Title: Cloudy GSA for load scheduling in cloud computing

Authors: Divya Chaudhary, Bijendra Kumar

PII: S1568-4946(18)30434-4
DOI: https://doi.org/10.1016/j.asoc.2018.07.046
Reference: ASOC 5014

To appear in: *Applied Soft Computing*

Received date: 15-10-2016
Revised date: 21-7-2018
Accepted date: 22-7-2018

Please cite this article as: Chaudhary D, Kumar B, Cloudy GSA for load scheduling in cloud computing, *Applied Soft Computing Journal* (2018), https://doi.org/10.1016/j.asoc.2018.07.046

Cloudy GSA for Load Scheduling in Cloud Computing

Divya Chaudhary*, Bijendra Kumar

Department of Computer Engineering
Netaji Subhas Institute of Technology, Dwarka, New Delhi, India
divyadabas@gmail.com, bizender@gmail.com

Highlights of the paper Cloudy-GSA

This work signifies:

- This paper gives a new Cloudy-GSA optimization method for solving the problem of load balancing
- The Proposed approach reduces the transfer time considerably as shown in tabulated and graphical analysis.
- The results are computed at various values and compared with the existing algorithm results.
- This algorithm reduces the total cost incurred by the system.
- The entire approach is elaborated with the help of the algorithm and the flowchart in a detailed manner.
- Thus, final result is reduction of the makespan along with cost i.e., the algorithm provides a positive result.

Abstract –Scheduling of load and data plays an important role in the efficient utilization of the resources from one cloudlet to another cloudlet in the cloud computing environment. Cloud computing is an incremental paradigm to brighten the world with its great vision of providing the power of distributed computing through virtual approach. Resource allocation plays an important role in the optimal handling of the load scheduling problem using static and meta-heuristic approaches. The Gravitational Search Algorithm (GSA) is a nature-inspired meta-heuristic optimization technique which is used for solving the load scheduling problem in the cloud computing environment and is based on Newton's gravitational law dealing with gravity. This paper proposes a near optimal load scheduling algorithm named Cloudy-GSA to minimize the transfer time and the total cost incurred in scheduling the cloudlets to the VMs. These are achieved by increased exploitation of VMs using the particles based on fitness values. The Cloudy-GSA algorithm is implemented on the CloudSim and has been compared with the existing popular algorithms.

The results of the algorithm are converged and statistically analysed over a set of iterations. As evident from the results, the proposed Cloudy-GSA algorithm minimizes the transfer time and the total cost for scheduling the load than the existing algorithms.

Keywords - Cloud Computing, Load Scheduling, Gravitational Search Algorithm, Swarm Intelligence, PSO

1. Introduction

The cloud computing is one of the increasing domains in the area of distributed and grid computing using the concept of virtualization. The computing refers to the processing of the tasks on the virtual machines in the system for the efficient functioning of the tasks. The computational power of networks is increasing day by day to make the world follow and know about the massive hidden potential present inside it. The cloud acts as a repository of the resources enabling the users with large capabilities and computing facilities like storage, processing, extraction and retrieval of the information dealing with a large number of heterogeneous tasks to produce better resource access to the users. The cloud is based on the pay-as-you-go or pay-as-you-use model for accessing the resources. Buyya et al. [1] state that the cloud provides resource and task scalability, on time resource execution, dynamic provisioning, fault tolerance and interoperability of the resources. It provides dynamic allocation of the cloudlets/ tasks to the virtual machines. The functionality of dynamic allocation is achieved by load scheduling of the cloudlets in a near optimal manner in the cloud. This is performed to achieve greater throughput, lesser execution and waiting time, less transfer time, and less cost of computation [2].

The load scheduling is defined as the process of providing, allocating and balancing of the load (tasks/ cloudlets to the virtual machines) in the cloud system efficiently. The main purpose is to reduce the transfer time and the total cost incurred for scheduling the load in the system [3]. The scheduling of the load is performed using various scheduling algorithms. The scheduling algorithms are specified on the basis of nature as static and dynamic algorithms. These are also classified as heuristic and non-heuristic algorithms. The meta-heuristic algorithms play an important role in scheduling the load by using a search mechanism. Vecchiola et al. [4] elaborate the problem solving approach having multiple objectives in the cloud. This method provides solution using an optimization strategy. The load scheduling based on the swarm intelligence methods provides a larger significance in the environment as they involve the real world behaviour of the swarms. A group of objects, particles, ants etc. following the mechanism for locating the food is followed to find the best solution in particle swarm optimization and ant colony optimization mechanisms. The physical laws of gravity are used for finding the near optimal solution among the group of objects. The scheduling of load in the cloud is an important problem that needs to be resolved using the more efficient algorithms than the existing algorithms like Segmented Min-Min, Tabu Search, Simulated Annealing, Genetic Algorithm, FCFS, PSO etc. [5]. In this paper, Newton's law of gravity is used for performing the scheduling of load. This is achieved by using the force and acceleration based on gravitational laws for locating the next particle for execution. The force is calculated based on the mass of the particle and the distance between the particles. Thus, we propose a Cloudy-GSA for load scheduling in cloud computing. It reduces the transfer time and the total cost of computation as compared to the existing algorithms using convergence and statistical analysis. Rest of the paper is organised as follows.

Section 2 provides a review on the load scheduling in cloud computing. In Section 3, the mathematical formulation and description of the gravitational search algorithm is specified. The proposed optimization approach Cloudy-GSA in cloud environment is given in Section 4. Section 5 illustrates the experimental setup used for implementing the Cloudy-GSA algorithm with the convergence and statistical analysis in a tabulated and graphical manner. Finally, section 6 provides the conclusion and future scope of advancement.

2. Load Scheduling in Cloud Computing

The cloud computing has been derived out from the parallel, distributed and grid computing. It supports a large number of applications in the system like processing of the tasks or applications to the corresponding resources. A cloud in general is an elastic and distributed system where storage space and resources are distributed throughout the network. The resources like the information, hardware and software can be accessed

in a shared manner from a distributed location in cloud [2]. It works on the pay-as-you-use model where the amount of the resource utilization is calculated on the server side and payment is made on the basis of the utilization by the end users. Khiyaita et al. [6] stated that architecture of a cloud supports two models namely: deployment model and service model as illustrated in figure 1. The deployment model illustrated the provisioning location of the resources along with the organizational structure. The clouds are distinguished on the above parameters as Private, Public, Community and Hybrid Cloud. The service model described the type of services available to the end user viz. Software as a Service (as applications accessible through standard interfaces), Platform as a Service (operation and development platforms) and Infrastructure as a Service (infrastructure based resources such as data storage, networking capacity and processing power).

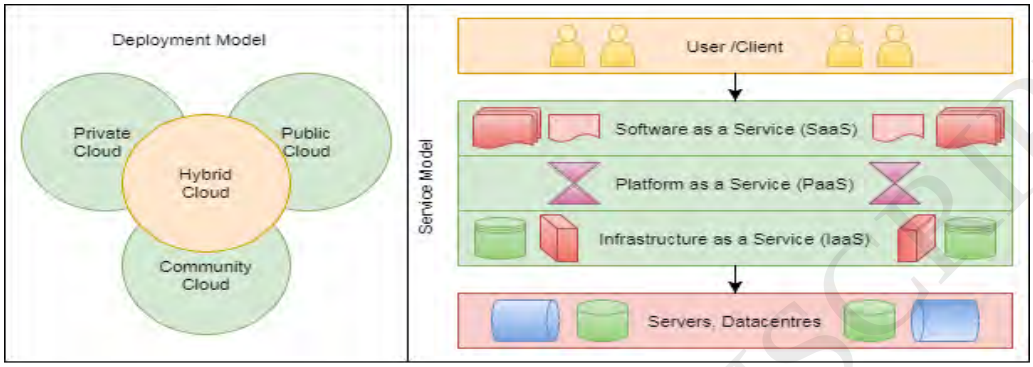


Figure1. The architecture specifying the service and deployment models in Cloud Computing [2].

It provides the users with enormous benefits like location independence, reduced costs and marketing time, wide network access, resilient computing, huge computational power, virtualization, elastic and highly scalable workout capabilities as given by Chaudhary et al. [7]. The tasks consumed a large amount of resources thereby dealing with computational complexity, costs, transfer time and response times. The system helps in the dynamic allocation of resources to the different heterogeneous tasks at diversified locations.

The load scheduling is needed to schedule the load from the overloaded virtual machines to under-utilized machines. Yu et al. [8] provided the workflow based scheduling methods over the nodes in the grid. These are based on the workflow of the task from one node to another node. A swarm based scheduling based on particle swarm heuristic is provided by Zhang et al. [9] in the grid computing. These tasks follow the nature of a flock of birds to move from one node to another. The particle swam strategy provides better utilization of the resources. The meta-heuristic specify a higher-level heuristic for finding, generating, or selecting a heuristic algorithm with inaccurate and incomplete data to provide a better solution to a problem. Tsai et al. [10] elaborated the various meta-scheduling algorithms in the cloud. The analysis of the load scheduling algorithms in similar manner is given by Chaudhary et al. [11] and Braun et al. [13]. The swarm based methods provide much efficient scheduling than other algorithms in the cloud. The swarm specifies a group of objects or particles in the search space. Pacini et al. [12] presented a detailed analysis of the swarm based methods of ants, bees or birds for finding the next best position on the basis of methods followed by them to locate the food. These provide maximum exploitation of the resources. The meta-heuristics are also elaborated using various applications in structural optimization by Yildiz et al. [14]. The robustness and scalability of finding the next particle in optimization methods is specified based on meta-heuristic swarm based strategies [15, 16]. Garg et al. [17] provided the network based capabilities for scheduling in the cloud.

Kennedy et al. [18] proposed a particle swarm optimization mechanism in a mathematical context. This is based on the behavior of the flock of birds to locate the food source. This optimization strategy led to higher exploitation of the resources with reduced computational cost. Tasgetiren et al. [19] and Yoshida et al. [20] used this optimization mechanism in flow shop scheduling for managing the tasks efficiently and for managing stability for reactive power and controlling the voltage. These strategies efficiently schedule the load but with high cost of computation. These constraints are applied on the particle swarm optimization by Zavala et al. [21] to decrease the cost of computation but leading to a more complex system. The PSO scheduling strategy is applied on the grids based on improved fitness values and parameter as given in [22, 23]. The fuzzy sets are used for evaluating the functions involved in scheduling. Izakian et al. [24] and Kang et al. [25] specified the

discrete PSO algorithm for resource allocation in heterogeneous environments. These are based on using different set of tasks but the complexity of algorithm is high and exploitation of nodes is low. Pandey et al. [26] proposed particle swarm optimization for workflow scheduling in cloud computing. The cloudlets are scheduled on the virtual machines based on the position generated by the particles. The cost of computation is reduced but exploitation of the resources is not optimal. To increase the exploitation of resources at reduced cost Kumar et al. [27] proposed the improved PSO approach in cloud computing environment.

3. Gravitational Search Algorithm

A gravitational search algorithm is a meta-heuristic bio-inspired search algorithm to find the nearest possible optimal path and to locate next specific solution as proposed by Rashedi et al. [28]. The meta-heuristic defines an advanced mechanism to locate, create and select a heuristic algorithm for a better solution using imprecise partial data. Heuristics are techniques which find near optimal solution at appropriate cost of computation without finalizing a proper optimal condition as shown by Karagöz et al. [29]. These heuristic algorithms play an important role in formulating the concepts as they replicate the physical and biological processes. All the bio-inspired swarm based algorithms depend on various real world objects like the ants, birds, bees etc. to locate their food. The gravitational search algorithm (GSA) based on the law of gravity, plays an important role in finding the best optimal path among the objects. The gravity has an affinity to accelerate masses towards each other. According to Newton's Law of gravitation, every object in the universe attracts every other object by a gravitational force. The gravity exists between any two objects in the entire universe. The objects are defined in the search space by various characteristics in the universe like masses (active or passive), inertia, force and their inter-distances. Each object calculates the force acting between the particles based on the acceleration. The two laws are reproduced here.

a) Newton's Law of Gravitation: Every particle attracts every other particle and the gravitational force between two particles is the product of their masses and inversely proportional to square of the distance between the two particles.

The gravitational force, F between two particles is directly proportional to the product of their masses, M_1 and M_2 and inversely proportional to the square of the distance R between them.

$$F = G\frac{M_1 M_2}{R^2} \tag{1}$$

b) Newton's Law of Motion: The force is directly proportional to inertial mass and acceleration. The velocity of the next particle depends on the initial particle velocity and change in velocity of the particles.

The acceleration, a of the particle depends on the force and the acting masses.

$$a = \frac{F}{M} \tag{2}$$

Thus, the heavier and closer particles exert more gravitational force than the lighter and distant particles. The higher the distance between the two particles, the lesser is the gravitational force between them.

The force acting on each particle shares that information with the corresponding objects present in the search space as given by Yildiz et al. [30]. The objects execute independently but they show a cumulative effect on the objects thus showing the results of the swarm intelligence by near optimal good solutions. This heuristic search algorithm explores the search area to identify the near optimal solution of the given problem. Rashedi et al. [31] showed that the two concepts searching and utilization are used for finding the best solution. In searching, entire search space is used to find the next best near optimal solution with optimal search space characteristic utilization to provide the final best solution. The objects worked on two principles [32] as shown in equations (1) and (2).

The fitness value of each object is calculated on the basis of mass of an object and the position and inertial mass of the corresponding object. The convergence in the results is achieved by clustering the particles such that ameliorated results are achieved. The GSA in [33] does not support load scheduling and convergence analysis on objects. The proposed work has addressed these issues.

4. Cloudy Gravitational Search Algorithm (Cloudy-GSA)

The proposed method Cloudy-GSA is based on the gravitational search algorithm in the cloud computing environment used to optimize the load. The objective of Cloudy-GSA is to achieve efficient load scheduling in cloud computing using higher exploitation of the virtual machines by the cloudlets. The aim is to reduce the total transfer time and the total cost of computation in the system. This is achieved using the fitness value of the particles in the cloud. The Cloudy-GSA overcomes all these shortcomings of higher transfer time and more number of computations by the other scheduling algorithms. Consider a system having N number of virtual machines, *NVM* to be used to schedule *NC* cloudlets. Total number of possible allocation of cloudlets to virtual machines is $(NVM)^{NC}$. To find the best position of the cloudlets on the virtual machines, we use Cloudy-GSA for scheduling. Let *NP* is the number of particles in the swarm in the *d* dimensional search space. The particles help in finding the best position of virtual machines for the cloudlets. These cloudlets are then executed on the virtual machines. Figure 2 illustrates 20 particles in the search space. Node 1 (light yellow colour) acts as the initial node or the particle. The force acting on all the nodes (dark yellow colour) in respect to node 1 is computed. The resultant force between any two nodes is taken to move from one particle to another. The calculation of the resultant force acting on the node 1 depends on the neighbouring active nodes (green colour) viz. 2, 19 and 20. The next particle is chosen based on the force.

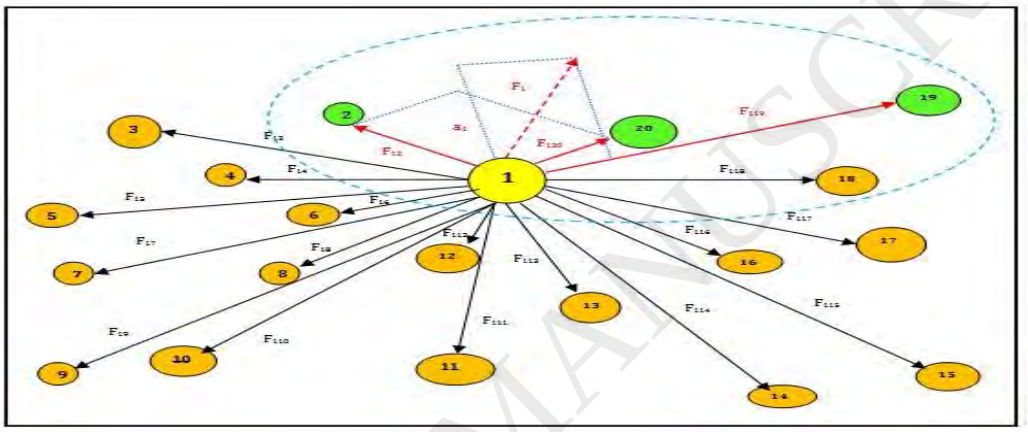


Figure2. Resultant force calculated on the particle 1 in respect of active particles 2, 19 and 20 in the search space.

The *NP* particles are defined in the *d* dimensional search space having a large set of possible solutions. The dimension values are derived by the virtual machines VMs. This algorithm uses the *best()* and *worst()* position (defined later) of the particles. The fitness value for each particle is calculated on the basis of the fitness function.

$$NP_i = \left(np_i^1, np_i^2, \ldots\ldots, np_i^n, \ldots\ldots\ldots, np_i^d\right) \qquad \forall\ i = 1\ to\ 25 \text{ and } n = 1\ to\ 10 \tag{3}$$

The fitness function depends on the execution cost and transfer cost parameters of the cloudlets and virtual machines. The execution and transfer cost of j^{th} cloudlet is specified as $Cost\_Exec(M)_j$ and $Cost\_Transfer(M)_j$ respectively. The total cost $Cost\_Total(M)_j$ which is the sum of the execution and transfer cost is computed for every cloudlet and then the minimization of the total maximum cost $Total\_Comp\_Cost\ (M)$ is performed. The fitness function $fitness\_particle_i^t$ of the particle *i* at an instant *t* of time is calculated as:

$$Cost\_Exec(M)_j = \sum_k w_{kj} \qquad \forall M(k) = j \tag{4}$$

$$Cost\_Transfer(M)_j = \sum_{k1 \in T} \sum_{k2 \in T} d_{M(k1),M\ (k2)} * e_{k1,k2} \qquad \forall\ M(k1) = j and\ M(k2) \neq j \tag{5}$$

$$Cost\_Total(M)_j = Cost\_Exec(M)_j + Cost\_Transfer(M)_j \tag{6}$$

$$Total\_Comp\_Cost\ (M) = \max\left(Cost\_Total(M)_j\right) \qquad \forall_j \in P \tag{7}$$

$$Minimize(Total\_Comp\_Cost\ (M) \quad \forall\ M) \tag{8}$$

So, the fitness value of each particle $fitness\_particle_i^t$ is generated. The masses active $Mass\_active_i$, passive $Mass\_passive_i$ and inertia $Mass\_inertia_i$ are calculated based on the fitness value. The *best()* stores the best fitness value i.e., minimum value among the particles. The *worst()* stores the maximum fitness value among the set of the particles in the search space. The $best(t)\ and\ worst(t)$ values of the particles for minimization of total cost and mass $M_i(t)$ are calculated as:

$$Mass\_active_i = Mass\_passive_i = Mass\_inertia_i = M_i, \qquad i = 1,2,3, \dots\dots, N \tag{9}$$

$$m_i(t) = \frac{fitness\_particle_i(t) - worst(t)}{best(t) - worst(t)} \tag{10}$$

$$M_i(t) = \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{11}$$

$$best(t) = \frac{min}{j\in\{1,\dots,N\}} fitness\_particle_j(t) \tag{12}$$

$$worst(t) = \frac{max}{j\in\{1,\dots,N\}} fitness\_particle_j(t) \tag{13}$$

The force of the particle in cloud is based on a gravitational constant, *G(t)* at a specific time instant. The *G(t)* determines the particle's potential and increases the efficiency of movement. The gravitational constant helps in the exponential rise in the search space area. It helps in the larger exploitation or usage of the resources. It depends on G_0 *(initial value)* and time instance *t*. It also depends on α (a random value) and maximum number of iterations *Max_Iter* specified.

$$G(t) = G_0 e^{-\alpha \frac{t}{Max\_Iter}} \tag{14}$$

The total force acting on the particle helps in finding the next near optimal particle position. It depends on the Euclidean distance $R_{ij}\ (t)$ between the two particles NP_i and NP_j in *d* dimensional search space. The static function $rand_j$ lies in the interval [0, 1). So, the total force $F_{ij}^d(t)$ acting on particle *i* with respect to particle *j* at particular instance of time *t* on the particles:

$$R_{ij}\ (t) = \left|\left| NP_i(t), NP_j(t) \right|\right|_2 \tag{15}$$

$$R\ (NPi, NPj) = \sqrt{(NP_1 - NP_1)^2 + \dots\dots + (NP_j - NP_i)^2 + \dots\dots + (NP_N - NP_N)^2} \tag{16}$$

$$R\ \left(NP_i, NP_j\right) = \sqrt{\sum_{i=1,\ j=1}^{N} (NP_j - NP_i)^2} \tag{17}$$

$$F_{ij}^d(t) = G(t) \frac{Mass\_passive_i\ (t) \times Mass\_active_j(t)}{R_{ij}(t) + \varepsilon} (np_j^d\ (t) - np_i^d(t)) \tag{18}$$

$$F_i^d(t) = \sum_{j=1, j\neq i}^{N} rand_j F_{ij}^d\ (t) \tag{19}$$

$$a_i^d(t) = \frac{F_i^d(t)}{Mass\_inertia_i(t)} \tag{20}$$

The acceleration $a_i^d(t)$ of particle *i* at a specific time *t* in the *d* dimensional space depends on force and inertial mass. Figure 3 illustrates the flowchart for the proposed Cloudy-GSA for load scheduling.

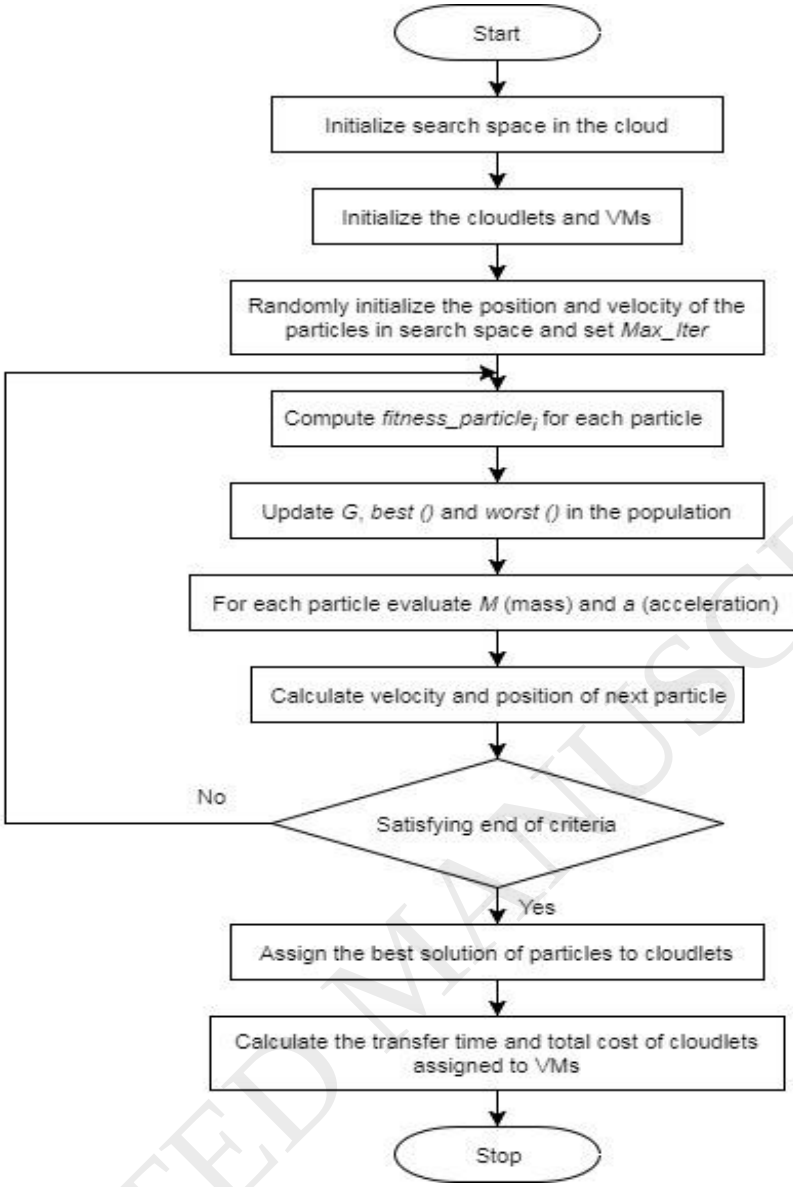


Figure 3. Flowchart depicting Cloudy-GSA Optimization approach for load scheduling in cloud

The next particle to be executed depends on the velocity of the particle. The velocity of the next iteration of the particle $vel_i^d(t+1)$ is calculated based on random uniform function $rand_i$ having value [0, 1] and velocity of particle and acceleration (eq. 21). The next position of the particle $np\_pos_i^d(t+1)$ is selected by old particle position at time *t* with velocity of the next particle.

$$vel_i^d(t+1) = rand_i \times vel_i^d(t) + a_i^d(t) \tag{21}$$

$$np\_pos_i^d(t+1) = np\_pos_i^d(t) + vel_i^d(t+1) \tag{22}$$

The process continues till the condition of maximum iterations is met. The best position of the particles is returned to the cloudlets for scheduling the load. The cloudlets are then allocated to respective VMs based on the positions to run on the data centres.

So, the entire Cloudy-GSA scheduling continues till all the cloudlets are executed on the respective virtual machines in the cloud to obtain minimum transfer time and minimum total cost of computation. The best near optimal results generated by Cloudy-GSA approach are better than the other scheduling algorithms like Segmented Min-Min, Genetic Simulated Annealing, Simulated Annealing (SA), Genetic Algorithm (GA), Tabu

Search, Min-Min, FCFS and PSO algorithm in cloud computing. The Cloudy-GSA overcomes the shortcomings of the existing algorithms like larger computation time by reducing the transfer time and the total cost of computation incurred by the cloud. The larger exploitation of the resources is performed by the cloudlets. The comparative analysis of the existing relevant algorithms and the proposed Cloudy-GSA algorithm is discussed in the next section.

5. Results and Analysis

The proposed Cloudy-GSA algorithm and the existing algorithms viz. Segmented Min-Min, Genetic Simulated Annealing, Simulated Annealing (SA), Genetic Algorithm (GA), Tabu Search, Min-Min, FCFS and PSO algorithm for load scheduling in the cloud are simulated in the CloudSim (Refer to **Appendix**) [13]. It offers the users with a built-in environment for implementation of algorithms similar to the real world scenario. The Network CloudSim Simulator is built on top of the CloudSim and is given by Garg and Buyya [17]. The results are compared on the basis of two metrics namely Transfer Time and Total Cost depicted in graphical manner. The simulation parameters are specified in Table 1 based on JSwarm package.

Table 1: Simulation Parameters

Parameter Description	Values
(x, y) co-ordinates	0 – 7
Number of particles	25
Number of VMs	8
Number of Cloudlets	10
Max_Iter	10-100 and 100– 1000
G_0	1
ram	2048
***B**andwidth*	10000
Storage	10000
Cloudlets	10
Datacentre Count	2
VM in each Datacentre	4

The Cloudy-GSA is used for scheduling the load in cloud. It takes 25 particles in the search space for finding the best position of 10 cloudlets on 8 virtual machines. The dynamic allocation of cloudlets on VMs takes place. The parameters are defined in the simulator statically. The cloudlets and VMs include the features and characteristics provided by the system like MIPS, bandwidth, transfer cost, execution cost are used for the calculation of the total time for computation. The proposed Cloudy-GSA utilizes the entire functionality of the system having complexity of O (n^2). The entire results are judged on a random large iteration set of values from 10 to 100 and 100 to 1000. The comparison in the table is for different runs. For each run of the iteration, the transfer time and total cost is taken several times. The average of the different runs for iteration is computed and the resultant value (average) is taken as final result for transfer time and for total cost of computation. The iterations are determined as after specific iterations the algorithm converges. Thus, the results produced would be similar to the existing for larger set of iterations. This has been performed for all the existing and proposed Cloudy-GSA algorithms. The convergence and statistical analysis of the algorithms is discussed further.

Figure 4 and 5 show the existing heuristic algorithms with the proposed Cloudy-GSA approach over an iteration set of values for transfer time incurred in load scheduling.

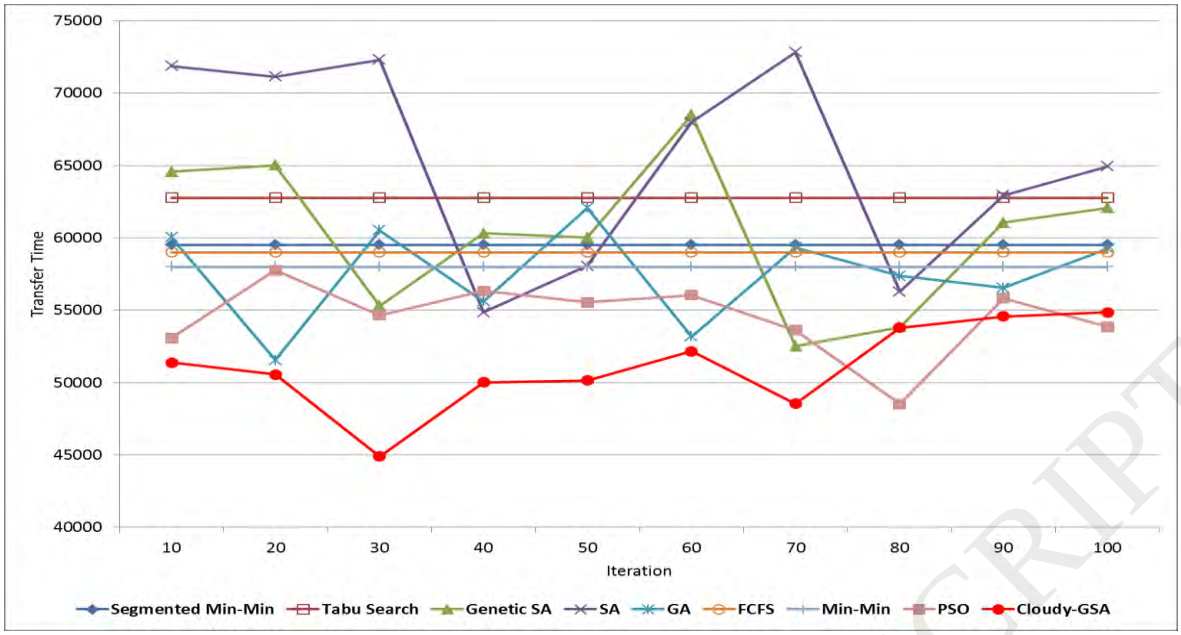


Figure 4. Transfer Time of existing and proposed Cloudy-GSA algorithm for scheduling over 10 to 100 iterations

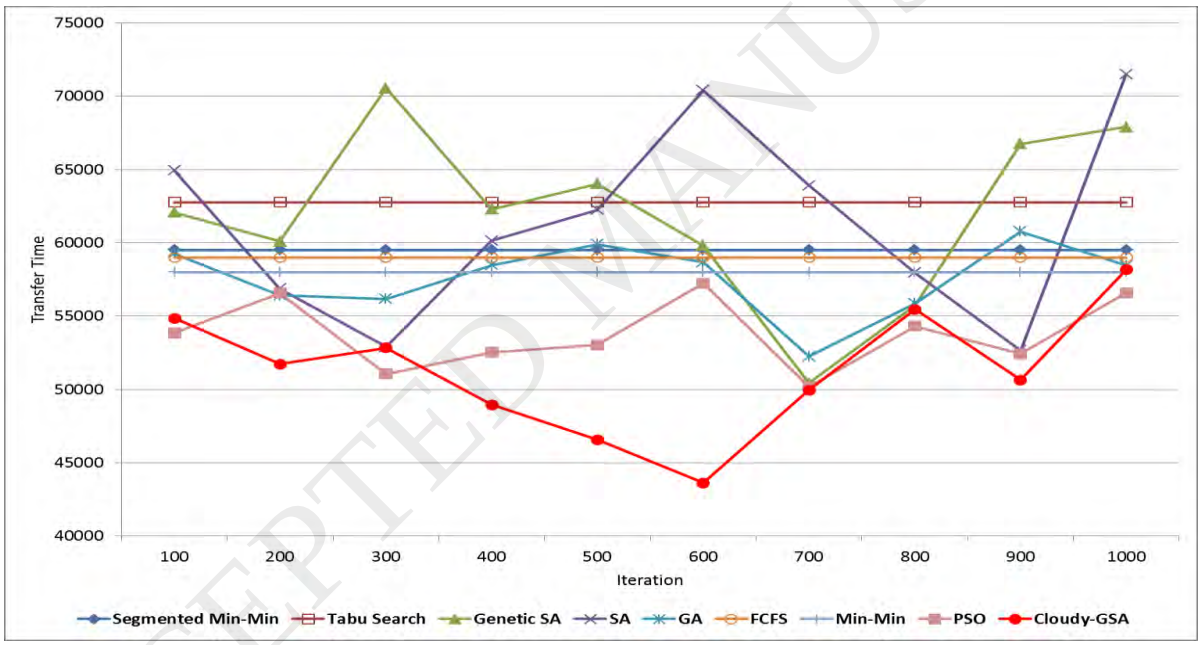


Figure 5. Transfer Time of existing and proposed Cloudy-GSA algorithm for scheduling over 100 to 1000 iterations

Figure 6 and 7 illustrate the total cost of computation for the proposed Cloudy-GSA optimization approach with the existing load scheduling algorithm over a set of iterations.

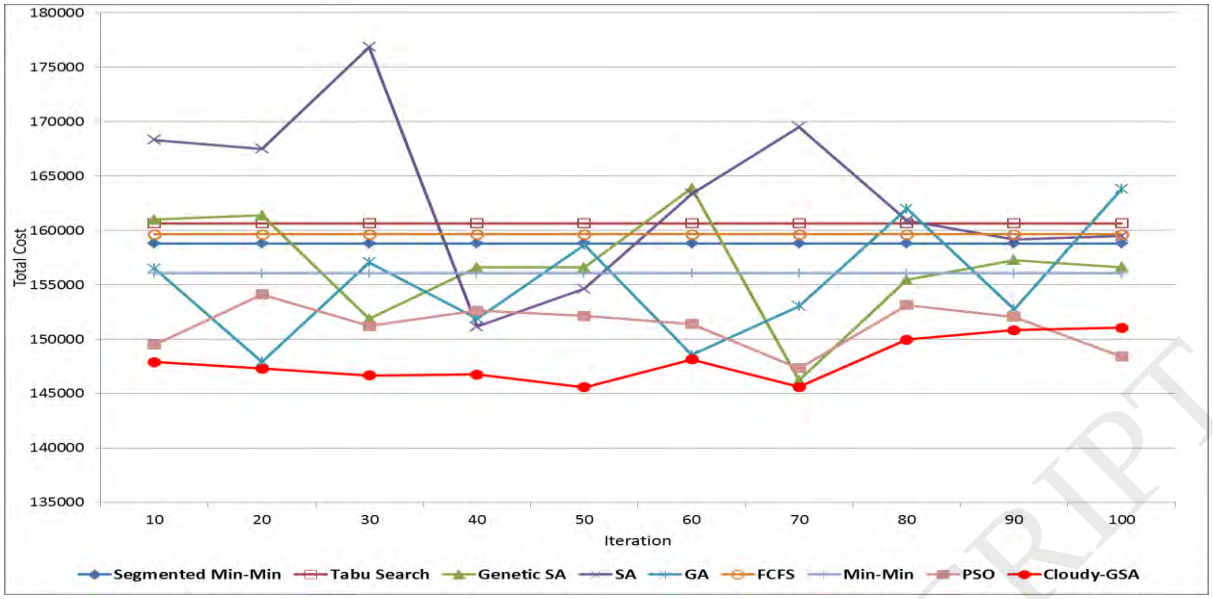


Figure 6. Total Cost of existing and proposed Cloudy-GSA algorithm for scheduling over 10 to 100 iterations

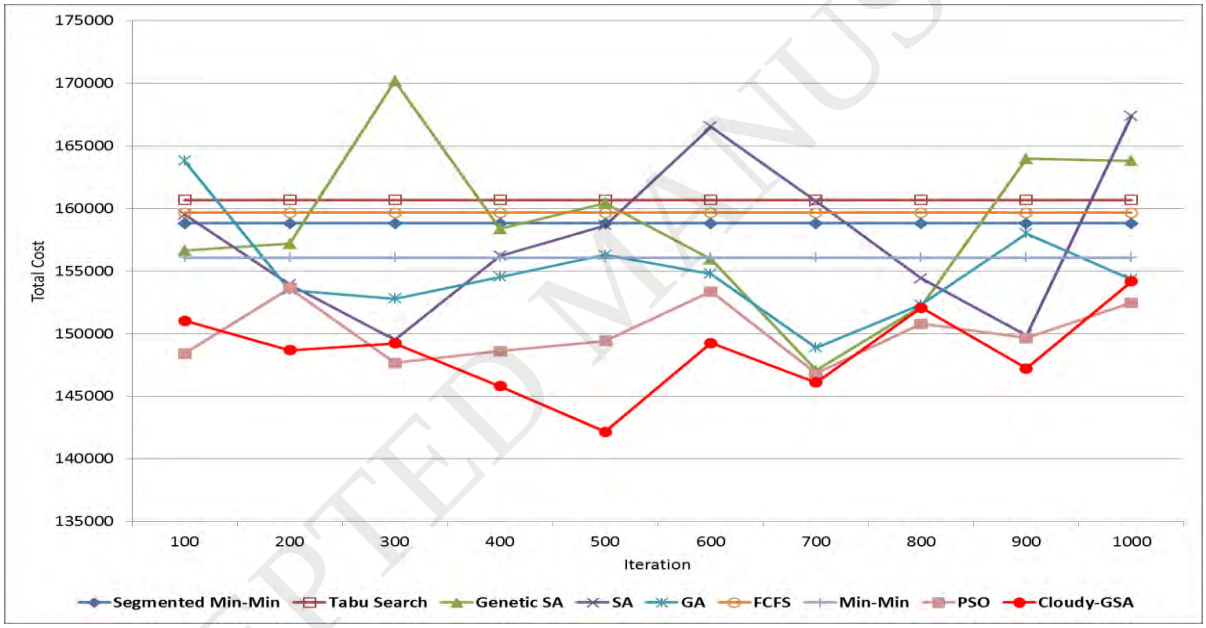


Figure 7. Total Cost of existing and proposed Cloudy-GSA algorithm for scheduling over 100 to 1000 iterations

Thus, the results state that the proposed Cloudy-GSA (gravitational search algorithm) for load scheduling in cloud reduces the transfer time and the total cost of computation than the existing algorithms. The results of proposed approach vary from iteration to iteration due to their heuristic nature and random function. In some iteration proposed Cloudy-GSA does not produce better results due to the randomized nature of the algorithm. However, the consistency of the algorithm is achieved by the convergence and the statistical analysis of the results set.

5.1. Convergence Analysis

The convergence analysis is performed for the heuristic simulation as finding the near optimal solution is tough. The converging behavior of the algorithm is analyzed mathematically for the set of iterations. We measured the mean of normalized displacement from the initial positions at different iterations on the result set of transfer time and total cost. The Mean of Normalized Displacement for transfer time and total cost at t^{th} iteration is

computed by ct_i^t , the normalized cloudlet transfer time or total cost at t^{th} iteration. Here, ct_i^0 refers at 0^{th} iteration and n is number of data vectors from 1 to 19 as:

$$= \frac{\sum_{i=1}^{n} ||ct_i^t - ct_i^0||_2}{n} \tag{23}$$

The results are depicted in figures 8 and 9. It is observed that the Cloudy-GSA approach converges within the thirteen iterations along with other heuristic algorithms in the cloud. We have analyzed the results of the other algorithms based on the transfer time and total cost of computation from figures 4 to 7.

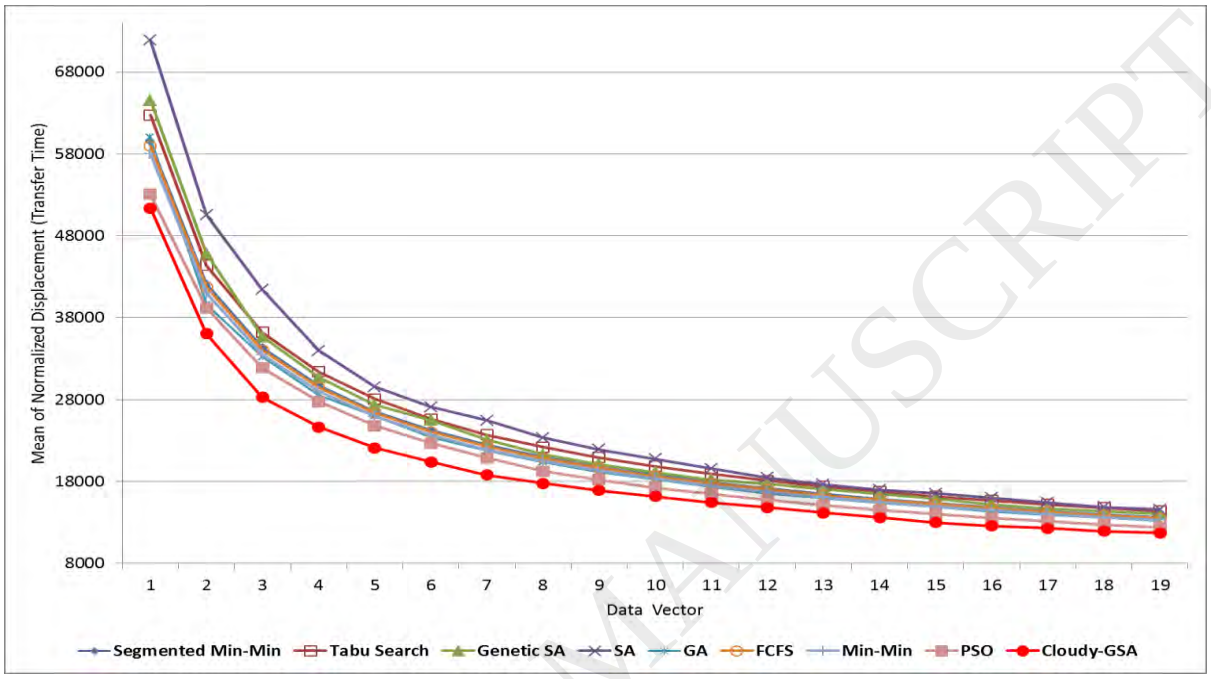


Figure 8. Mean of Normalized Displacement convergence analysis based on Transfer Time for the result set

Secondly, we have also calculated the mean of pairwise distances at different iterations on the result set of the transfer time and the total cost from tables. It is based on the ct_i^t and ct_j^t on the normalized transfer time or total cost data at i^{th} and j^{th} iterations respectively where n is number of data vectors ranging from 1 to 19. The Mean of Normalized Pairwise Distance at t^{th} iteration is:

$$= \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} [i \neq j] ||ct_i^t - ct_j^t||_2}{n^2 - n} \tag{24}$$

The results of normalized pairwise distance are illustrated in figures 10 and 11 for transfer time and total cost respectively. It is observed that the proposed Cloudy-GSA and the existing algorithms converge within the nine iterations. The algorithms minimize the mean of the pairwise distance between the iterations to converge the results and based on these the behaviour is predictable for large iteration set of values in case of large number of cloudlets and virtual machines.

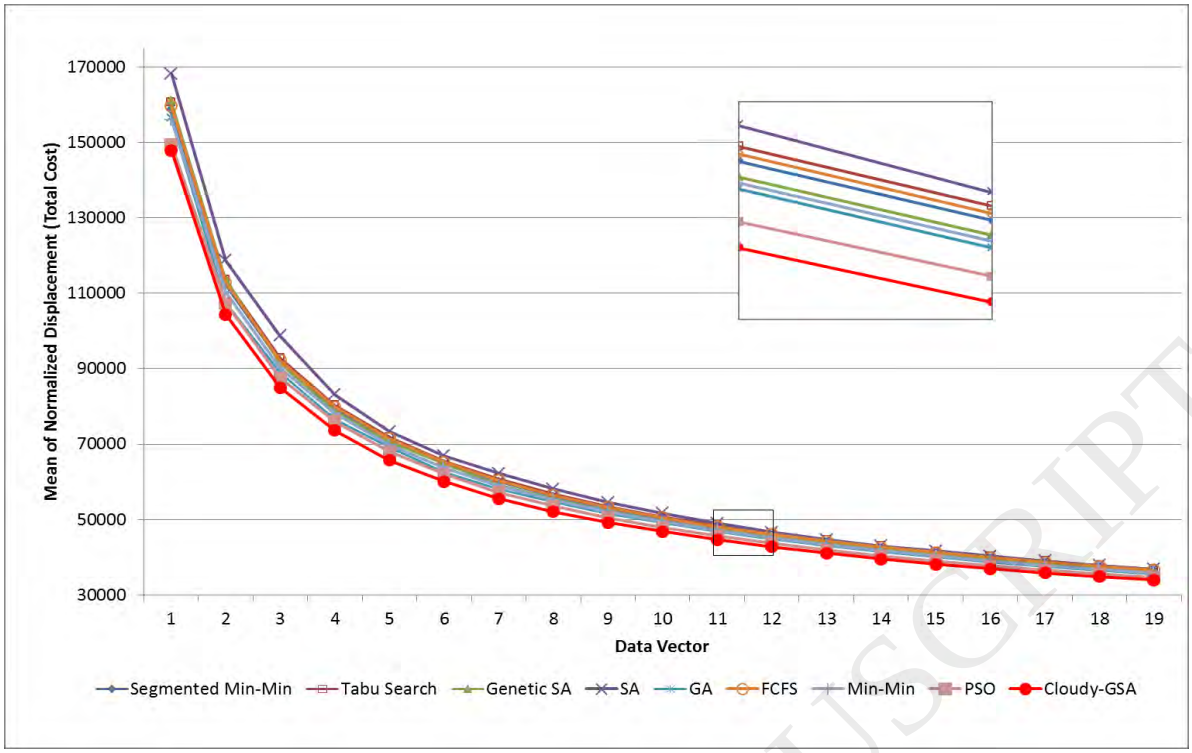


Figure 9. Mean of Normalized Displacement convergence analysis based on Total Cost for the result set

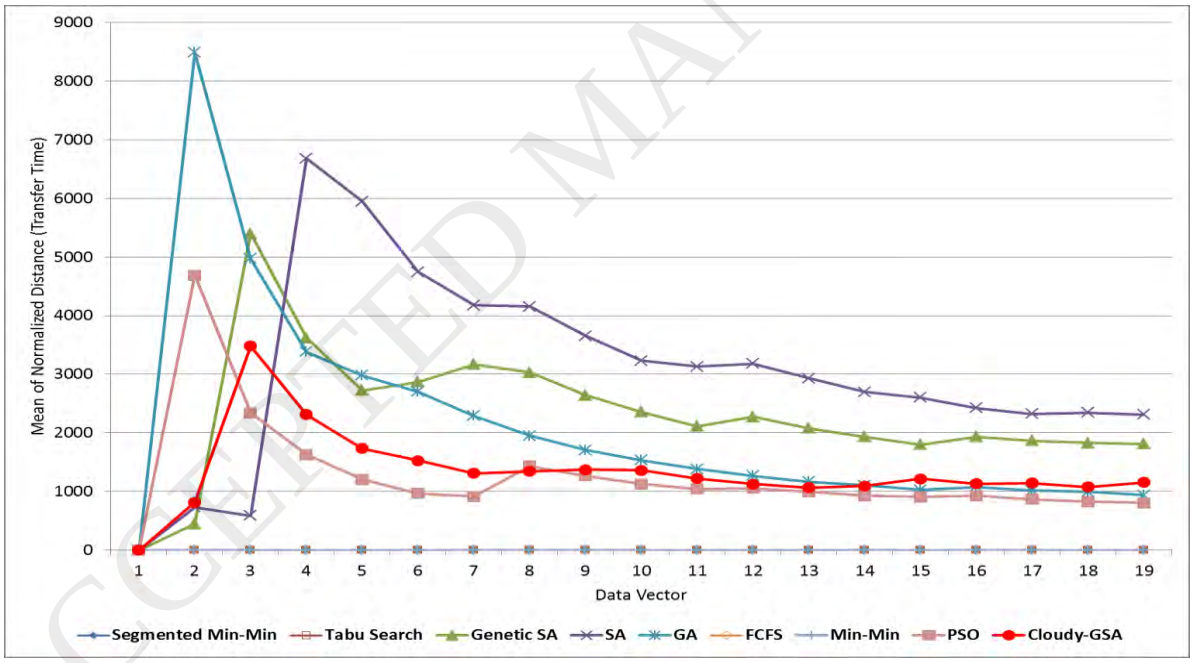


Figure 10. Mean of Normalized Pairwise Distance convergence analysis based on Transfer Time

5.2. Statistical Analysis

The statistical analysis of the proposed Cloudy-GSA approach and the existing scheduling algorithms provide us with the proof of efficiency of the proposed algorithm over others. The metrics of mean, standard deviation, minimum value and maximum value are specified and computed for the algorithms based on the result set of transfer time and the total cost of computation. The metrics of the transfer time are provided in the table 2 for the transfer cost of the cloudlets over the virtual machines. Table 3 specifies the total cost incurred by the cloudlets for executing on the virtual machines.

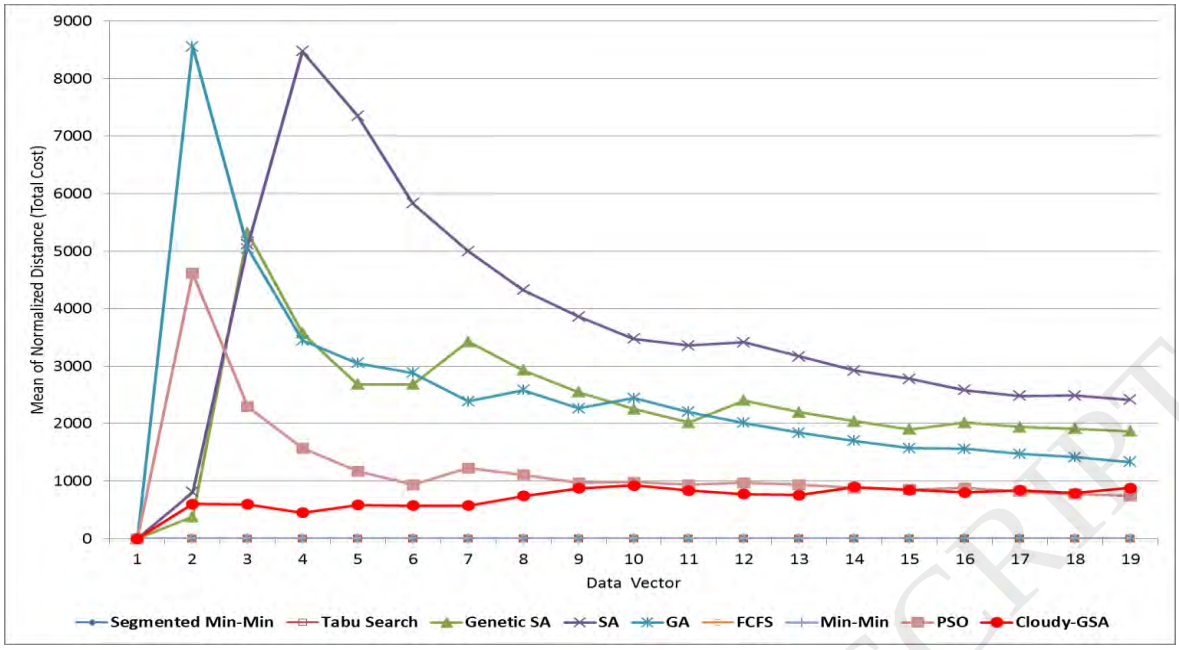
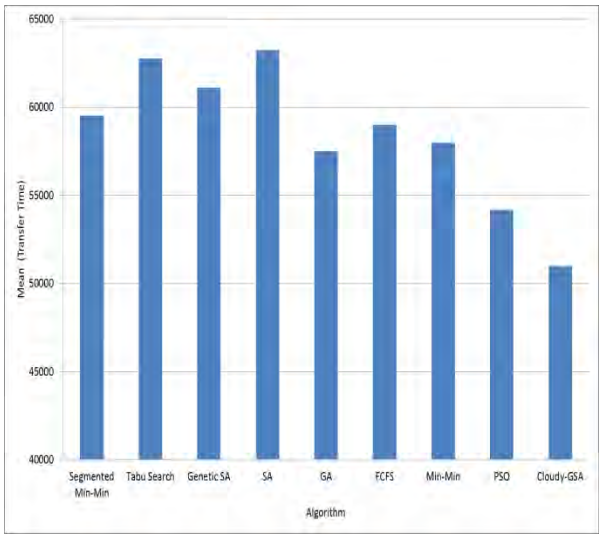


Figure 11. Mean of Normalized Pairwise Distance convergence analysis based on Total Cost

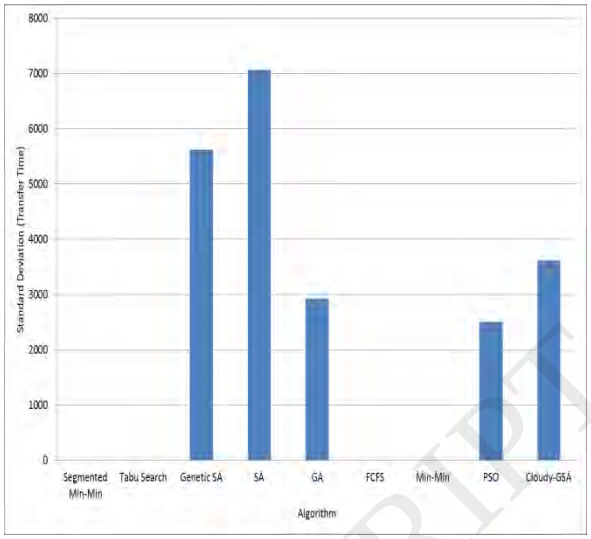
Table 2: Statistical Analysis of Transfer Time

	Segmented Min- Min	Tabu Search	Genetic SA	SA	GA	FCFS	Min-Min	PSO	*Cloudy-GSA*
Mean	59524.76	62768.22	61098.7632	63255.019	57502.225	59000	58000	54170.38	50995.59
Standard Deviation	7.48E-12	0	5617.15397	7068.0319	2926.831	0	0	2505.163	3624.817
Maximum	59524.76	62768.22	70559.41	72822.8	62075.51	59000	58000	57759.06	58166.84
Minimum	59524.76	62768.22	50432.35	52661.52	51562.33	59000	58000	48525.77	43632.41

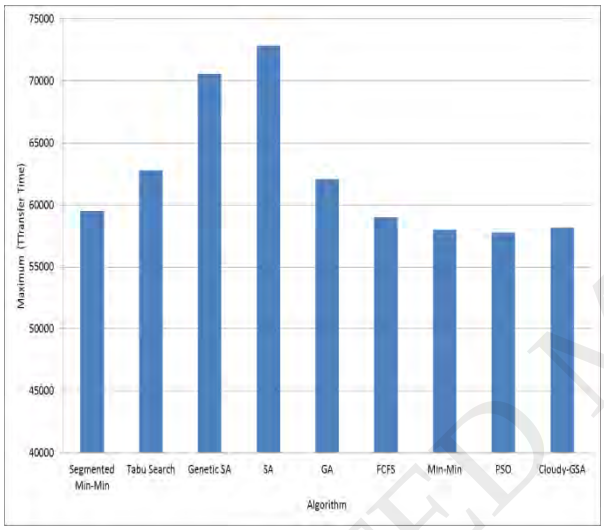
Figures 12 and 13 illustrate the mean, standard deviation, minimum and maximum for all the load scheduling algorithms in the cloud computing for transfer time and total cost respectively. The mean and standard deviation depict the superiority of the proposed Cloudy-GSA for load scheduling in cloud computing over the existing scheduling algorithms like Segmented Min-Min, Tabu Search, Genetic Simulated Annealing, Simulated Annealing, Genetic algorithm, Min-Min, FCFS, and Particle Swarm Optimization. The proposed algorithm outperforms in terms of the maximum exploitation of the resources on the virtual machines by the cloudlets.



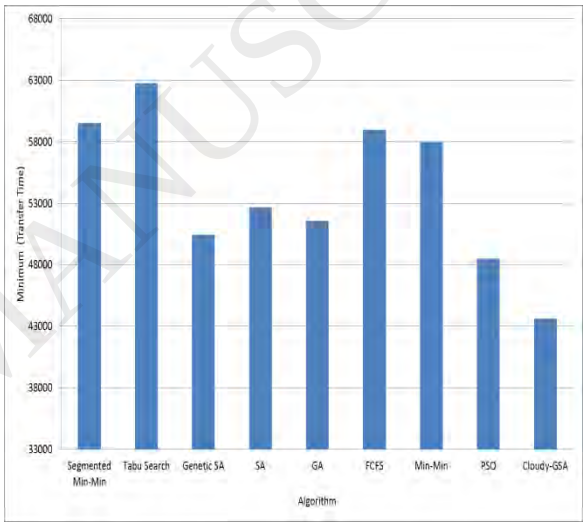
(a) Mean



(b) Standard Deviation



(c) Maximum

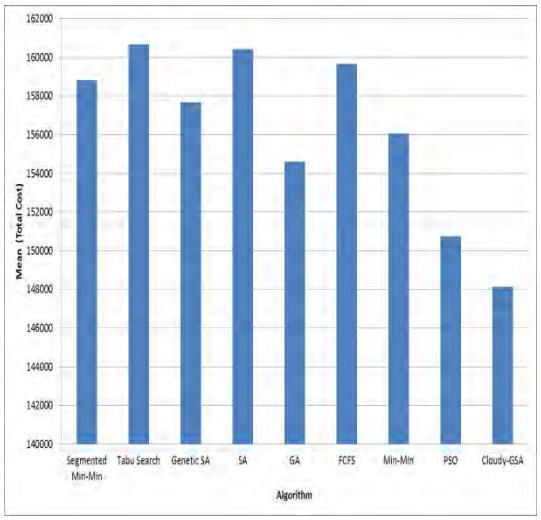


(d) Minimum

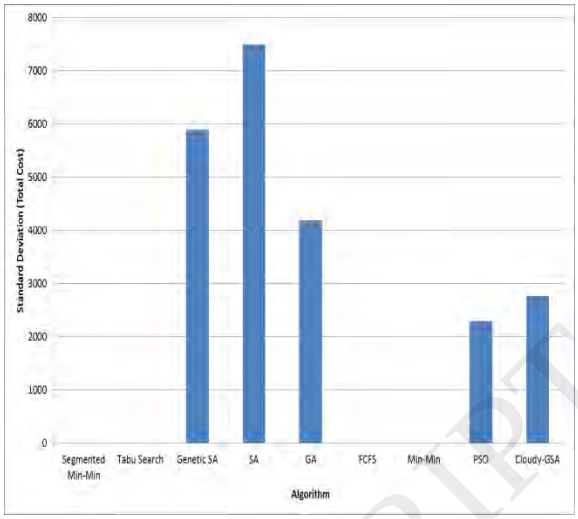
Figure 12. Statistical Analysis of transfer time of proposed Cloudy-GSA and existing algorithms.

Table 3: Statistical Analysis of the Total Cost

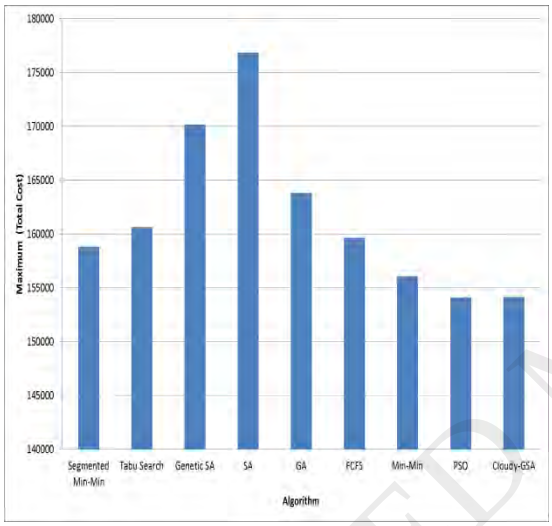
	Segmented Min- Min	Tabu Search	Genetic SA	SA	GA	FCFS	Min-Min	PSO	*Cloudy-GSA*
Mean	158807	160665.4	157687.8	160423.0	154617.62	159662.3	156066.6	150759.4	148137.5
Standard Deviation	0	2.99E-11	5887.616	7494.170	4192.3283	2.99E-11	2.99E-11	2290.429	2764.070
Maximum	158807	160665.4	170164.1	176870.7	163806.78	159662.3	156066.6	154132.5	154176.8
Minimum	158807	160665.4	146216.5	149520.3	147935.8	159662.3	156066.6	146814.2	142157.8



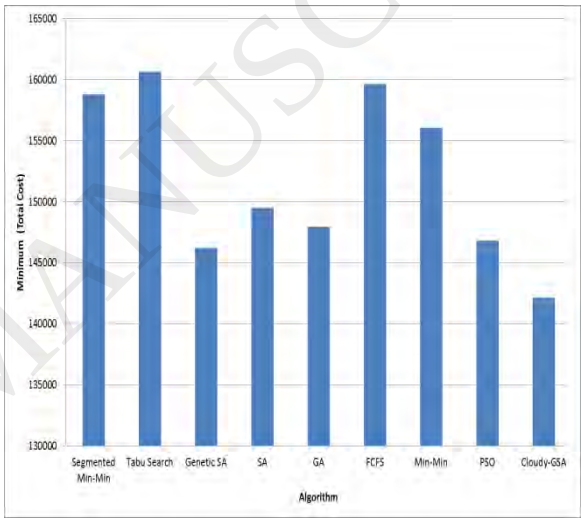
(a) Mean



(b) Standard Deviation



(c) Maximum



(d) Minimum

Figure 13. Statistical Analysis of total cost of proposed Cloudy-GSA and existing algorithms

The Cloudy-GSA optimization method based on principle of gravitational search algorithm provides greater load scheduling than the previous ones and reduces the cost of computation. The stochastic algorithms perform standard deviation. The proposed Cloudy-GSA provides more movement than PSO within the particles. The algorithm converges after a specific set of iterations. This approach can be used for faster processing of the user requests of data or information by the server in the cloud in the real world scenario. It involves faster allocation, storage and retrieval of the data.

6. Conclusion And Future Work

This paper elaborated the major problem of load scheduling in the cloud computing environment between cloudlets and VMs. The meta-heuristic based swarm intelligence technique for load scheduling in the cloud is discussed. The concepts of load scheduling algorithms with proposed Cloudy-GSA on load scheduling has been explained. The proposed Cloudy-GSA approach reduces the transfer time and total cost of the system along with maximum utilization of VMs. This has been achieved using the fitness values of particles and force acting on the particles in search space. At a bigger context, proposed approach provides higher cost optimization than the existing Segmented Min-Min, Tabu Search, Genetic Algorithm, Simulated Annealing, Genetic Simulated Annealing, FCFS, Min-Min and Particle Swarm Optimization scheduling algorithms. The results of

the proposed approach have been compared with the existing discussed scheduling algorithms which are heuristic and non-heuristic in nature in a detailed manner on a large set of iterations. Thus, the proposed Cloudy-GSA optimization method generates far better results in terms of transfer time and total cost of execution time based on the convergence statistical analysis. The future work aims to minimize the total cost by using improved fitness function considering other parameters for better minimized results in the cloud computing environment, based on swarm intelligence to further reduce the total cost.

References

[1] Buyya R., Pandey S., Vecchiola C.: Cloudbus toolkit for market-oriented cloud computing. In CloudCom '09: Proceedings of the 1st International Conference on Cloud Computing, vol. 5931, pp.: 24–44, LNCS Springer (2009).

[2] http://en.wikipedia.org/wiki/Cloud_computing, 19:43, 23 February 2017

[3] http://en.wikipedia.org/wiki/Load_balancing_(computing), 19:58, 23 February 2017

[4] Vecchiola C., Kirley M., Buyya R.: Multi-objective problem solving with offspring on enterprise clouds. 10th International Conference on High-Performance Computing in Asia-Pacific Region, pp. 132–139, IEEE (2009).

[5] Chaudhary D., Kumar B.: Analytical study of load scheduling algorithms in cloud computing. IEEE International Conference on Parallel, Distributed and Grid Computing, pp.: 7 – 12, IEEE (2014).

[6] Khiyaita A., Bakkali E., Zbakh M., Kettani D.E.: Load Balancing Cloud Computing: State of Art. 2012 National Days of Network Security and Systems (JNS2), pp.: 106-109 IEEE (2012).

[7] Chaudhary D., Chhillar R.S.: A New Load Balancing Technique for Virtual Machine Cloud Computing Environment. International Journal of Computer Applications, vol.69, issue 23, pp.: 37-40 (2013).

[8] Yu J., Buyya R., Ramamohanarao K.: Workflow Scheduling Algorithms for Grid Computing. Metaheuristics for Scheduling in Distributed Computing Environments, vol. 146, pp.: 173–214. Springer Heidelberg (2008).

[9] Zhang L., Chen Y., Sun R., Jing S., Yang B.: A task scheduling algorithm based on pso for grid computing. International Journal of Computational Intelligence Research, vol. 4, no. 1, pp.: 37-43, IJCIR (2008).

[10] Tsai C.W., Joel J., Rodrigues P. C.: Metaheuristic Scheduling for Cloud: A Survey. IEEE Systems Journal, vol. 8, no. 1, March, pp.: 279-291, IEEE (2014).

[11] Chaudhary D., Kumar B.: An analysis of the load scheduling algorithms in the cloud computing environment: A survey. IEEE 9th International Conference on Industrial and Information Systems, pp.: 1 – 6, IEEE (2014).

[12] Pacini E., Mateos C., Garino C. G.: Distributed job scheduling based on Swarm Intelligence: A survey. Computers and Electrical Engineering, vol. 40, pp.: 252–269, Elsevier Ltd (2013).

[13] Braun T.D., Siegel H.J., Beck N.: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems[1]. Journal of Parallel and Distributed Computing, vol. 61, pp.: 810-837, Elsevier (2001).

[14] Yildiz B.S., Lekesiz H., Yildiz Ali R.: Structural Optimization Using Meta-Heuristic Algorithms In Automotive Industry. The 17th International Conference on Machine Design and Production, July 12 - July 15 (2016).

[15] Kiani M., Yildiz Ali R.: A Comparative Study of Non-traditional Methods for Vehicle Crashworthiness and NVH Optimization. Archives of Computational Methods in Engineering, vol. 23, issue 4, pp.: 723-734, Springer (2016).

[16] Yildiz Ali R.: Comparison of evolutionary based optimization algorithms for structural design optimization. Engineering Applications of Artificial Intelligence, vol. 26, issue 1, pp.: 327-333, Elsevier (2013).

[17] Garg S. K., and Buyya R.: Network CloudSim: Modelling Parallel Applications in Cloud Simulations. 4th IEEE/ACM International Conference on Utility and Cloud Computing (UCC), Melbourne, Australia, (2011).

[18] Kennedy J., Eberhart R.: Particle swarm optimization. IEEE International Conference on Neural Networks, vol. 4, pp.: 1942–1948, IEEE (1995).

[19] Tasgetiren M. F., Liang Y.C., Sevkli M., Gencyilmaz G.: A particle swarm optimization algorithm for makespan and total flow time minimization in the permutation flowshop sequencing problem. European Journal of Operational Research, vol. 177(3), pp.:1930–1947, March (2007).

[20] Yoshida H., Kawata K., Fukuyama Y., Nakanishi Y.: A particle swarm optimization for reactive power and voltage control considering voltage stability. International Conference on Intelligent System Application to Power System, pp.: 117–121, IEEE (1999).

[21] Zavala A. E. M., Aguirre A. H., Diharce E. R. V., Rionda S. B.: Constrained optimisation with an improved particle swarm optimisation algorithm. International Journal of Intelligent Computing and Cybernetics, vol. 1(3), pp.:425–453, Springer (2008).

[22] Mathiyalagan P., Dhepthie U., Sivanandam S.: Grid scheduling using enhanced PSO algorithm. International Journal of Computer Science Engineering, vol. 02(02), pp.:140–145, IJCSE (2010).
[23] Liu H., Abraham A., Hassanien A.: Scheduling jobs on computational Grids using a fuzzy particle swarm optimization algorithm. Future Generation Computer Systems, vol. 26(8), pp.:1336-1343, Elsevier (2010).
[24] Izakian H., Ladani B., Abraham A., Snasel V.: A discrete particle swarm optimization approach for Grid job scheduling. International Journal of Innovative Computing Information Control, vol. 6(9), pp.:4219-4233 (2010).
[25] Kang Q., He H.: A novel discrete particle swarm optimization algorithm for meta-task assignment in heterogeneous computing systems. Microprocessor and Microsystems, vol. 35(1), pp.:10–17, Elsevier (2011).
[26] Pandey S., Buyya R. et al., "A Particle Swarm Optimization based Heuristic for Scheduling Workflow Applications in Cloud Computing Environments", 24th IEEE International Conference on Advanced Information Networking and Applications, pp.: 400-407, IEEE (2010).
[27] Kumar D., Raza Z.: A PSO Based VM Resource Scheduling Model for Cloud Computing. IEEE International Conference on Computational Intelligence & Communication Technology (CICT), pp.: 213-219, IEEE (2015).
[28] Rashedi E., Nezamabadi-pour H., Saryazdi S.: GSA: A Gravitational Search Algorithm. Information Sciences, vol. 179, pp.: 2232–2248, Elsevier (2009).
[29] Karagöz S., Yıldız Ali R.: A comparison of recent metaheuristic algorithms for crashworthiness optimisation of vehicle thin-walled tubes considering sheet metal forming effects. International Journal of Vehicle Design, vol. 73 (1/2/3), pp.:179-188, InderScience (2017).
[30] Yıldız Ali R, Kurtuluş E., Demirci E., Yıldız B.S., Karagöz S.: Optimization of thin-wall structures using hybrid gravitational search and Nelder-Mead algorithm. Materials Testing, vol. 58 (1), pp.:75-78, Carl Hanser Verlag (2016).
[31] Rashedi E., Nezamabadi-pour H., Saryazdi S.: Filter modeling using gravitational search algorithm. Engineering Applications of Artificial Intelligence, vol. 24, pp.: 117–122, Elsevier (2011).
[32] Yildiz B.S., Lekesiz H., Yildiz Ali R.: Structural design of vehicle components using gravitational search and charged system search algorithms. Materials Testing, vol. 58 (1), pp.:79-81, Carl Hanser Verlag (2016).
[33] Wong K. C., Peng C., Li Y., Chan T. M.: Herd Clustering: A synergistic data clustering approach using collective intelligence. Applied Soft Computing, vol. 23, pp.: 61-75, Elsevier (2014).

APPENDIX

The simulation parameters for the proposed Cloudy-GSA and the existing algorithms are as:

Parameter Description	Values
Cloudy GSA	
(x, y) co-ordinates	0 – 7
Number of particles	25
Number of VMs	8
Number of Cloudlets	10
Max_Iter	10-100 and 100– 1000
G_0	1
ram	2048
Bandwidth	10000
Storage	10000
Cloudlets	10
Datacentre Count	2
VM in each Datacentre	4
Genetic Algorithm	
Virtual Machines	8
Cloudlets	10
VM in each Datacentre	4
Particle Count	25

Dimensions in position of each particle	Cloudlets
Probability of Crossover	0.04
α	20
Probability of Mutation	0.04
Genetic Simulated Annealing	
Probability of Mutation	0.04
Probability of Mapping	0.5
Cooling rate	0.9 of current value
Temperature	100000.0
Min-Min Algorithm	
No of VMs	8
No of tasks	10
Simulated Annealing	
Probability of Mapping	0.5
Cooling rate	0.9 of current value
Temperature	100000.0
Tabu Search	
Hops initialized	0
Limit Hops	1000
Segmented Min-Min Algorithm	
Segments	3 (Minimum, Average and Maximum)
PSO	
Number of particles	25
Number of VMs	8
Number of Cloudlets	10
α	0.4