Received: February 15, 2017

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Multiple Objective–Based Modified Exponential Gravitational Search Algorithm to VMM Strategy for Load Balancing in Cloud

Vijayakumar Polepally[1]* **Kaila Shahu Chatrapati[2]**

[1]*Kakatiya Institute of Technology and Science, Warangal, India*
[2]*JNTUH College of Engineering, Manthani, India*
* Corresponding author's Email: vijayakumarpolepally@gmail.com

Abstract: Cloud computing is a trending topic in the field of science and technology since the internet dependent services have been growing rapidly. In this environment, there are a lot of immense infrastructures and resources to satisfy the internet users. When a large number of service requests reach at a particular time, load balancing becomes a necessity. Load balancing involves the effective migration of the resources from the loaded physical machine to the other physical machine. For the effective migration, a method named Modified Exponential Gravitational Search Algorithm based on Virtual Machine Migration strategy (MEGSA-VMM) has been proposed that uses the gravitational concepts for performing the frequency-based velocity computations. MEGSA algorithm is the integration of the gravitational search algorithm and exponential weighted moving average theory. Also, the quality-of-Service (QoS) constraints considered for VM migration are migration cost, migration time, resource usage and energy. Simulation of the proposed method and the comparison of the results obtained, with the traditional methods like Ant-Colony Optimization (ACO), Gravitational Search Algorithm (GSA) and Exponential Gravitational Search Algorithm (EGSA) is performed. The proposed method is found to achieve an optimum migration with a minimum energy at a rate of 0.26 and minimum migration cost at a rate of 0.015.

Keywords: Cloud Computing, Load Balancing, Modified Exponential Gravitational Search Algorithm (MEGSA), VM Migration strategy (VMM), Fitness Function.

1. Introduction

Load balancing [1, 2] is the strategy to solve the uneven distribution of the resources in the network, which is the cause of increased data processing and data storage over the internet. Due to this uneven distribution, the overload and underload problems exist, which paved the way for load balancing [3, 4]. Load balancing [5, 6] is a term that contributes to the proper distribution of the resources among all the virtual machines available in the cloud [7-9]. Also, it does not place any single virtual machine loaded, but it distributes the workload among the multiple virtual machines [10, 11]. This distribution causes the improvement in the performance of the system and results in the proper resource utilization. Thus, load balancing is the trending strategy that contributes to achieving better resource utilization and better response time. Load balancing can be performed in two ways namely, distributed and non-distributed load balancing. In distributed balancing, load from the overloaded virtual machine gets distributed to the other virtual machines, whereas in the non-distributed load balancing, only one virtual machine is involved in load balancing [12-15].

The virtual machines are controlled using the large servers present in the data centers, and they share a lot of resources provided by the servers. These resources may get loaded in certain cases, and a perfect model is required, which computes the exact physical host for load balance [16]. Load balancing with the VMM strategy is the only solution to achieve the perfect load balance in the network. VMM strategy is simply the migration of the virtual machine from the loaded physical machine to the other physical machine. Also, it is essential to determine the physical host to undergo migration. Various technologies have been employed to determine the exact physical host to migrate the loaded virtual machine. After determining the physical host, load balancing is done [17]. However, this physical position does not hold good for the varying load. Therefore, it is clear that for varying load, the physical host varies, which poses the need to determine the optimum location [18, 19].

In this paper, a method named MEGSA-VMM is proposed, which is a gravitational search algorithm. In this strategy, the migration of the virtual machines from the loaded physical machine to the unloaded physical machine is done. The main contributions of the paper include the following:

MEGSA-VMM strategy: The modified exponential gravitational search algorithm that depends on the virtual machine migration strategy is the major contribution of this paper. Here, the virtual machines are migrated based on the available resources in the physical machines. The proposed MEGSA algorithm utilizes the maximum resources with minimum energy. The energy constraint computations are frequency-based calculations and promoted towards efficient VMM.

Fitness Function: Fitness function is another significant contribution of the paper that determines the efficient location to perform the migration. The fitness calculation uses the maximization concept, which means the value obtained by the computation of the fitness function should be maximum. The fitness function depends on the migration cost, QoS and the energy of consumption.

The major advantages of the proposed MEGSA-VMM model are, this method achieves the optimum position in minimal time and minimum energy and contributed to the maximum resource utilization.

The organization of this paper is described as follows: In Section 2, we present the system model of the proposed MEGSA-VMM algorithm which includes various models, such as energy model, QoS model, load model, and migration cost model. Section 3 describes our proposed MEGSA-VMM algorithm for load balancing in cloud computing. Results and discussion are presented in Section 4. The conclusion of the proposed method is presented in Section 5.

2. System model

This section presents the detailed description of the system model. It describes the cloud computing environment, and it provides the detailed explanation regarding the objectives and functions of energy modeling, load modeling, and QoS modeling.

Fig. 1 describes the system model of the cloud computing. The data center receives the user request and processes it. Depending on the time of arrival of the request, the queued task gets delivered to the users through the user interface. A user interface is a place, where the user request arrives, and the user fetches the required data. Consider, there is N number of physical machine present in the data center, which can be represented as,

$$P = \{P_1, P_2, P_3, \cdots, P_{i,} \cdots, P_N\} \tag{1}$$

where, P_i represents the i^{th} physical machine. $P_{1,}$ $P_{2,}$ P_3... are the physical machines present in the network. P_N represents the N^{th} physical machine. $|P|$ is the total number of physical machines present in the data center. The number of virtual machines corresponding to the physical machine present in the data center is given by,

$$V^i \equiv \{V_1^i, V_2^i, \cdots, V_j^i, \cdots, V_k^i\} \tag{2}$$

where, k is the total number of virtual machines present in the data center. $V_{1,}$ $V_{2,}$ V_3... are the number of virtual machines present in the network. V_j and V_k represent the j^{th}, k^{th} virtual machines respectively. $|V^i|$ is the total number of virtual machines corresponding to the i^{th} physical machine present in the data center.

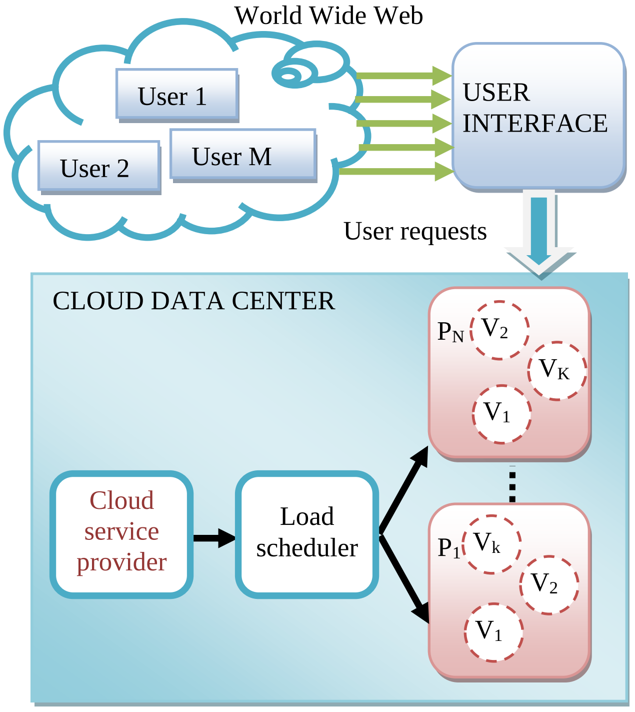


Figure. 1 Shows the system model of cloud computing

2.1 Energy model

The Energy model of the system is shown below,

$$e = \frac{1}{|P| \times |V|}\left[\sum_{i=1}^{|P|}\sum_{j=1}^{|V|} N_{ij}\, e_{max} + \left(1 - N_{ij}\right) R_{ij}^{U}\, e_{max}\right] \quad (3)$$

where, R^{U}_{ij} is the resource utilized by the i^{th} physical machine of the j^{th} virtual machine, e_{max} is the maximum energy consumed, e is the total energy consumed during migration. $|P|$ is the total number of physical machines present in the data center and $|V|$ is the total number of virtual machines present in the data center

The resources utilized by the i^{th} physical machine of the j^{th} virtual machine is shown below,

$$R_{ij}^{U} = \frac{1}{2}\left[\frac{C_{ij}^{U}}{C_{ij}} + \frac{M_{ij}^{U}}{M_{ij}}\right] \quad (4)$$

where, C^{U}_{ij} is the number of CPU utilized in the i^{th} physical machine of the j^{th} virtual machine. C_{ij} is the total number of CPU in the i^{th} physical machine of the j^{th} virtual machine, M^{U}_{ij} is the memory used by the i^{th} physical machine of the j^{th} virtual machine, and M_{ij} is the total memory available in the i^{th} physical machine of the j^{th} virtual machine.

2.2 QoS model

The QoS of the cloud network is formulated from the following equation as,

$$QoS^{t} = \frac{1}{2}\left[QoS^{c} + QoS^{m}\right] \quad (5)$$

where, QoS^{t} is the total quality of service of the cloud network, QoS^{c} is the quality of service of the CPU in the i^{th} physical machine and QoS^{m} is the quality of service of the memory in the i^{th} physical machine.

The computation of the QoS for the CPU and memory of the i^{th} physical machine is given using the expressions below,

$$Qos^{C} = \frac{1}{|P|}\sum_{i=1}^{|P|}\frac{C_{i}^{d}}{C_{i}} \quad (6)$$

where, C^{d}_{i} corresponds to the unutilized CPU in the i^{th} physical machine and C_i corresponds to the total CPU in the i^{th} physical machine. $|P|$ is the total number of physical machines present in the data center.

$$Qos^{M} = \frac{1}{|P|}\sum_{i=1}^{P}\frac{M_{i}^{d}}{M_{i}} \quad (7)$$

where, M^{d}_{i} is the unutilized memory present in the i^{th} physical machine and M_i is the total memory present in the i^{th} physical machine. $|P|$ is the total number of physical machines present in the data center.

2.3 Load model

During this process, load calculation of the overall cloud network is essential, and the calculation of load as in Eq. (8). Initially, a load threshold is fixed for detecting the loaded condition. Let L^{th} be the load threshold, and L_D be the calculated load of the network. When the calculated load is found less than the threshold load, this states that the network is not overloaded, but when the calculated load is found greater than the threshold load, the condition states that the system is overloaded. Suppose, when the virtual machine V_1 of the physical machine P_1 is executing a program in the user interface, the condition is said to as overload. Once the execution completes, the status of V_1 becomes unloaded. Thus, Eq. (8) shows the expression of the network load.

$$L_{M} = \frac{1}{|P| \times |V|}\left[\sum_{i=1}^{|P|}\sum_{j=1}^{|V|}\left(\frac{C_{ij}^{U}}{C_{ij}} + \frac{M_{ij}^{U}}{M_{ij}}\right)\right] \times \frac{1}{2} \quad (8)$$

where, C^{U}_{ij} is the CPU utilized by the j^{th} virtual machine in the i^{th} physical machine, C_{ij} is the total CPU utilized in i^{th} physical machine of the j^th virtual machine, M^{U}_{ij} is the memory used by the i^{th} physical machine of the j^{th} virtual machine, M_{ij} is the total memory available in the i^{th} physical machine of the j^{th} virtual machine, $|P|$ is the total number of physical machines present in the network and $|V|$ is the total number of virtual machines present in the network. L_M represents the load of the network. .

2.4 Migration cost model

Migration cost is the key factor in the virtual machine migration. Moreover, the profit of the service provider increases. M_C is the migration cost of the virtual machine and the formula of the migration cost is,

$$M_{C} = \frac{1}{|P|}\sum_{i=1}^{|P|}\left(\frac{M_{i}}{F \times |V^{i}|}\right) \quad (9)$$

where, M_i is the total number of virtual machine movements in the i^{th} physical machine. $|V^{i}|$ is the total number of virtual machines present in the i^{th}

physical machine and F is the scaling factor, and the scaling factor is 10. $|P|$ is the total number of physical machines present in the data center.

3. Proposed method based on MEGSA-VMM strategy

The proposed work that is based on the MEGSA-VMM is presented below. Fig. 2 shows the block diagram of the proposed method.

3.1 Load balancing algorithm

Fig. 3 presents the load balancing algorithm, which explains the detailed steps involved in load balancing. The migration cost of the physical machine one is assumed to be 1. In the load determination step, the load value of the cloud network is determined using the Eq. (8). Then, compare the load value of the system obtained through computations with the threshold value of 0.8.

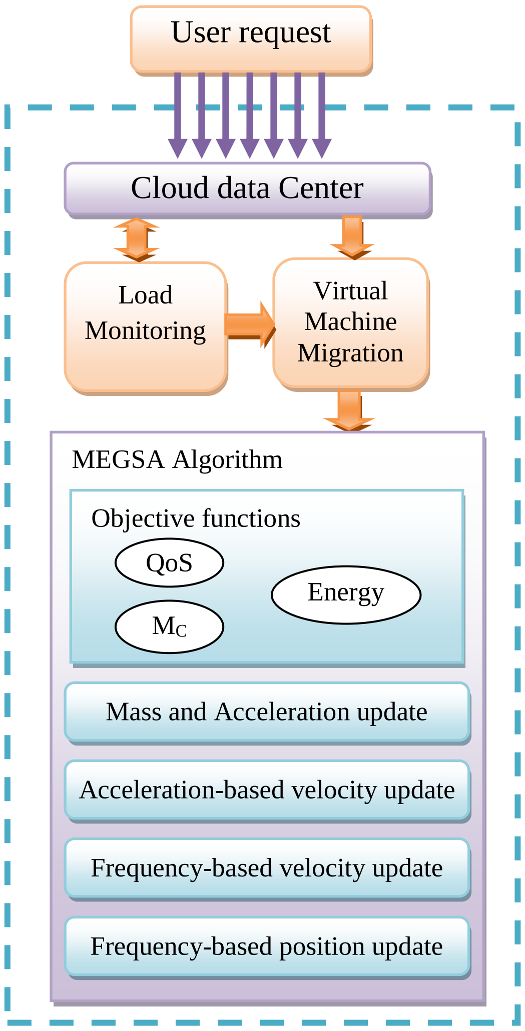

User request
Cloud data Center
Load Monitoring
Virtual Machine Migration
MEGSA Algorithm
Objective functions
QoS
Mc
Energy
Mass and Acceleration update
Acceleration-based velocity update
Frequency-based velocity update
Frequency-based position update

Figure. 2 Block diagram of the proposed MEGSA-VMM

3.2 MEGSA algorithm for VM migration strategy

EGS algorithm employed for the efficient migration of the virtual machines did not consider the energy constraint. Therefore, a proposed modified algorithm named Modified Exponential Gravitational Search Algorithm for VMM (MEGSA-VMM) is used that considers the energy constraint in optimization.

3.2.1. Solution encoding

In the cloud computing environment, there are a large number of physical machines and virtual machines. Let us consider that there are two physical machines present in the cloud environment. Eq. (10) presents the physical machines present in the network.

$$\text{Physical machines, } P = \{P_1, P_2\} \qquad (10)$$

where, P_1 and P_2 are the physical machines.

Consider, there are two virtual machines present in the physical machine, P_1 and the total virtual machine present in P_1 is, $V^1 = \{V_1, V_2\}$. There are three virtual machines are present in the physical machine, P_2. The virtual machines present in P_2 is $V^2 = \{V_3, V_4, V_5\}$. Fig. 4 shows the migration principle. Suppose, if the virtual machine V_2 is loaded, it gets migrated to the other unloaded physical machine P_2 as shown in the Fig. 4. Thus, the unloaded V_5 migrates to P_1.

	Load Balancing Algorithm
Step 1	Initialization of the cloud model
Step 2	Initialize the migration cost of physical machine
Step 3	Load determination
Step 4	if (Load > 0.8)
Step 5	Perform VMM.
Step 6	Migration cost update.
Step 8	End

Figure. 3 Load balancing algorithm

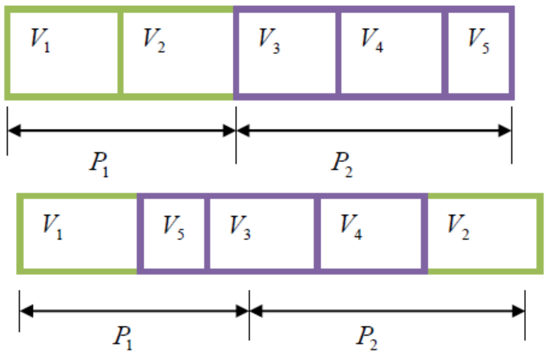


Figure. 4 VMM strategy

3.2.2. Fitness calculation

Fitness calculation is an important parameter in the VMM strategy, which identifies the best migrating location. Fitness is the key factor for performing the effective migration of the virtual machines without affecting the system performance. Eq.(11) shows the formula for performing the fitness calculation.

$$fit = \alpha(1 - M_C) + \beta QoS + \gamma(1 - e) \qquad (11)$$

where, M_c is the migration cost of the virtual machines. QoS is the quality of service. α,β,γ are the constants which are fixed here as 0.3, 0,3 and 0.4. e is the total energy consumed during migration.

The fitness calculation uses the maximization concept, which means the value obtained by the computation of the fitness function should be maximum. The fitness function depends on the migration cost, QoS and the energy of consumption. The value of the migration cost varies between 0 and1. Therefore, *(1-M$_c$)* in the Eq. (11) highlights that the migration cost reduces and lies below 1 and therefore, the maximum migration cost is 1. Also, energy consumption is another significant factor that contributes to a better fitness value. The energy consumed while utilizing the resources should be minimum. However, the resource utilization should remain maximum.

Fitness calculation is possible when the resources present in the loaded physical machine are greater than the old resources present in the machine else the fitness function is zero.

$$Fitness = \begin{Bmatrix} fit; Resource\ of\ L_M > Old\ Resources \\ 0; Otherwise \end{Bmatrix} \qquad (12)$$

where, L_M represents the loaded physical machine. The Eq. (11) determines the best fit values, and the other factors like the migration cost and the quality of service are determined for the best fit value.

3.2.3. MEGSA algorithm

The Modified Exponential Search algorithm uses two approaches, namely the Exponential Weighed moving average theory and the gravitational theory.

Step 1: Parameter Initialization: Initialization of the parameter step involves initializing the positions of the virtual machines. Let Y denote the position, and the following expression gives the position of the i^{th} physical machine.

$$y_i = \left\{y_i^1, y_i^2, \cdots, y_i^g, \cdots, y_i^{|P|}\right\} \qquad (13)$$

where, y_i is the position of the i^{th} physical machine. $i=1, 2, 3,... N$. N denotes the number of physical machines present in the cloud network.y^g_i is the position of the i^{th} physical machine in g^{th} dimension.

Step 2: Calculating the fitness values: The fitness function obtained from Eq. (11) is utilized in this step to determine the fitness value. The fitness value is essentially an important parameter to perform a highly efficient migration of virtual machines.

Step 3: Determining the best and worst fits: From the fitness value obtained from the previous step, the best fit values and the worst fit values are determined. The fitness value is based on the maximization problem. Below formula determines the best, and worst fit values from the fitness calculation carried out in step2,

$$bestfit(T) = \max_{j\in\{1,\cdots,N\}} fit_j(T) \qquad (14)$$

$$worstfit(T) = \min_{j\in\{1,\cdots,N\}} fit_j(T) \qquad (15)$$

Step 4: Mass Calculation: The parameter calculations depend on the gravitational theory to compute the mass, acceleration, and force. According to this gravitational concept, each agent exhibits a mass value and a corresponding acceleration.

$$M_i(T) = \frac{m_i(T)}{\sum_{j=1}^{N} m_j(T)} \qquad (16)$$

where, $m_i(T) = \frac{fit_j(T) - worst(T)}{best(T) - worst(T)}$ (17)

where, $m_i(T)$ is the mass of the i^{th} agent, and $m_j(T)$ is the mass of the j^{th} agent.

The gravitational law of motion explains the acceleration of the agent. According to the law of motion, the acceleration is defined as the ratio of the total force acting on the i^{th} agent to the inertial mass of the i^{th} agent. Therefore, the acceleration of the agent i is given by

$$a_i^g(t) = \frac{f_i^g(T)}{M_i^I(T)} \qquad (18)$$

where, $a^g_i(t)$ is the acceleration of the i^{th} agent on g^{th} dimension. $M^I_i(T)$ is the inertial mass of the i^{th} agent. and $f^g_i(T)$ is the force acting on the i^{th} agent on g^{th} dimension. The total force acting on the i^{th} agent is calculated based on the law of gravity, which states that there exists a force of attraction between any two objects. The attraction is mainly due to the gravity, and this gravity is directly proportional to the product of masses and inversely proportional to the square of the distance between the objects. Thus, the expression for total force is as follows,

$$f_i^g(T) = \sum_{i=1, j\neq 1}^{N} r_j f_{ij}^g(T) \tag{19}$$

where, $$f_{ij}^g(T) = G_C(T) \times \frac{m_i^P(T) \times m_j^a(T)}{D_{ij}(T)+\varepsilon}\left(y_j^g(T) - y_i^g(T)\right) \tag{20}$$

where, $f^g_i(T)$ is the force acting on the i^{th} agent on g^{th} dimension. $G_C(T)= fucn\ (G_S,\ T)$. g is the dimension of the system, G_C is the gravitational constant, r_j is the random number, G_S is the initial value of gravitation, $m^p_i(T)$ - Passive gravitational mass with respect to the agent i, $m^a_j(T)$ - Active gravitational mass with respect to the agent i, $D_{ij}(T)$ - Euclidian distance between the agent i and j respectively. $f^g_{ij}(T)$ is the force acting on the i^{th}, j^{th} *agents on* g^{th} dimension. $y^g_j(T)$ is the velocity of the i^{th} agent on g^{th} dimension, and $y^g_j(T)$ is the velocity of the j^{th} agent on g^{th} dimension.

Step 5: Velocity calculation based on acceleration: By using the acceleration calculated from the previous step, the velocity y is calculated using the formula given below,

$$V_{T+1}^a = r_j \times V^a(T) + a_i^g(T) \tag{21}$$

where, r_j is the random number. V^a_{T+1} is the velocity value based on the acceleration value.

$a^g_i(T)$ is the acceleration of the i^{th} agent. $V^a(T)$ is the acceleration-based velocity value of the previous iteration.

Step 6: Velocity calculation based on frequency: Velocity is calculated based on the frequency, and the position of the previous iteration, and the velocity of the previous iteration. Velocity computed in this step is required to update the position of the virtual machine. The migration is performed in an efficient way, in other words, the time required for the migration of the virtual machines contribute a lot towards the system performance. The result of migration is that it reduces the interaction of the virtual machines among themselves causing the minimal consumption of energy. The velocity update depends on the best fit values, and this velocity effectively contributed towards determining the optimum VMM with minimal migration time and minimal energy consumption. Frequency based velocity calculation is done from the below formula,

$$V_{T+1}^f = V_T^f + (y_T - y_{best}) \times f_T \tag{22}$$

where, V^f_{T+1} is the velocity value based on the frequency value. V^f_T is the velocity of the previous VM location that is calculated based on the frequency.

y_{best} is the best set of position values obtained with a high fitness function.

$$f_T = f_{min} + \Delta f \times \beta \tag{23}$$

where, $\Delta f=f_{max}-f_{min}$ is computed by taking the difference between f_{max} and f_{min}. f_{max} is the maximum frequency, f_{min} is the minimum frequency, and β is constant.

Step 7: Position update based on the velocity: Normally, the position is computed based on the velocity-based acceleration calculation. The position computation is carried out depending on the conventional GSA [12]. The position for the VMM based on GSA is given below,

$$y(T+1) = y(T) + V(T+1) \tag{24}$$

It clearly indicates that the position update depends on the position and velocity of the previous iteration. Rearranging the above equation we get,

$$y(T+1) - y(T) = V(T+1) \tag{25}$$

According to the algorithm EWMA, the position to perform the migration is based on the position value obtained using the formula shown below,

$$Y(T+1) = \frac{1}{w}[Y^W(T+1) - (1-w) \times Y^W(T)] + V(T+1) \tag{26}$$

Where, y^W is the exponential weighted agent. $y(T+1)$ is the velocity of the solution obtained during the present iteration. $Y\ (T+1)$ is the optimum position to undergo VMM. w is the positive constant.

The existing method of position update determined only the optimum position and did not consider the energy constraint. Hence, a method is proposed to perform the optimum migration considering the energy constraint. This method uses the velocity values of the previous step obtained from the Eq. (20) and (21), to determine the position. The position to perform migration is calculated using a formula given below,

$$Y_{T+1} = \frac{1}{w}\left[e_{T+1} - (1-w) \times e_T\right] + \frac{1}{2}\left[V^{a}_{T+1} + V^{f}_{T+1}\right] \tag{27}$$

where, e_{T+1} and e_T are the energy values respectively, V^{a}_{T+1} is the velocity value based on the acceleration value, V^{f}_{T+1} is the velocity value based on the frequency value, w is the positive constant.

Step 8: Iteration: Iterations are carried out to determine the best migrating location with minimal time. When the best values are attained to meet the required conditions, it performs the position update and the migration.

4. Results and discussion

4.1 Experimental set up

The performance of the proposed MEGSA-VMM algorithm for load balancing is evaluated using the experiment. Experimentation is done in the personal computer with Intel Core i-3 processor 4GB RAM and Windows 8 operating system. The experimentation is carried out using the cloudsim tool with JAVA. The performance evaluation of the proposed MEGSA-VMM algorithm for load balancing is done using two setups.

Setup 1: The set up1 comprises of three physical machines with 12 virtual machines for resource allocation.

Setup2: The cloud set up 2 is made up of 5 physical machines and 19 virtual machines.

4.2 Methods taken for comparison

The performance of the proposed MEGSA-VMM algorithm is compared with the other existing methods like the ACO [13], EGSA-1, EGSA-2 and GSA [20] regarding load, QoS, migration cost and energy. ACO is an optimization algorithm, which performs load balancing using the ant-colony concept. Gravitational search theory is the fundamental concept involved in EGSA. EGSA-1 is similar to the proposed algorithm but the first two objectives are kept in the fitness function by applying the α and β as 0.4 and 0.6. EGSA-2 is similar to the proposed algorithm but the first two objectives are presented in the fitness function by applying the α and β as 0.6 and 0.4.

4.3 Performance evaluation

In this section, the performance criteria, namely QoS, migration cost, load, and energy are determined to compare the results of the proposed MEGSA with other existing methods like the ACO, GSA and EGSA and the discussions are presented below.

4.3.1. Setup 1

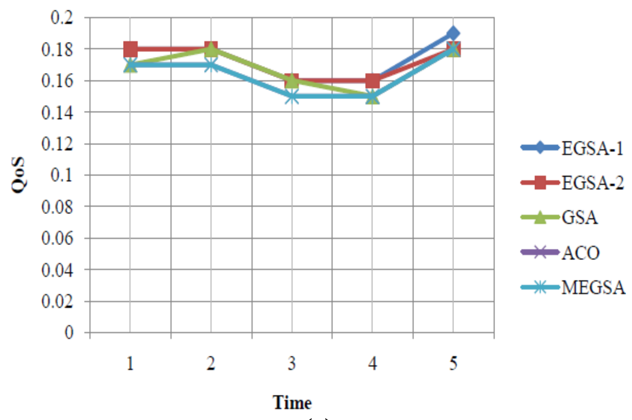
This section presents the evaluation of the proposed method in terms of migration cost, QoS, energy, and load for cloud set up1. Fig. 5 depicts the performance comparison of the proposed MEGSA with other algorithms.

The QoS obtained for the algorithms are shown in Fig. 5(a). When the time is t=1 sec, the value of QoS obtained for the algorithms ACO, GSA, EGSA (α=0.4, β =0.6), EGSA (α=0.6, β =0.4) are 0.16, 0.18, 0.18 and 0.18 respectively whereas, for the proposed MEGSA, the value of QoS is 0.19. One can understand that the QoS value for the proposed MEGSA is improved when compared to the other existing algorithms. Fig. 5 (b) depicts the variation of migration cost with respect to time. The migration cost should be low for the effective VM migration. The migration cost for the algorithms ACO, GSA, EGSA (α =0.4, β =0.6), EGSA (α =0.6, β =0.4) and MEGSA are 0.024, 0.024, 0.002, 0.002 and 0.002 respectively when the time is 2 secs. When the time is 0.3 secs, the migration cost for the algorithms ACO, GSA, EGSA (α=0.4, β =0.6), EGSA (α=0.6, β =0.4) and MEGSA are 0.012, 0.024, 0.002, 0.002 and 0.002 respectively. These values prove that the migration cost of the proposed MEGSA is found to be minimum when compared with the other algorithms. Also, it is clear that as time increases from t=1 to 5 secs, the migration cost reduces.

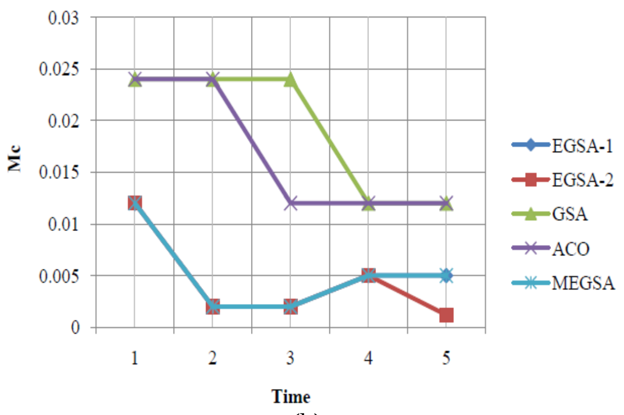
Fig. 5(c) figures out the load of the cloud set up1 for the algorithms, namely ACO, GSA, EGSA, and MEGSA. For performing the effective VM migration, the load should be minimum. When t=1 sec, the load value obtained is 0.1863, 0.1863, 01657, 0.1657 and 0.1551 respectively for ACO, GSA, EGSA (α =0.4, β =0.6), EGSA (α =0.6, β =0.4) and MEGSA. It is clear that the load is the minimum for the proposed MEGSA as compared to the existing methods. Additionally, as time increases, the load value reduces from 0.1551 to 0.1549, which is very much low when compared to the other methods. Fig. 5(d) shows the energy curve obtained for the algorithms. The energy consumed should be low with the maximum resource utilisation. When *t*=2 secs, the energy value is 0.29, 0.28, 0.27, 0.27 and 0.26 respectively for the algorithms ACO, GSA, EGSA(α =0.4, β =0.6 and α =0.6, β =0.4) and MEGSA. It is evident that the energy consumed while using the proposed MEGSA is found to be less when compared to the other methods and as time increases, the energy value reduces. Energy efficiency obtained using the proposed MEGSA promotes effective VM migration with maximum resource utilization.

4.3.2. Set up 2

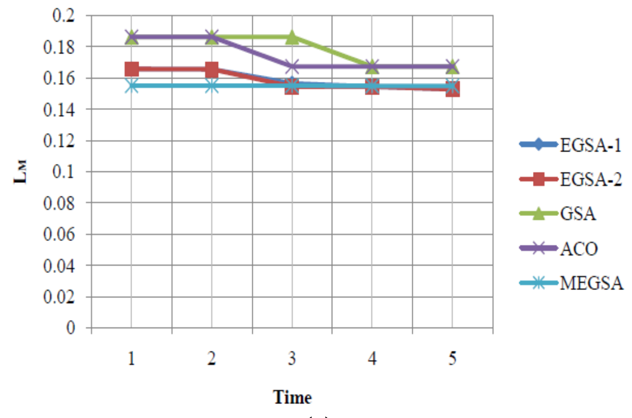
Fig. 6(a) shows the variation of QoS for the proposed MEGSA and the existing algorithms. The QoS of the network should be maximum for efficient resource utilization. When t=3 secs, QoS obtained for the algorithms ACO, GSA, EGSA (α =0.4, β =0.6), EGSA (α =0.6, β =0.4) and MEGSA are 0.21, 0.22, 0.22, 0.22 and 0.23 respectively. Fig. 6(b) depicts the variation of migration cost for the various algorithms. The cost for performing the migration should be minimum to perform the migration. When t=2 secs, the migration cost obtained with ACO, GSA, EGSA (α =0.4, β =0.6), EGSA (α=0.6, β =0.4) and MEGSA are 0.023, 0.023, 0.016, 0.016 and 0.015 respectively. Fig. 6(c) clearly visualizes the variation of the load with respect to time. The load in the network should be minimum to perform the migration and to achieve maximum resource utilization. Fig. 6(d) visualizes the variation of energy with respect to time and the comparison of the various algorithms.



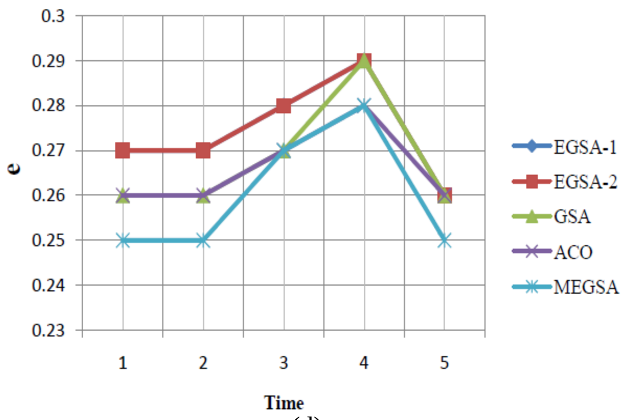
(a)



(b)



(c)



(d)

Figure. 5 Performance comparison of the proposed MEGSA with EGSA, GSA, and ACO for the setup1: (a) QoS Vs Time (b) Migration cost Vs time (c) Load Vs Time (d) Energy Vs Time

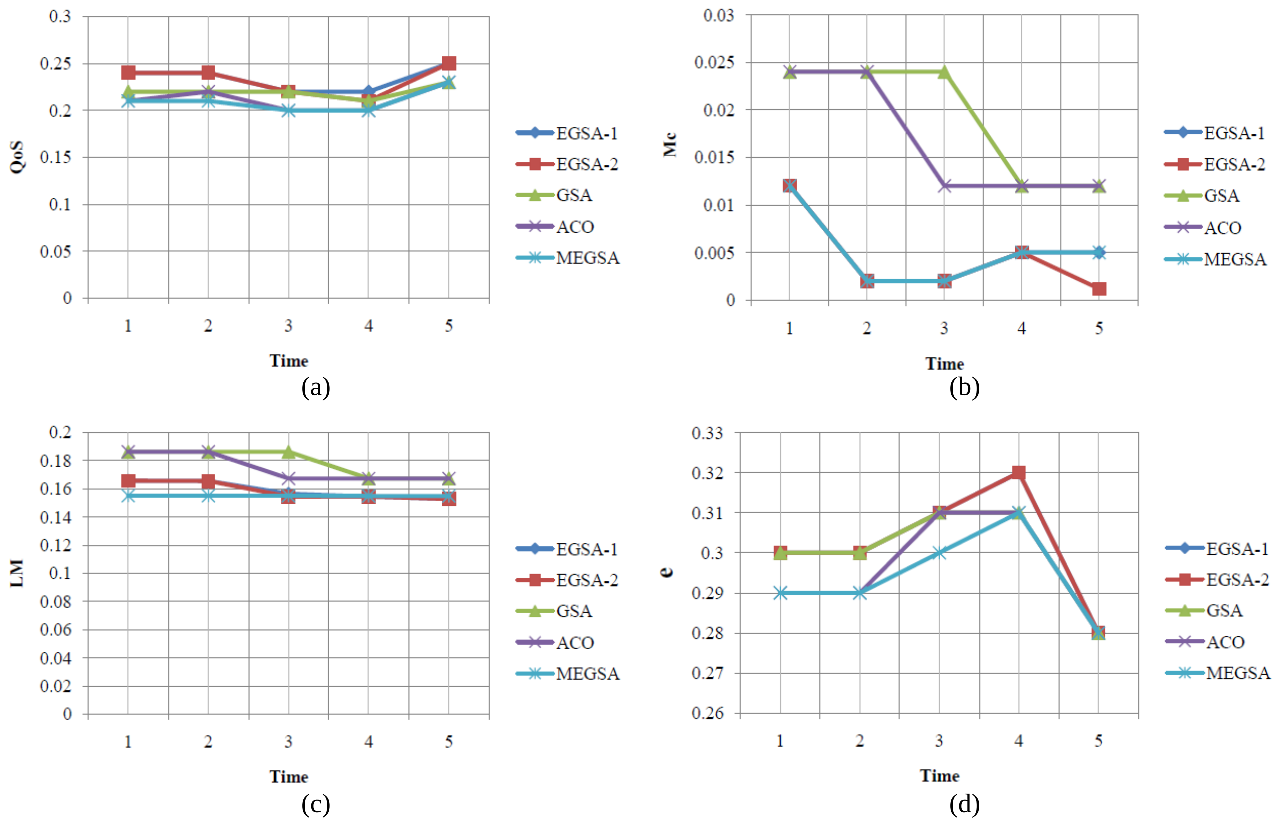


Figure. 6 Performance comparison of the constraints for cloud set up 2: (a) QoS Vs Time (b) Migration cost Vs time (c) Load Vs Time (d) Energy Vs Time

Table 1. Comparative analysis of the proposed method

METHODS	QoS	M_c	L_M	*e*
MEGSA	**0.19**	**0.015**	**0.1551**	**0.26**
EGSA-1 (α=0.4,β =0.6),	0.18	0.016	0.1657	0.27
EGSA-2 (α=0.6, β =0.4)	0.18	0.016	0.1657	0.27
GSA	0.16	0.023	0.1863	0.28
ACO	0.16	0.023	0.1863	0.29

4.4 Discussion

Table 1 shows the comparative analysis of the proposed MEGSA with the other algorithms. MEGSA is proposed to perform effective load balancing with the required criteria. In other words, the optimum position determined using the proposed work reduces the interaction between the nearby virtual machines, which reduces the energy consumption. Also, it promotes maximum resource utilization with good quality. In numerical terms, the minimum migration cost rate of 0.015 is required to perform the migration with good quality at a rate of 0.19 that consumed very less energy at a rate of 0.26.

5. Conclusion

In this proposed work, the MEGSA-VMM strategy has been used that contributed a lot towards performing the virtual machine migration in an effective way. The VMM strategy employed in this algorithm is used in load balancing. Effective migration has been achieved through the determination of the optimum position using MEGSA. This optimum position depends upon the migration cost, QoS, load, and energy. The results prove that the proposed MEGSA achieved the optimum position in minimal time and minimum energy at a rate of 0.26, which is less compared to the other methods. Similarly, MEGSA contributed to the maximum resource utilisation at a good QoS rate of 0.19 with a minimal load of 0.1551. Thus, it is evident that MEGSA is better when compared to other traditional methods. The energy parameter has been taken into consideration, which is the major contribution of MEGSA. The energy consumed and the migration cost during the migration is maintained at a minimal rate, which is the major fitness factor. In future, the load balancing can be achieved by assigning the task optimally to the relevant VMs instead of migrating the VMs.

References

[1] K.Dasgupta, B.Mandal, P.Dutta, J.Kumar Mondal, and S.Dam, "A Genetic Algorithm (GA) based Load Balancing Strategy for Cloud

Computing", *Procedia Technology*, Vol. 10, pp. 340 – 347, 2013.

[2] L.D. DhineshBabu and P.V.Krishna, "Honey bee behaviour inspired load balancing of tasks in cloud computing environments", *Applied Soft Computing*, Vol. 13, No.5, pp. 2292–2303, 2013.

[3] W.T. Wen, C.D. Wang, D.S. Wu and Y.Y.Xie, "An ACO-Based Scheduling Strategy on Load Balancing in Cloud Computing Environment", In: *Proc. of International Conference on Frontier of Computer Science and Technology*, pp. 364 - 369, 2015.

[4] L.Cheng and F.C. M. Lau ,"Offloading Interrupt Load Balancing from SMP Virtual Machines to the Hypervisor", *IEEE Transactions on Parallel and Distributed Systems*, Vol.27, No.11, pp.3298 - 3310, 2016.

[5] X.Xu, L. Cao, X. Wang, X. Xu, L.Cao and X.Wang, "Resource pre-allocation algorithms for low-energy task scheduling of cloud computing", *Journal of Systems Engineering and Electronics*,Vol.27, No.2, pp.457 - 469, 2016.

[6] Q.Qi, J. Wang, Qi Li, T.Li and Y.Cao ,"Resource orchestration for multi-task application in home-to-home cloud", *IEEE Transactions on Consumer Electronics*,Vol. 62, No.2,pp.191 - 199, 2016.

[7] S.Ningning, G.Chao , A.Xingshuo and Z. Qiang, "Fog computing dynamic load balancing mechanism based on graph repartitioning", *China Communications*,Vol. 13, No. 3,pp.156 - 164, 2016.

[8] R.K.Naha and M.Othman, "Cost-aware service brokering and performance sentient load balancing algorithms in the cloud", *Journal of Network and Computer Applications*, Vol. 75, pp.47-57, 2016.

[9] A.S.Milani and N.J.Navimipour, "Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends", *Journal of Network and Computer Applications*, Vol.71, pp.86-98, 2016.

[10] S.Chander, P. Vijaya and P.Dhyani, "DOFL: Kernel Based Directive Operative Fractional Lion Optimisation Algorithm for Data Clustering", *International review on computers and softwares*, Vol.11, No.8, 2016.

[11] N. Moganarangan, R.G. Babukarthik, S. Bhuvaneswari, M.S. SaleemBasha and P. Dhavachelvan, "A novel algorithm for reducing energy-consumption in cloud computing environment: Web service computing approach", *Journal of King Saud University - Computer and Information Sciences*,Vol.28, No.1, pp.55-67, 2016.

[12] S.D.GopaMandal, K.Dasgupta and P. Dutta, "Genetic Algorithm and Gravitational Emulation Based Hybrid Load Balancing Strategy In Cloud Computing", In: *Proc. of Third International Conference on Computer, Communication, Control and Information Technology* (C3IT), 2015.

[13] Y.Gao, H.Guan, Z.Qi, Y.Hou and L.Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing", *Journal of Computer and System Sciences*, Vol.79, No.8, pp.1230-1242, 2013.

[14] H.Zhong, Y.Fang and J.Cui, "LBBSRT: An efficient SDN load balancing scheme based on server response time", *Future Generation Computer Systems*, Vol.68, pp.183-190, 2017.

[15] J. S Chou, N. T Ngo, W.K. Chong and G.E. Gibson Jr., "16 – Big data analytics and cloud computing for sustainable building energy efficiency", *Start-Up Creation*, pp.397-412, 2016.

[16] D. Merkle, M. Middendorf and H. Schmeck, "Ant colony optimization for resource-constrained project scheduling", *IEEE Transactions on Evolutionary Computation*,Vol.6,No.2,pp.333- 346, 2002.

[17] S.G. Domanal and G.R.M.Reddy, "Optimal load balancing in cloud computing by efficient utilization of virtual machines", In: *Proc. of the Sixth International Conference on Communication Systems and Networks* (COMSNETS) ,pp.1 - 4, 2014.

[18] S.Jiang, Z.Ji and Y.Shen, "A novel hybrid particle swarm optimization and gravitational search algorithm for solving economic emission load dispatch problems with various practical constraints", *International Journal of Electrical Power & Energy Systems*, Vol.55, pp.628-644, 2014.

[19] F.F.Matos, J.Celestino and A.R.Cardoso, "VBalance: A selection policy of virtual machines for load balancing in cloud computing", In: *Proc. of the IEEE Symposium on Computers and Communication* (ISCC), pp.770 - 775, 2016.

[20] Y.Liu and L.Ma, "Improved gravitational search algorithm based on free search differential evolution", *Journal of Systems Engineering and Electronics*, Vol.24, No.4, pp.690 - 698, 2013.