies from 0 to 1, and it is assumed as 1 in this work. If $v_k(i)$ and $f_k(i)$ are the required voltage supply and the frequency and $ET_{(i,j)}$ is the processing time (in seconds) to complete the subtask, then the power consumption to execute the subtask of task T_j can be given as:

$$E_{ij} = \lambda f(i)_j \left[v(i)_j\right]^2 ET_{(i,j)}. \tag{2}$$

The total power consumption to complete all the tasks in a virtual machine can be calculated using Eq. (3).

$$E_{Total} = \sum_{i=1}^{m} \sum_{j=1}^{n} \lambda f(i)_j \left[v(i)_j\right]^2 ET_{(i,j)}. \tag{3}$$

3.2 Model for Makespan

The makespan is the total time required to complete all the tasks and subtasks. The processing time and waiting time (idle time) are considered in this work to calculate the makespan. The time required to complete a subtask on a specific resource is called as the processing time and it is calculated based on the size of the database (in GB) of the subtask and the processing capacity (in GB/h) of the resource. As the DVFS system is used to regulate the energy consumption by controlling the CPU frequency, the processing time of the application may critically vary according to the level of the DVFS system. The processing time is calculated using Eq. (4) based on the concept proposed by Hsu and Kremer [38].

$$T_P(in\ \text{hr.}) = \frac{DBS_{subtask}(in\ GB)}{PC_{resource}\left(in\,\frac{GB}{hr.}\right)} \times \left[\left(\frac{f_i^{\max}}{f_s^{ST}} - 1\right)\gamma^i_{CPU} + 1\right] \tag{4}$$

where DBS is the size of the database, PC is the processing capacity, f_i^{max} is the maximum frequency of the resource i, f_s^{ST} is the frequency required to execute the subtask ST_s and γ^i_{CPU} is the CPU bound. The CPU bound is usually varied from 0 to 1 and it is assumed as 1 in this work, which means that processor utilization is high. When a subtask is randomly assigned to a resource that does not finish the previous subtask assigned to it, the resource may be idle for some time, and it is considered as the idle time or waiting time of the resource (T_W). The total time required for the resource i to execute m number of tasks can be found using Eq. (5).

$$T_i = \sum_{j=1}^{m} T_P^i + \sum_{j=1}^{m} T_W^i. \tag{5}$$

In a cloud service, if r number of resources is used, then the makespan is given by

$$Makespan = \max(T_1, T_2, T_3, \ldots, T_r). \tag{6}$$

4 Formulation of Optimization Problem

The goal of the optimization process is to identify the best solution from a set of available solutions. In order to compare all the acceptable solutions that are existing in the design space, a design function or a criterion is needed and this design function or criterion is called the objective function. The optimization problem that has only one objective function is called as single objective optimization problem. However, real world engineering problems, like the task scheduling of a cloud service, shall have more than one design function. If all those design functions are crucial, the problem is considered as a multi-objective optimization problem. The parameters, which are varied to maximize or minimize those design functions, are known as design variables.

In this paper, task scheduling is initially carried out to enhance any one of the objective functions (Makespan or Energy consumption) in order to study the effectiveness of the proposed concept. In a single objective optimization problem, the secondary function is treated as the design constraint. Minimizing the makespan and energy consumption are considered as the design objectives of the single objective optimization problem. The maximum limits for the secondary objectives are considered as the design constraint. Completing the subtasks in a sequential manner is set as the design constraint in both single objective and multi-objective optimization problem. The simple genetic algorithm is used to find the optimum solution for the single objective optimization problem. Later, the non-domination solutions are obtained for the dual objectives using the non-domination sorting genetic algorithm (NSGA-II). The dual objective functions are

$$\begin{aligned}
&\text{Minimize the energy consumption, } \phi_1 = \sum E_{ij} \\
&\quad \text{where, } E_{ij} \text{ is the energy required to complete task j on virtual machine i} \\
&\quad i \text{ and } j \text{ vary as follows:} \\
&\quad i = 1,2,3,\ldots,M \\
&\quad j = 1,2,3,\ldots,J \\
&\text{Minimize the makespan, } \phi_2 = max\left(t_1, t_2, t_3, \ldots, t_i\right) \\
&\quad where,\ t_i \text{ is the makespan for the virtual machine } i \\
&\text{Subject to the following constraints:} \\
&\quad \text{Completion of subtasks } \left(ST_k^j\right) \text{ in a sequential manner} \\
&\qquad \text{where } j \text{ is the task number and } k \text{ is the subtask number} \\
&\quad \text{makespan } \phi_2 \le Max.\ Makespan \\
&\quad \text{energy consumption } \phi_1 \le Max.\ Energy
\end{aligned} \tag{7}$$

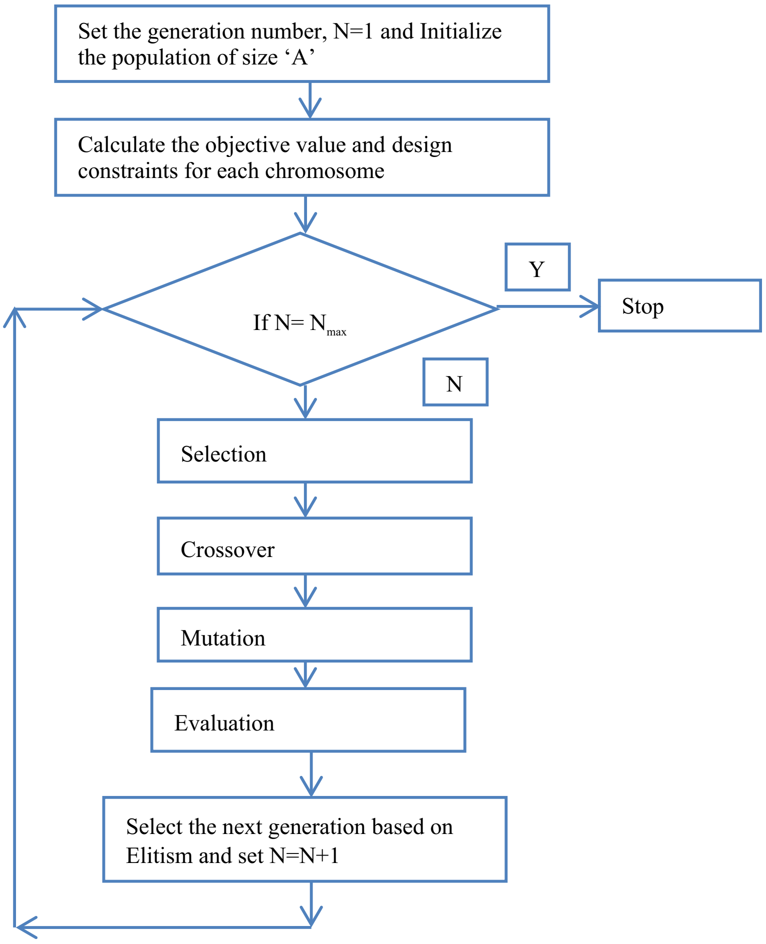
Set the generation number, N=1 and Initialize the population of size 'A'
Calculate the objective value and design constraints for each chromosome
If N= N_{max}
Y
Stop
N
Selection
Crossover
Mutation
Evaluation
Select the next generation based on Elitism and set N=N+1

Fig. 1 Step-by step procedure of SGA

4.1 The Simple Genetic Algorithm

The step-by-step procedure of the simple genetic algorithm is shown in Fig. 1. The initial population of size *N* is generated in a random manner. Each solution in the current population is called a chromosome, which is nothing but the allocation of resources to execute the set of tasks and their subtasks in the cloud service. An example of a chromosome, which is generated during the initial population, is given in Table 1. The sample problem consists of five resources and three tasks where each task comprises three subtasks. The real encoding is used to represent each gene (resource number) in order to avoid the errors occurring during the encoding and decoding process. The genes are randomly generated during the initial population within the specified range to execute the subtasks of each task.

Table 1 An example for a chromosome

Tasks	T_1			T_2			T_3		
	ST_1	ST_2	ST_3	ST_1	ST_2	ST_3	ST_1	ST_2	ST_3
Resources	1	5	1	4	2	2	4	3	5

Table 2 Uniform crossover operator

Crossover probabilility = 0.85; mixing ratio = 0.5; random number = 0.88									
Before crossover									
Parent 1	1	5	1	4	2	2	4	3	5
Parent 2	3	4	2	1	4	3	5	1	3
After crossover									
Child 1	1	*4*	1	*1*	2	*3*	*5*	3	*3*
Child 2	3	*5*	2	*4*	4	*2*	*4*	1	*5*

The numbers altered during the crossover process are given in italics

The objective functions and design constraints (makespan/energy consumption) of each chromosome (solution) in the current population are calculated using Eqs. (3) and (6). If the stopping criterion is satisfied, the algorithm is stopped and the best design (solution) is presented. In general, the maximum number of generations is considered as the stopping criterion in SGA. If the stopping criterion is not satisfied, the genetic operators are applied on the current population in order to evolve a new population. The genetic algorithm works with a set of operators known as selection, crossover and mutation to reproduce the offspring. The selection is the first operator applied on the current population to select the healthy chromosomes (solutions) to act as the parents. As the healthy chromosomes have a very high probability for being selected as parents, the genetic algorithm is generally concerned with the maximization problem. However, it may also be applied for the minimization problem either by altering the sign of the objective function or taking the reciprocal value of the objective function. The roulette wheel selection method, which was successfully used by many researchers [36, 39] has been used in this work due to its expediency and efficacy.

The crossover and mutation are applied to the selected parents based on the user defined probabilities in order to alter the genes and to identify better offspring. The uniform crossover operator is employed in this work in order to obtain the changes in the gene level. The uniform crossover exchanges the genes between the parents in a random manner. The number of genes exchanged between the parents is decided based on the mixing ratio defined by the user. An example of uniform crossover is given in Table 2. The crossover probability is set in order to retain some healthy chromosomes in the matting pool by setting them free from crossover operation. For example, if the crossover probability is set as 0.85, a random number is generated between 0 and 1 and the chromosomes are permitted for crossover, if the random number is more than 0.85 (crossover probability). If the random number is below

Table 3 Uniform mutation operator

Crossover probabilility = 0.05; random number = 0.2									
Before mutation	1	4	1	1	2	3	5	*3*	3
After mutation	2	3	3	1	1	3	1	*4*	3

The numbers altered during the crossover process are given in italics

0.85, the same parent chromosomes are permitted to enter into the child population without doing any modification. In the example given in Table 2, the sample crossover probability is set as 0.85, and the random number generated is assumed as 0.88. Hence, the uniform crossover is performed on the parent chromosomes by randomly selecting and exchanging five genes of parent 1 with the respective genes of parent 2 (the user defined mixing ratio is assumed as 0.5).

Following the crossover, mutation is performed to prevent the algorithm from getting stuck in any local optimum value. In the mutation process, a gene of a chromosome is randomly chosen and replaced by a new gene that is arbitrarily chosen within the specified range of design variables. The uniform mutation is employed in this work, and an example is given in Table 3. A user defined mutation probability is also to be set in order to avoid performing mutation in all the chromosomes. On the other hand, if the mutation is performed in all the chromosomes, the search becomes a random walk and consumes the high computational cost for the convergence. An example for a uniform mutation operator is given in Table 3, where the mutation probability is set as 0.05 and the random number generated is assumed as 0.2. Therefore, a randomly selected gene is altered by a new gene in the chromosome. After the mutation, the objective function and design constraints of the chromosomes existing in the new population are calculated and the fitted chromosomes are sorted based on their objective values. A method called elitism is used to retain the better chromosomes of the current population for the next generations. In elitism method, a specified size of the best chromosomes is identified from the parent population and is used to replace the worst chromosomes of the same size in the child population. This prevents the optimization technique from wasting time with rediscovering the partial solutions. In this research work, the best 5 solutions are identified in each generation and are permitted to take part in the matting pool of the next generation. This cyclic process continues until the stopping criterion is satisfied or the solutions in the current population converge.

4.2 The Non-dominated Sorting Genetic Algorithm (NSGA-II)

The Non-dominated Sorting Genetic Algorithm (NSGA-II) proposed by Deb [35] is used to obtain the non-domination (Pareto optimal) solutions for the multi-objective optimization problem. The pseudo code of NSGA-II is as follows:

Step 1: Set the number of generation (N) as 1 and initialize the population of size A

Step 2: Evaluate the objective functions and design constraints for each chromosome

Step 3: Sort the chromosomes based on each objective function and calculate the crowding distance using Eqs. (8) and (9)

Step 4: Rank the solutions based on the front in which they are existing and by crowding distance

Step 5: Obtain the non-domination solutions (for which the rank is assigned as one)

Step 6: While $N < N_{max}$

- Select the parents using tournament selection
- Apply crossover and mutation
- Evaluate the objective functions
- Combine the offspring and current population
- Sort and rank the chromosomes
- Select the chromosomes of size A as the next generation
- Set $N = N + 1$.

The population of NSGA-II is initialized in a random manner and the objective values of each chromosome are calculated. The chromosomes are sorted based on each objective function and they are ranked using a fitness assignment system, which favors non-dominated solutions. Further, the crowding distance is also calculated for each chromosome in order to indicate the closeness of a design to its neighbors when they are ranked. Usually, very large crowding distances are assigned to the designs existing in the boundary of the front as given in Eq. (8) and the crowding distances of the remaining designs are calculated using Eq. (9).

$$\text{Crowding distance of boundary solutions,} \quad d_1^m = d_S^m = \infty \tag{8}$$

$$\text{Crowding distance of other solutions,} \quad d_j^m = I_j^m + \frac{I_{j+1}^m - I_{j-1}^m}{f_{\max}^m - f_{\min}^m} \tag{9}$$

where m is the number of objective functions, s is number of solutions existing in a front, f_{max} and f_{min} are the maximum and minimum values of the objective functions (m), I is the vector in which the solutions are sorted in worst order, based on the objective functions (m).

The solutions existing in the first front (ranked as 1) are absolutely non-dominant solutions. Likewise, the solutions available in the second front are dominated only by the solutions existing in the front one and the ranks are assigned to the other fronts in a similar fashion. The tournament selection method is used to select the parents from the current population based on their rank and crowding distance. The crowding distance plays a major role when two parent chromosomes, which are compared in tournament selection, have equal rank. The crowding distance is a measure of closeness of an individual to its neighbors. If a chromosome has large crowding distance, it will result in better diversity in the population and so the chromosome having the higher crowding distance is considered as the better one. For example, if *p* and *q* are the two parent chromosomes having equal rank, then the chromosome *p* will be selected during the tournament selection if the crowding distance of *p* is greater than the crowding distance of *q*. The uniform crossover and mutation methods are applied on the selected chromosomes to evolve the offspring. The elitism method is used to combine the parent and offspring population in order to provide the best designs for the next generation. The process continues until the stopping condition is satisfied.

4.3 Artificial Neural Network (ANN)

A multilayer feed forward backpropagation neural network is used in this work. The multilayer network has more than one input layer. As the output of each neuron in the proposed network propagates from the input side to the output side it is called a feed forward network. Further, the gradient vector is found in the opposite direction to the flow of output of each neuron, and so the term backpropagation is used. The neural network is prepared in four steps as it is shown in Fig. 2. In the first step, the sample data are randomly generated using Cloudsim and MATLAB code and the desired targets of those sample data are calculated. The network objects such as the number of layers, number of neurons in each layer, the transfer function used in each layer, and the training algorithm of the network are optimally chosen in step two. The network is trained in the next step by using 80% of the sample data prepared in step one and verified in step four using the remaining 20% of the sample data.

The training algorithm of the network has to find a set of weights and biases to the network by which the network can map any set of input data to the concerned outputs. The input data together with the desired targets are provided to the network, and hence, the supervised training algorithm is used to train the network. The weights and biases are to be adjusted to minimize the performance function during the training of the network. The mean square error (*mse*) is generally used as the

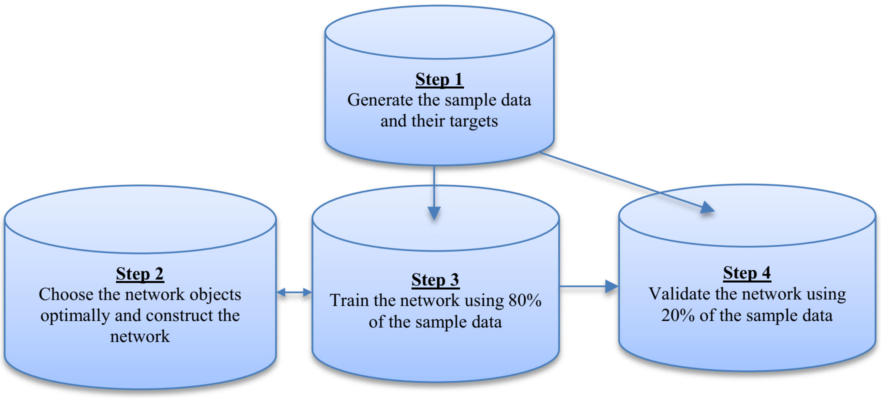
Step 1
Generate the sample data and their targets
Step 2
Choose the network objects optimally and construct the network
Step 3
Train the network using 80% of the sample data
Step 4
Validate the network using 20% of the sample data

Fig. 2 Steps involved in the construction of ANN

Table 4 Various training algorithms

Sl. no	Name of the training algorithm	Abbreviation
1	Resilient backpropagation	RP
2	Scaled conjugate gradient	SCG
3	Conjugate gradient with Powell/Beale restarts	CGB
4	Fletcher–Powell conjugate gradient	CGF
5	Polak–Ribiere conjugate gradient	CGP
6	Levenberg–Marquardt	LM
7	BFGS quasi-Newton	BFG

performance function and is obtained by getting the average squared error between the known outputs (t_k) and the network forecast (a_k) as shown in Eq. (10). The different supervised training algorithms suggested by Demuth and Beale [40] are given in Table 4 and are considered in this work.

$$mse = \frac{1}{N}\sum_{k=1}^{N} e_k^2 = \frac{1}{N}\sum_{k=1}^{N} (t_k - a_k)^2. \tag{10}$$

As each training algorithm listed in Table 4 has its own merits and demerits, the best one has to be selected based on the trial and error method. The Resilient Backpropagation (RP) algorithm finds the direction of the weight update based on the sign of the derivative alone, and the magnitude of the derivative has no impact on the weight update. So, this algorithm can permit only slight alterations in the weights and biases. However, RP is normally a faster training algorithm than the other steepest descent algorithms. The conjugate gradient algorithms search towards the conjugate directions, and hence, the convergence is quicker in the steepest descent directions. Four different conjugate gradient algorithms are considered in this work and all four algorithms initiate their search in the steepest descent direction as given in Eq. (11).

$$p_0 = -g_0 \tag{11}$$

where p_0 and g_0 are initial search direction and initial gradient, respectively. Consequently, a line search is used as given in Eq. (12) to find the optimum distance and to move towards the current search direction.

$$x_{k+1} = x_k + \alpha_k p_k \tag{12}$$

where x_k is the vector of weight and bias of the current search and α_k is the learning rate of the current search. The current search direction (p_k) is acquired by adding the current steepest descent direction (g_k) to the previous search direction (p_{k-1}) as shown in using Eq. (13).

$$p_k = -g_k + \beta_k p_{k-1}. \tag{13}$$

Each conjugate gradient algorithm differs in calculating the constant β. The Fletcher–Reeves conjugate gradient algorithm calculates the constant by taking the ratio between the averages squared of the current gradient with the previous gradient.

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}. \tag{14}$$

The Polak–Ribiere conjugate gradient algorithm works based on the inner product of the previous alteration in the gradient (Δg_{k-1}) with the current gradient divided by the average squared of the previous gradient.

$$\beta_k = \frac{\Delta g_{k-1}^T g_k}{g_{k-1}^T g_{k-1}}. \tag{15}$$

Generally, the network remembers the training sample data and executes them well during the training. However, the same network may be overfit when it is used for the new inputs, and hence, Demuth and Beale [40] suggested a technique known as regularization to improve the generalization of the network. The performance function is modified in this method by adding the mean square error with the mean squared weights and biases (*msw*) of the network. The improved performance function used in this work is given in Eq. (16).

$$\begin{aligned} &mse reg = \beta(mse) + (1-\beta)msw \\ &\text{where } msw = \frac{1}{n}\sum_{j=1}^{n} w_j^2 \end{aligned}. \tag{16}$$

This improved performance function allows the network to have weights and biases in smaller amounts in order to reduce the chances for overfitting. The performance ratio (β) of the improved performance function varies from 0 to 1. If the performance ratio is nearer to one, the improved performance function acts like the mean square error performance function and may lead to overfitting. On the other hand, if the performance ratio is reduced to zero, the mean square error will be snubbed and the network may not efficiently fit the training data. The value of the performance ratio has been chosen as 0.2 in this work based on the trial and error method.

5 Numerical Results and Discussion

In cloud service, several heterogeneous resources, which are located at different places, are employed by the cloud users based on a pay-per-use method. The cloud users usually desire to complete their tasks in less time and with minimum cost. Likewise, the cloud providers yearn for high profit by reducing the cost of energy consumption. Hence, the task scheduling is essential to identify the best sequence of tasks to be executed in the corresponding resources. The subtasks of various tasks can simultaneously run at different resources with the constraint of executing the subtasks in a sequential manner. For instance, if a task *j* has *s* number of subtasks, the subtask ST_k can only be executed after the completion of the subtask ST_{k-1}. In addition to task scheduling, the DFVS technology is also employed to reduce the energy consumption. The features of the resources considered in this paper are listed in Table 5, whereas the details of the sample tasks and subtasks are listed in Table 6.

5.1 Construction of ANN

The performance of the neural network greatly depends on the selection of the network objects like the number of layers, number of neurons in hidden layers, type of transfer functions, and training algorithm. Yuen and Lam [41], Bolanca et al. [42], Kermanshahi and Iwamiya [43] and Chakraborty [44] recommended that the network objects are to be identified either based on the experience or rule of thumb

Table 5 The details of resources

Resource ID	Processing speed in GB/h	SLA gear	$f_{\min}$ (MHz)	$f_{\max}$ (MHz)	Voltage
R1	1.4	4	1441	1800	1.3
		3	1081	1440	1.2
		2	721	1080	1.1
		1	361	720	1
		0	0	360	0.9
R2	0.85	4	1361	1700	1.65
		3	1021	1360	1.45
		2	681	1020	1.25
		1	341	680	1.05
		0	0	340	0.85
R3	0.6	4	1281	1600	1.15
		3	961	1280	1.05
		2	641	960	0.95
		1	321	640	0.85
		0	0	320	0.75
R4	0.74	4	1121	1400	1.2
		3	841	1120	1.1
		2	561	840	1
		1	281	560	0.9
		0	0	280	0.8
R5	0.64	4	1601	2000	1.5
		3	1201	1600	1.4
		2	801	1200	1.3
		1	401	800	1.2
		0	0	400	1

Table 6 The details of tasks and subtasks

	ST_1	ST_2	ST_3	ST_4	ST_5	ST_6	ST_7	ST_8	ST_9	ST_{10}
Size of the subtasks (S) in GB and required frequency (f) in MHz										
T_1										
S	0.5	0.8	1.2	1.5	0.4	1.8	0.9	2.1	0.6	0.4
F	150	300	300	350	125	450	450	300	200	125
T_2										
S	0.4	1.2	0.4	0.6	1.3	1.2	0.9	1.5	0.6	1.4
F	125	150	125	200	250	300	450	450	200	250
T_3										
S	0.8	0.5	0.2	0.3	0.8	0.4	0.9	1.2	1.3	1.5
F	300	150	100	125	300	125	450	400	250	450
T_4										
S	1.3	0.2	0.6	0.85	0.94	0.86	1.3	1.9	2.1	0.3
F	900	100	200	400	700	350	830	250	600	125
T_5										
S	1.3	0.3	0.8	0.56	0.42	0.25	0.26	1.3	0.8	0.65
F	900	125	300	175	140	125	130	550	300	325

only. The neural network constructed with three layers (an input layer, a hidden layer and an output layer) is used in this work. Sigmoid transfer functions are used in the input layer and hidden layers, since the inputs and targets have been normalized between -1 and 1. The sigmoid transfer function compresses an infinite range of inputs into a finite range of output. A tan-sigmoid transfer function has been chosen for the first two layers, and a linear transfer function has been used in the last layer. Seven networks have been constructed and trained separately using the seven different training algorithms given in Table 4. After the successful training, each network was validated using the new sample data. The maximum error and average error rates have been calculated in the percentage for each network and were compared in Fig. 3. The results show that the prediction of the network constructed with a CGF type training algorithm is better than the prediction of the networks constructed with other training algorithms.

5.2 Minimization of Makespan and Energy Consumption Using SGA

The single objective optimization is performed initially using the simple genetic algorithm in order to minimize the energy consumption and makespan independently. The optimization process is carried out for the cloud services that consist of a different number of tasks (1000, 1250, 1500, 1750 and 2000). The completion of subtasks in a sequential manner and maximum limits for the energy consumption and makespan are considered as design constraints. The efficiency of the optimization procedure, in which the machine learning algorithm (ANN) is enabled to predict the resources based on the task size and features is investigated. The

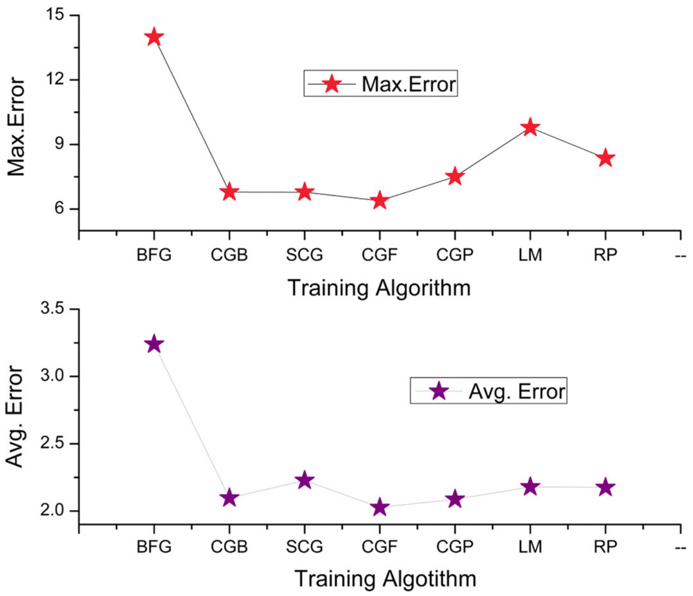

Fig. 3 Comparison of training algorithms

simple genetic algorithm is constructed based on the procedure given in Fig. 1 and is run several times to find its optimum control parameters. The control parameters of SGA such as the crossover probability, mutation probability, number of generations, and population size are optimally varied to find the GA control parameters. The crossover probability varies from 0.6 to 0.9 with an increment of 0.5 and the mutation probability varies from 0.05 to 0.1 with an increment of 0.01. Likely, the population size varies from 30 to 60 with an increment of 10. The results obtained by the genetic algorithm with these different configurations have been compared with one another and the optimum set of control parameters is identified and listed in Table 7. The maximum number of generations is chosen as 350 based on the convergence of GA solutions.

The optimum makespan and energy consumption obtained for the cloud services consists of a various number of tasks as shown in Figs. 4 and 5. The Fig. 4 shows that the optimization procedure, in which DVFS is not incorporated offers a better solution than the optimization procedure constructed with DVFS. The reason for this is that the optimization procedure functioning without DVFS permits the virtual machines to run at their maximum frequency. While the resources are operated at

Table 7 Control parameters for SGA

Control parameter	Value
Crossover probability	0.8
Mutation probability	0.07
Number of generations	350
Population size	40

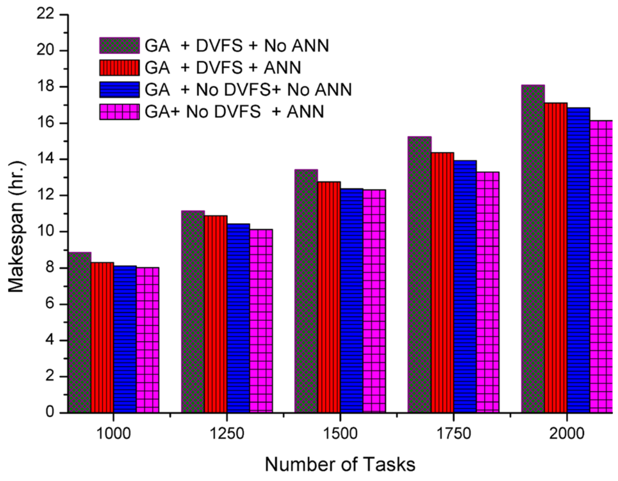

Fig. 4 Optimum makespan obtained using SGA

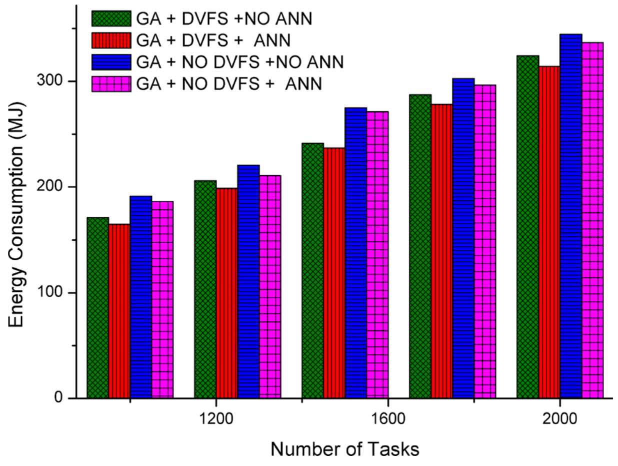

Fig. 5 Optimum energy consumption obtained using SGA

their maximum frequency, the maximum voltage is consumed and less time is required for processing the tasks. However, the energy consumption critically increases due to a very high operating voltage. Figure 5 shows that the optimization technique in which DVFS is incorporated consumes less energy, as the resources are operated at the minimum required voltage and frequency. Similarly, Figs. 4 and 5 also prove that the optimization procedure constructed with ANN finds a better solution than the optimization procedure constructed without ANN. The main reason behind this is that the resources are chosen in a random manner only based on their availability when ANN is not used in GA. On the other hand, the resources

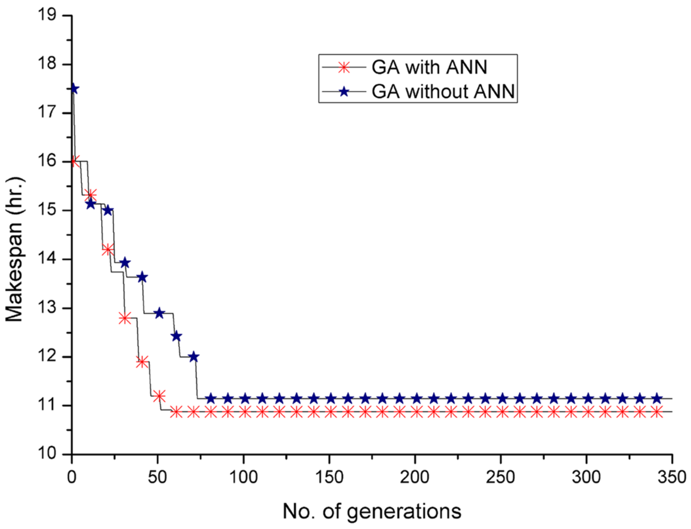

Fig. 6 Convergence of solutions during the makespan minimization problem

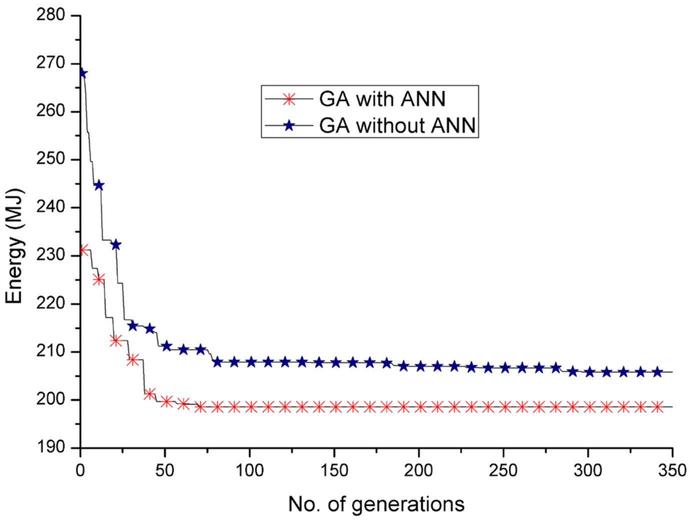

Fig. 7 Convergence of solutions during the energy consumption minimization problem

are chosen based on the predicted completion time and energy consumption while ANN is used in the optimization process.

Additionally, the convergence of solutions obtained during the single objective optimization process are ploted in Figs. 6 and 7, which show that the optimum solutions are obtained in a minimum number of generations while ANN is enabled in the optimization procedure. Thus, the proposed concept also reduces the computational cost of GA.

5.3 Multi-objective Optimization to Minimize the Operating Cost and Makespan

In the single objective optimization problem, the major design function is considered as the objective function and all other design functions are treated as design constraints, and therefore, the single objective optimization problem can satisfy either the cloud user (by providing the minimum makespan) or the cloud provider (by reducing the energy consumption). Thus, the multi-objective optimization is performed to yield the benefits simultaneously to the cloud users and cloud providers. While the other multi-objective optimization procedures lead to a single solution, the NSGA offers a set of solutions called non-domination solutions or Pareto-optimal solutions. The set of non-domination solutions obtained using NSGA-II is shown in Fig. 8 for the dual objectives makespan and energy consumption. The non-domination solutions were also obtained with the support and without the support of ANN. Each solutions existing in the Pareto-optimal set (with ANN and without ANN) is not dominated by the other solutions in the same set. For example, the makespan and energy consumption of the upper boundary solution obtained using NSGA and ANN are 12.9 h and 219.5 MJ, respectively. Likewise, the makespan and energy consumption of the lower boundary solution of the same Pareto-optimal set is 29.95 h and 190 MJ, respectively. However, these two solutions cannot be compared with each other, as the first one is optimum only for the makespan value while the second one is optimum only on account of energy consumption.

Similarly, the other solutions existing in the Parteo-optimal set are also being optimum based on the weightages offered by the cloud users and providers on makespan and energy consumption. However, these weightages will vary from time to time depends upon the priority of the tasks, availability of resources and other factors. Hence it is necessary to find as many solutions as possible in the Pareto optimal set. Figure 8

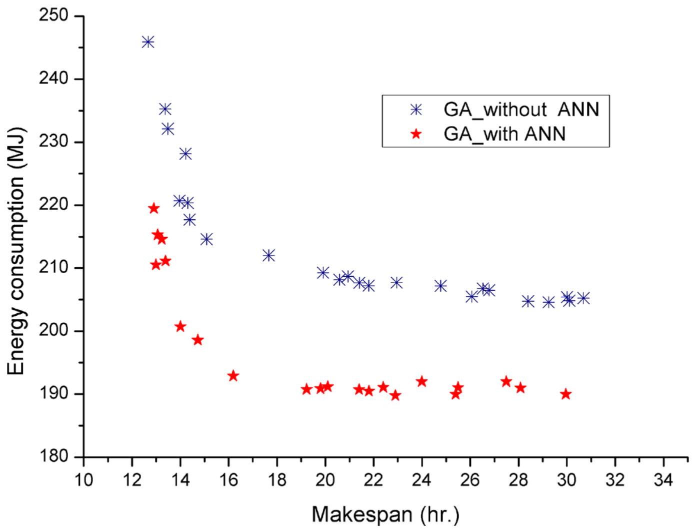

Fig. 8 Pareto optimal designs

shows that a wide range of solutions exists in the Pareto optimal sets by which the cloud users and providers can select the most crucial solution based on their needs. Figure 8 also shows that the solutions obtained by NSGA without the support of ANN are completely dominated by the solutions obtained by NSGA with the support of ANN.

6 Conclusion

Minimizing makespan and energy consumption of the cloud service is considered as the major objective of this paper. The energy consumption was effectively controlled by optimally regulating the frequency and voltage of the virtual machine by using the DVFS system. As makespan and energy consumption are the contradictory objectives, the non-domination solutions were sought using the non-domination optimization procedure. Furthermore, the artificial neural network was also used in the optimization procedure to predict the appropriate resource based on the characteristics of the tasks and available resources. Initially, the single objective optimizations were carried out to minimize the makespan and energy consumption individually and the efficiency of the proposed optimization method was investigated. The numerical results were obtained for a different number of tasks and the results were compared between the optimization techniques in which DVFS and ANN were optionally used. The results show that the energy consumption was effectively minimized by incorporating DVFS in the optimization procedure. However, the makespan critically increased, as DVFS was mainly concerned with reducing the energy consumption. Hence, the optimization problem was converted as a multi-objective and the non-domination solutions were obtained using NSGA-II with the support of ANN and without the support of ANN. The numerical results obtained in the multi-objective optimization problem show that a wide range of solutions exists in the Pareto-optimal front by which the cloud user and provider can benefit by choosing the more appropriate one based on the time varying factors such as the priority of the tasks and availability of resources. Furthermore, the results obtained in single objective and multi-objective optimization problems show that the optimization procedure constructed with ANN finds a better solution than the optimization procedure constructed without ANN, as ANN predicts the resources based on the forecasted completion time and energy consumption. The computational complexity of the genetic algorithm usually increases with the number of generations. However, it is proven in this paper that the number of generations shall be critically reduced by using ANN in the optimization procedure. Additionally, the computational cost is minimized by optimally choosing the control parameters of GA such as number of generations, population size, crossover probability, and mutation probability.

References

1. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R.: CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Softw. Pract. Exp. **41**(1), 23–50 (2011)

2. Wu, C.M., Chang, R.S., Chan, H.Y.: A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. Future Gener. Comput. Syst. **37**, 141–147 (2014)
3. Kliazovich, D., Bouvry, P., Khan, S.U.: GreenCloud: a packet-level simulator of energy-aware cloud computing data centers. J. Supercomput. **62**(3), 1263–1283 (2012)
4. Jin, X., Zhang, F., Fan, L., Song, Y., Liu, Z.: Scheduling for energy minimization on restricted parallel processors. J. Parallel Distrib. Comput. **81**, 36–46 (2015)
5. Piątek, W., Oleksiak, A., Da Costa, G.: Energy and thermal models for simulation of workload and resource management in computing systems. Simul. Model. Pract. Theory **58**, 40–54 (2015)
6. Ding, Y., Qin, X., Liu, L., Wang, T.: Energy efficient scheduling of virtual machines in cloud with deadline constraint. Future Gener. Comput. Syst. **50**, 62–74 (2015)
7. Mustafa, S., Nazir, B., Hayat, A., Madani, S.A.: Resource management in cloud computing: taxonomy, prospects, and challenges. Comput. Electr. Eng. **47**, 186–203 (2015)
8. Lei, H., Zhang, T., Liu, Y., Zha, Y., Zhu, X.: SGEESS: smart green energy-efficient scheduling strategy with dynamic electricity price for data center. J. Syst. Softw. **108**, 23–38 (2015)
9. Pedram, M.: Energy-efficient datacenters. IEEE Trans. Comput. Aided Des. Integr. Circuits Syst. **31**(10), 1465–1484 (2012)
10. 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)
11. Quan, D.M., Mezza, F., Sannenli, D., Giafreda, R.: T-Alloc: a practical energy efficient resource allocation algorithm for traditional data centers. Future Gener. Comput. Syst. **28**(5), 791–800 (2012)
12. Castane, G.G., Nunez, A., Llopis, P., Carretero, J.: E-mc 2: a formal framework for energy modelling in cloud computing. Simul. Model. Pract. Theory **39**, 56–75 (2013)
13. Zheng, X., Cai, Y.: Energy-aware load dispatching in geographically located internet data centers. Sustain. Comput. Inform. Syst. **1**(4), 275–285 (2013)
14. Wang, L., Zhang, F., Arjona Aroca, J., Vasilakos, A.V., Zheng, K., Hou, C., Li, D., Liu, Z.: GreenDCN: a general framework for achieving energy efficiency in data center networks. IEEE J. Sel. Areas Commun. **32**(1), 4–15 (2014)
15. Kim, N., Cho, J., Seo, E.: Energy-credit scheduler: an energy-aware virtual machine scheduler for cloud systems. Future Gener. Comput. Syst. **32**, 128–137 (2014)
16. Luo, L., Wu, W., Tsai, W.T., Di, D., Zhang, F.: Simulation of power consumption of cloud data centers. Simul. Model. Pract. Theory **39**, 152–171 (2013)
17. Hammadi, A., Mhamdi, L.: A survey on architectures and energy efficiency in data center networks. Comput. Commun. **40**, 1–21 (2014)
18. Rodero, I., Jaramillo, J., Quiroz, A., Parashar, M., Guim, F., Poole, S.: Energy-efficient application-aware online provisioning for virtualized clouds and data centers. In: Presented at the IEEE International Conference on Green Computing, pp. 31–45 (2010)
19. Kessaci, Y., Melab, N., Talbi, E.G.: A multi-start local search heuristic for an energy efficient VMs assignment on top of the OpenNebula cloud manager. Future Gener. Comput. Syst. **36**, 237–256 (2014)
20. Luo, Y., Zhou, S.: Power consumption optimization strategy of cloud workflow scheduling based on SLA. WSEAS Trans. Syst. **13**, 368–377 (2014)
21. Guo-ning, G., Ting-Lei, H., Shuai, G.: Genetic simulated annealing algorithm for task scheduling based on cloud computing environment. In: Presented at the International Conference on Intelligent Computing and Integrated Systems (2010)
22. Priyanto, A.A., Adiwijaya, W.: Implementation of ant colony optimization algorithm on the project resource scheduling problem. Faculty of informatics, Institute of Technology Telkom, Bandung (2008)
23. Preve, N.: Balanced job scheduling based on ant algorithm for grid network. Int. J. Grid High Perform. Comput. **2**(1), 34–50 (2010)
24. Banerjee, S., Mukherjee, I., Mahanti, P.K.: Cloud computing initiative using modified ant colony framework. World Acad. Sci. Eng. Technol. **56**, 221–224 (2009)
25. Feller, E., Rilling, L., Morin, C.: Energy-aware ant colony based workload placement in clouds. In: Presented at the IEEE/ACM 12th International Conference on Grid Computing, pp. 26–33 (2011)
26. Pandey, S., Wu, L., Guru, S.M., Buyya, R.: A particle swarm optimization-based heuristic for scheduling workflow applications in cloud computing environments. In: Presented at the IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 400–407 (2010)

27. Tayal, S.: Tasks scheduling optimization for the cloud computing systems. Int. J. Adv. Eng. Sci. Technol. **5**(2), 111–115 (2011)
28. Ajila, S.A., Bankole, A.A.: Using machine learning algorithms for cloud client prediction models in a web VM resource provisioning environment. Trans. Mach. Learn. Artif. Intell. **4**(1), 28–51 (2016)
29. Bala, A., Chana, I.: Prediction-based proactive load balancing approach through VM migration. Eng. Comput. **32**(4), 1–12 (2016)
30. Kumar, N., Patel, P.: Resource management using feed forward ANN-PSO in cloud computing environment. In: Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, p. 57 (2016)
31. Islam, S., Keung, J., Lee, K., Liu, A.: Empirical prediction models for adaptive resource provisioning in the cloud. Future Gener. Comput. Syst. **28**(1), 155–162 (2012)
32. Suresh, S., Sujit, P.B., Rao, A.K.: Particle swarm optimization approach for multi-objective composite box-beam design. Compos. Struct. **81**(4), 598–605 (2007)
33. Omkar, S.N., Khandelwal, R., Ananth, T.V.S., Naik, G.N., Gopalakrishnan, S.: Quantum behaved Particle Swarm Optimization (QPSO) for multi-objective design optimization of composite structures. Expert Syst. Appl. **36**(8), 11312–11322 (2009)
34. Omkar, S.N., Mudigere, D., Naik, G.N., Gopalakrishnan, S.: Vector evaluated particle swarm optimization (VEPSO) for multi-objective design optimization of composite structures. Comput. Struct. **86**(1), 1–14 (2008)
35. Deb, K.: Multi-objective Optimization Using Evolutionary Algorithms. Wiley, Hoboken (2001)
36. Nicholas, P.E., Padmanaban, K.P., Babu, M.C.: Multi-objective optimization of laminated composite plate with diffused layer angles using non-dominated sorting genetic algorithm (NSGA-II). Adv. Compos. Lett. **23**(4), 96–105 (2014)
37. Bolanos, R., Echeverry, M., Escobar, J.: A multiobjective non-dominated sorting genetic algorithm (NSGA-II) for the Multiple Traveling Salesman Problem. Decis. Sci. Lett. **4**(4), 559–568 (2015)
38. Hsu, C.H., Kremer, U.: The design, implementation, and evaluation of a compiler algorithm for CPU energy reduction. ACM SIGPLAN Not. **38**(5), 38–48 (2003)
39. Tao, F., LaiLi, Y., Xu, L., Zhang, L.: FC-PACO-RM: a parallel method for service composition optimal-selection in cloud manufacturing system. IEEE Trans. Ind. Inform. **9**(4), 2023–2033 (2013)
40. Demuth, H., Beale, M.: Neural Network Toolbox User's Guide. The Mathworks, Natick (2000)
41. Yuen, K.V., Lam, H.F.: On the complexity of artificial neural networks for smart structures monitoring. Eng. Struct. **28**(7), 977–984 (2006)
42. Bolanca, T., Ukic, S., Peternel, I., Kusic, H., Bozic, A.L.: Artificial neural network models for advanced oxidation of organics in water matrix-comparison of applied methodologies. Indian J. Chem. Technol. **21**(1), 21–29 (2014)
43. Kermanshahi, B., Iwamiya, H.: Up to year 2020 load forecasting using neural nets. Int. J. Electr. Power Energy Syst. **24**(9), 789–797 (2002)
44. Chakraborty, D.: Artificial neural network based delamination prediction in laminated composites. Mater. Des. **26**(1), 1–7 (2005)

A. Sathya Sofia is working as an Assistant Professor in the Department of CSE, PSNA College of Engineering and Technology, India. She received B.E. (ECE) and M.E. (CSE) from Anna University in 2005 and 2007 respectively. She is currently pursuing her Ph.D. in Anna University, Chennai. Her research interests include Cloud Computing, Computer networks and Soft computing.

P. Ganeshkumar is a Professor in Department of IT, PSNA College of Engineering and Technology, India. He completed B.E. (EEE) in Madurai Kamaraj University, in 2001, M.E. (CSE) from Bharathiyar University and Ph.D. in the department of Information and Communication Engineering at Anna University, Chennai. His area of interest are Cloud Computing, Wireless Networks and Distributed Systems.