Applied Soft Computing Journal 99 (2021) 106876

Contents lists available at ScienceDirect

Applied Soft Computing Journal

journal homepage: www.elsevier.com/locate/asoc

Resource provisioning in scalable cloud using bio-inspired artificial neural network model

Pradeep Singh Rawat [a,*,1], Priti Dimri [b,1], Punit Gupta [c,1], G.P. Saroha [d,1]

[a] Uttarakhand Technical University, Dehradun, India
[b] Govind Ballabh Pant Institute of Engineering & Technology, Pauri Garhwal, India
[c] Manipal University Jaipur, India
[d] Maharshi Dayanand University Rohtak, India

ARTICLE INFO

Article history:
Received 20 January 2020
Received in revised form 18 September 2020
Accepted 1 November 2020
Available online 5 November 2020

Keywords:
AFs
ANN (Artificial Neural Network)
ACS (Artificial Task Scheduler)
BB–BC (Big-Bang Big-Crunch)
GA
Scheduling
Task

ABSTRACT

Resource assignment is one of the emerging research area in the cloud scenario. Cloud computing provides a shared pool of resources in a distributed environment. It supports the features of utility-based computing. Efficient task provisioning on virtual machines is the major concern in an extensible cloud computing environment. The task provisioning minimizes the performance metrics total completion time (ms), average start time, average finish time, average execution time, scheduling time, and simulation time respectively. The scheduling is an important problem which becomes more complicated when various parameters consider. The key issue in virtual machine level scheduling is execution time overhead and scalability in a real-time scenario. Our objective is to make an optimal schedule of tasks on a virtual machine inside the datacenter using neural-bio inspired GA-ANN technique. This work presents a scheduler based on a genetic approach and an artificial neural network. The presented approach performs optimal scheduling of tasks on an appropriate virtual machine. The reliability of the system improves by reducing the number of tasks failed. The presented work uses a genetic algorithm to generated huge data sets and trains the neural model using the data set generated by using a genetic approach. The accuracy of the model is improved using back propagation with 98% accuracy. The set of experiments are performed using a scalable cloud computing environment. The presented bio-inspired technique is compared against nature-inspired, bio-inspired cost-aware BB–BC, GA-Cost, and GA-Exe based efficient task scheduling techniques. The results are obtained using real workload logs and synthetic data sets. Results indicate that the proposed GA-ANN bio-inspired predictive approach outperforms the considered nature-inspired scheduling approaches. The proposed algorithm is compared using various performance metrics total completion time, average start time, average finish time, and the fault rate, execution time, and scheduling time respectively. The proposed model reduces the fault rate by 82.63%, successfully completed tasks count improves by 26.81% and execution time improves by 10.66% and scheduling time improves by 69.94%. The scheduling time improves by 85.76% with an increasing number of iterations and constant numbers of tasks. Hence the presented GA-ANN scheduling technique outperformed the GA cost, GA EXE, and BB–BC COST scheduling approaches.

1. Introduction

In the current years, cloud computing has grown up an identifiable part of internet technologies. The real cloud service providers such as Amazon, Google, and Microsoft have high-performance datacenter across the globe. Effective utilization of datacenter resources is the key focus of different classes of users. The researchers are focusing on static, dynamic, and nature-inspired meta-heuristic approaches that improve the efficiency of the datacenter. Cloud computing supports the utility base computing paradigm which takes the benefit from storage, computing, and network resources at a datacenter. The shared pool of resources and rapid elasticity and scalability features fulfill the objective function. The objective function depends on time and fault performance metrics. Cloud computing provides an extensible infrastructure, platform to run the tasks. Consumer higher the resources using pay-as-you-go pricing models. The agreement of service availability between service providers and consumers assure service improvements and resource availability. The service

* Corresponding author.
E-mail address: pradeep_kec09@yahoo.com (P.S. Rawat).
[1] All authors have equally contributed to their part to update the article.

https://doi.org/10.1016/j.asoc.2020.106876
1568-4946/

level agreement may be violated when the resource disruption occurs per day or per month. Our primary concern is the efficient assignment of tasks on virtual machines using scalable simulation [1,2]. In the recent decade, cloud computing has grown-up as an indistinguishable part of present-day internet technology. The cloud computing paradigm follows the features of utility computing where IaaS, PaaS, and SaaS service delivery model are provided using the pay-as-you-go pricing scheme and service level agreement. The infrastructure resources (storage, computing, network), grouped as a single data center unit. The datacenter unit provides the set of virtual machines for the assignment of tasks independently. With the help of virtualization technology, scalable cloud infrastructure can be used effectively. It offers huge computing power, storage, and bandwidth to the virtual machines. The assignment of the tasks on the virtual machine is the major concern for systematic resource management. The static, dynamic, nature-inspired heuristic approaches are normally used to generate the optimal schedule. The multi-objective performance metrics are considered which are time-aware. The poor resource management results in huge operational costs which depend on the execution duration of the cloud resources. Therefore, task scheduling on virtual machines is considered. Recently investigators are concentrating on the use of nature-inspired and artificial intelligence-based scheduling approaches. The meta-heuristic techniques take more time than the deterministic algorithm which raises the execution overhead [3]. He challenging issues with these approaches are i.e. (1) increase execution overhead. (2). It cannot handle the dynamic features of a scalable cloud scenario. This work conveys these concerns with the use of the neural base feature classifier. The classification power of artificial neural networks benefited the scheduling concern in the cloud. The scheduling approach takes the benefits of the classification power of the intelligent computing paradigm. A neural model is designed to learn which virtual machine is appropriate for the incoming requests. Once the network gets trained to sue the data sets generated using GA, the input, output layers cannot extend. The recursive neural network with minimum error rate overcome the constraint. The number of layers and perceptron in each layer depends on the complexity of the cloud infrastructure.

The key contributions of our proposed time and fault aware GA-ANN based independent task scheduling technique summarized as follows. The presented GA-ANN approach reduces the average start time (ms), average finish time (ms), execution time (ms), and scheduling time (ms), and fault rate (tasks failure count) and completed tasks count. It also reduces the execution overhead and improves the operational cost. The ANN-based task scheduler (ATS) schedules incoming requests on an appropriate virtual machine [4]. The scheduling prediction of ANN depends on training data sets. The accuracy of ANN to predict the resources is better than other machine learning techniques [5]. Further bio-inspired based time aware scheduler used to generate huge data sets. The list of independent tasks select as input and schedule them on a scalable cloud scenario. The traditional approaches have low convergence of hyperparameters. The basic objective of the genetic aware solution for hyperparameter optimization use to consider the various neural network. The bio-inspired approach uses the fitness function for the selection of an optimal solution. The performance is evaluated using the comparison between GA-ANN and cost-efficient GA-Cost [6], GA-Exe [6], and BB–BC COST [6–8]. The execution overhead and cost are used as evaluation metrics. The execution time and fault aware GA-ANN independent task scheduler use the multi-layer neural model to select a virtual machine for the scheduling of submitted tasks. It also adapts the current status of a scalable cloud scenario which is equivalent to an input of a neural network. The trained neural model predicts the task allocation on virtual machines in a scalable cloud scenario. Finally, the presented time-aware and fault aware GA-ANN technique simulation is performed using cloudsim 3.0.

1.1. Organization of work

Section 2 covers the related work done in meta-heuristic algorithms and energy-aware scheduling in a scalable cloud scenario. Section 3 illustrates the motivation of the work. Section 4 illustrates the presented genetic trained ANN-based Task Scheduler Policy. Section 5 illustrates the simulation environment and the results of the developed model using real-time workload data sets and fabricated data sets. Finally, the article is concluded in Section 6.

2. Related works

Most recently resources have focused on independent tasks scheduling on a virtual machine using various optimization objectives. The optimization objectives included time, cost and energy consumption, scalability, and availability. This aims to reduce the execution overhead and resource cost. The static, dynamic, and meta-heuristic approaches are good enough to generate the best possible solution in a task execution overhead. The simple heuristic techniques suffer from generating local optimal solutions. The local optimal solution is obtained for energy [9–23]. In the previous work, resource management problems are solved using machine learning algorithms, ANN, reinforcement learning [24,25]. To find an optimal virtual machine configuration parameters are automated using reinforcement learning [26]. Kalra and Singh [3] developed a multi-objective bio-inspired technique for scheduling of tasks where ANN is used to achieve an objective function. Berral et al. [27] focused on the linear regression model to predict power utilization and time but there is an overhead cost of the tasks. It may increase the completion time of tasks. The authors focused on the benefits of the predictive model. Still, there is a requirement to do some heuristic or meta-heuristic which utilize the schedule prediction effectively. However, our primary focus is to propose a GA-ANN technique that focuses directly on execution overhead and resource utilization cost. Tripathy et al. [28] presented a search optimization using objective function makespan, energy consumption. There is a quick convergence in search optimization. Agarwal [29] developed an augmented neural network using heuristic scheduling. However, in our work GA is used to train the perceptron model which works in optimum ways. Once the perceptron model is trained with huge data sets, GA-ANN does not require meta-heuristic techniques to take the scheduling decisions.

Dam et al. [30] presented an evolution-based approach, which distributes the tasks on a shared pool of computing, storage, and network resources. The existing provisioning techniques incorporate first come first serve (FCFS), a local probing approach e.g. stochastic hill climbing (SHC), a genetic approach (GA). Dam et al. [30] emphasized only on limited performance measures without an artificial neural network classifier. Kousalya and Balasubramanie [31] studied an adaptive nature-inspired ant approach which is used in grid infrastructure. Tawfeek MA et al. [32] modeled the scheduling approach which depends on the nature-inspired iterative optimization method. The performance is measured using response time, makespan, and resource cost($). MadadyarAdeh and Bagherzadeh [33] studied the nature-inspired meta-heuristic technique i.e. ant colony optimization. Singh et al. [34] presented a taxonomy and review on task scheduling in a cloud computing environment using meta-heuristic approaches. An assessment using an evolution strategy for provisioning the

tasks on a scalable cloud environment. The author introduced terminology and qualified exploration. The efficient exploration of task scheduling on a virtual machine is based on nature-inspired techniques. The nature-inspired evolution approach is suitable for suggesting a better solution for scheduling SaaS modeler on virtual machines [35]. Yu and Buyya [36] developed a genetic approach that deals the resource assignment concern in a scalable cloud. The deadline and budget are used as quality constraint parameters. Sawant [37] demonstrated a bio-inspired virtual machine scheduling strategy that focuses on system resource distribution. Our fundamental objective distinct on time and cost tradeoff in a scalable IaaS cloud scenario. Researchers focused only on meta-heuristic techniques and their performance metrics. Our motivations include the convergence rate improvement of task allocation techniques. The convergence rate and optimization metrics improve using a bio-inspired technique. The bio-inspired analogy includes the SaaS modeler in high-performance computing. It covers a single perceptron, and multi perceptron layer neural network model. This is known as a neural network classifier. It has a wide application in performance improvement of the scalable cloud scenario. The prominent features of the ANN model are inherited in a high-performance computing paradigm. It covers Utility, Grid, Cloud, and distributed computing environment. The high-performance computing invents put the demand of the artificial neural network. The performance of the bio-inspired technique improves using ANN (Artificial Neural Network). The systematic assignment of tasks on the virtual machine is a major concern in scalable cloud infrastructure. This section covers static, dynamic, and meta-heuristic approaches in a scalable cloud scenario. Researchers have a center of attention on the different quality of service parameters i.e. time, fault, completed tasks count. Liu et al. [10] studied effective ant colony based optimization techniques. The authors incorporated the performance evaluation parameters i.e. agreement of service availability management, minimal overhead at a datacenter in distinct time zones, and energy reduction at the data center. Quality of service and resource employment enhanced by computing power, storage, and network parameters. Lu and Gu [38] illustrated an adaptive load aware resource allocation technique based on a nature-inspired methodology. Authors grasp that the flexible resource assignment in a scalable cloud scenario attains the goal. The performance metrics unutilized time and total completion time of the tasks taken into account for finer employment of cloud resources at the infrastructure and platform level. The meta-heuristic approach assigns the requests on the virtual machines. Sun et al. [39] studied the prominent features and the attainment method based on ant colony optimization. Mathiyalagan et al. [40] outlined that the grid scheduling concern can be successfully solved using the nature-inspired approach. Liu and Wang [41] developed a flexible ant colony approach to run the scientific application in grid infrastructure. Experimental evaluation reveals the model efficiency at virtual machine level scheduling and load sharing among the shared infrastructure and platform resources. The authors evaluated the performance with existing approaches. The ACO acquires a new state transition rule for obtaining appropriate scheduling outcomes. Chiang et al. [42] presented enhanced performance parameters of scheduling. The accomplishment of the quality of service is evaluated against the genetic scheduling method. The author concentrated on workflow depend on virtual machine level scheduling. Pop et al. [43] developed a distributed scheduling approach for dependent task scheduling on virtual machines. The developed technique accompanies the characteristic of the biology base genetic approach. Zheng et al. [44] demonstrated an optimized scheduling approach to acquire an optimal solution in task scheduling. The author explores the feasibility of the placement of the virtual machine. Babu and Krishna [45] developed honey bee behavior base techniques to balance the priority of tasks on a virtual machine. The performance is evaluated using the average execution time of the task and the waiting time of the tasks. Jana et al. [46] presented a modified particle swarm intelligence approach in a cloud computing environment. The authors focused on performance metrics average scheduling length and the ratio of successful execution. Kumar and Patel [47] presented a hybrid approach using a neural network with particle swarm optimization. The performance is measured using makespan and cost. Bacanin et al. [48] illustrated the hybridized monarch butterfly optimization approach. The authors focused on the accuracy improvement of the neural network design. Sreenu and Sreelatha [49] presented an approach of tasks scheduling on virtual machines i.e. whale scheduler which follows the features of the whale optimization algorithm. The makespan and cost are considered as performance metrics. Shirani and Safi-Esfahani [50] presented an improved dragonfly approach for the scheduling of tasks on a virtual machine. The performance is measured using execution time, response time, and reduction of service-level agreement violation. P. S. Rawat et al. [6] proposed a metaheuristic task scheduling algorithm for the cloud using the Big-Bang Big-Crunch algorithm the algorithm is inspired by astrology which assumes that the universe is evolved from a big bang and will converge to a single point. The algorithm is inspired by the Genetic algorithm only the change is an extra phase in which the worst solution is deleted reducing the search space of the problem, with this in each step the problem size reduces resulting in the single optimal solution in the last iteration. The solution can be found in less time as compared to a genetic algorithm with all the advantages of a genetic algorithm.

Our motivation comes from the convergence rate improvement of task scheduling techniques. The classification power of the perceptron model improves the performance using large data sets generated using a genetic approach. The high-performance computing paradigm generates better results using artificial neural networks. The approaches discussed above can be compare to the bases of optimization that can be achieved using those algorithms. Based on the existing survey static and dynamic algorithms provide the least optimization as compared to nature-inspired metaheuristic algorithms in a general scenario without any specific objective like cost, network performance, or power. In the category of metaheuristic algorithm genetic algorithm, Big-Bang Big Crunch [6] or a modified genetic algorithm is one of the most advanced algorithms which aim to provide global optimal and optimal performance as compared to many algorithms like ACO [7], PSO [46,51], honey bee [45], butterfly [48], whale algorithm [49], dragonfly [50] and many more. So the focus of our work is to further improve the performance of existing GA using artificial neural networks to overcome the drawback of existing algorithms in terms of performance and optimization that can be evaluated in terms of execution time and scheduling time to find the global optimal results.

3. Motivation for work

The performance gain is the focus of the meta-heuristic, static, and dynamic task scheduler policies. The quality of service is measured using a scalable cloud. In the reviewed articles, authors focused on static, dynamic, and nature-inspired and astrology based techniques. The state of art techniques provides the direction to develop a classification features based model based which is trained using data sets generated using the bio-inspired technique. The reviewed task scheduling technique provides an avenue for performance improvement. It helps in developing a task provisioning approach that improves the quality of service of the task scheduling challenges. The Genetic approach provides

Table 1
Performance metrics.

SL. No.	Meta-Heuristic Technique/Static/Dynamic	Performance metrics for task scheduler policy	Data sets 1	Data sets 2 [52,53]
1	BB–BC COST [6,8]			Real workload (Real workload file (The San Diego Supercomputer Center (SDSC) Blue Horizon logs))
2	GA cost [6,8]	1. Average start time (ms) 2. Average finish time (ms) 3. Fault rate (failure count)	Fabricated data sets	Real workload (Real Workload File (The San Diego Supercomputer Center (SDSC) Blue Horizon logs))
3	GA-EXE [6,8]	4. Execution time (ms) 5. Tasks completed (success count) 6. Scheduling time (ms)		Real workload (Real Workload File (The San Diego Supercomputer Center (SDSC) Blue Horizon logs))
4	GA-ANN (Proposed)			Real workload (Real Workload File (The San Diego Supercomputer Center (SDSC) Blue Horizon logs))

huge data sets for the training of the neural network. Hence Fig. 2 exhibits the neural network model for real-time scheduling of tasks on virtual machines. The metaheuristic techniques with artificial neural network (ANN) focus on performance measurement parameters exhibited in Table 1.

It includes time aware of performance metrics. The performance metrics claim the global optimality of the proposed model. The optimal solution measures using the parameters with input layer data sets from real workload (SDSC-BLUE-2000-4.swf) logs [52,53]. The original workload data sets contain the logs of the San Diego Supercomputer Center (SDSC) Blue Horizon)), and synthetic data sets are also used for modeling and simulation in scalable cloud aura.

4. Proposed genetic trained ANN-based task scheduler policy

This section covers a neural network task scheduler that is trained using training data sets. The training data set is prepared using a bio-inspired genetic approach. The genetic trained neural network acts as a classifier that predicts an appropriate virtual machine for the scheduling of tasks. The scheduling decision takes using the perceptron model which is trained using huge data sets generated by using a modified genetic algorithm. It is a bio-inspired, human brain based complex, nonlinear parallel system. High-performance computing puts the demand for the ANN model along with nature-inspired techniques. It assigns the task on virtual machines with adjustment of the bias parameters of a node. The ANN-based task assignment on virtual machines in a scalable cloud is an emulation of an artificial neural task scheduler.

4.1. Artificial neural network-based task scheduler

This subsection presents a classification features based artificial neural network which is trained using a data set generated using a biology base technique. Most of the researchers use artificial intelligence techniques such as genetic algorithm and ant colony and particle swarm optimization to solve the task scheduling problem. The solution may converge at a local or global optimal point. Still, there is an opportunity to improve the indistinguishable objectives. It includes metrics time (average finish time, makespan, simulation time). Nature-inspired population-based evolution techniques serve a local optimal and optimal global solution due to the convergence rate. The performance metrics improve further using a meta-heuristic technique with artificial neural networks. Therefore, the presented perceptron model provides an optimized task scheduling results. The proposed technique in which the ANN model is trained using a genetic approach. The artificial neural network model is formulated of multiple-input, hidden, and output nodes similar to the biological neurons of the human brains. The training process covers the error correction mechanism using weight adjustment. The ATS scheduler trained using 80% of total data sets. The trained ATS is used for the real-time scheduling of tasks on virtual machines. The proposed model aims to improve, makespan, average finish time, average simulation time, tasks failed and tasks completed respectively.

The artificial neural network provides better results than simple cost and time-aware genetic technique and costs aware BB–BC techniques as shown in the experiment section [8]. The fitness selection is performed for the development of the optimal schedule. The trained neural network is used for the prediction of output. The predicted output generates at the output layer at the end of the learning or training process which provides accuracy and correctness. Researches primarily focused on solving the task scheduling challenges in a scalable cloud scenario. Most of them have used a genetic algorithm, ant colony [41], and Big-Bang Big-Crunch cost-aware [6,8] methodology to find an optimal assignment of the virtual machines [7,37]. Our objective is to enhance the quality of service of the neural network with minimum error i.e. the predicted value should be closer to the desired output.

The proposed ANN-based task scheduler uses the vector of task and vector of virtual machines as an input to the scheduler with basic parameters to the model. The time-aware neural network model is divided into the following steps:

1. Initialization of ANN Scheduler
2. Training data set preparation using a genetic approach
3. Neural model designing
4. Neural model training
5. Error Backpropagation and correction
6. Task scheduling on virtual machines

1. Initialization of ANN Scheduler

In this phase, a digital neural network is initialized. The neural network includes an input layer (of two neurons), two hidden layers (three nodes in each hidden layer), and an output layer (of one neuron). The neural network learns from huge data sets generated using a genetic approach in a scalable cloud environment. The hyperparameters weights and bias values alter which produces the optimal neural model. In the presented model takes input (X) i.e. the set of tasks and set of virtual machines list in a scalable cloud scenario and produces the scheduling decision or virtual machine identity as output (Y). The ANN scheduler design is described in Section 3.

2. Training data set preparation using a genetic approach

This phase focuses on the learning process of digital neural networks indicated as the training process. The supervised and unsupervised training process are used in a scalable cloud environment. In this manuscript, the supervised training process is

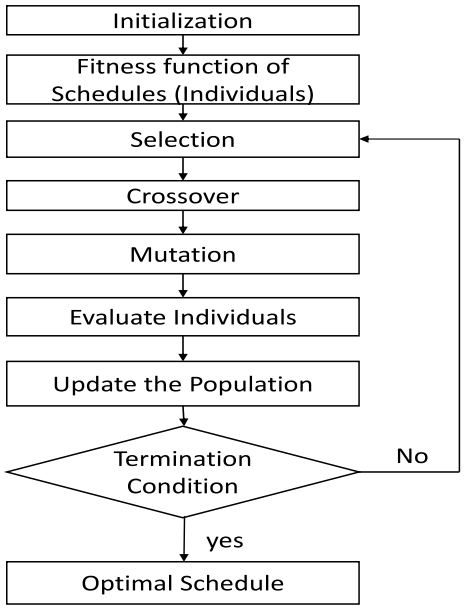

Fig. 1. Flow diagram of the Genetic approach.

primarily focused. In the supervised training process, the input features along with desired outputs are fed to the ANN-based task scheduler which enhances the accuracy of the neural network model. The genetic algorithm provides huge data sets for supervised training or learning of the perceptron model. The training process inherits the configuration parameters e.g. fitness function of the schedules mapped on virtual machines. The training data sets prepared using a genetic algorithm provides the input to the neural model. The proposed genetic approach is defined by the fitness function shown in Eq. (1). The fitness function derives the output of the proposed genetic approach where the fitness function is the function of network cost, execution time, and failure rate in a virtual machine.

In the domain of artificial intelligence, a bio-inspired genetic algorithm (GA) is a meta-heuristic technique that pretends the biological evolution procedure. This bio-inspired technique generates useful solutions to optimization and search problems. Genetic algorithms belong to the class of evolutionary algorithms [54] which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover. The individual solution of the genetic approach is encoded in the form of a virtual machine-id. A collection of schedules of the tasks on virtual machine form string i.e. known as a chromosome. Initially, schedules are created randomly, which exhibits different chromosomes in a solution space [55]. A fitness function value of the individual schedule represents the degree of fairness of the schedule. Based on the essence of survival of the fittest [56], some of the schedules are selected and each schedule is assigned several copies that go into the mating pool. Crossover (multi-point) and mutation (swap base) operators are applied to these chromosomes or schedules. The selection, crossover, and mutation operations continue until a termination condition is satisfied [57]. Fig. 1 exhibits the flow diagram of the genetic approach. Prediction scheduling performance is based on a genetic approach and a neural network. Subsequent basic steps are used in the neural network training approach. For this section total number of tasks is divided into two parts of A-20% and B-80%, out of this A part 80% of this is given to the proposed GA for generating training data set and 20% of A is for testing the neural network.

$$F(x) = \alpha^* Network\_cost + \beta^* Total\_Execution\_time + \gamma^* \frac{1}{Failure\_rate_i} \quad (1)$$

where $\alpha + \beta + \gamma = 1$

$$Total\_Execution\_time = \sum_{task\_i=1}^{n} \frac{Task\_Length_i}{MIPS_j} \quad (2)$$

$$Network\_cost_i = Network\_delay_i \quad (3)$$

$$Failure\_rate_i = \frac{task\_failed\_count_i}{\Delta T} \quad (4)$$

The failure rate is defined as the number of tasks failed over a period of time in a virtual machine, where failure may occur in a VM at any point due to may reason, software or hardware glitch, overloading, storage, or network failure. The failure is simulated in the model using the Poisson distribution for random generation

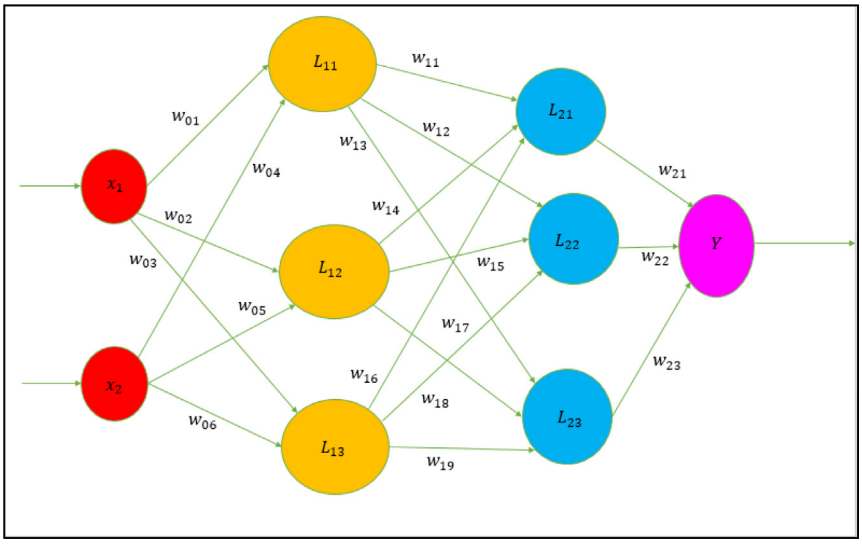

Fig. 2. Three layer neural network architecture.

of fault in VM. Task failed count concern to the tasks failed over a period of time Δ T.

The training data set preparation bio-inspired GA (Genetic algorithm) is divided into 3 phases as follows

1. Initialization
2. Selection
3. Crossover and Mutation

2.1 Initialization

In the initialization step, a huge number of chromosomes (schedules) are randomly generated to form an initial set of schedules or set of chromosomes i.e. known as population. The population size is the set of tasks to be scheduled. Total populations contain several possible solutions equalize population size which provides accuracy in the training process of neural networks. Traditionally, the population is generated randomly which allows the entire range of possible schedules which include best, average, and worst-case schedules of tasks on virtual machines with the worst fitness value.

2.2 Selection

Individuals or two best chromosomes (schedules) are chosen to have maximum fitness function values. The fitness function values of chromosomes or schedules are measured using equation 1. Certain selection techniques rate the objective function value of each solution and select the optimal solutions using a superior way. Other methods rate only a random sample of the population, former approaches increase the execution overhead [58]. The individual schedule selects using fitness function evaluation.

2.3 Crossover and Mutation

Once the fittest schedule or chromosomes selection process completes, the second-generation population of tasks and virtual machine schedules produce using a genetic operator's crossover and mutation [59]. Each new individual to be produced, using a pair of "parent" solutions. A multi-point crossover is performed which generates a new schedule of tasks on virtual machines. A "child" solution is produced using the above technique of crossover, a new solution is created which shares the characteristics of its parent nodes using swap base mutation. Finally the next generation chromosomes or population of a chromosome which is different from the initial generation. Generally, the average fitness will be increased via this method, since only the optimal schedule from the first generation are selected for breeding, along with a small proportion of less fit solution. To find optimal schedule selection, the crossover and mutation phases are repeated using several evolutions. The outcome of this phase is a schedule of a set of tasks which is considered as a training data set. The training data set includes task size, utilization of each virtual machine, and the identity of the finally chosen virtual machine.

3. Neural model designing

The neural model design phase focuses on multi-layer neural network applications in a scalable cloud computing environment. The supervised learning based neural network model is influenced by the functioning of the human brain and how it processes the input features. The human brain is made of billions of neurons which are again connected forming a network of neurons. Electric signals that travel through a designed neural network. Similarly, an artificial neural network based task scheduler is a digital model of the human brain. The learning methodology is used to build a neural network model design layout. The following key steps are followed. Initialize parameters/define hyperparameters and loop for several iterations which include a. Forward propagation, b. Compute cost function which depends on scheduling parameters in a scalable cloud environment. c. Backward propagation, d. Update parameters (using parameters, and grads from backpropagation), 4. Use trained parameters to predict the output. Each node of the neural model uses the summing function and activation function to produce an output signal. Fig. 2 illustrates the neural network architecture with an input layer (of two nodes), two hidden layers (each hidden layer having three-neurons), and an output layer (of one neuron). The complexity of the brain impressive system depends on the number of datacenters, virtual machines, and user requests. The number of layers and the number of neurons in each layer depends on the cloud infrastructure configuration. The neural network model can learn the complex relationships of consequences in the external scenario by adjusting the weights and bias values. In this manuscript scalable cloud environment use the genetic algorithm from which it takes input (x1) is a task list and (x2) virtual machine list and scheduling results as output (Y) which predicts the virtual machine patterns. The neural network shown in Fig. 2 is a sample neural network where the proposed algorithm also uses the same model with input layer 2 hidden layer and output layers but the network with extending itself to a multilayered deep neural network depending on the configuration and resources in the cloud environment. The weights shown in the figure are auto-adjusted during the training phase which decides the neural connections using the activation function and weight correction mechanism using Error backpropagation and correction phase.

Table 2 exhibits the criterion of the neural network model. The criterion includes synaptic weights and bias values associated with each layer. The weights and bias values are unknown parameters that are adjusted using a differentiation between target output and the desired output. The learning mechanism defines the connection strength between neurons of the neural network model. The mathematical representation of the neural network is shown in the following equations.

$$z_i^{[l]} = w^{[l]} a_i^{[l-1]} + b^{[l]} \tag{5}$$

$$a_i^{[l]} = g^{[l]}(z^{[l]}) \tag{6}$$

where $i = 1, 2, 3$

where Eqs. (5), (6) measures the parameters value at individual neurons. $z_i^{[l]}$ represents the output of the node i at layer l before activation function and $a_i^{[l]}$ illustrates the output of the neuron i at layer l after using the activation function. In the case of our proposed model, the value of $l = 0,\ 1,\ 2,\ 3$, where $l = 0$ represents the input layer,. In our research work, the Leaky ReLU activation function is used at two hidden layers $l = 1,\ 2$ and the sigmoid function is used at the output layer $l = 3$.

Eq. (7) exhibits the output using an activation function on hidden layer 1 of the neural network shown in Fig. 2.

$$a_i^{[1]} = g^{[1]}(z^{[1]}) \tag{7}$$

where $i = 1, 2, 3$ Eqs. (8), (9), (10) represents the activation function value at neuron i. The equations measure the value of the activation parameter for all the layers $l = 0,\ 1,\ 2,\ 3$.

For hidden layer 1

$$a_{1,2,3}^{1} = Leaky\ \mathrm{ReLU}(z_{1,2,3}^{[1]}) \tag{8}$$

$$a_{1,2,3}^{2} = Leaky\ \mathrm{ReLU}(z_{1,2,3}^{2}) \tag{9}$$

At the output layer i.e. l= 3 predicted output is measured using Eq. (10).

$$a_1^3 = Sigmoid(z_1^3) \tag{10}$$

where $L_{21}, L_{22}, L_{23}, L_{12}, L_{13}$ measures using equation numbers (11), (12), and (13) respectively. The variable w_1, w_2, w_3 denotes the synaptic weights, b_3 denotes the bias value at the summing

Table 2
Parameters of the artificial neural network model.

SL. NO.	Variable type	Neural network parameters (Unknown variables)	Layers size	Known variables
1	Weight	w_{01}, w_{02} w_{03} w_{04} w_{05} w_{06} w_{11} w_{12} w_{13} w_{14} w_{15} w_{16} w_{17} w_{18} w_{19} w_{21} w_{22} w_{23}	$n_x = 2$ $n_{h1} = 3, n_{h2} = 3$ $n_y = 1$	INPUT: x_1, x_2
2	Bias	b_1, b_2, b_3		TARGET OUTPUT: Y

point. The variate L_{11}, L_{12}, L_{13} measures the output of the neurons at the hidden layer 1 of the neural model and x_1, x_2 denotes the input patterns of the neural network.

$$L_{11} = z_1^1 = Leaky\ \text{ReLU}(w_{01}x_1 + w_{04}x_2) + b^1 \tag{11}$$

$$L_{12} = z_2^1 = Leaky\ \text{ReLU}(w_{02}x_1 + w_{05}x_2) + b^1 \tag{12}$$

$$L_{13} = z_3^1 = Leaky\ \text{ReLU}(w_{03}x_1 + w_{06}x_2) + b^1 \tag{13}$$

Similarly, Eqs. (14), (15), (16), L_{21}, L_{22}, L_{23} measures the output of the neurons at hidden layer 2 of the neural network.

$$L_{21} = z_1^2 = Leaky\ \text{ReLU}(w_{11}{}^*L_{11} + w_{14}{}^*L_{12} + w_{16}{}^*L_{13}) + b^2 \tag{14}$$

$$L_{22} = z_2^2 = Leaky\ \text{ReLU}(w_{12}{}^*L_{11} + w_{15}{}^*L_{12} + w_{17}{}^*L_{13}) + b^2 \tag{15}$$

$$L_{23} = z_3^2 = Leaky\ \text{ReLU}(w_{13}{}^*L_{11} + w_{18}{}^*L_{12} + w_{19}{}^*L_{13}) + b^2 \tag{16}$$

The Eq. (17) measures the output of the neurons at layer 3 of the neural network. Fig. 2 exhibits the third layer of the neural network which acts as an output layer of the proposed neural model.

$$Y = z_1^3 = sigmoid(w_{21}{}^*L_{21} + w_{22}{}^*L_{22} + w_{23}{}^*L_{23}) + b^3 \tag{17}$$

4. Neural Model Training

Phase 3 provides a digital neural network model that is trained using data sets generated using the proposed Genetic algorithm. The trained digital neural network is validated using 20% of the generated data sets using a genetic algorithm.

The neural network training includes:

1. Give an initial range for the edge weights and biases.
2. Generate the data sets using a genetic algorithm and train the neural network model
3. Once training of a network is completed, its testing is performed using 20% of the data set. This is an essential step that measures the efficiency of the neural network model. If desired results are not obtained, the network must be redesigned and trained again and validate the redesigned neural network. So it is a trial and error process. The edge weight and bias parameters adjust if there is a difference between the predicted output and the desired output.
4. Once the Neural based model is validated, and the range of errors is established. The network runs to predict the values. Hence increasing the nodes in a layer and increasing the layers allows the network to cope up with the complex scheduling problems. It also increases the computational space required. The activation function of the nodes in the hidden layers measures using the Leaky ReLU Activation function and the activation function of the nodes in the output layer is measured using the nonlinear activation function (sigmoid). The mathematical expressions are shown in Eqs. (18), (19) respectively.

Leaky Relue *Activation* function $=$

$$f(x) = \{\ .01^*\ x\ if\ x\ <\ 0\ ,\ \ x\ \ if\ \ x\ \geq 0\ \} \tag{18}$$

$$f(x) = \frac{1}{1 + e^{-x}}\ \ \forall\ x \in (0, 1) \tag{19}$$

where $f(x)$ represents the activation function applied at the hidden layer and output layer respectively. The quality of service measures using optimization criteria makespan and cost pay for resource usages. Eq. (18) presents the linear activation function which includes the Simple ReLU and Leaky ReLU function for transforming the summing values at the nodes of the hidden layers. The Neural network-based model trains using the requests vector which is the subset of the prepared data sets using a genetic algorithm. The proposed GA-ANN-based resource provisioning model includes two inputs, one output, and two hidden layers having three nodes at the individual hidden layer.

5. Error Backpropagation and correction

In this phase, the general backpropagation learning method focuses on the difference between the desired output and the predicted output using a trained neural network model. The fundamental objective of error backpropagation is that each time the network forecasts an output. We equate the forecasted output i.e. predicted output to the desired output and calculates error. The bias values and weights of communication links are altered in the direction of lower error distance. The error correction learning mechanism is followed which improves the quality of service. The computation is performed at each node of the hidden layer using the activation function. The basic idea of the backpropagation is the comparison of the predicted output and desired output as shown in Eq. (20). The error signal activates a control mechanism for corrective adjustment of synaptic weights.

$$e_k(n) = d_k(n) - y_k(n) \tag{20}$$

where the parameter $d_k(n)$ denotes the desired output, $y_k(n)$ denotes the predicted output, and $e_k(n)$ is the error distance and k represents the kth output neuron. Where n denotes the time step.

6. Task scheduling on virtual machine

In this phase, the proposed ANN-based tasks scheduler schedules the task on a virtual machine. Once the network is trained then there is no need for a bio-inspired meta-heuristic technique.

4.2. Algorithm of the proposed ATS (ANN-task scheduler)

Fig. 3 illustrates the flow diagram of the presented GA-ANN tasks scheduling approach. The flow diagram shows the flow of steps of the training data set preparation using a genetic approach and optimal scheduling using a trained neural network.

4.2.1. Pseudo code of neural network parameters updates

The Section 4.2.1 exhibits the pseudo-code for the hyperparameters learning rate, the number of layers, and the number of neurons in a given layer and update of the weights of neurons connection and bias values of neurons of the given layers in a neural network model. The trained neural network model performs real-time scheduling of tasks on virtual machines using an optimal neural network model. The connection strength and bias values are updated using the pseudo-code. The process is repeated until the predicated output is approximately equal to the desired output of the neural network model.

Table 3
Configuration parameters of datacenter.

Datacenter ID	Memory (GB)	RAM (GB)	Processor (PE)	Hosts	CORE
Datacenter- D1	1 000 000	128	6	3	4
Datacenter- D2	1 000 000	128	6	3	4
Datacenter- D3	1 000 000	128	6	3	4
Datacenter- D4	1 000 000	128	6	3	4
Datacenter- D5	1 000 000	128	6	3	4

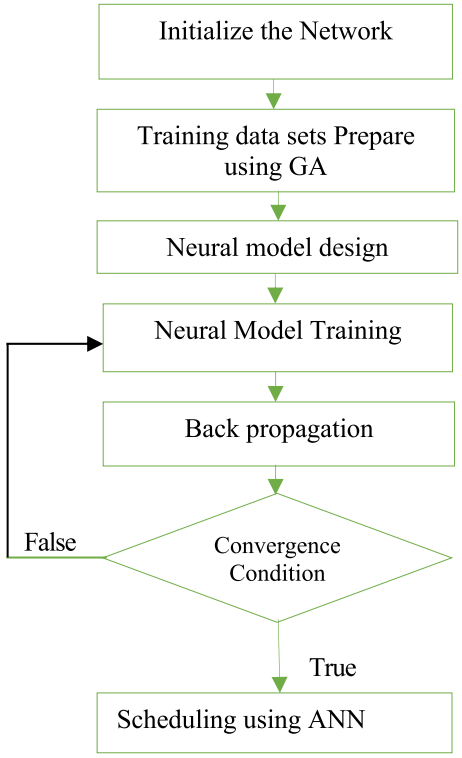

Fig. 3. Proposed GA-ANN (Genetic-Artificial Neural Network) flow diagram.

Algorithm: Evaluation

1. Input: set of observations (x, y)
2. for i=1:n
3. Relue activation function to apply on Hidden Layers
4. A random distribution of the weight and biases
5. Repeat:
6. For each (xi, yi) do
7. compute h(xi)
8. if(predicted output != desired output) then
9. update the parameters
10. . $w_d = w_d - \alpha * \frac{\partial L(y_i, h(x_i))}{\partial w_d}$
11. . $b_d = b_d - \alpha * \frac{\partial L(y_i, h(x_i))}{\partial b_d}$
13. end if
14. end for
15. Ensure: stopping criteria is satisfied.

Output: total cost found

5. Results and discussions

This work focuses on performance enhancement using ANN-based virtual machine scheduling in a scalable cloud scenario. The bio-inspired technique brings about output for the artificial neural network. The optimal results are generated using the bio-inspired technique. The ANN improves the performance of the optimal scheduling problem using makespan (ms) as a performance metric. Results are generated using different approaches with ANN for quality of service enhancement. Data sets are used for the training and validation, which gives a clue about the observations. It provides optimal scheduling with a minimum error rate. Table 3 reveals the configuration of the datacenter and Table 4, exhibits the configuration of three types of VM's taken into consideration for simulation. Table 6 illustrates the organization of the link between datacenter and cloud controller with its bandwidth and latency. For performance comparison of proposed algorithm existing metaheuristic algorithms are taken into consideration which are GA-EXE, GA-cost, BB–BC COST. These are algorithm inspired from a genetic algorithm which is proven to find the global best solution in any scenario where other existing algorithms like ACO, PSO, and many more discoursed in Section 2 fails to find a global solution and stuck in a locally optimal solution.

Table 4
Virtual machine configuration.

Virtual machine type	Storage (MB)	MIPS per processor	RAM (MB)	PE
VM 1	1000	500	1024	1
VM 2	1000	400	512	1
VM 3	1000	600	2048	1

Table 5.1 illustrates the simulation parameters of the neural bio-inspired approach and hyperparameters of the neural network architecture. The genetic approach acts as a supervisor of the neural network model which solves hyperparameters optimization challenges. The chromosomes are the hyperparameter values of the multi-layer perceptron model. The hyperparameters form the chromosomes. The individual chromosomes contain multiple genes that depend on hyperparameters. The hyperparameters choice depends on gene values. The implementation of the GA hyperparameter optimization is experimented using population sizes i.e. 100. The hyperparameter values are listed in Table 5.1 which provides the combination of genes generates i.e. unique chromosomes generates for the neural network. The performance of the human brain impressed neural networks depends on the selection of numerous of hyperparameters. Once we select the hyperparameters efficient hyperparameter optimization performs using a genetic approach. The hyperparameters control the values of weight and bias values. Table 5.2 illustrates the simulation parameters of cost-aware BB–BC, cost-aware genetic, and execution time aware genetic approach. We reach the neural network architecture shown in Fig. 2 using the improvement of the fitness function shown in Eq. (1) using 100 iterations.

5.1. Results using fabricated data sets

5.1.1. Simulation results comparisons using performance metrics average start time

This subsection illustrates the variations of the average start time with several cloud task scenarios. The user requests or cloud tasks vary from one thousand to ten thousand. The performance metric average start time is used for the comparative

Table 5.1
Simulation parameters of GA-ANN & Hyperparameter of the neural network.

Hyperparameters	Values
Number of iterations	100,150, 200, 250, 300, 350, 400, 450, 500
Layers	Input layer:1, Hidden layer: 2 Output layer:1
Learning rate	0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9
Mutation rate	0.15
Population size	100, 150, 200, 250, 300, 350, 400, 450,500
Evolution	100
Number of neurons input layer	2
Number of neurons output layer	1
Number of hidden layers	2
Number of hidden unit	6
Activation function (hidden layers)	*Leaky* Relue *Activation* function = f(x) = { $\alpha * x$ if $x \leq 0$, x if $x > 0$ } $\forall\ \alpha = 0.01$
Activation function (output layer)	$f(x) = \frac{1}{1+e^{-x}}$ $\forall\ x \in (0, 1)$

Table 5.2
Simulation parameters of GA-Exe, GA-Cost, and BB–BC COST.

Parameters	GA-EXE	GA-Cost	BB–BC COST
Number of iterations	100	100	100
Mutation rate	0.15	0.15	0.15
Population size	100	100	100
Evolution	100/150/200	100/150/200	100/150/200
Fitness function	Least execution time	Least executional cost	Least executional cost

Table 6
Network parameter.

Datacenter	Bandwidth (Gbps)	Latency (ms)
D1	10	10
D2	10	8
D3	10	6
D4	10	10
D5	10	8
D6	10	6

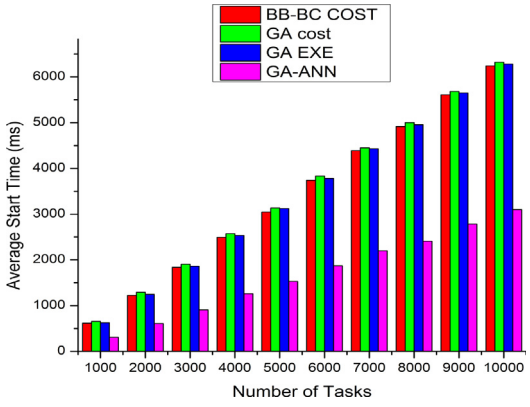

Fig. 4(a). Average start time (ms) vs. number of tasks with 5 VM's.

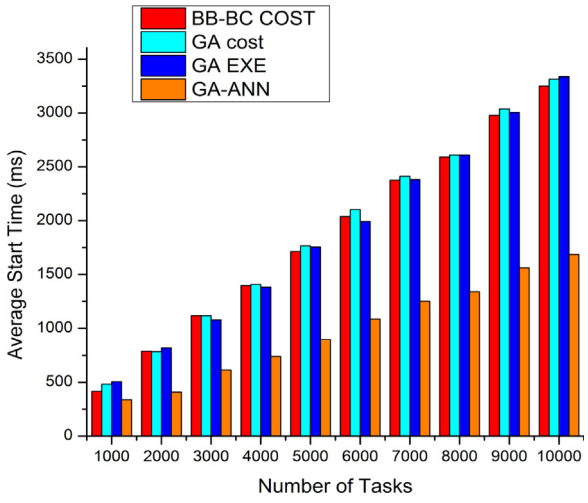

Fig. 4(b). Average start time (ms) vs. number of tasks with 10 VM's.

study of the task provisioning techniques. The resource provisioning technique GA-ANN outperforms the cost-aware and time-aware bio-inspired genetic approach using the performance metric average start time (ms).

Figs. 4(a) and (b) show the simulation results of the presented GA-ANN technique for the average start time with varying virtual machine count. Fig. 4 compares the variations of the average start time on increasing the user tasks with 5 VM's and 10 VM's correspondingly. The tasks or user requests are mapped from different geographical areas across the globe. Results are compared against bio-inspired time aware and cost-aware genetic approaches.

5.1.2. Simulation results comparisons using performance metrics average finish time

This Section 5.1.2 illustrates the variations of the average finish time with several tasks. The submitted tasks vary from one thousand to ten thousand. The performance metric average finish time is used for the comparative study of the resource provisioning techniques. The resource provisioning technique GA-ANN provides optimal results than a genetic approach using the performance metric average finish time (ms).

Figs. 5(a) and (b) demonstrate the simulation results of the GA-ANN provisioning technique using average finish time with varying virtual machine count. Fig. 5 compares the variations of the average finish time on increasing the number of tasks with 5 VM's and 10 VM's correspondingly. The tasks or user requests are mapped from different geographical areas across the globe.

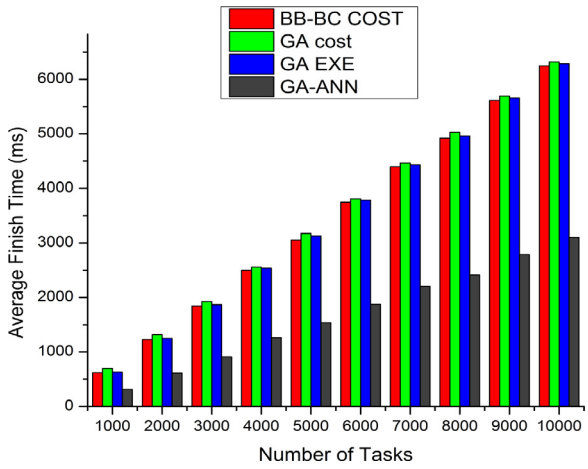

Fig. 5(a). Average finish time (ms) vs. number of tasks with 5 VM's.

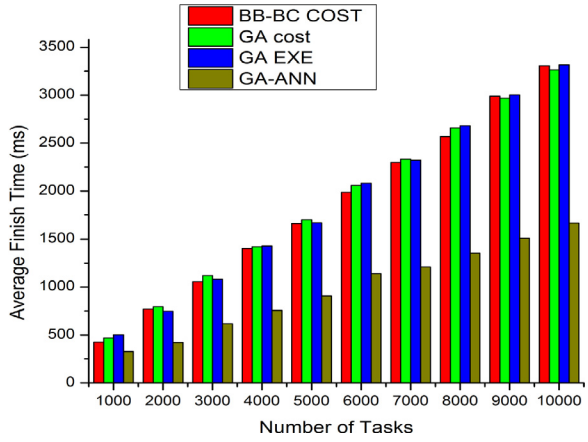

Fig. 5(b). Average finish time (ms) vs. number of tasks with 10 VM's.

5.1.3. Simulation results comparisons using performance metrics execution time

The Section 5.1.3 exhibits the variations of execution time with several tasks. The submitted tasks vary from one thousand to ten thousand. The performance metric execution time is used for the comparative study of the resource provisioning techniques. The resource provisioning technique GA-ANN provides optimal results than a genetic approach using the performance metric execution time (ms).

Figs. 6(a) and (b) reveal the simulation results of the presented resource provisioning technique GA-ANN for the execution time with varying VM count. Fig. 6 compares the variations of the execution time on increasing the user requests (tasks) with 5 VM's and 10 VM's correspondingly. The tasks or user requests are mapped from different geographical areas across the globe.

5.1.4. Simulation results comparisons using performance metric completed tasks

This subsection exhibits the variations of completed tasks with several tasks assigned on virtual machines. The submitted tasks vary from one thousand to ten thousand. The performance metric tasks completed count is used for the comparative study of the resource provisioning techniques. The resource provisioning technique GA-ANN provides optimal results than time aware and costs aware genetic approaches using the performance metric completed tasks count.

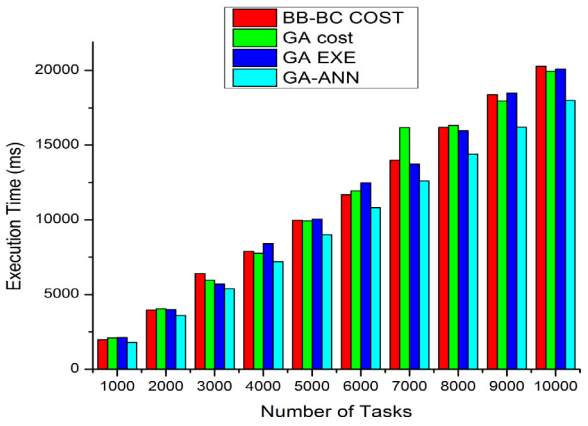

Fig. 6(a). Execution time (ms) vs. number of tasks with 5 VM's.

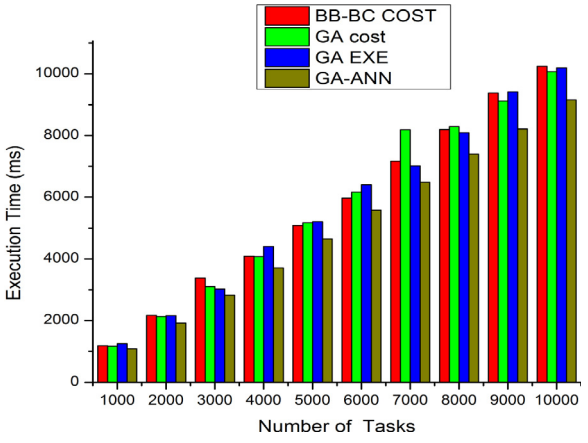

Fig. 6(b). Execution time (ms) vs. number of tasks with 10 VM's.

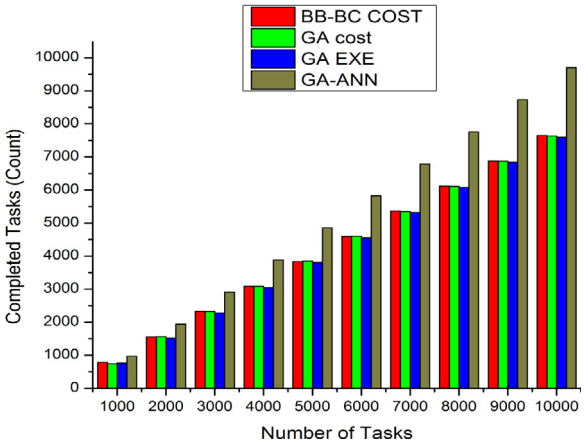

Fig. 7(a). Tasks completed (count) vs. number of tasks with 5 VM's.

Figs. 7(a) and (b) reveal the simulation results of the presented resource provisioning techniques for the completed tasks with varying VM count. Fig. 7 compares the variations of the completed tasks on increasing the user requests (tasks) with 5 VM's and 10 VM's correspondingly. The tasks or user requests are mapped from different geographical areas across the globe.

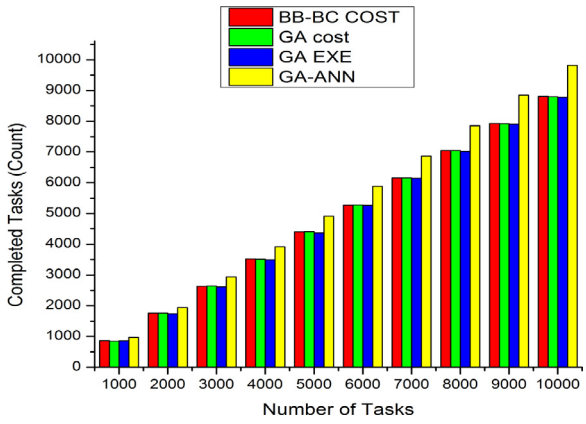

Fig. 7(b). Tasks completed (count) vs. number of tasks with 10 VM's.

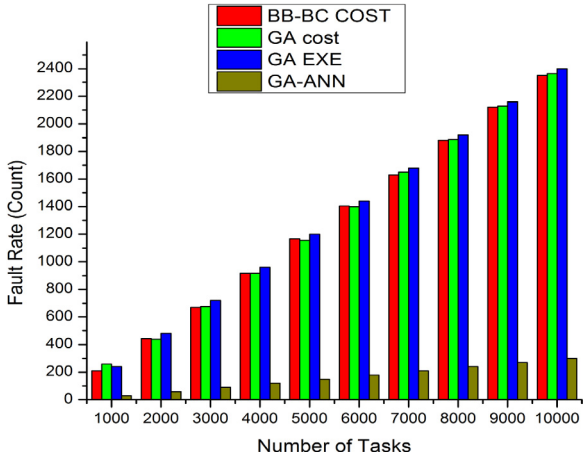

Fig. 8(a). Fault rate (count) vs. number of tasks with 5 VM's.

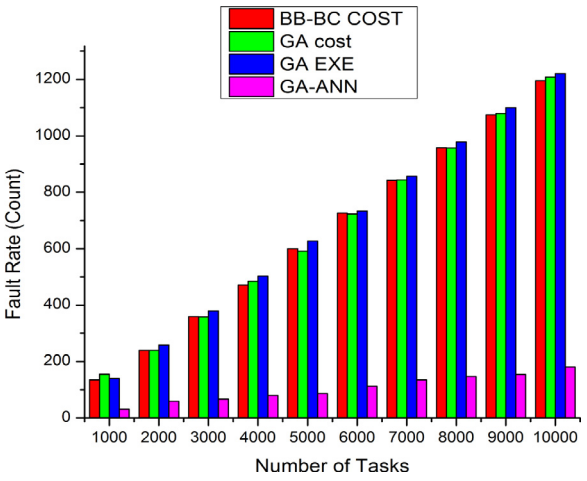

Fig. 8(b). Fault rate(count) vs. number of tasks with 5 VM's.

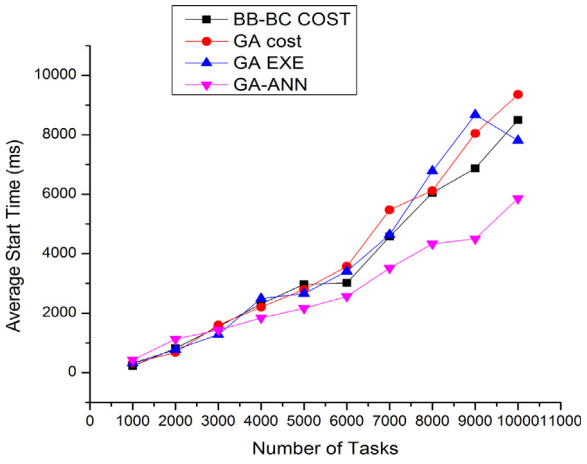

Fig. 9(a). Average start time (ms) vs. number of tasks with 5 VM's.

5.1.5. Simulation results comparisons using performance metric fault rate

This subsection describes the variations of the fault rate with various user request scenarios. The user requests or number of tasks vary from one thousand to ten thousand. The performance metric fault is used for the comparative study of the resource provisioning techniques. The resource provisioning technique GA-ANN outperforms time aware and costs aware genetic approach using the performance metric fault.

Figs. 8(a) and (b) reveal the simulation results of the developed GA-ANN provisioning technique for the fault rate or failure count with varying VM count. Fig. 8 compares the variations of the failure count on increasing the user requests (tasks) with 5 VM's and 10 VM's correspondingly. The tasks or user requests are assigned to the virtual machine from different geographical areas across the globe.

5.2. Results using real workload file (The San -Diego Supercomputer Center (SDSC) Blue Horizon Logs)

The performance of the Neural with Genetic approach and existing task provisioning techniques measures using real workload data sets. In this work, the performance is measured using data sets from a system having 144-node IBM SP, with eight processors per node. The log duration is April 2000 to Jan 2003. The number of jobs included in the log file, i.e., 250, 440.

5.2.1. Simulation results comparisons using performance metric average start time

Section 5.2.1 shows the simulation results using ten different scenarios. The tasks increase from one thousand to ten thousand. It exhibits the average start time (ms) variations of the proposed GA-ANN and compared against BB–BC COST, GA-Cost, and GA-Exe, respectively. Requests are generated using Real Workload File (The San Diego Supercomputer Center (SDSC) Blue Horizon logs).

Figs. 9(a) and (b) reveal the simulation results of the presented GA-ANN resource provisioning technique using average start time with varying VM count. Fig. 9 compares the variations of the average start time on increasing the user requests (tasks) with 5 VM's and 10 VM's correspondingly. The tasks or user requests are mapped from different geographical areas across the globe. The results are obtained using real workload logs.

5.2.2. Simulation results comparisons using performance metric average finish time

Section 5.2.2 illustrates the simulation results using ten different scenarios. The tasks increase from one thousand to ten thousand. It shows the average finish time (ms) variations of the presented GA-ANN and compared against BB–BC COST, GA-Cost, and GA-Exe, respectively. Requests are generated using Real Workload File (The San Diego Supercomputer Center (SDSC) Blue Horizon logs).

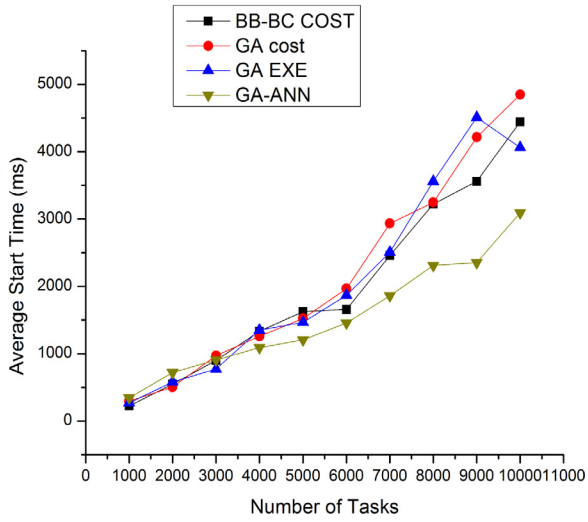

Fig. 9(b). Average start time (ms) vs. number of tasks with 10 VM's.

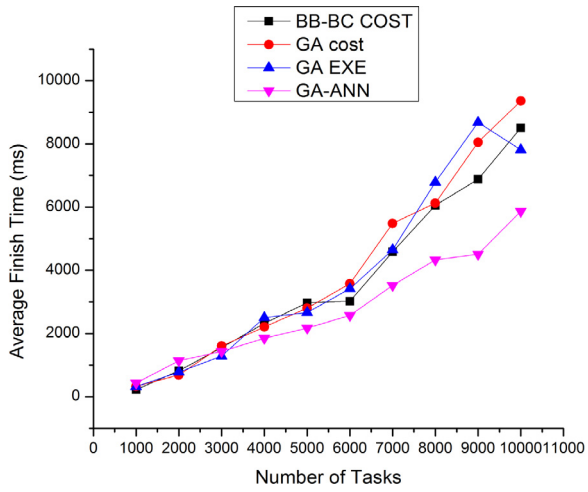

Fig. 10(a). Average finish time (ms) vs. number of tasks with 5 VM's.

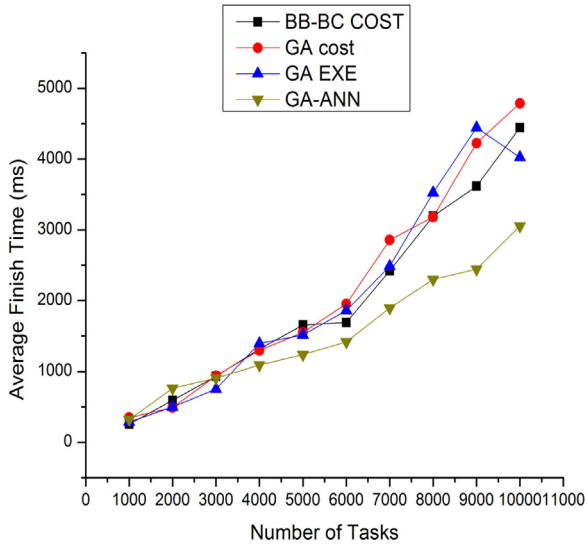

Fig. 10(b). Average finish time (ms) vs. number of tasks with 10 VM's.

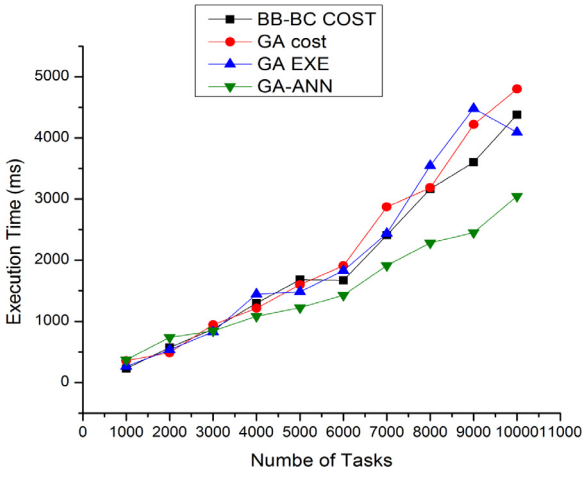

Fig. 11(a). Execution time (ms) vs. number of tasks with 5 VM's.

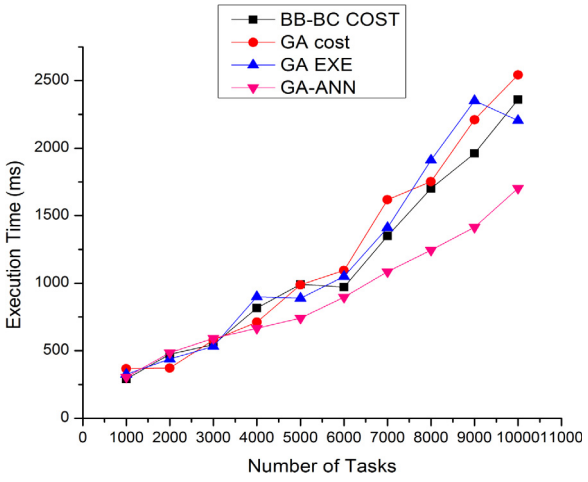

Fig. 11(b). Execution time (ms) vs. number of tasks with 10 VM's.

Figs. 10(a) and (b) illustrates the simulation results of the presented GA-ANN resource provisioning technique using average finish time with varying VM count. Fig. 10 compares the variations of the average finish time with increasing the user requests (tasks) with 5 VM's and 10 VM's correspondingly. The tasks or user requests are mapped from different geographical areas across the globe.

5.2.3. Simulation results comparisons using performance metric execution time

Section 5.2.3 exhibits the performance metric execution time (ms). The tasks are submitted on four virtual machines. The execution time is measured using tasks, which are generated using a real workload data sets logs. The number of requests varies from 1000 to 10 000. The proposed GA-ANN outperforms the existing meta-heuristic techniques.

Figs. 11(a) and (b) show the simulation results of the presented GA-ANN resource provisioning technique using performance metric execution time with varying VM count. Fig. 10 compares the variations of the execution time on increasing the user requests (tasks) with 5 VM's and 11 VM's correspondingly. The tasks or user requests are mapped from different geographical areas across the globe.

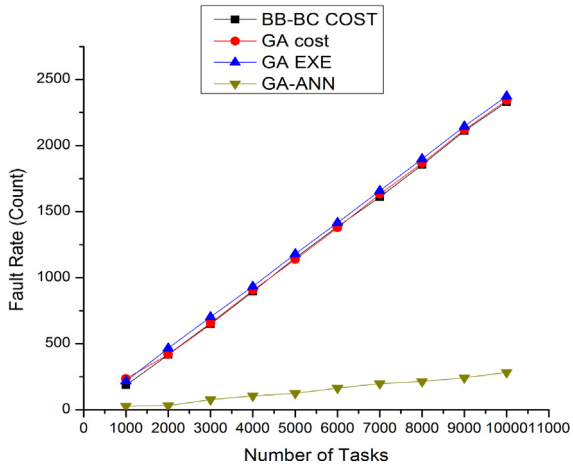

Fig. 12(a). Fault rate vs. number of tasks with 5 VM's.

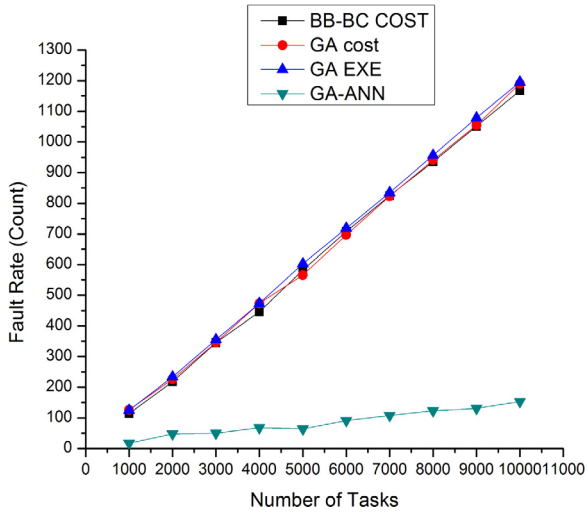

Fig. 12(b). Fault rate vs. number of tasks with 10 VM's.

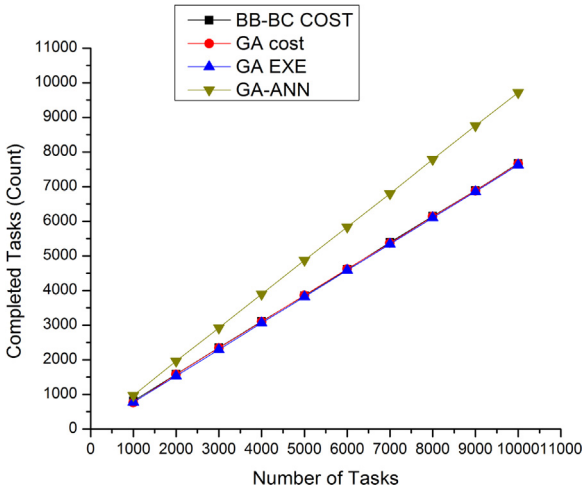

Fig. 13(a). Completed tasks (count) vs. number of tasks with 5 VM's.

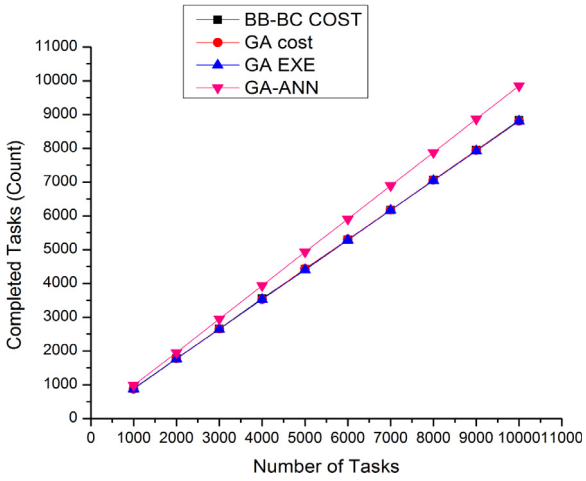

Fig. 13(b). Completed tasks (count) vs. number of tasks with 5 VM's.

5.2.4. Simulation results comparisons using performance metric fault rate

This sub-section describes the performance metric fault rate. It illustrates the variations of fault rate of the tasks executed virtual machines. The fault rate varies with an increasing number of user requests. The proposed GA-ANN technique is compared against three nature-inspired approaches. The tasks vary from 1000 to 10 000. The results exhibit that the proposed Neural-GA outperforms meta-heuristic techniques.

Figs. 12(a) and (b) demonstrates the simulation results of the proposed GA-ANN resource provisioning technique using performance metrics fault rate with varying VM count. Fig. 12 compares the variations of the fault rate on increasing the user requests (tasks) with 5 VM's and 10 VM's correspondingly. The tasks or user requests are mapped from different geographical areas across the globe.

5.2.5. Simulation results comparisons using performance metric completed tasks (count)

This sub-section describes the performance metric completed tasks count. It illustrates the variations of completed tasks from the tasks submitted on virtual machines. The completed task count varies with an increasing number of user requests. The proposed GA-ANN technique is compared against three nature-inspired approaches (GA cost, BB–BC COST, GA EXE). The number of requests varies from 1000 to 10 000. The results exhibit that the proposed Neural-GA outperforms meta-heuristic techniques.

Figs. 13(a) and (b) show the simulation results of the presented GA-ANN resource provisioning technique using performance metrics completed tasks count with varying VM count. Fig. 13 compares the variations of the completed task count on increasing the user requests (tasks) with 5 VM's and 10 VM's correspondingly. The tasks or user requests are mapped from different geographical areas across the globe.

Fig. 14 shows the simulation results of the presented GA-ANN resource provisioning technique using performance metric scheduling time (ms) with a varying number of tasks. It compares the variations of scheduling time on increasing user requests (tasks). The tasks or user requests are mapped from different geographical areas across the globe in a scalable scenario.

Fig. 15 illustrates the comparison of the simulation results of the presented GA-ANN resource provisioning technique using performance metric scheduling time (ms) with a varying number of iterations which can also be referred as a number of evolutions. It compares the variations of scheduling time on an increasing number of iterations. The presented GA-ANN approach outperforms the cost-aware BB–BC, GA, and execution time aware genetic approaches respectively.

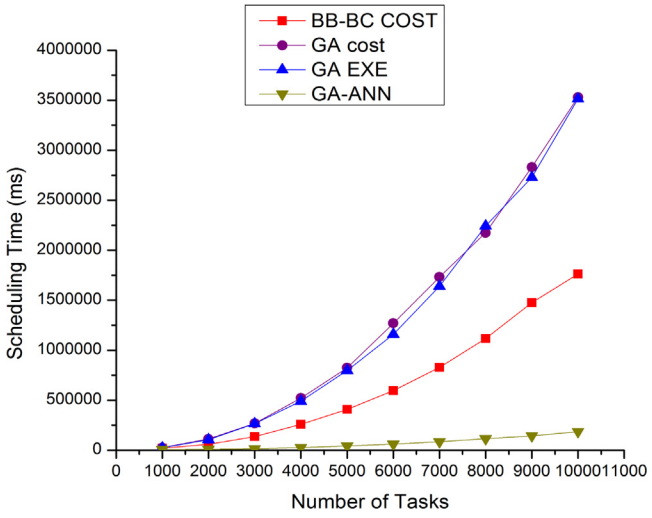

Fig. 14. Scheduling time vs. the number of tasks.

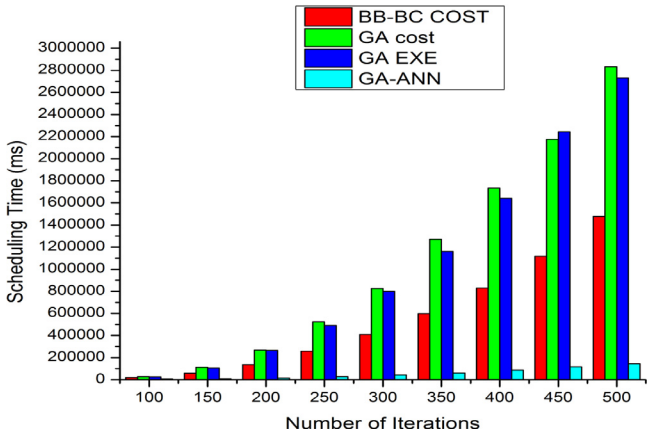

Fig. 15. Scheduling time vs. number of iterations with 1000 tasks.

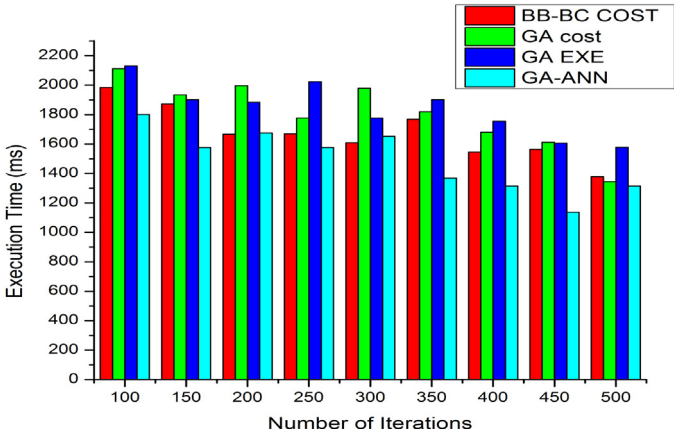

Fig. 16. Execution time vs. number of iterations with 1000 tasks.

Fig. 16 illustrates the simulation results comparisons of the presented GA-ANN resource provisioning technique using performance metric execution time (ms) with a varying number of iterations. It compares the variations of execution time on an increasing number of iterations. The number of iterations varies from 100 to 500 with an increment of 50. The presented GA-ANN approach outperforms the cost-aware BB–BC, GA, and execution time aware genetic approaches respectively.

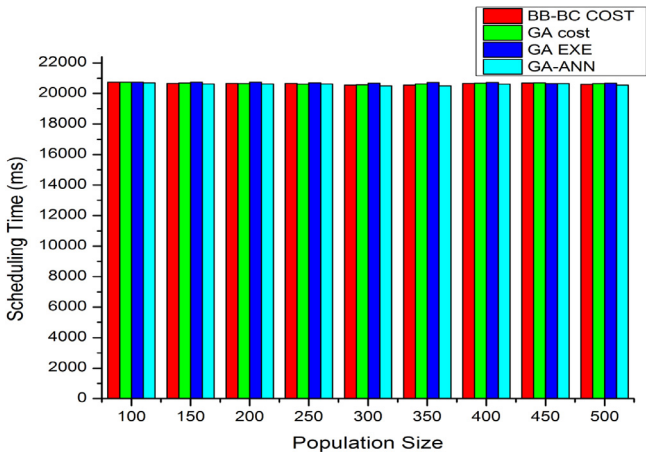

Fig. 17. Scheduling time vs. population size with 100 iterations & 1000 tasks.

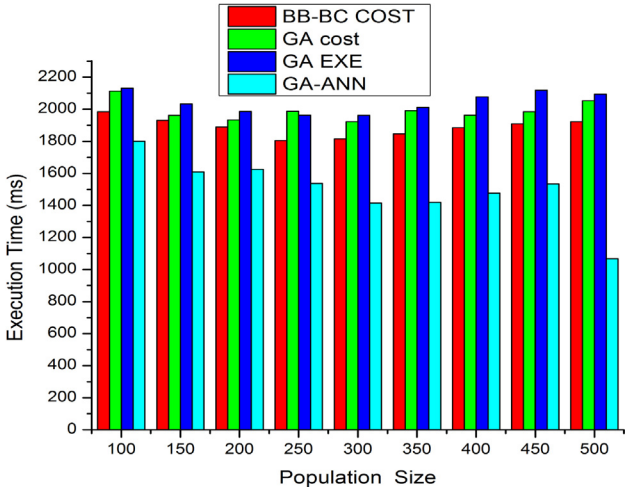

Fig. 18. Execution time vs. population size with 100 iterations & 1000 tasks.

Fig. 17 illustrates the comparison of the simulation results of the presented GA-ANN resource provisioning technique using performance metric scheduling time (ms) with varying population size. It compares the variations of scheduling time on increasing population size from 100 to 500. The presented GA-ANN approach presents better results than cost-aware BB–BC, GA, and execution time aware genetic approaches respectively but at the same time change in population size has not made a large change in scheduling time.

Fig. 18 illustrates the comparison of the simulation results of the presented GA-ANN resource provisioning technique using performance metric execution time (ms) with a varying population size from 100 to 500. The presented GA-ANN approach outperforms the cost-aware BB–BC, GA, and execution time aware genetic approaches respectively. The presented GA-ANN performance improves with increasing population size and finds a better solution than existing algorithms. This is because with an increase in population side more random populations are generated which increases the possibility to find a better solution.

6. Conclusion & future works

The task scheduler policy provides optimal scheduling of tasks on a virtual machine in a scalable cloud environment. The optimal

scheduling depends on the scheduling policy of tasks assignment on virtual machines. In this work, a neural bio-inspired scheduling approach is presented that follows the well-known features of the human brain. The performance of the proposed GA-ANN technique is assessed using real data sets and fabricated data sets and scaling the resources and increasing the tasks. The performance metrics include average start time (ms), average finish time (ms), execution time (ms), fault rate (failure count), and completed tasks (successfully completed tasks count), execution time (ms), and scheduling time (ms). The training, validation, and prediction is performed using real-time data sets from standard workload file and fabricated data sets at IaaS, PaaS, and SaaS level. The results and discussions section illustrates that the proposed GA-ANN technique outperforms the existing nature-inspired GA cost, GA EXE, and BB–BC COST meta-heuristic techniques in a scalable cloud scenario. The performance is measured using the tasks increasing from 1000 to 10 000 and variations of the population size and variation of the number of iterations from 100 to 500 respectively. The proposed neural bio-inspired GA-ANN approach performs better in the real, fabricated, and scalable scenario. This work focuses only on task scheduling on the virtual machine. In future work, the presented GA-ANN approach will be performed for virtual machine scheduling on the host inside the datacenter. The efficiency of the model will be evaluated and tested on real cloud scenarios using Amazon, Microsoft, and Google cloud.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

[1] R. Buyya, J. Broberg, A.M. Goscinski, Cloud Computing: Principles and Paradigms, Vol. 87, John Wiley & Sons, 2011.

[2] R. Buyya, Introduction to the ieee transactions on cloud computing, IEEE Trans. Cloud Comput. 1 (1) (2013) 3–21, http://dx.doi.org/10.1109/TCC.2013.13.

[3] M. Kalra, S. Singh, A review of metaheuristic scheduling techniques in cloud computing, Egypt. Inform. J. 16 (3) (2015) 275–295, http://dx.doi.org/10.1016/j.eij.2015.07.001.

[4] M. Maqableh, H. Karajeh, et al., Job scheduling for cloud computing using neural networks, Commun. Netw. 6 (03) (2014) 191, http://dx.doi.org/10.4236/cn.2014.63021.

[5] S. Islam, J. Keung, K. Lee, A. Liu, Empirical prediction models for adaptive resource provisioning in the cloud, Future Gener. Comput. Syst. 28 (1) (2012) 155–162, http://dx.doi.org/10.1016/j.future.2011.05.027.

[6] P.S. Rawat, P. Dimri, P. Gupta, Learning-based task scheduling using big bang big crunch for cloud computing environment, Recent Adv. Comput. Sci. Commun. (Former.: Recent Pat. Comput. Sci.) 13 (2) (2020) 137–146, http://dx.doi.org/10.2174/2213275912666190204125712.

[7] O.K. Erol, I. Eksin, A new optimization method: big bang-big crunch, Adv. Eng. Softw. 37 (2) (2006) 106–111, http://dx.doi.org/10.1016/j.advengsoft.2005.04.005.

[8] P.S. Rawat, P. Dimri, S. Kanrar, G.P. Saroha, Optimize task allocation in cloud environment based on big-bang big-crunch, Wirel. Pers. Commun. (2020) 1–44, http://dx.doi.org/10.1007/s11277-020-07651-1.

[9] Y. Changtian, Y. Jiong, Energy-aware genetic algorithms for task scheduling in cloud computing, in: 2012 Seventh ChinaGrid Annual Conference, IEEE, 2012, pp. 43–48, http://dx.doi.org/10.1109/ChinaGrid.2012.15.

[10] X. Liu, Z. Zhan, C. Author, K. Du, W. Chen, Energy Aware Virtual Machine Placement Scheduling in Cloud Computing Based on Ant Colony Optimization, 2014, pp. 41–47, http://dx.doi.org/10.1145/2576768.2598265.

[11] M.H. Ferdaus, M. Murshed, R.N. Calheiros, R. Buyya, Virtual machine consolidation in cloud data centers using aco metaheuristic, in: European Conference on Parallel Processing, Springer, 2014, pp. 306–317, http://dx.doi.org/10.1007/978-3-319-09873-9_26.

[12] L. Chimakurthi, et al., Power efficient resource allocation for clouds using ant colony framework, arXiv preprint arXiv:1102.2608.

[13] G. Wu, M. Tang, Y.-C. Tian, W. Li, Energy-efficient virtual machine placement in data centers by genetic algorithm, in: International Conference on Neural Information Processing, Springer, 2012, pp. 315–323, http://dx.doi.org/10.1007/978-3-642-34487-9_39.

[14] X. Wang, Y. Wang, H. Zhu, Energy efficient multi-job scheduling model for cloud computing and its genetic algorithm, Math. Probl. Eng. (2012) http://dx.doi.org/10.1155/2012/589243.

[15] G. Shen, Y.-Q. Zhang, A shadow price guided genetic algorithm for energy aware task scheduling on cloud computers, in: International Conference in Swarm Intelligence, Springer, 2011, pp. 522–529, http://dx.doi.org/10.1007/978-3-642-21515-5_62.

[16] J. Kolodziej, S.U. Khan, F. Xhafa, Genetic algorithms for energy-aware scheduling in computational grids, in: 2011 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing, IEEE, 2011, pp. 17–24, http://dx.doi.org/10.1109/3pgcic.2011.13.

[17] F. Tao, Y. Feng, L. Zhang, T.W. Liao, Clps-ga: A case library and pareto solution-based hybrid genetic algorithm for energy-aware cloud service scheduling, Appl. Soft Comput. 19 (2014) 264–279, http://dx.doi.org/10.1016/j.asoc.2014.01.036.

[18] A.-p. Xiong, C.-x. Xu, Energy efficient multi resource allocation of virtual machine based on pso in cloud data center, Math. Probl. Eng. (2014) http://dx.doi.org/10.1155/2014/816518.

[19] S. Wang, Z. Liu, Z. Zheng, Q. Sun, F. Yang, Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized datacenters, in: 2013 International Conference on Parallel and Distributed Systems, IEEE, 2013, pp. 102–109, http://dx.doi.org/10.1109/icpads.2013.26.

[20] A. Beloglazov, J. Abawajy, R. Buyya, Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing, Future Gener. Comput. Syst. 28 (5) (2012) 755–768, http://dx.doi.org/10.1016/j.future.2011.04.017.

[21] Z. Wang, K. Shuang, L. Yang, F. Yang, Energy-aware and revenue-enhancing Combinatorial Scheduling in Virtualized of Cloud Datacenter, J. Converg. Inf. Technol. 7 (1) (2012) 62–70, http://dx.doi.org/10.4156/jcit.vol7.issue1.8.

[22] S. Yassa, R. Chelouah, H. Kadima, B. Granado, Multi-objective approach for energy-aware workflow scheduling in cloud computing environments, Sci. World J. (2013) http://dx.doi.org/10.1155/2013/350934.

[23] H.F. Sheikh, I. Ahmad, D. Fan, An evolutionary technique for performance-energy-temperature optimized scheduling of parallel tasks on multi-core processors, IEEE Trans. Parallel Distrib. Syst. 27 (3) (2015) 668–681, http://dx.doi.org/10.1109/TPDS.2015.2421352.

[24] S.U. Khan, I. Ahmad, A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids, IEEE Trans. Parallel Distrib. Syst. 20 (3) (2008) 346–360, http://dx.doi.org/10.1109/TPDS.2008.83.

[25] S. Mostafavi, F. Ahmadi, M.A. Sarram, Reinforcement-learning-based foresighted task scheduling in cloud computing, arXiv preprint arXiv:1810.04718.

[26] C.-Z. Xu, J. Rao, X. Bu, A unified reinforcement learning approach for autonomic cloud management, J. Parallel Distrib. Comput. 72 (2) (2012) 95–105, http://dx.doi.org/10.1016/j.jpdc.2011.10.003.

[27] J.L. Berral, I. Goiri, R. Nou, F. Julia, J. Guitart, R. Gavalda, J. Torres, Towards energy-aware scheduling in data centers using machine learning, in: Proceedings of the 1st International Conference on Energy-Ecient Computing and Networking, 2010, pp. 215–224, http://dx.doi.org/10.1145/1791314.1791349.

[28] B. Tripathy, S. Dash, S.K. Padhy, Dynamic task scheduling using a directed neural network, J. Parallel Distrib. Comput. 75 (2015) 101–106, http://dx.doi.org/10.1016/j.jpdc.2014.09.015.

[29] A. Agarwal, H. Pirkul, V.S. Jacob, Augmented neural networks for task scheduling, European J. Oper. Res. 151 (3) (2003) 481–502, http://dx.doi.org/10.1016/s0377-2217(02)00605-7.

[30] S. Dam, G. Mandal, K. Dasgupta, P. Dutta, An ant colony based load balancing strategy in cloud computing, in: Advanced Computing, Networking and Informatics-Volume 2, Springer, 2014, pp. 403–413, http://dx.doi.org/10.1007/978-3-319-07350-7_45.

[31] K. Kousalya, P. Balasubramanie, Online grid scheduling using ant algorithm, Int. J. Eng. Technol. 1 (1) (2009) 21.

[32] M.A. Tawfeek, A. El-Sisi, A.E. Keshk, F.A. Torkey, Cloud task scheduling based on ant colony optimization, in: 2013 8th International Conference on Computer Engineering & Systems (ICCES), IEEE, 2013, pp. 64–69, http://dx.doi.org/10.1109/ICCES.2013.6707172.

[33] M. MadadyarAdeh, J. Bagherzadeh, An improved ant algorithm for grid scheduling problem using biased initial ants, in: 2011 3rd International Conference on Computer Research and Development, Vol. 2, IEEE, 2011, pp. 373–378, http://dx.doi.org/10.1109/ICCRD.2011.5764154.

[34] P. Singh, M. Dutta, N. Aggarwal, A review of task scheduling based on meta-heuristics approach in cloud computing, Knowl. Inf. Syst. 52 (1) (2017) 1–51, http://dx.doi.org/10.1007/s10115-017-1044-2.

[35] S. Kaur, A. Verma, An efficient approach to genetic algorithm for task scheduling in cloud computing environment, Int. J. Inf. Technol. Comput. Sci. (IJITCS) 4 (10) (2012) 74, http://dx.doi.org/10.5815/ijitcs.2012.10.09.
[36] J. Yu, R. Buyya, Scheduling scientic workflow applications with deadline and budget constraints using genetic algorithms, Sci. Program. 14 (3, 4) (2006) 217–230.
[37] S. Sawant, A Genetic Algorithm Scheduling Approach for Virtual Machine Resources in a Cloud Computing Environment, 2011, http://dx.doi.org/10.31979/etd.et9e-73fz.
[38] X. Lu, Z. Gu, A load-adapative cloud resource scheduling model based on ant colony algorithm, in: 2011 IEEE International Conference on Cloud Computing and Intelligence Systems, IEEE, 2011, pp. 296–300, http://dx.doi.org/10.1109/CCIS.2011.6045078.
[39] J. Sun, S.-W. Xiong, F.-M. Guo, A new pheromone updating strategy in ant colony optimization, in: Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No. 04EX826), Vol. 1, IEEE, 2004, pp. 620–625, http://dx.doi.org/10.1109/ICMLC.2004.1380766.
[40] P. Mathiyalagan, S. Suriya, S. Sivanandam, Hybrid enhanced ant colony algorithm and enhanced bee colony algorithm for grid scheduling, Int. J. Grid Util. Comput. 2 (1) (2011) 45–58, http://dx.doi.org/10.1504/IJGUC.2011.03998.
[41] A. Liu, Z. Wang, Grid task scheduling based on adaptive ant colony algorithm, in: 2008 International Conference on Management of E-Commerce and E-Government, 2008, pp. 415–418, http://dx.doi.org/10.1109/ICMECG.2008.50.
[42] C.-W. Chiang, Y.-C. Lee, C.-N. Lee, T.-Y. Chou, Ant colony optimisation for task matching and scheduling, IEE Proc.-Comput. Digit. Tech. 153 (6) (2006) 373–380, http://dx.doi.org/10.1049/ip-cdt:20050196.
[43] F. Pop, C. Dobre, V. Cristea, Genetic algorithm for dag scheduling in grid environments, in: 2009 IEEE 5th International Conference on Intelligent Computer Communication and Processing, IEEE, 2009, pp. 299–305, http://dx.doi.org/10.1109/ICCP.2009.5284747.
[44] Z. Zheng, R. Wang, H. Zhong, X. Zhang, An approach for cloud resource scheduling based on parallel genetic algorithm, in: 2011 3rd International Conference on Computer Research and Development, Vol. 2, IEEE, 2011, pp. 444–447, http://dx.doi.org/10.1109/ICCRD.2011.5764170.
[45] D.B. LD, P.V. Krishna, Honey bee behavior inspired load balancing of tasks in cloud computing environments, Appl. Soft Comput. 13 (5) (2013) 2292–2303, http://dx.doi.org/10.1016/j.asoc.2013.01.025.
[46] B. Jana, M. Chakraborty, T. Mandal, A task scheduling technique based on particle swarm optimization algorithm in cloud environment, in: Soft Computing: Theories and Applications, Springer, 2019, pp. 525–536, http://dx.doi.org/10.1007/978-981-13-0589-4_49.
[47] N. Kumar, P. Patel, 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, 2016, pp. 1–6, http://dx.doi.org/10.1145/2905055.2905115.
[48] N. Bacanin, T. Bezdan, E. Tuba, I. Strumberger, M. Tuba, Monarch butterfly optimization based convolutional neural network design, Mathematics 8 (6) (2020) 936, http://dx.doi.org/10.3390/math8060936.
[49] K. Sreenu, M. Sreelatha, W-scheduler: whale optimization for task scheduling in cloud computing, Cluster Comput. (2019) 1–12, http://dx.doi.org/10.1007/s10586-017-1055-5.
[50] M.R. Shirani, F. Safi-Esfahani, Dynamic scheduling of tasks in cloud computing applying dragonfly algorithm, biogeography-based optimization algorithm and mexican hat wavelet, J. Super-Comput. (2020) 7–020, http://dx.doi.org/10.1007/s1122, http://dx.doi.org/10.1007/s11227-020-03317-8.
[51] Y. Zhang, L. Wu, A hybrid ts-pso optimization algorithm, J. Converg. Inf. Technol. 6 (5) (2011) 169–174, http://dx.doi.org/10.4156/jcit.vol6.issue5.18.
[52] https://www.cse.huji.ac.il/labs/parallel/workload/l_sdsc_blue/.
[53] Parallel Workloads Archive: https://www.cse.huji.ac.il/labs/parallel/workload/.
[54] M.-Y. Cheng, A.F. Roy, Evolutionary fuzzy decision model for construction management using support vector machine, Expert Syst. Appl. 37 (8) (2010) 6061–6069, http://dx.doi.org/10.1016/j.eswa.2010.02.120.
[55] J. Sasikala, M. Ramaswamy, Optimal gamma based fixed head hydrothermal scheduling using genetic algorithm, Expert Syst. Appl. 37 (4) (2010) 3352–3357, http://dx.doi.org/10.1016/j.eswa.2009.10.015.
[56] Y. Zhang, L. Wu, A genetic ant colony classifier system, in: 2009 WRI World Congress on Computer Science and Information Engineering, Vol. 5, IEEE, 2009, pp. 744–748, http://dx.doi.org/10.1109/CSIE.2009.748.
[57] I.J. Graham, K. Case, R. Wood, Genetic algorithms in computer-aided design, J. Mater Process. Technol. 117 (1–2) (2001) 216–221, http://dx.doi.org/10.1016/S0924-0136(01)01144-X.
[58] Y. Zhang, L. Wu, Face pose estimation by chaotic artificial bee colony, Int. J. Digit. Content Technol. Appl. 5 (2) (2011) 55–63.
[59] S.K. Shukla, M. Tiwari, Soft decision trees: A genetically optimized cluster oriented approach, Expert Syst. Appl. 36 (1) (2009) 551–563, http://dx.doi.org/10.1016/j.eswa.2007.09.065.