JOURNAL OF CRITICAL REVIEWS
ISSN- 2394-5125 VOL 7, ISSUE 13, 2020

HYBRIDIZATION OF METAHEURISTIC ALGORITHMS FOR LOAD SCHEDULING IN CLOUD COMPUTING ENVIRONMENT

[1]J. ROBERT ADAIKALARAJ, [2]T. VENGATTARAMAN
[1]Assistant Professor, Department of Computer Applications, St. Joseph's College of Arts and Science (Autonomous), Cuddalore
[2]Department of Computer Science, Pondicherry University, Puducherry, India
[1]j.rubertraj@gmail.com, [2]vengattaramant@gmail.com

Abstract

Load scheduling defines the process of offering, assigning and balancing the load (tasks/ cloudlets to the virtual machines) in the cloud system effectively. The major intention is to minimize the transfer time and the total cost incurs in the load scheduling of the system. The traditional load scheduling techniques necessitate massive amount of resources and mechanisms, which are dynamic in processing, thereby increases the response time, waiting time and the total computation cost. This paper presents an efficient load scheduling technique called hybridization of binary tree optimization with Gravitational Search Algorithm (BTO-GSA) algorithm to minimize the computation time. The total computation time cost comprises of execution cost and transferring cost. It operates on hybrid Splitting Point Selection technique based GSA to search the optimal positions of the particles in the search space. The use of BTO algorithm depends upon the mathematical tree subject and enhances the outcome and searching speed by continuously eliminates the portions of the search space with minimum fitness for minimizing and purifying the search space. The BTO-GSA model has been implemented using CloudSim simulator and a detailed comparative result takes place under several aspects. The simulation outcome indicated that the BTO-GSA algorithm has offered superior performance over the compared methods in a significant way.

Keywords: Binary tree optimization, CloudSim, Cloud computing, Load scheduling, Swarm intelligence

1. Introduction

Nowadays, Cloud Computing (CC) has been developed rapidly in the field of distributed and grid computing by applying virtualization models. Here, the term computing means the functions which are executed in the virtual machines (VM) of a system in order to enhance the working function of a device. CC used here is treated as a repository of the resources that enables the individuals with massive potential as well as processing facilities such as memory, processing, extraction, and information retrieval about the heterogeneous operations to provide best resources for the users. CC is relied on pay-as-you-go or pay-as-you-use method for the resources consumption. The CC offers resources, task scalability, timely resource implementation, dynamic maintenance, fault tolerance and interoperability of resources. Also, it is helpful in allocating the tasks to VMs in a dynamic manner. This dynamic allocation can be attained using load scheduling process in an optimal fashion. Hence, it is mainly computed to accomplish maximum throughput, minimum implementation and waiting time, lower transfer time, as well as least computational cost.

Load scheduling is mainly applied to perform of the operations like task allocation, providing and management of load or tasks to the VM in the cloud effectively. The major aim of this model is to limit the transferring duration as well as the overall cost involved in the load scheduling system. This action has been processed under the application of diverse scheduling methods. These scheduling models are used accordingly to the static and dynamic techniques. Furthermore, it is divided into heuristic and non-heuristic methodologies. The meta-heuristic approach is a mandatory objective while performing the load scheduling process with the help of search framework. [1] expanded the problem solving technique with massive objectives of the cloud. It offers the conclusion for the applied problem by employing optimization criteria. It depends upon the SI models, which simulates the real nature of the swarms. The collection of objects, particles, ants and so forth follows the approach of placing the food that is applied for food discovery or optimal solution. The allocation of load in a cloud is one of the major complexities which have to be solved by applying effective models instead of using previous methods like Segmented Min-Min, Tabu Search, Simulated Annealing (SA), Genetic Algorithm (GA), particle swarm optimization (PSO), and so on. [2,22-30].

[3] pointed that the structure of cloud withstands 2 modules like Deployment model and Service model as shown in Fig. 1. Initially, deployment model states the location of resource maintenance with the organized architecture. Also, clouds are differentiated on the basis of metrics as Private, Public, Community and Hybrid Cloud. Secondly, service model defined the class of available resources for the people. Software as a Service (SaaS) is the application which can be applied by standard interfaces, Platform as a Service (PaaS) is defined as the task and deploying environments. At last,

Infrastructure as a Service (IaaS) is the infrastructure relied resources like memory space, networking ability and computational power.

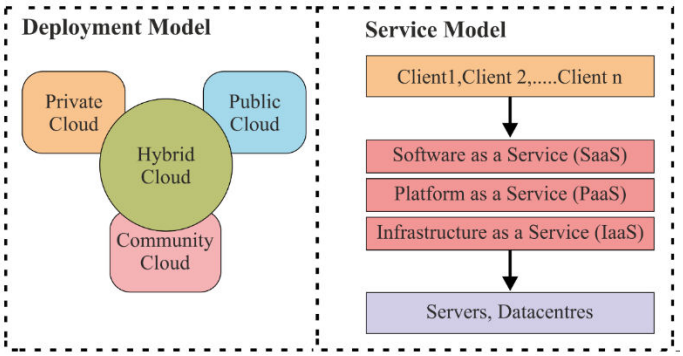

Fig. 1. Service and Deployment Models of CC

There are several benefits offered in these models like location independence, minimum cost and advertising time, extensive network access, resilient computing, maximum computation power, virtualization, flexible and scalable workout potentials as presented by Chaudhary et al. [4]. These operations applied maximum number of resources to resolve the computational difficulty, cost, transfer time and response time. It is useful in the dynamic resources allocation for diverse heterogeneous tasks at various locations. The load scheduling has to allocate the load on the basis of heavily loaded VM to minimum loaded VM. Yu et al. [5] offered the workflow based scheduling models across the nodes present in a grid. It depends upon the process workflow of nodes which might be varied.

A swarm based scheduling is relied on the PSO approach [6] has been proposed in grid computing. The PSO applies the behavior of flock of birds to migrate from one node to other. The PSO approach applies the resources in an optimal manner. The meta-heuristic specifies a high level heuristic to find, produce, or select a heuristic approach with inaccuracy and partial data to offer the good solution for the issue. Tsai et al. [7] extended diverse metaheuristic scheduling models in the cloud. The investigation of load scheduling techniques is presented by Chaudhary et al. [8]. The SI relied frameworks offers maximum efficiency in scheduling when compared with alternate models of cloud. The swarm defines a collection of particles in the search space. Pacini et al. [9] implied a brief examination of swarm based techniques like ants, bees or birds for identifying the upcoming best position according to the models used to find the food. It provides higher utilization of resources. Additionally, meta-heuristic algorithm is extended under the application in structural optimization by Yildiz et al. [10]. The strength and scalability of discovering the future particle in optimization frameworks depends upon the meta-heuristic swarm based principles [11].

Garg et al. [12] defined the network based potentials to perform the load scheduling in the cloud system. Kennedy et al. [13] projected a PSO algorithm in an arithmetic content. It has been evolved from the nature of flock of birds in order to place the food source. This optimization models tends to exploit the resources with minimum computation cost. Tasgetiren et al. [14] employed this optimization method in flow shop scheduling to control the cloudlets in an effective manner and managing stability for reactive power as well as management of the voltage. Such strategies schedule the load correctly, however with expensive computation. The PSO scheduling has been utilized on the grids which depend upon the enhanced fitness measures as well as applied attributes in [15]. The fuzzy sets were also deployed to estimate the functions applied to perform the scheduling task. Kang et al. [16] pointed out that the discrete PSO technology offers resource allocation in heterogeneous platforms. It is based on the application of diverse operations; therefore, the difficulty of the model is maximum processing and minimum node exploitation. Pandey et al. [17] devised the PSO in workflow scheduling in CC. The tasks were allocated on VM according to the position of particles. The expense of computation has been minimized but application of the resources is not so optimal. In order to enhance the resource utilization at cheaper rate Kumar et al. [18] applied the improved PSO (IMPSO) method in CC platform.

This paper introduces a new load scheduling algorithm known as hybridization of binary tree optimization with Gravitational Search Algorithm (BTO-GSA) to lessen the computation time, which includes the execution cost and transferring cost. It operates on hybrid Splitting Point Selection technique based GSA for the identification of optimum positions of the particles in the search space. The application of BTO algorithm depends upon the mathematical tree subject, improvises the outcome and searching speed by incessantly eliminates the portions of the search space with

minimum fitness for minimizing and purifying the search space. The BTO-GSA model has been implemented using CloudSim simulator and the results are examined under several aspects.

2. The Proposed BTO-GSA Algorithm

The BTO-GSA algorithm is mainly focused on reducing the overall computational cost and allocates the cloudlets (load or task) to VM effectively. The computational cost is comprised with transfer and execution cost of the loads. The BTO-GSA model accomplishes maximum search space application as well as user convenience when compared with previous technologies. It depends upon bio-inspired SI algorithm, scientific and law of gravitation heuristic in CC. BTO-GSA is defined as the memory-driven techniques that save the optimal position value of particles that resolves the issues of classical GSA. The consecutive particle can be estimated on the basis of GA and GSA. It comprises a fitness function (FF) computation, Splitting Point Selection, new-gravitational constant function, new cost determination function, memory-relied velocity and position estimation. The cloudlets and VM metrics such as MIPS, bandwidth, implementation cost, and transferring cost were applied for the FF estimation. The flowchart of the BTO-GSA algorithm is depicted in Fig. 2.

In the CC platform, it performs the task of load allocation on VM. The data center gets $(\text{VMs})^{\text{Cloudlets}}$ possibilities of implementing the cloudlets on the corresponding VM. When 3 loads are performed on 2 VMs, then the possibility is $(2)^3$ that is 8.The M particles are described in d dimensional search space which has a maximum number of feasible solutions. Hence, a meta-heuristic algorithm is essential in Newton's law of gravitation in order to identify the particle's positions. Thus, the better solution for the cloudlets can be found under effective allocation of particles on the basis of BTO-GSA scheduling principle. This model offers the optimal position to the load in order to implement on VM. The particles M have been initiated with the application of CloudSim tool as given below:

$$\mathrm{M_i} = (\mathrm{m_i^1}, \mathrm{m_i^2}, \dots\dots, \mathrm{m_i^n}, \dots\dots\dots, \mathrm{m_i^d})$$

$$\forall i = 1\ to\ 25\ and\ n = 1\ to\ 10 \qquad (1)$$

The FF module is used to estimate the fitness value of the particles present in the searching space. The initial particle is applied in a random manner in the search space after selecting the future particle according to the best FF values. It is based on the bandwidth, MIPS, implementation cost as well as transfer cost of loads and VMs. Suppose $\mathrm{Ct_{exec}(M)_j}$ is the overall execution cost of particles assigned to estimate the VM resource PCj. It is evaluated by the addition of weights allocated on the nodes and implementation cost of cloudlets k is measured on VM resource $\mathrm{PC_j}$ under the mapping of particles M of each cloudlet that is assigned for all resources.

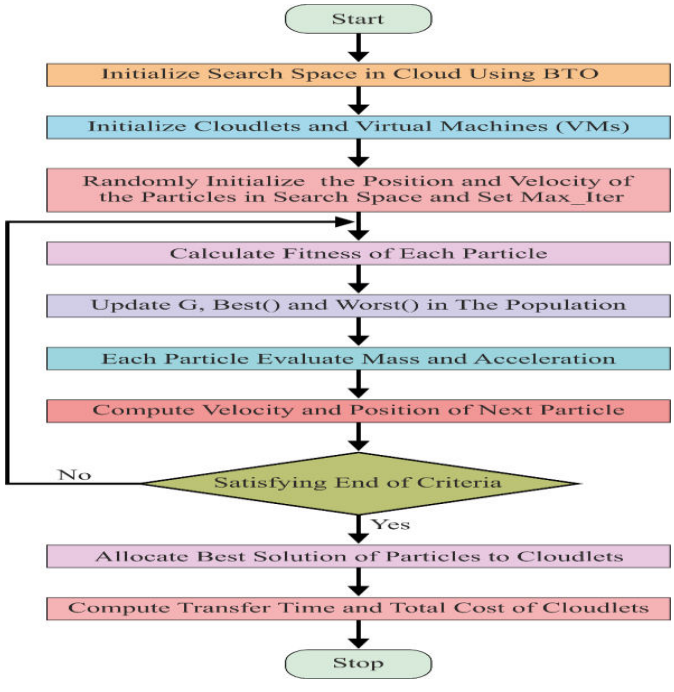

Fig. 2. Flowchart of Proposed BTO-GSA

Assume $\mathrm{Ct}_{\mathrm{trans}}(\mathrm{M})_{\mathrm{j}}$ is the total transfer cost among cloudlets which are declared to estimate VM resource PCj and few of them were not assigned in the mapping M. Consequently, the product of resultant size $\mathrm{e}_{\mathrm{l1,l2}}$ by means of 1 particle $\mathrm{l1} \in \mathrm{k}$ to task $\mathrm{l2} \in \mathrm{l}$ and the expense of transmission between the source in which l1 is mapped $(\mathrm{M(l1)})$ to alternate resources where l2 is mapped $(\mathrm{M(l2)})$. The maximum cost of data communication among 2 resources is depicted as $\mathrm{dM(l1), M(l2)}$ and these particles are reliant to one another. Here, more than 2 particles were executed on the similar resource, and communication cost becomes 0. Hence, the overall cost is summarized for all particles M with the application of execution as well as transfer cost and it is limited to compute the FF measures. Thus, the FF module is expressed as,

$$\mathrm{Ct}_{\mathrm{exec}}(\mathrm{M})_{\mathrm{j}} = \sum_{\mathrm{l}} \omega_{\mathrm{lj}}\,, \forall \mathrm{M}\ (\mathrm{l}) = j \tag{2}$$

$$\mathrm{cr}_{\mathrm{trans}}(\mathrm{M})_{\mathrm{j}} = \sum_{\mathrm{l1}\in\mathrm{T}} \sum_{\mathrm{l2}\in\mathrm{T}} \mathrm{d}_{\mathrm{M(l1),M(l2)}} * \mathrm{e}_{\mathrm{l1,l2}} \tag{3}$$

$$\forall \mathrm{M(k1)} = j \; and \; \mathrm{M(l2)} \neq \mathrm{j} \tag{4}$$

$$\mathrm{Ct}_{\mathrm{total}}(\mathrm{M})_{\mathrm{j}} = \mathrm{Ct}_{\mathrm{exec}}(\mathrm{M})_{\mathrm{j}} + \mathrm{Ct}_{\mathrm{trans}}(\mathrm{M})_{\mathrm{j}} \tag{5}$$

$$\mathrm{Cost}_{\mathrm{Total}}(\mathrm{M}) = \max\left(\mathrm{ct}_{\mathrm{total}}(\mathrm{M})_{\mathrm{j}}\right), \forall_{\mathrm{j}} \in \mathrm{M} \tag{6}$$

$$\mathrm{Minimize}(\mathrm{Cost}_{\mathrm{total}}(\mathrm{M}), \forall \mathrm{M}) \tag{7}$$

2.1. Binary Tree Optimization algorithm

The FF of every particle is created using the above-mentioned equations. The BTO algorithm is applied achieve effective search space exploitation. Tree optimization algorithm [19] utilizes a tree for reducing the landscape of search (i.e., search space) for improvising the effectiveness and searching speed. The removal process depends upon the identification and elimination of bad regions of landscape which have low fitness over other regions with improved fitness. It results to small search space and make the searching process easier as well as precise. It converges to a region of search space that is adequate for searching it precisely and possibly the preferable solution to the optimization issue. BTO algorithm applies a binary tree to divide the area of search space. A binary tree is defined as a tree in which all nodes are classified into 2 edges. Binary TBO technique is composed with diverse parts as defined in the following.

- **Vertical & Horizontal Split**: The initial landscape is classified into 2 regions, namely, vertical and horizontal. This selection could be interchanged among vertical and horizontal choices, that is equated as:

$$Split\ Orientation = \begin{cases} Vertical & if\ was\ Horizontal, \\ Horizontal & if\ was\ Vertical. \end{cases} \tag{8}$$

An alternate model for this procedure is to select the vertical or horizontal split with a probability $p(e.g., p = 0.5)$:

$$Split\ Orientation = \begin{cases} Vertical & if\ r \in [0, p], \\ Horizontal & if\ r \in [p, 1], \end{cases} \tag{9}$$

where r implies the uniform arbitrary value from the range $\lfloor 0,1 \rfloor$, where $r \sim U(0,1)$.

The BTO technique could be expanded for search space with maximum dimensions. For high-dimensional search spaces, a division has to be processed in single dimension and upcoming iteration applies other dimension to perform the splitting task.

- **Splitting Point Selection**: After selecting the orientation of split, the splitting point is present in the respective area. The randomly selected feasible points from the region is applied to select the point within a specific range, that is moderately away from edges of residual region, and eliminate the splitting in larger and smaller regions. The residual region is used to split $[R1; R2]$, and:

$$Splitting\ point \leftarrow U(L_1, L_2)\,, \tag{10}$$

where $L_1 = (30\% \times (R_2 - R_1)) + R_1$ and $L_2 = (70\% \times (R_2 - R_1)) + R_1$.

The values 30% and 70% are hyper-parameters, and $U(\alpha, \beta)$ implies the uniform random value from the range $[\alpha, \beta]$.

- **Update global best:** Every meta-heuristic algorithm stores the best solution in memory. Once the 2 classified regions are identified, the results attained imply an optimal conclusion. When the optimal solutions performs quite-well than global best found, the finally saved global best would be swapped into that is formulated as :

$$GB_i = \begin{cases} B_i & if\ B_i > GB_{i-1}, \\ GB_{i-1} & otherwise, \end{cases} \tag{11}$$

where B_i refers to the optimal solutions of 2 regions in the i-th iteration and GB denotes the global best solution identified.

- **Enters into a Region with a Probability:** This process can be assumed as the complexity of a model. Selecting a region for entering into is one of the significant processes as the alternate region would be eliminated from the search space that cannot be identified whenever required till the approach gets implemented. Initially, it is better to select an optimal conclusion. It is pointed that, the better solutions of regions might be considered as local solutions and it is far apart from the required global best. Besides, the global best is present in the region with poor solution and the sub-algorithm is not present in the target region. In order to eliminate the problem, a probabilistic selection method has to be employed. Hence, probabilistic decision leads to additional issue. Once the decision method selects a region which is not the preferable global best, the algorithm then converges to an outlier or local solution. To resolve this issue, the execution of the algorithm gets repeated for several rounds to make sure about the solution. . Note that this re-running approach is common in metaheuristic algorithms because of their randomness manner and un-sureness.

In order to compute the probabilistic decision, the problem type such as minimization or maximization is highly significant. Also, the main complexity is cost reduction, which requires lower function. At this point, a decision has been created:

$$P_1 = \begin{cases} 1 - \dfrac{B_1}{B_1 + B_2} & if\ B_1 > 0 \& B_2 > 0, \\ \dfrac{B_1}{B_1 + B_2} & if\ B_1 < 0 \& B_2 < 0, \\ 1 - \dfrac{BB_1}{BB_1 + BB_2} & otherwise, \end{cases} \tag{12}$$

where B_1 and B_2 are the best solutions of regions 1 and 2. P_1 implies the probability of getting into the region 1, where BB_1 and BB_2 can be represented by:

$$BB_1 = B_1 + |\min(B_1, B_2)| + 1,$$

$$BB_2 = B_2 + |\min(B_1, B_2)| + 1, \tag{13}$$

For making a simple and an easy implementation of this approach, the decision enters into the region as determined by:

$$Enter\ to\ = \begin{cases} Region\ 1\ if\ r < P_1, \\ Region\ 2\ if\ r \geq P_1, \end{cases} \tag{14}$$

where r shows a uniform arbitrary value such as, $r \sim U(0,1)$. While entering into the target region, the alternate region is eliminated from a search space and boundaries of landscape are extended by:

$$\begin{cases} y_{\text{mn}} = SP & if\ horizontal\ split, entering\ up \\ y_{\text{mx}} = SP & if\ horizontal\ split, entering\ down \\ x_{\text{mn}} = SP & if\ vertical\ split, entering\ right \\ x_{\text{mx}} = SP & if\ vertical\ split, entering\ left \end{cases} \tag{15}$$

where SP is the splitting point. The process involved in the BTO algorithm is provided in Algorithm 1.

Algorithm I: Pseudo code of BTO algorithm

Begin: Setting up parameter

While termination criteria is unsatisfied **do**

α=Size of region/Iteration count

If Number of Branches More **then**

For j from 1 to d **do**

For i from 1 to α **do**

splitting point randomly [L_1, L_2]

End For

End For

Else

Exchange split on proportions

For i from 1 to α **do**

splitting point randomly [L_1, L_2]

End For

End If

If Best of iteration is superior to Global best **then**

Update Global best (i.e. Assign Global best ← Best of iteration)

End If

For all regions do

Compare best results in each region and set chance of entering the region

End For

Discard outer portion from search space

End While

If Execute again **then**

Goto Begin of process

Else

Return the Global best

End If

2.2. Process involved in GSA

The FF values of a particle could be evaluated under the application of best (z) and worst (z) measures produce the mass M_{ati} (active), M_{psi} (passive) and M_{iti} (inertia). $fit_i(z)$ are named as the fitness rates of the particle for each iteration, in which t implies the overall count of particles.

$$M_{ati} = M_{psi} = M_{iti} = M_i, i = 1,2,3, \dots \dots, N \quad (16)$$

$$m_i(z) = \frac{fit_i(z) - worst(z)}{best(z) - worst(z)} \quad (17)$$

$$M_i(z) = \frac{m_i(z)}{\sum_{j=1}^{N} m_j(z)} \tag{18}$$

The load scheduling is assumed to be the minimization issue, thus the fitness has been estimated on the basis of best (z) and worst (z) measures.

$$best(z) = \frac{\min}{j \in \{1, \ldots., N\}} fit_j(z) \tag{19}$$

$$worst(z) = \frac{mc \downarrow x}{j \in \{1, \ldots., N\}} fit_j(r) \tag{20}$$

Where, $j \in \{1, \ldots., N\}$ is the particle j from N particles. The gravitational constant $G(z)$ acts as a force at a particular moment is described by particle's potential and applied to improve the movement among them. It is processed on the basis of G_0 (initial value), β(global best fitness), $pbest_i^z$ and time z.

$$G(z) = G_0 e\left(\beta \frac{1}{f(pbest_i^z)}\right) + 1 \tag{21}$$

$$\beta = f(gbest_i) \tag{22}$$

The force on the particle is treated on the mass and Euclidean distance $R_{ij}(z)$ in d dimensional search space among the particles M_i and M_j. A static function $rand_j$ belongs to the interval of $[0,1]$. Thus, the cumulative force which acts on a particle i corresponds to particle j at a particular instant t on the particles, as applied to identify the upcoming possible positions.

$$F_{ij}^d(z) = G(t) \frac{M_{pi}(z) \times M_{aj}(z)}{R_{ij}(z) + \varepsilon} (x_j^d(z) - x_i^d(z)) \tag{23}$$

$$R_{ij}(z) = ||M_i(z), M_j(z)||_2 \tag{24}$$

$$R(M_i, M_j) = R(M_j, M_i) = \sqrt{\sum_{i=1,j=1}^{N} (M_j - M_i)^2} \tag{25}$$

$$F_i^d(z) = \sum_{j=1, j \neq i}^{N} rand_j F_{ij}^d(z) \tag{26}$$

$$d_i^d(z) = \frac{F_i^d(z)}{M_{iti}(z)} \tag{27}$$

The simulation of particle i depends upon the force as well as inertial mass. The particle which has to be placed in the search space can be estimated with the application of velocity and particle. The velocity of subsequent round of a particle $v_i^a(z+1)$ is determined by applying the arbitrary uniform function ω which exists from $[0,1]$ along with velocity of a particle and acceleration. The arbitrary measures c_1, c_2 and r_1, r_2 with $pbest_i^z$ and $gbest^z$ assists in exploring the optimization between the particles. The position of subsequent particle $x_i^d(z+1)$ has chosen a particle position $x_i^d(z)$ and velocity $v_i^d(z+1)$ of upcoming particle.

$$v_i^d(z+1) = \omega \times v_i^d(z) + a_i^d(z) + [c_1.r_1.(pbest_i^z - x_i^d(z)) + c_2.r_2.(gbest^z - x_i^d(r))] \tag{28}$$

$$x_i^d(r+1) = x_i^d(r) + v_i^d(z+1) \tag{29}$$

The process is repeated till the maximum condition has been fulfilled. The global best position outcomes are provided to perform the load scheduling on VM in CloudSim simulator. Then, the cloudlets assigned to concern data centres. The overall computational cost is calculated under the application of parallel computing model. BTO-GSA limits the computation cost in the CC by offering efficient load scheduling of tasks. It makes use of maximum search space area and gives higher user satisfaction. This model provided higher application as well as minimum cost by using VMs.

3. Experimental Validation

The BTO-GSA algorithm is intended to optimize cost while scheduling the loads to the data centres by the cloudlets and VMs. The proposed BTO-GSA algorithm is simulated using CloudSim tool. A set of methods used for comparison are PSO, Cloudy GSA and LIGSA-C techniques [20, 21]. A series of experimental analysis takes place under the existence of 10, 15 and 20 cloudlets. The execution of the algorithm is conducted in the iterations of 10 to 1000.

3.1. Result analysis

Table 1 and Fig. 3 have provided a detailed comparison of the results offered by BTO-GSA algorithm with other algorithms under varying number of iterations. The table values indicated that the Genetic-SA algorithm has offered ineffective results with the maximum transfer time over the compared methods. At the same time, the FCFS and Min-Min method has offered slightly better performance over the Genetic-SA model, by offering slightly lower transfer time. Also, the PSO and C-GSA algorithms have tried to exhibit competitive results with the moderate transfer time. However, the BTO-GSA algorithm has resulted to a maximum transfer time over the compared methods under 1000 iterations.

Table 1 Transfer Time Analysis of BTO-GSA algorithms over 1000 rounds

Iterations	Genetic-SA	FCFS	Min-Min	PSO	C-GSA	BTO-GSA
100	62000	58500	58200	54500	55000	54000
200	60000	58500	58200	57000	51500	51000
300	70500	58500	58200	51000	52250	51550
400	62500	58500	58200	52500	48000	47000
500	64200	58500	58200	53500	47000	46000
600	59500	58500	58200	56800	38000	37800
700	50500	58500	58200	50500	50000	48000
800	55550	58500	58200	55500	55500	53000
900	67000	58500	58200	52500	50100	50000
1000	68000	58500	58200	56500	71550	56000

For instance, under the iteration of 100, the BTO-GSA algorithm has attained lower transfer time with the minimal value of 54000 whereas the Genetic-SA algorithm has offered maximum transfer time of 62000. At the same time, the other existing methods such as FCFS, Min-Min, PSO and C-GSA algorithms have resulted to a low transfer time of 58500, 58200, 54500 and 55000 respectively. Similarly, under the maximum iteration of 1000, the proposed BTO-GSA algorithm has exhibited lower transfer of 56000 whereas the C-GSA algorithm has achieved maximum transfer time of 71550. Along with that, the other existing methods such as Genetic-SA, FCFS, Min-Min and PSO and C-GSA algorithms leads to a moderate transfer time of 62000, 58500, 58200 and 54500 respectively.

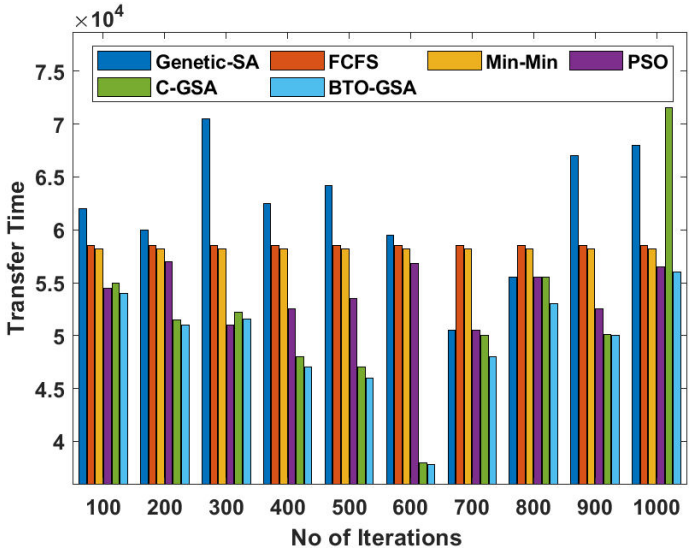

Fig. 3. Transfer Time Analysis of BTO-GSA algorithm

Table 2 and Fig. 4 have depicted a brief comparison of results provided by BTO-GSA model with other algorithms under varying number of iterations. The table values pointed that the PSO algorithm has shown ineffective results with the higher cost for 10 cloudlets compared to other models. Simultaneously, the C-GSA and LIGSA-C methods have offered moderate performance over PSO model, by resulting to a lower cost for 10 cloudlets. In addition, the HG-GSA algorithms have attempted to showcase the results with considerable cost for 10 cloudlets. Therefore, the BTO-GSA algorithm has given a greater cost for 10 cloudlets over the other methods under 1000 iterations. For example, under the iteration of 10, the BTO-GSA technology has reached to a minimum cost for 10 cloudlets with the least value of 19000.85 while the PSO algorithm has accomplished higher cost of 144670.35. Simultaneously, the other traditional approaches like C-GSA, LIGSA-C and HG-GSA methodologies have shown as somewhat manageable cost of 22033.56, 27842.76 and 19307.62 correspondingly. Likewise, under the maximum iteration of 1000, the presented BTO-GSA algorithm has obtained lower cost for 10 cloudlets of 18990.90 and the PSO algorithm has attained best cost for 10 cloudlets of 150183.34. In line with this, the other existing methods namely, C-GSA, LIGSA-C and HG-GSA models provides the acceptable cost for 10 cloudlets of 19879.76, 18100.89 and 19585.39 correspondingly.

Table 2 Total Cost Analysis of 10 Cloudlets in Existing with Proposed Algorithms

Iterations	PSO	C-GSA	LIGSA-C	HG-GSA	BTO-GSA
10	144670.35	22033.56	27842.76	19307.62	19000.85
20	154718.39	22231.08	22364.73	22127.08	21900.03
30	146023.00	22646.84	22364.73	20917.08	20250.80
40	151117.03	25236.80	29819.64	16894.16	15980.10
50	153934.60	21804.13	28550.08	19307.62	18870.25
60	146671.56	28238.39	22977.63	19237.54	18920.24
70	150630.78	24134.52	25685.22	18361.30	17950.00
80	144320.10	27754.70	26092.18	19360.00	18760.28
90	150043.44	28324.33	27754.70	19754.70	19200.65
100	155538.95	29378.09	27309.16	19307.62	18900.10
200	143757.94	22364.73	25289.42	18637.28	18200.17
300	155469.76	26501.04	23337.08	20290.97	19600.50
400	154534.97	27754.70	29023.44	19754.70	19300.35
500	148303.41	22239.87	24134.52	18035.19	17800.63
600	143482.75	22364.73	25410.00	18745.79	18126.60
700	148181.19	33288.01	25229.59	19879.76	19270.56
800	145980.43	26069.20	25705.82	19760.96	19150.45
900	145713.85	19760.96	22927.79	20514.34	19580.15
1000	150183.34	19879.76	18100.89	19585.39	18990.90

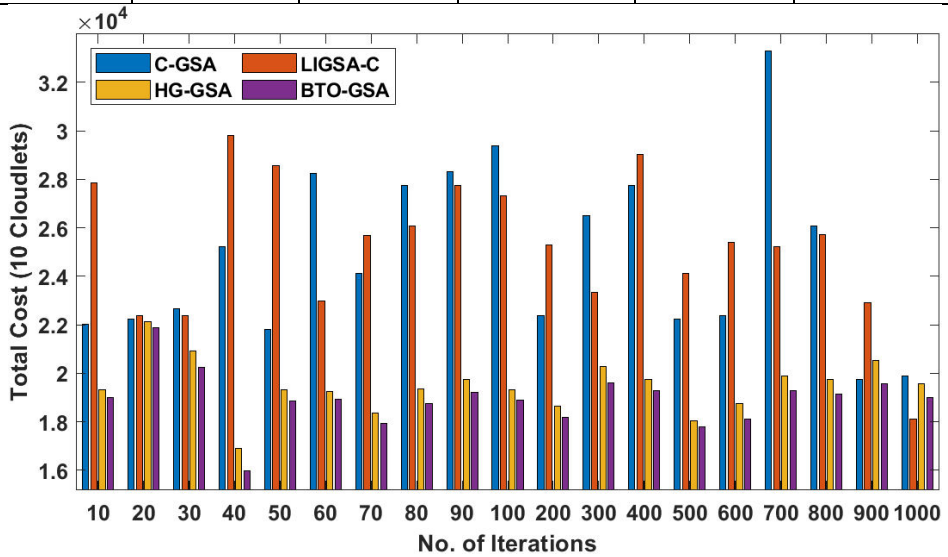

Fig. 4. Cost Analysis of 10 Cloudlets of BTO-GSA algorithm

Table 3 and Fig. 5 have offered an extended comparison of the results exhibited by BTO-GSA algorithm with other methods under different number of iterations. The table values implied that the PSO algorithm has provided poor results with the higher cost for 15 cloudlets over the compared methods. Meanwhile, the C-GSA and LIGSA-C methodology has given reasonable performance when compared with PSO model, leads to a moderate cost for 15 cloudlets. Furthermore, the HG-GSA algorithm has shown results with the slightly lower cost over previous algorithms for 15 cloudlets. Thus, the BTO-GSA algorithm has attained a greater cost for 15 cloudlets when compared with alternate techniques under 1000 iterations. For illustration, under the iteration of 10, the BTO-GSA model has accomplished a lower cost for 15 cloudlets with the least value of 27105.36 while the PSO algorithm has reached maximum cost for 15 cloudlets of 251281.52. Concurrently, the conventional approaches like C-GSA, LIGSA-C and HG-GSA algorithms have obtained moderate cost of 37857.08, 32257.32 and 27754.70. On continuing with, under the iteration of 1000, the proposed BTO-GSA algorithm has defined to a minimum cost for 15 cloudlets of 25840.56 and the PSO algorithm has derived maximum cost for 15 cloudlets of 245697.97. In line with this, the previous techniques like C-GSA, LIGSA-C and HG-GSA algorithms have produced to a gradual cost for 15 cloudlets of 34995.05, 40039.80 and 27124.43 respectively.

Table 3 Cost Analysis of 15 Cloudlets in Existing with Proposed Algorithms

Iterations	PSO	C-GSA	LIGSA-C	HG-GSA	BTO-GSA
10	251281.52	37857.08	32257.32	27754.70	27105.36
20	248417.27	35963.57	28937.14	32304.61	31102.25
30	248960.09	29040.00	28961.42	25341.25	21020.14
40	247411.82	38166.08	38475.07	27754.70	21356.20
50	249097.51	32648.05	33050.35	32053.00	31000.86
60	252863.34	38681.68	36268.23	25341.25	24920.35
70	233301.08	41898.99	36731.93	27567.72	27230.51
80	240275.47	31897.65	35784.15	30058.65	28500.34
90	238414.15	36863.29	30352.66	26451.61	25230.36
100	246693.71	38786.14	31826.26	32027.90	31032.20
200	257046.55	31383.81	37946.69	30390.74	28982.98
300	241913.93	30846.27	36969.75	30421.03	29352.05
400	250686.57	43246.33	38615.23	26682.65	21346.63
500	255247.49	30421.03	34466.45	32676.34	29850.15
600	255640.52	41051.12	38202.02	28961.42	25980.98
700	240621.62	42062.43	32053.00	26547.97	25680.65
800	234732.64	35731.24	31027.56	25705.82	22300.32
900	245081.61	37408.51	38177.92	28961.42	26780.00
1000	245697.97	34995.05	40039.80	27124.43	25840.56

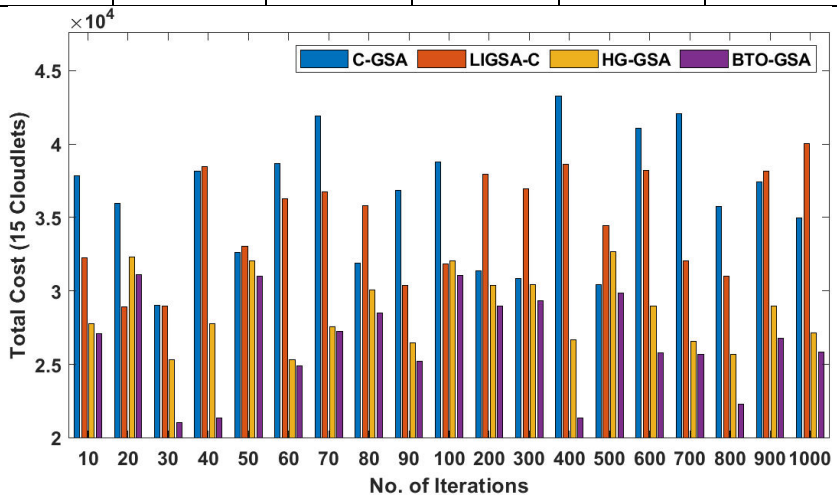

Fig. 5. Cost Analysis for 15 Cloudlets of BTO-GSA algorithm

Table 4 and Fig. 6 have depicted a detailed comparison of the results attained by BTO-GSA with other algorithms under several numbers of iterations. The table values denote that the PSO algorithm has provided worst results with the greater cost than alternate technologies. Meanwhile, the C-GSA and LIGSA-C methods have reached manageable outcome over the PSO model, by resulting to a slightly reduced cost for 20 cloudlets. In addition, the HG-GSA algorithms have attempted to implement the results with the reasonable cost for 20 cloudlets. However, the BTO-GSA algorithm has resulted to a higher cost for 20 cloudlets than the related methods under 1000 iterations. For example, under the iteration of 10, the BTO-GSA approach has accomplished to least cost of 35450.20 while the PSO algorithm has offered maximum cost of 368649.60. Simultaneously, the conventional methods such as C-GSA, LIGSA-C and HG-GSA algorithms have obtained a low cost for 20 cloudlets of 44876.48, 44087.03 and 36863.29 correspondingly. Likewise, under the maximum iteration of 1000, the projected BTO-GSA algorithm has implied lower cost for 20 cloudlets of 35632.77 and the PSO algorithm has derived maximum cost for 20 cloudlets of 363249.97. On the same way, the previous models like C-GSA, LIGSA-C and HG-GSA algorithms provides manageable cost for 20 cloudlets of 52452.21, 51730.62 and 36201.78 correspondingly.

Table 4 Cost Analysis of 20 Cloudlets in Existing with Proposed Algorithms

Iterations	PSO	C-GSA	LIGSA-C	HG-GSA	BTO-GSA
10	368649.60	44876.48	44087.03	36863.29	35450.20
20	354478.05	41712.67	45876.72	37233.37	36560.12
30	345540.58	46517.10	45524.08	40483.47	38450.65
40	356553.37	44648.86	43818.19	38517.03	37620.35
50	353046.85	50765.56	49868.66	34390.46	33160.42
60	355353.89	47062.32	46108.80	41002.00	40562.32
70	351187.75	44648.86	45980.00	41028.68	40250.79
80	357996.22	36958.02	42285.25	36604.12	35120.52
90	371324.59	47707.21	39759.52	37341.23	35940.11
100	368448.35	43551.33	42368.31	38615.23	37462.36
200	359849.89	48400.00	35498.52	34390.46	33280.12
300	347655.15	44876.48	42082.11	39821.96	38502.32
400	350307.00	44648.86	46404.60	32027.90	31089.85
500	365131.36	39521.91	45492.86	40855.59	39580.37
600	366225.26	51872.51	39434.34	44085.78	43250.12
700	369217.78	47914.30	39521.91	41690.19	40980.85
800	366704.32	42284.40	46171.69	34390.46	33890.75
900	362957.68	47195.22	50498.53	41028.68	40089.65
1000	363249.97	52452.21	51730.62	36201.78	35632.77

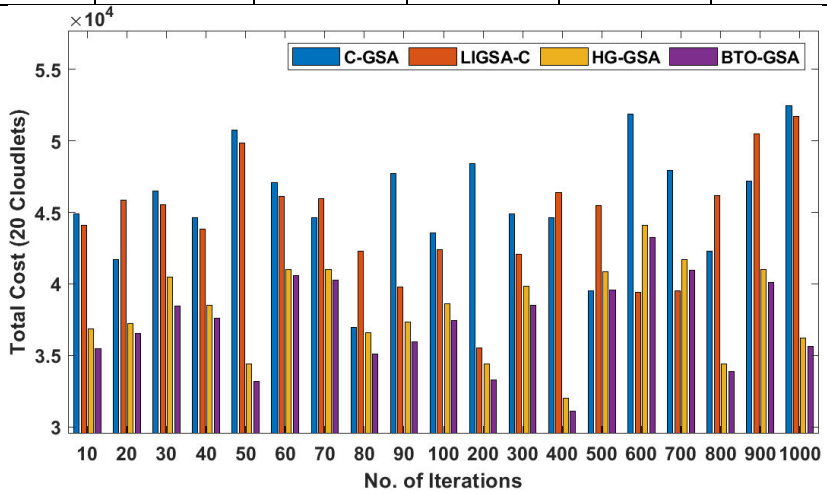

Fig. 6. Cost Analysis of 20 Cloudlets of BTO-GSA algorithm

Table 5 and Fig. 7 showcase a detailed comparison of the results accomplished by BTO-GSA algorithm with alternate algorithms under numerous numbers of iterations. The table values represented that the PSO algorithm has provided worst results with the higher distance cost for 20 cloudlets than the compared methods. Concurrently, the C-GSA and LIGSA-C methods have reached slightly better results over the PSO model, by offering a slightly minimum distance cost for 20 cloudlets. Additionally, the HG-GSA algorithms have attempted to display results with the considerable distance cost for 20 cloudlets. But, the BTO-GSA algorithm has derived a least distance cost for 20 cloudlets than the methods under 19 iterations. For example, under the iteration of 1, the BTO-GSA, PSO, C-GSA, LIGSA-C and HG-GSA frameworks has accomplished cost for 20 cloudlets with the value of 0. Likewise, under the maximum iteration of 19, the deployed BTO-GSA algorithm has signified least distance cost of 470 while the PSO algorithm has attained maximum distance cost of 1550. In line with this, the other methods like C-GSA, LIGSA-C and HG-GSA algorithms tends to generate reasonable distance cost of 1540, 1100 and 600 respectively.

Table 5 Mean of Normalized Distance (Total Cost) for 20 Cloudlets

Data Vector	PSO	C-GSA	LIGSA-C	HG-GSA	BTO-GSA
1	0	0	0	0	0
2	10000	200	5400	2800	2000
3	5200	300	3000	1750	1700
4	3700	1300	3000	1900	1300
5	3200	1000	2500	1700	1750
6	2800	1700	2100	1600	1480
7	2200	1600	1900	1200	1050
8	2100	1650	1700	1000	750
9	1990	1700	1650	900	660
10	1990	1650	1600	900	600
11	1990	1600	1550	850	800
12	1980	1580	1550	820	780
13	1900	1570	1550	800	700
14	1800	1530	1500	780	670
15	1800	1480	1450	700	600
16	1700	1550	1200	680	560
17	1650	1520	1100	660	500
18	1600	1530	1000	630	480
19	1550	1540	1100	600	470

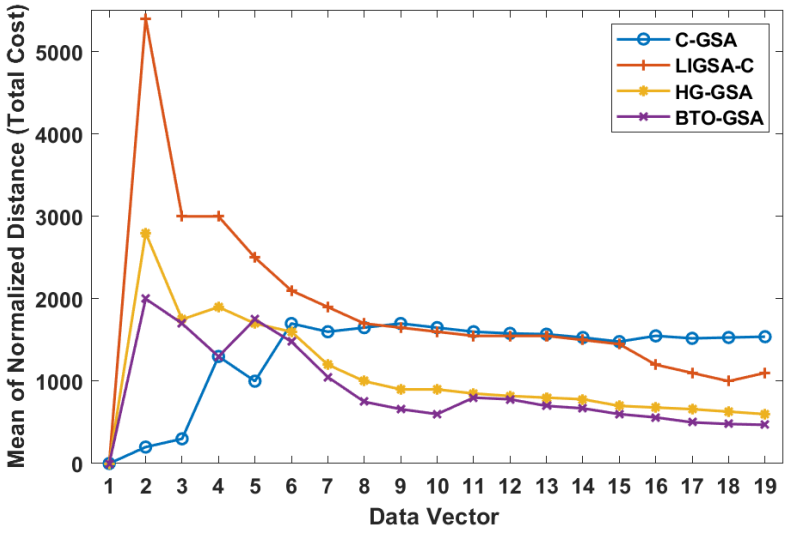

Fig. 7. Mean of Normalized Distance of BTO-GSA algorithm

Table 6 and Fig. 8 depict a brief comparison of the results accomplished by BTO-GSA algorithm with other models under diverse number of iterations. The table values show that the PSO algorithm has given poor results with the maximum displacement cost for 20 cloudlets over the other methods. Simultaneously, the C-GSA and LIGSA-C methods have afforded better performance than the PSO model with a minimum displacement cost for 20 cloudlets. Moreover, the HG-GSA algorithms have attempted to exhibit moderate results with the considerable displacement cost for 20 cloudlets. Thus, the BTO-GSA algorithm has achieved superior displacement cost for 20 cloudlets over existing methods under 19 iterations. For sample, under the iteration of 1, the BTO-GSA algorithm has accomplished least displacement cost for 20 cloudlets with the low value of 33000 and the PSO algorithm has provided maximum displacement cost for 20 cloudlets of 350000. Concurrently, the other methods like C-GSA, LIGSA-C and HG-GSA models have attained as minimum displacement cost for 20 cloudlets of 45000, 44900 and 37000 correspondingly.

Table 6 Mean of Normalized Displacement (Total cost) for 20 Cloudlets

Data Vector	PSO	C-GSA	LIGSA-C	HG-GSA	BTO-GSA
1	350000	45000	44900	37000	33000
2	256000	30500	33000	26000	22000
3	200000	25500	26000	22000	20000
4	158000	22500	22600	19000	16000
5	152000	20500	20500	17000	13000
6	145000	18000	18000	15100	12000
7	142000	17500	17500	14950	11500
8	136000	16000	16100	14000	12500
9	130000	15000	15000	13000	10000
10	125000	14800	14800	12250	9900
11	120000	14000	13950	11500	8300
12	110500	13000	12950	11000	7500
13	105000	12500	12450	10500	7500
14	100000	12000	12000	10000	6000
15	98000	11500	11400	10000	5900
16	95000	11000	10800	9950	5500
17	87000	10800	10700	9900	4400
18	84000	10500	10300	9850	4300
19	80000	10100	10050	9800	4200

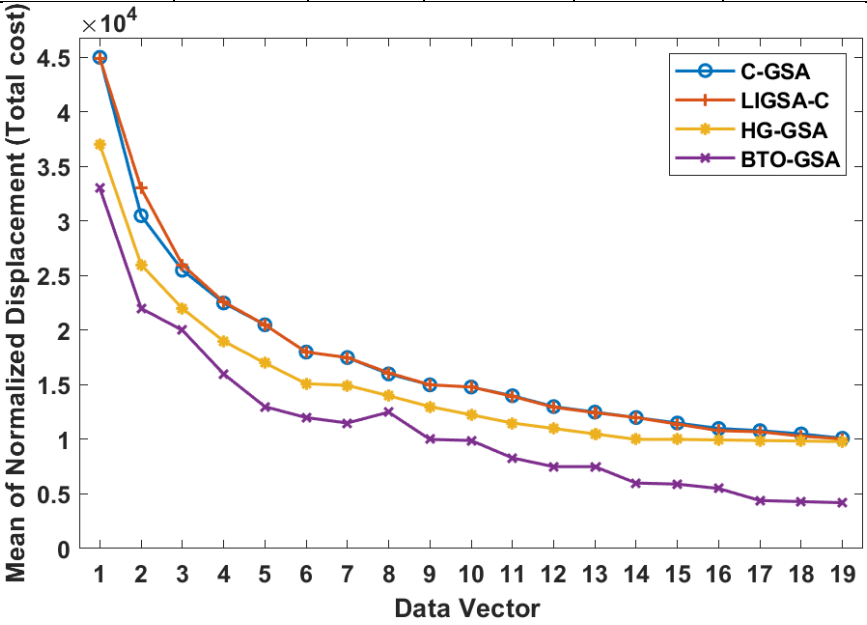

Fig. 8 Mean of Normalized Displacement of BTO-GSA algorithm

Along with that, under the maximum iteration of 19, the developed BTO-GSA algorithm has shown displacement of lower cost of 4200 whereas the PSO algorithm has reached best displacement cost of 80000. Similarly, the previous techniques namely, C-GSA, LIGSA-C and HG-GSA algorithms leads to a manageable displacement cost of 10100, 10050 and 9800 respectively.

Table 7 and Fig. 9 show the statistical analysis of the total cost under diverse number of cloudlets. Fig. 9a implies the statistical analysis of the total cost of cloudlets with respect to mean. Under the existence of 10 cloudlets, the BTO-GSA algorithm has accomplished minimum total cost with the lower value of 18934.40 while the PSO model has given higher total cost of cloudlets of 149119.78. Meanwhile, the previous models like C-GSA, LIGSA-C and HG-GSA algorithms have exhibited high total cost of cloudlets of 24842.39, 25258.91 and 19462.06 correspondingly. Concurrently, under the presence of 15 cloudlets, the BTO-GSA algorithm has reached lower total cost of cloudlets with the minimal value of 26558.57 and the PSO algorithm has showcased greater total cost of cloudlets of 246493.94. Meanwhile, the other existing approaches like C-GSA, LIGSA-C and HG-GSA algorithms have depicted a high total cost of cloudlets of 36260.44, 34744.37 and 28638.27 correspondingly. Likewise, under the presence of 20 cloudlets, the presented BTO-GSA algorithm has implied total cost of cloudlets of 37203.93 while the PSO algorithm has attained higher total cost of cloudlets of 359677.77. In line with this, the other existing methods namely, C-GSA, LIGSA-C and HG-GSA technologies results in considerable total cost of cloudlets of 45663.91, 44342.72 and 38240.61 correspondingly.

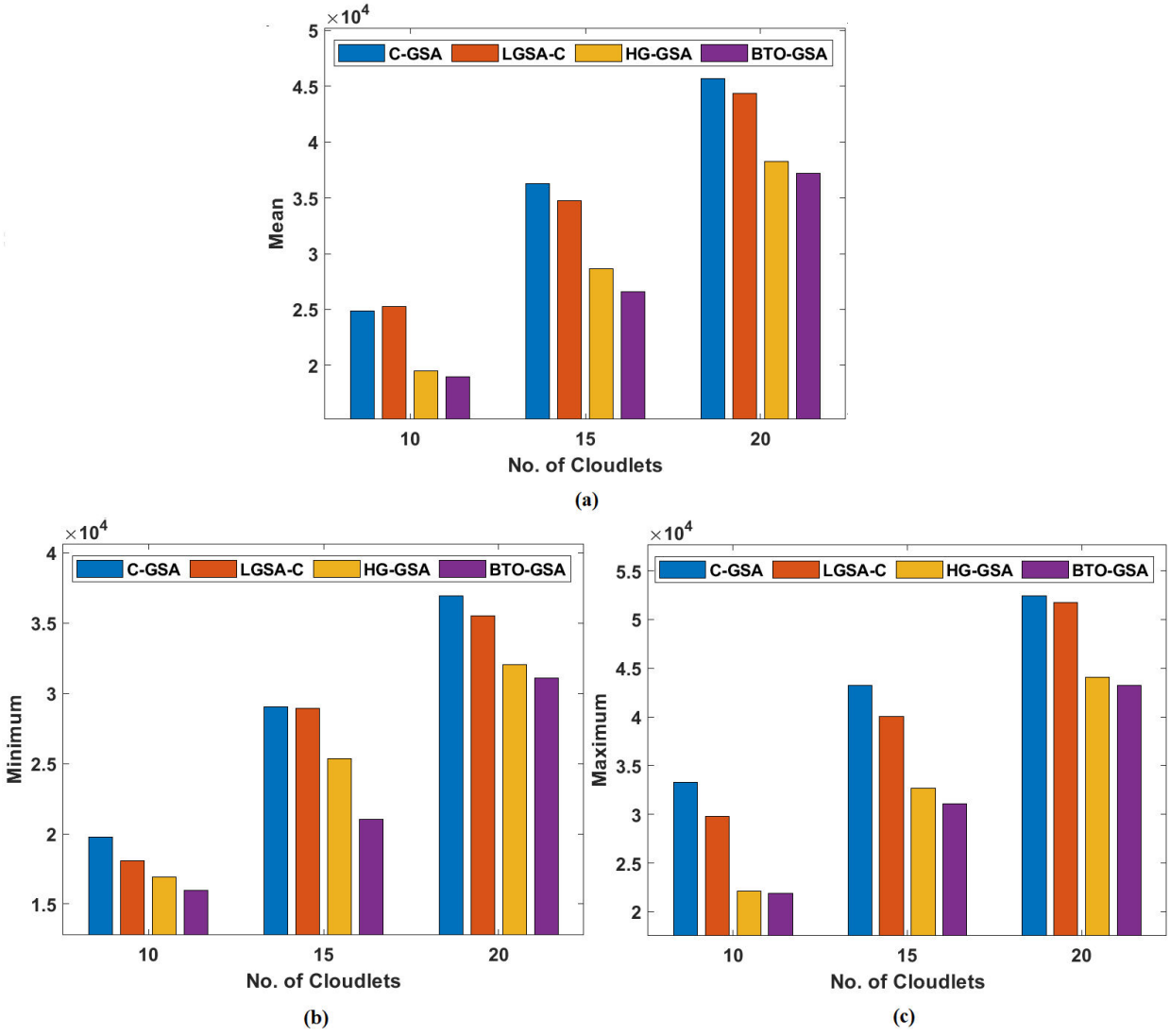

Fig. 9. Statistical Results Analysis of BTO-GSA algorithm a) Mean b) Minimum c) Maximum

Table 7 Statistical results analysis of BTO-GSA algorithm

Parameter	No. of Cloudlets	PSO	C-GSA	LGSA-C	HG-GSA	BTO-GSA
Mean	10	149119.78	24842.39	25258.91	19462.06	18934.40
	15	246493.94	36260.44	34744.37	28638.27	26558.57
	20	359677.77	45663.91	44342.72	38240.61	37203.93
Minimum	10	143482.75	19760.96	18100.89	16894.16	15980.10
	15	233301.08	29040.00	28937.14	25341.25	21020.14
	20	345540.58	36958.02	35498.52	32027.90	31089.85
Maximum	10	155538.95	33288.01	29819.64	22127.08	21900.03
	15	257046.55	43246.33	40039.80	32676.34	31102.25
	20	371324.59	52452.21	51730.62	44085.78	43250.12

Fig. 9b signifies the statistical investigation of the total cost of the cloudlets interms of Minimum. Under the iteration of 10, the BTO-GSA method has reached minimum total cost of cloudlets with the least value of 15980.10 while the PSO algorithm has attained maximum total cost of cloudlets of 143482.75. Meanwhile, the other existing schemes like C-GSA, LIGSA-C and HG-GSA models have achieved a moderate total cost of cloudlets of 19760.96, 18100.89 and 16894.16 correspondingly. Meanwhile, under the iteration of 15, the BTO-GSA approaches has accomplished a lower total cost of cloudlets with the least value of 21020.14 and the PSO algorithm has shown good total cost of cloudlets of 233301.08. Likewise, the other previous technologies like C-GSA, LIGSA-C and HG-GSA algorithms have attained minimum total cost of cloudlets of 29040.00, 28937.14 and 25341.25 correspondingly. In line with this, under the maximum iteration of 20, the projected BTO-GSA algorithm has attained total cost of 31089.85 and the PSO algorithm has reached to a higher total cost of cloudlets of 345540.58. On the same way, the other existing algorithms like C-GSA, LIGSA-C and HG-GSA algorithms resulted in a high total cost of cloudlets of 36958.02, 35498.52 and 32027.90 correspondingly.

Fig. 9c illustrates the statistical examination of the total cost of the cloudlets with respect to maximum. Under the iteration of 10, the BTO-GSA algorithm has accomplished a minimum total cost of cloudlets with the least value of 21900.03 and the PSO algorithm has provided maximum total cost of 155538.95. Concurrently, the classical methods like C-GSA, LIGSA-C and HG-GSA algorithms have attained a high total cost of cloudlets of 33288.01, 29819.64 and 22127.08 correspondingly. Meanwhile, under the iteration of 15, the BTO-GSA approach has attained lower total cost of 31102.25 and the PSO algorithm has given maximum total cost of 257046.55. Simultaneously, the existing methods like C-GSA, LIGSA-C and HG-GSA algorithms have shown a high total cost of 43246.33, 40039.80 and 32676.34 correspondingly. In line with this, under the maximum iteration of 20, the projected BTO-GSA algorithm has generated a least total cost of 43250.12 while the PSO algorithm has reached to a maximum total cost of cloudlets of 371324.59. Similarly, the other existing models like C-GSA, LIGSA-C and HG-GSA methodologies tends to a considerable total cost of cloudlets of 52452.21, 51730.62 and 44085.78 correspondingly. The above-mentioned tables and figures clearly stated that the BTO-GSA algorithm is superior to other algorithms under diverse aspects.

4. Conclusion

This paper has devised a new load scheduling algorithm using hybridization of metaheuristic algorithm, called BTO-GSA algorithm. The proposed BTO-GSA algorithm make use of Splitting Point Selection technique based GSA to identify the optimum positions of the particles in the search space. The application of BTO algorithm relied on the mathematical tree subject, improvises the outcome and searching speed by incessantly eliminates the portions of the search space with minimum fitness for minimizing and purifying the search space. The BTO-GSA model has been implemented using CloudSim simulation. The outcome from the experiments pointed out that the BTO-GSA model is superior to other methods in a significant way. In future, the performance of the BTO-GSA model can be improvised by the use of deep learning methods.

References

[1] 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).
[2] 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).
[3] 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).
[4] 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).
[5] 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).
[6] 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).
[7] 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).
[8] 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).
[9] 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).
[10] 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).
[11] 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).
[12] 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).
[13] Kennedy J., Eberhart R.: Particle swarm optimization. IEEE International Conference on Neural Networks, vol. 4, pp.: 1942–1948, IEEE (1995).
[14] 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).
[15] 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).
[16] 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).
[17] 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).
[18] 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).
[19] Ghojogh, B., Sharifian, S. and Mohammadzade, H., 2018. Tree-based optimization: A meta-algorithm for metaheuristic optimization. *arXiv preprint arXiv:1809.09284*.
[20] Chaudhary, D. and Kumar, B., 2019. Cost optimized Hybrid Genetic-Gravitational Search Algorithm for load scheduling in Cloud Computing. Applied Soft Computing, 83, p.105627.
[21] Chaudhary, D. and Kumar, B., 2018. Cloudy GSA for load scheduling in cloud computing. Applied Soft Computing, 71, pp.861-871.
[22] A. Francis Saviour Devaraj, Mohamed Elhoseny, S.Dhanasekaran, E. LaxmiLydia, K. Shankar, "Hybridization of firefly and Improved Multi-Objective Particle Swarm Optimization algorithm for energy efficient load balancing in Cloud Computing environments", Journal of Parallel and Distributed Computing, Volume. 142, Page(s): 36-45, August 2020.
[23] Mohamed Elhoseny, K. Shankar, S. K. Lakshmanaprabu, Andino Maseleno, N. Arunkumar, "Hybrid optimization with cryptography encryption for medical image security in Internet of Things", Neural Computing and Applications - Springer, October 2018. https://doi.org/10.1007/s00521-018-3801-x

[24] Lakshmanaprabu S.K, Sachi Nandan Mohanty, Sheeba Rani S, Sujatha Krishnamoorthy, Uthayakumar J, K. Shankar, "Online clinical decision support system using optimal deep neural networks", Applied Soft Computing, Volume 81, Page(s): 1-10, August 2019.
[25] M. Sivaram, E. Laxmi Lydia, Irina V. Pustokhina, Denis A. Pustokhin, Mohamed Elhoseny, Gyanendra Prasad Joshi, K. Shankar, "An Optimal Least Square Support Vector Machine Based Earnings Prediction of Blockchain Financial Products", IEEE Access, Volume. 8, Page(s): 120321-120330, June 2020.
[26] Shweta Sankhwar, Deepak Gupta, K. C. Ramya, S. Sheeba Rani, K. Shankar, S. K. Lakshmanaprabu, "Improved grey wolf optimization-based feature subset selection with fuzzy neural classifier for financial crisis prediction", Soft Computing, September 2019. In Press. https://doi.org/10.1007/s00500-019-04323-6
[27] K. Shankar, Lakshmanaprabu S. K, Ashish Khanna, Sudeep Tanwar, Joel J.P.C.Rodrigues, Nihar Ranjan Roy, "Alzheimer detection using Group Grey Wolf Optimization based features with convolutional classifier", Computers & Electrical Engineering, Volume 77, Pages 230-243, July 2019.
[28] K. Shankar, Lakshmanaprabu S.K, Deepak Gupta, Andino Maseleno, Victor Hugo C. de Albuquerque, "Optimal Features Based Multi Kernel SVM Approach for Thyroid Disease Classification", The Journal of Supercomputing - Springer, June 2018. In press: https://doi.org/10.1007/s11227-018-2469-4
[29] S. Famila, A. Jawahar, A. Sariga, K. Shankar, "Improved artificial bee colony optimization based clustering algorithm for SMART sensor environments", Peer-to-Peer Networking and Applications, August 2019. In Press. https://doi.org/10.1007/s12083-019-00805-4
[30] K. Karthikeyan, R. Sunder, K. Shankar, S. K. Lakshmanaprabu, V. Vijayakumar, Mohamed Elhoseny, Gunasekaran Manogaran, "Energy consumption analysis of Virtual Machine migration in cloud using hybrid swarm optimization (ABC–BA)", The Journal of Supercomputing - Springer, September 2018. https://doi.org/10.1007/s11227-018-2583-3