Research Article

Load Balancing In Cloud Computing Environment Using Quasi Oppositional Dragonfly Algorithm

J.Robert Adaikalaraj [a], and T. Vengattaraman [b]

[a] Research Scholar, Department of Computer Science, Bharathiar University, Coimbatore, Tamilnadu, India
[b] Department of Computer Science, Pondicherry University, Puducherry, India

Article History: Received: 10 January 2021; Revised: 12 February 2021; Accepted: 27 March 2021; Published online: 28 April 2021

Abstract: In cloud computing (CC) environment, load balancing of tasks remains as an important problem of distributing resources from a data center to make sure that every virtual machine (VM) have balanced load to attain optimum utilization of its abilities. Load balancing in CC environment is considered as a non-polynomial (NP) problem and metaheursitic algorithms can be applied to resolve it. This paper presents a new quasi oppositional dragonfly algorithm for load balancing (QODA-LB) in CC environment to achieve optimal resource scheduling. The proposed QODA-LB algorithm derives an objective function using three variables namely execution time, execution cost, and load. Based on the derived objective function, the QODA-LB algorithm allocates tasks to VM with respect to its capacity. Besides, the QODA-LB algorithm incorporates quasi oppositional based learning (QOBL) concept to improve the convergence rate of classical dragonfly algorithm (DA). Detailed set of experimentations were carried out to ensure the effective performance of the QODA-LB algorithm and the results are examined under several aspects. The simulation outcome has depicted optimal load balancing performance and demonstrated better results compared to state of art methods.

Keywords: Cloud computing, load scheduling, dragonfly algorithm, oppositional based learning

1. Introduction

Load balancing in Cloud computing (CC) is assumed to be a challenging and helpful study in dividing the operations over Virtual Machines (VM) at data centers. Here, CC is a significant approach of on-demand resource distribution by Internet. Cloud is composed of massive interlinked system in a multifarious fashion, in which all files and fields are applied. CC is combined with distributed and parallel computing principle to provide resource distribution like software, hardware, data and files according to the requirement of alternate machines on cloud. It provides ‘‘pay as you need” module in shared system. Here, the user does not need any computational environment to calculate an operation and needs the Internet connection to apply the resources by spending money for specific time. As a result, it limits the amount of purchasing software package that does not require full-time and CC offers the service of dynamic feature of resources. VMs are considered as processing units in CC that computes and distributes the resources whenever essential dynamically at the time of implementing an operation.

In CC, massive VMs are linked by placing the resources in pre-emptive as well as non- pre-emptive fashion, finally they are not shared equally and few VMs are not capable to accomplish the resources. If a task is provided in cloud, VMs has to implement the operation rapidly and minimize the implementation time complexity and here; all VMs should be executed in parallel fashion. It shows the requirement of scheduling tasks and finishes the implementation with accessible resources. If multiple tasks are allocated to massive VMs, then it is implemented simultaneously to finish the allocated works. If the scheduler is assigning the operation for VMs, it should ensure that all operations are not invoked in single VM which alternate VMs are free. Therefore, it is role of scheduler in which all user tasks has to be equally managed over all VMs in CC. In order to eliminate the issue of load balancing in all VMs, which needs smart load balancing model for enhancing the response time of implementation of allocated operations by saving higher application of provided resources, as shown in Fig. 1.

Load balancing is to allocate input task over the VMs in uniform manner. The key objective of load balancing is to eliminate the burden from all VMs and declare it to alternate minimum weighted VMs. It enhances the effectiveness as well as throughput of a system. Developers have solved the load balancing issues using heuristic as well as meta-heuristic models. Load balancing with protection is reported in a distributed network (Ezumalai, R., 2010). Also, there are 3 models which have been applied for solving the load balancing as well as security issues in a distributed network. Initially, it offers a structure for mobile agent to spread all nodes in a distributed network. Followed by, it gives a structure to reform the load between the peers to offer optimal function and consequently, it offers network security. A Hierarchical load balancing approach is presented for resolving the distribution of loads in Grid computing platform. The present node details are used while sharing the input task in dynamic manner from VMs and merits of this application which limits the maximum response time for Grid application. The speed of a method has validated by comparing with Minimal Completion Time as well as Perfect Information on Arrival.

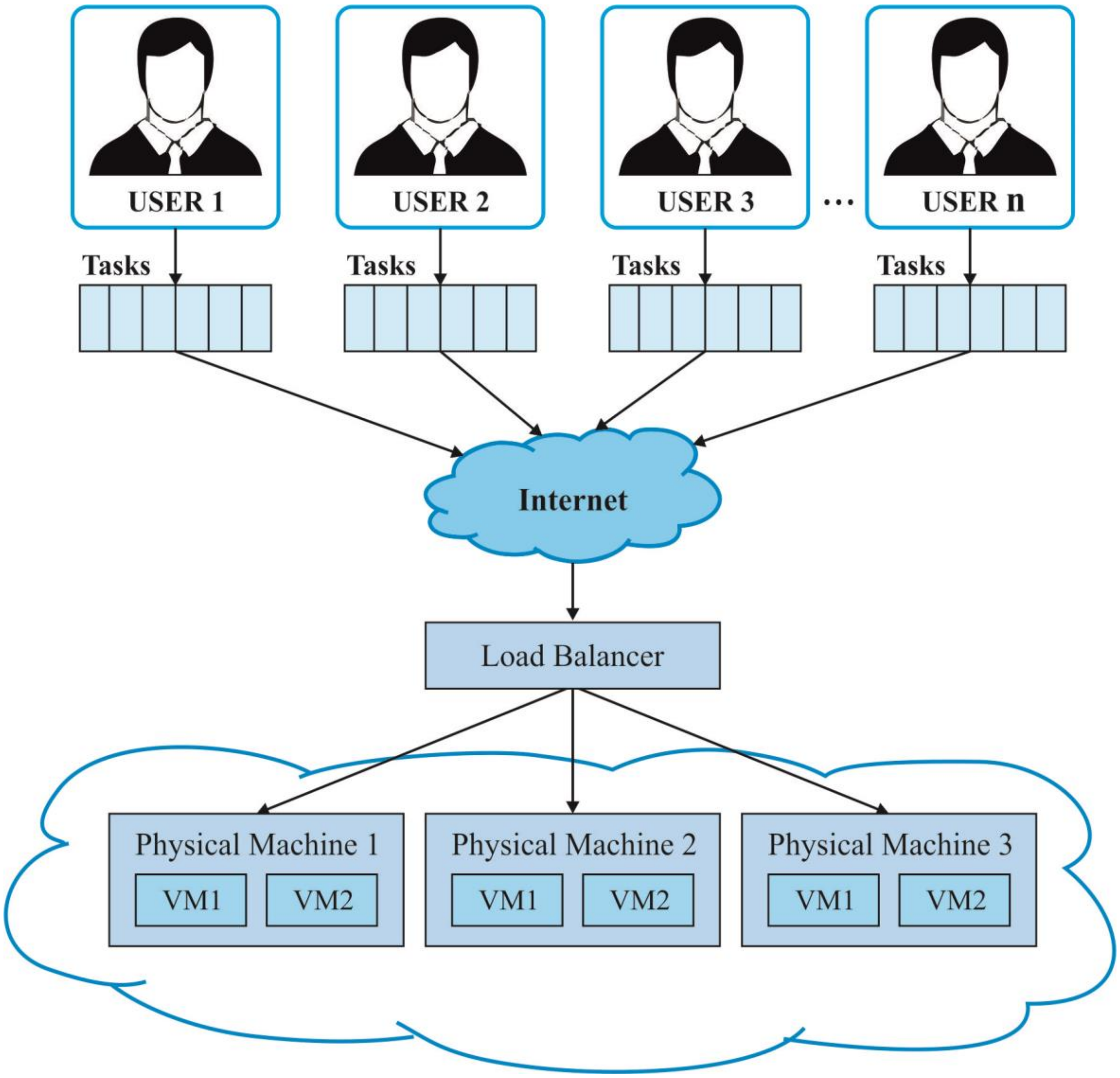

Figure 1. Load balancing in CC platform

Honey bee nature based relied load balancing is solved in non-pre-emptive autonomous tasks on VMs (Krishna, P. 2013). The newly developed method is helpful in resolving load balancing over VM to enhance the throughput, limiting the waiting time of tasks by assuming the preferences of task and consequently, the associations is performed with alternate models like weighted Round Robin (RR), FIFO and Dynamic Load Balancing to provide effectiveness of a system with respect to throughput and limit the response time of VMs. Dynamic load balancing is solved in a distributed virtual platform by heat diffusion (Deng, Y., 2014). It projects 2 dynamic load balancing models, initially, it applies local and global load balancing in distributed virtual platform by using heat diffusion and secondly it examined 2 functional aspects like load balancing factor as well as convergence threshold.

Extended model of Particle Swarm Optimization (PSO) is developed to resolve the load balancing problem in CC system (Zhu, Y., 2016). It has enhanced the implementation of speed and effectiveness. Tabu search (TS) approach is developed for resource management in CC model. Dynamic optimized resource scheduling management is deployed according to the factors which are comprised of deadline constraint, cost constraint and optimal solution. TS method is applied for resolving the resource assignment by prioritization as well as task grouping. TS method is used for optimizing the positions of cloud data centers as well as software units like data routing and network link abilities (Larumbe, F., 2013).

The makespan and higher application of resource is limited by simple scheduling approach in grid computing platform (Alharbi, F., 2012). It presents a load balancing model where load is shared uniformly and reduces the response time. Fuzzy logic (FL) is implied for effective load balancing and minimizes the cost and power in Geo-Distributed several data Centers. It showcases best offline geographical load balancing by using FL inference system under the input data mapping non-linearly like current application of renewable power, accessibility of electric cost as well as power utilization to produce the requests by data center.

Load rebalancing is performed for distributed file systems in cloud system where it optimizes the network traffic by improving the bandwidth of a system (Hsiao, H.C., 2013). Complete distribution of load balancing rebalancing model is developed to solve the load imbalance and consequently, the newly developed approach is related with previous centralized approach. An online algorithm with Lyapunov optimization strategy is proposed for load scheduling as well as eco-aware power management for cloud data centers (Deng, X., 2016). The main theme of this approach is to reduce the time average eco-aware power cost of CC data centers at the time of assuring quality-of-experience (QoE) constraints. Honey bee method is executed for limiting the makespan and allocates the resource for extending the throughput in CC platform (Vasudevan, S.K., 2016). It is considered as dynamic approach which has been applied for identifying the variations in nature among dependence and independence operations limit the makespan for all tasks by using task preferences. The load of a server is managed by applying self-adaptive Randomized Optimization. Power consumption has been lowered at the time of allocating resources to each task in CC platform. Load prediction and requirement of resources is defined by Enhanced Exponentially Weighted Moving Average. Soft computing methods are projected to resolve the dynamic load balancing in CC environment (Mondal, B., 2015).

The load balancing method for CC is performed by a Genetic Algorithm (GA). Resource allocation is done in homogeneous as well as heterogeneous CC platforms (Mishra, Sambit Kumar, 2018). Researchers have depicted the makespan and power application at the time of resource allocation. A static load balancing approach is developed for resolving the load balancing (Song, S., 2014). It employs static data for balancing the load with no influence of load of cluster node which contains inferior adaptive capability. Bayes scheme is executed for prolonged load balancing. Improved weighted round-robin approach is deployed for resolving the load balancing in case of non-pre-emptive dependent tasks (Devi, D. 2016).

The function of load balancing is realized by various load balancing modules and results are related with respect to throughput and speed (Kanakala, V.R.T., 2015). Various scheduling approaches are deployed for Map Reduce environment that involves in load balancing for CC network (Selvi, R.T., 2016). Adaptive task allocation scheduler is projected for maximizing the function of Map Reduce in a dissimilar cloud network (Yang, S.J., 2015). The deadline reduction scheduler is developed to reduce the deadline of a task in which it is collapsed while computing the massive data like video and image in Map Reduce frameworks (Hwang, Eunji, 2012). An independent agent which depends upon load balancing approach is presented for balancing a load by VMs with the help of 3 agents like Channel agent, load agent and migration agent (Singha, A., 2015).

This paper introduces an effective quasi oppositional dragonfly algorithm for load balancing (QODA-LB) in CC environment for optimal resource scheduling. The proposed QODA-LB algorithm derives an objective function utilizing three variables namely execution time, execution cost, and load for allocating tasks to VM with respect to its capacity. Besides, the QODA-LB algorithm incorporates quasi oppositional based learning (QOBL) concept to improve the convergence rate of classical dragonfly algorithm (DA). A series of simulations were takes place to ensure the effective performance of the QODA-LB algorithm and the results are examined under several aspects.

2. The Proposed QODA-LB Technique

Fig. 2 illustrates the system model of presented load balancing approach. A main aim of the projected model is for allocating all operations to VM according to the load capability of VM. The load balancing method guides for eliminating the tasks from overloaded VM and allocates to VM that is under a loaded stage. The projected approach is composed of massive data centers named as Physical Machines (PM) and the PM contains few VMs to precede the user's tasks. Every user of CC has diverse number of tasks to compute VM. Here, the loads are allocated to VM under the application of load balancing approach. The presented framework verifies the load of all VM in CC. The task of VM is based on computation time of all loads.

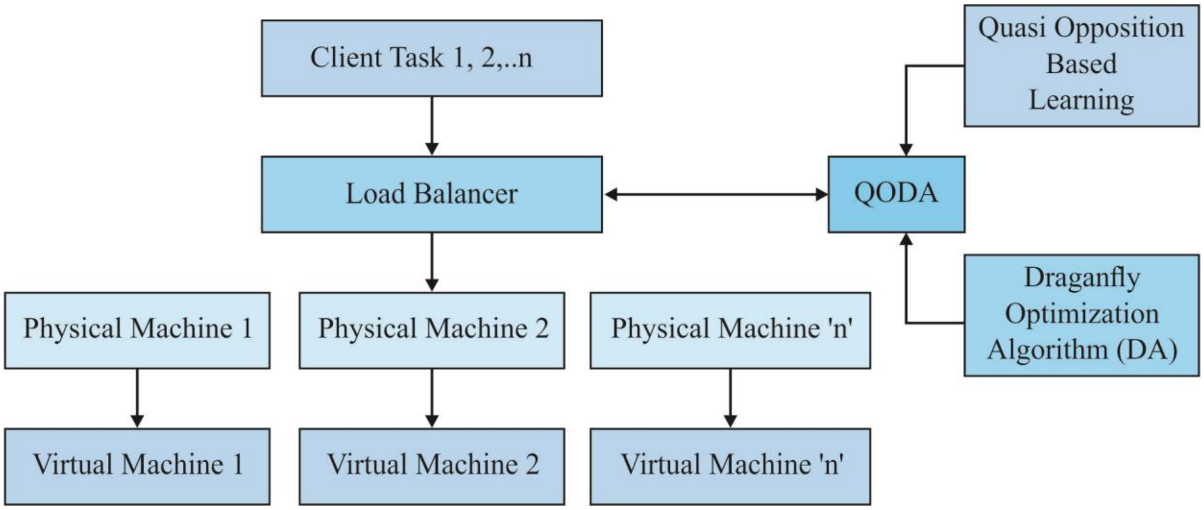

Figure 2. Proposed System Architecture

2.1 Problem definition with solution framework

Assume cloud C, that have "n" number of PM as well as PH is comprised of m" number of VM.

$$C = \{PM_1, PM_2, \ldots, PM_n\} \tag{1}$$

where C indicates the cloud, PM_1 shows the initial and PM while PM_n implies the nth PM which is represented in the following:

$$PM_n = \{VM_1, VM_2, \ldots, VM_m\} \tag{2}$$

where VM_1 is a first VM and VM_m refers the final VM. Likewise, icount of users is loaded in cloud and user is composed of icount of task. A user is represented in the following;

$$U_i = \{T_1, T_2, \ldots, T_j\} \tag{3}$$

The main theme of these models is to limit the execution time and expense of the task and to accomplish a balanced load entire VMs in CC system. Hence, it is operated using 3 objectives namely, limitation of task Execution Time, reduction of Execution Cost and finally, to scatter the burden for all VM in CC.

The overall execution time (ET) is defined with the help of Eq. (4).

$$ET = \frac{1}{\max(ET) \times numberoftask} \sum_{j=1}^{Numberoftask} (ETofcorresponding \text{ VM } \times Sizeofthetask) \tag{4}$$

The execution cost (EC) is determined using Eq. (5).

$$EC = \sum_{j=1}^{Nmberoftask} \left[\frac{Executiontime \times Communicationtime}{Numberoftask}\right] \tag{5}$$

Lode is determined using Eq. (3).

$$Load = \frac{1}{N\imath xmberoftask} \sum_{i=1}^{Numberoftask}$$

$$\left[\frac{sizeofVM - \left((TotalsizeofVM - FreespaceofVM) + Sizeoftask\right)}{SizeofVM}\right] \quad (6)$$

In order to accomplish the appropriate scheduling, load balancing should be performed accurately. In absence of balanced load, a system consumes higher time and cost to implement the operation. In order to resolve the issue, an effective multi-objective based load balancing scheme has been deployed (Neelima, P. 2020).The projected multi-objective function (MOF) is described in Eq. (7).

$$MOF = \min\left[\alpha_1\,(ET) + \alpha_2(EC) + \alpha_3(1 - Load)\right] \quad (7)$$

2.2 Proposed Load Scheduling Algorithm

The key objective of presented model is used for allocating a task to VM under the application of QODA-LB that reduces the overall ET and EC at the time of balancing load. The load is vital attribute of scheduling, where the method of scattering the load over diverse nodes of allocated modelfor enhancing resource application as well as task response time. In order to overcome these issues, multi-objective based load balancing has been deployed with the help of QODA-LB. DA is defined as meta-heuristic approach that is evolved by static and dynamic swarming nature of dragonflies. The dragonflies swarm for 2 objectives like Hunting (static swarm) and migration (dynamic swarm). First, massive dragonflies swarm at the time of roaming in longer distances and various territories those results in exploration phase.

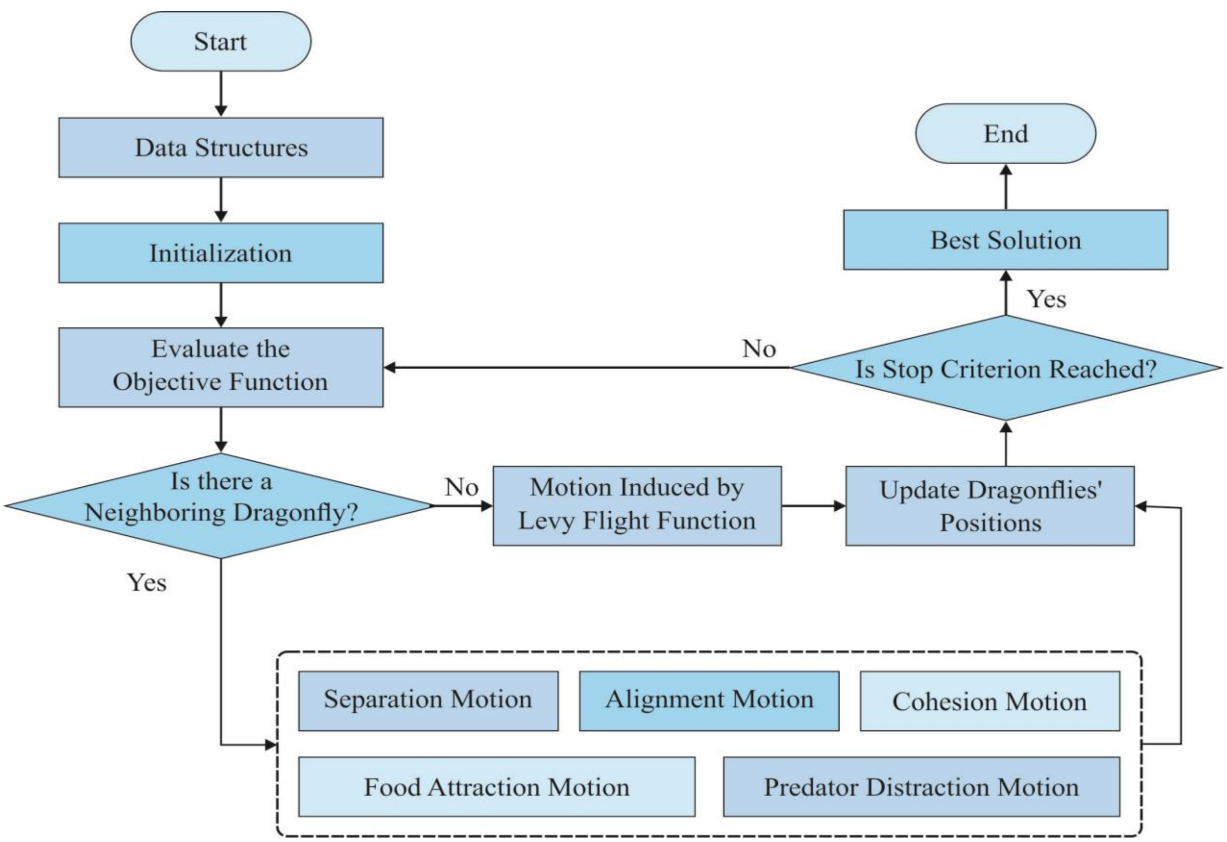

Figure. 3. Flowchart of Dragonfly algorithm

For static swarm, dragonfly's shifts in higher swarms and bearing by close to deployments and immediate modifications in flying direction, that leads to exploitation stage. For the enhancement of exploring capability and to eliminate the local optimal, QOBL is embedded throughDA. The flowchart of DA approach is showcased in Fig. 3. The periodical steps of presented multi- objective load balancing relied task arrangement are provided in the following;

Step 1: Initialization

A population of dragonflies is invoked arbitrarily. A population is composed of a group of results. The result is developed according to the count of user task as well as VM. At the initial stage, the tasks are allocated to VM in random manner. Followed by, according to the Fitness Function (FF) the results are maximized. A first result is provided in Eq. (1). A length of solution implies the supremacy of a task and dragonfly refers the available nodes. There are 10 operations and 5 nodes, and length of population is 10 and evaluation of all dragonflies might be 1, 2, 3, 4, and 5. A sample solution encoded has been offered in Eq. (8).

$$P_i = \{4,3,3,1,5,2,5,3,2,4\} \tag{8}$$

The previous function is showcased as 1 which is allocated toVM_4, task 2 is declared to VM_3 and task 10 is declared to VM_4. According to the solution encoding, population matrix (SM) has been implemented with 0 or 1The framework depends upon the association between task and VM. The task is related through VM i denotes PM(i,j) =1, or PM(i,j) =0. However, the column contains a single component for 1, otherwise0.

Firstly, the initial population is depending upon dragonflies.

Step 2: Fitness calculation

Once the solution is generated, the fitness of all solutions is calculated. The FF is provided in Eq. (9).

$$MOF = \min\left[\alpha_1\,(ET) + \alpha_2(EC) + \alpha_3(1 - Load)\right] \tag{9}$$

Step 3: Update using dragonfly algorithm

In order to enhance the solution, 5 major factors have been applied such asseparation, alignment, cohesion, attraction to food and distraction from opponent. Separation is estimated with the help of Eq. (10).

$$SP_i = -\sum_{j=1}^{N} X - X_j \tag{10}$$

where X- implies the location of present individual, X_j shows the location of jth neighboring individual and N represents the count of neighboring individuals. An alignment is measured utilizing Eq. (11).

$$A_i = \frac{\sum_{\mathrm{j=1}}^{\mathrm{N}} V_j}{N} \tag{11}$$

where y_j indicates the velocity of jth neighbouring individual. Hence cohesion is evaluated by Eq. (12).

$$C_i = \frac{\sum_{\mathrm{j=1}}^{\mathrm{N}} X_j}{N} - X \tag{12}$$

where X denotes the location of recent individual, N implies count of neighborhoods and X_j indicates the location of the jth neighbouring individual. Attraction to food source is determined with the given function:

$$F_i = X^+ - X \tag{13}$$

where X depicts the place of present individual and X^+ refers the place of food source. A direction visible an enemy is measured by:

$$E_i = X^- - X \tag{14}$$

where X signifies a place of recent individual, and X^- illustrates the location of the enemy.

Once the position is calculated, then velocity vector is determined by Eq. (15).

$$\Delta X_{k+1} = (sS_i + aA_i + cC_i + fF_i + eE_i) + w\Delta X_i \tag{15}$$

where s represents the separation weight, S_i is a separation of ith individual, a defines alignment weight, A_i implies alignment of ith individual, c refers the cohesion weight, C_i signifies cohesion of ith individual, f illustrates food factor, F_i showcases food source of ith individual, e denotes enemy factor, E_i refers position of enemy of the ith individual, w demonstrates inertia weight, and k depicts the iteration counter. Once the step vector is measured, the position vectors are estimated in the following:

$$X_{k+1} = X_k + \Delta X_{k+1} \tag{16}$$

where k is the present iteration.

Step 4: Select best solution

When the DA and QODA are compared, DA has optimal fitness value (DA_best) which is lower than DA fitness value (FA_best) , the best place of DA is interchanged by QODA. Otherwise, AODA fitness value is minimum than DA fitness value, the position is changed by DA.

Step 5: Termination criteria

The iteration is terminated if a better solution is attained. The consequent solution is submitted to CC platform.

2.3 Quasi-oppositional based learning (Q-OBL)

It accomplishes maximum attention from developers in Computation Intelligence (CI). The main aim of OBL in evolutionary processing is to maximize the solution accuracy and simulate the convergence rate to reach global solution. It contains higher probability in optimization method for generating suboptimal solution. Here, recent populations as well as inverse values are produced at the same time and results in optimal candidate solution. Hence, inverse value is produced correctly at mirror position of recent population. From (Guha, D., 2017), the opposite population contains optimal chance to attain global optimal solution when compared to randomly provided population. The OBL is obtained by describing 2 of significant mathematical features:

2.3.1 Opposite number

It is referred as a mirror point of candidate solution from center of search space. When Xis a number in real plane with search interval [a,b], the parallel opposite number (oX) in 1D search space is described by (17).

$$OX = a + b - X \tag{17}$$

where X implies the randomly invoked candidate solution, a and b are lower and higher limits of search space. The previous definition is improved to d-dimensional search space and formalized by (18).

$$OX_j = a_j + b_j - X_j \tag{18}$$

where $j = 1,2,---,d$ and $X_j = \{X_1, X_2, ---, X_d\} \in R$.

2.3.2 Quasi-opposite number

Furthermore, the mechanism of OBL based learning is improved to quasi-oppositional relied learning that showcases that quasi-opposite number is nearby global optimal solution when compared to opposite number. The quasi-opposite number is meant to be among the center of search space (^(a_j+b_j) /_2) while opposite number (a_j+b_j-X_j), is depicted by (19).

$$QOX_j = rand\left(\frac{a_j + b_j}{2}, a_j + b_j - X_j\right) \tag{19}$$

The pseudo code to accomplish quasi-opposite value is given in the following:

$C = {}^{a_j + b_j}/_{2};$

if$(oX < C)$

$QOX = C + (oX - C) * r_1;$

else

$QOX = OX + (C - OX) * r_1;$

end

where r_1 denotes a randomly produced value from (0,1).

2.3.3 Jumping rate

It assists the DA to move from recent solution to fresh candidate solution where it has optimal fitness value than present one. According to the jumping rate, as said in (20), novel population is developed and then quasi-opposite population has been determined. The selection of jumping value guides DA to eliminate sub-optimal solution and stimulate the DA to attain globally optimized solution. Basically, the jumping rate is decided from [0,0.6].

$$J_r = \left(J_{r,max} - J_{r,min}\right) - \left(J_{r,max} - J_{r,min}\right)\left(\frac{NFC_{max} - NFC}{NFC_{max}}\right) \tag{20}$$

where $J_{r,mae}$ and $J_{r,\min}$ are lower and higher value of jumping rate, $NFC_{\max}$ refers higher value in generation and NFC denotes the value of function call at present iteration.

3. Performance Validation

The performance validation of the QODA-LB algorithm takes place under several aspects and the proposed model is simulated using CloudSim tool. Since the goal of the QODA-LB algorithm is to allocate the tasks to VMs depending upon the capacity (load) of VM. The task is allocated on VM depending upon execution time, execution cost and load. For the validation of the experimental results of the QODA-LB algorithm, a series of simulations were carried out under diverse configurations namely (i) PM = 5, VM = 15 and 50 tasks, (ii) PM = 10 and VM = 30 and 75 tasks, (iii) PM = 20 and VM = 50 and 100 tasks.

3.1. Execution Time Analysis

Table 1 and Figs. 4-6 shows the execution time analysis of the QODA-LB algorithm under varying number of PMs, VMs and tasks. Fig. 4 shows the execution time analysis of the QODA-LB algorithm under 5 PMs, 15 VMs and 50 tasks. The experimental results stated that FA algorithm has demonstrated poor results by exhibiting maximum execution time. At the same time, the DA and ADA has tried to show certainly acceptable results by attaining slightly lower execution time. However, the QODA-LB model has exhibited better performance by attaining minimum execution time. For instance, under the iteration of 10, the QODA-LB algorithm has resulted to a minimum execution time of 0.32ms whereas higher execution time of 0.63s, 0.42s and 0.38s are incurred by the FA, DA and ADA techniques respectively.Likewise, under the iteration of 20, the QODA-LB model has lead to generate lower execution time of 0.33ms while maximum execution time of 0.65s, 0.43s and 0.39s are obtained by the FA, DA and ADA methods. Also, under the iteration of 30, the QODA-LB algorithm has provided least execution time of 0.33ms and higher execution time of 0.67s, 0.45s and 0.4s are attained by the FA, DA and ADA approach correspondingly. On the other side, under the iteration of 40, the QODA-LB model has shown lower execution time of 0.34ms while high execution time of 0.68s, 0.47s and 0.41s are obtained by the FA, DA and ADA techniques methods. Moreover, under the iteration of 50, the QODA-LB algorithm has demonstrated to a lower execution time of 0.35ms while greater execution time of 0.69s, 0.48s and 0.42s are achieved by the FA, DA and ADA techniques respectively.

Table 1 Execution Time (ms) Analysis

PM = 5, VM = 15 and Task = 50				
Iterations	FA	DA	ADA	QODA-LB
10	0.63	0.42	0.38	0.32
20	0.65	0.43	0.39	0.33
30	0.67	0.45	0.4	0.33
40	0.68	0.47	0.41	0.34
50	0.69	0.48	0.42	0.35
PM = 10 and VM = 30 and Task = 75				
Iterations	FA	DA	ADA	QODA-LB
10	0.93	0.86	0.73	0.65
20	0.95	0.87	0.75	0.67
30	0.95	0.87	0.76	0.68
40	0.95	0.88	0.77	0.68
50	0.95	0.89	0.78	0.69
PM = 20 and VM = 50 and Task = 100				
Iterations	FA	DA	ADA	QODA-LB
10	0.95	0.90	0.78	0.67
20	0.98	0.90	0.79	0.68
30	0.99	0.93	0.80	0.71

40	0.99	0.95	0.80	0.72
50	0.99	0.96	0.81	0.73

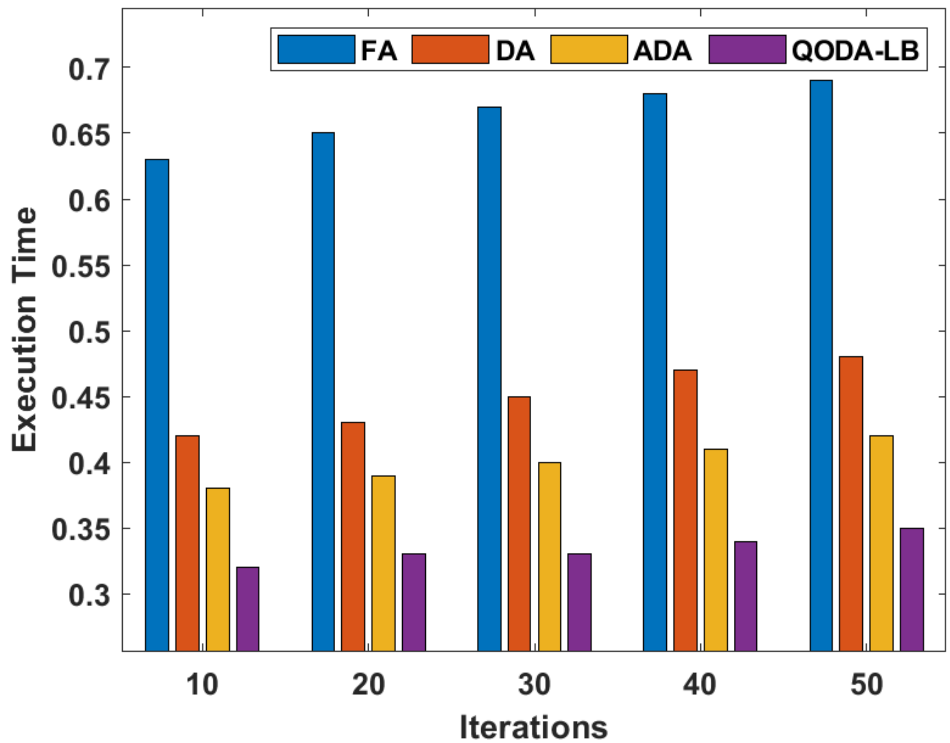

Figure 4. Execution time analysis of QODA-LB model under 5 PMs, 15 VMs and 50 tasks

Fig. 5 displays the execution time analysis of the QODA-LB algorithm under 10 PMs, 30 VMs and 75 operations. The experimental results have illustrated that FA algorithm has showcases inferior results by showing higher execution time. Meantime, the DA and ADA has attempted to show moderate results by accomplishing better execution time. Hence, the QODA-LB model has presented has showcases optimal function by reaching lower execution time. For sample, under the iteration of 10, the QODA-LB scheme has exhibited to a lower execution time of 0.65ms and higher execution time of 0.93s, 0.86s and 0.73s are obtained by the FA, DA and ADA techniques correspondingly. In line with this, under the iteration of 20, the QODA-LB framework has shown a lower execution time of 0.67ms while higher execution time of 0.95s, 0.87s and 0.75s are obtained by the FA, DA and ADA methods respectively. In addition, under the iteration of 30, the QODA-LB approach has shown a least execution time of 0.68ms and higher execution time of 0.95s, 0.87s and 0.76s are accomplished by the FA, DA and ADA methodologies respectively. Followed by, under the iteration of 40, the QODA-LB approach has depicted to a lower execution time of 0.68ms and higher execution time of 0.95s, 0.88s and 0.77s are achieved by the FA, DA and ADA techniques correspondingly. Moreover, under the iteration of 50, the QODA-LB approach has depicted a lower execution time of 0.69ms whereas maximum execution time of 0.95s, 0.89s and 0.78s are incurred by the FA, DA and ADA approaches respectively.

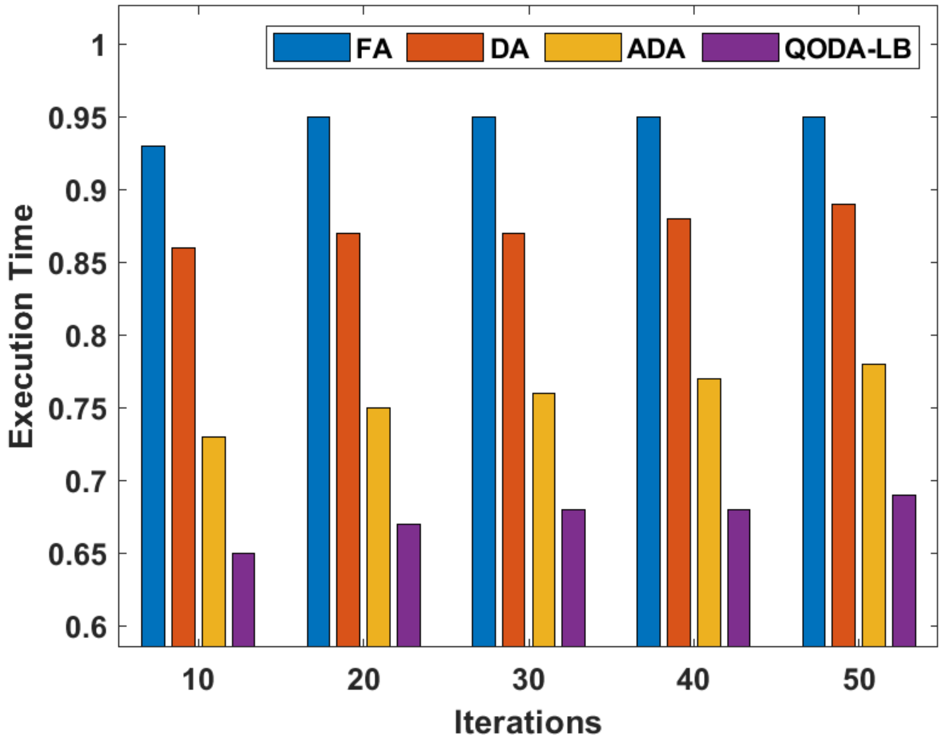

Figure 5. Execution time analysis of QODA-LB model under 10 PMs, 30 VMs and 75 tasks

Fig. 6 displays the execution time analysis of the QODA-LB method under 20 PMs, 50 VMs and 100 tasks. The experimental results stated that FA model has shown worst results by illustrating higher execution time. Concurrently, the DA and ADA has attempted to showcase better results by reaching minimum execution time. Therefore, the QODA-LB approach has implied moderate performance by reaching lower execution time. For sample, under the iteration of 10, the QODA-LB technique has showcased at least execution time of 0.67ms while maximum execution time of 0.95s, 0.90s and 0.78s are obtained by the FA, DA and ADA models correspondingly. Likewise, under the iteration of 20, the QODA-LB approach has provided to a lower execution time of 0.68ms and higher execution time of 0.98s, 0.90s and 0.79s are attained by the FA, DA and ADA methodologies correspondingly. In addition, under the iteration of 30, the QODA-LB approach has offered to a lower execution time of 0.71ms while maximum execution time of 0.99s, 0.93s and 0.80s are achieved by the FA, DA and ADA methodologies respectively. On the other side, under the iteration of 40, the QODA-LB method has generated to a lower execution time of 0.72ms whereas greater execution time of 0.99s, 0.95s and 0.80s are obtained by the FA, DA and ADA methods respectively. Furthermore, under the iteration of 50, the QODA-LB framework has showcased to a lower execution time of 0.73ms whereas high execution time of 0.99s, 0.96s and 0.81s are incurred by the FA, DA and ADA techniques respectively.

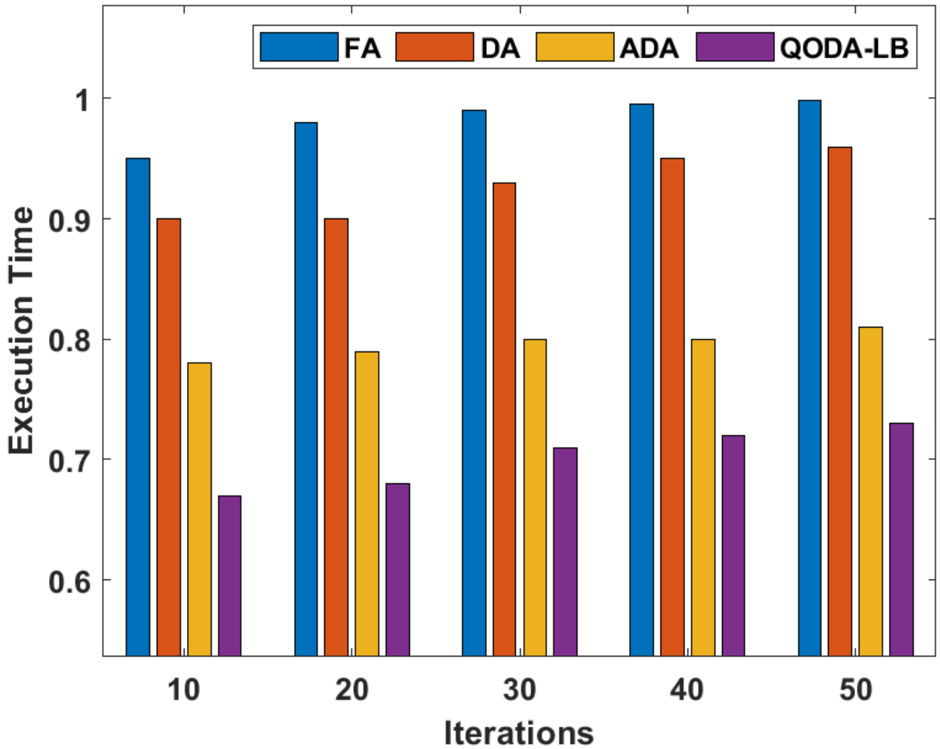

Figure6. Execution time analysis of QODA-LB model under 20 PMs, 50 VMs and 100 tasks

3.2. Execution Cost Analysis

Table 2 and Figs. 7-9demonstrates the execution cost analysis of the QODA-LB approach under varying number of PMs, VMs and tasks. Fig. 7 depicts the execution cost analysis of the QODA-LB algorithm under 5 PMs, 15 VMs and 50 tasks. The experimental results have pointed FA algorithm has shown worst results by showing higher execution cost. Simultaneously, the DA and ADA has attempted to depict considerable results by achieving lower execution cost. Hence, the QODA-LB approach has implied moderate function by accomplishing least execution cost. For sample, under the iteration of 10, the QODA-LB technology has provided to a lower execution cost of 0.69 while maximum execution cost of 0.80, 0.76 and 0.75 are obtained by the FA, DA and ADA methodologies correspondingly. Likewise, under the iteration of 20, the QODA-LB scheme has shown least execution cost of 0.51 while high execution cost of 0.70, 0.67 and 0.60 are accomplished by the FA, DA and ADA techniques correspondingly. Furthermore, under the iteration of 30, the QODA-LB algorithm has shown to a low execution cost of 0.67 while high execution cost of 0.79, 0.78 and 0.74 are incurred by the FA, DA and ADA methodologies correspondingly. Followed by, under the iteration of 40, the QODA-LB model has showcased a minimal execution cost of 0.68 whereas maximum execution cost of 0.80, 0.77 and 0.74 are obtained by the FA, DA and ADA techniques respectively. Furthermore, under the iteration of 50, the QODA-LB approach has provided to a low execution cost of 0.68 whereas higher execution cost of 0.80, 0.77 and 0.76 are incurred by the FA, DA and ADA techniques.

Table 2 Execution Cost Analysis

PM = 5, VM = 15 and Task = 50				
Iterations	FA	DA	ADA	QODA-LB
10	0.80	0.76	0.75	0.69
20	0.70	0.67	0.60	0.51
30	0.79	0.78	0.74	0.67
40	0.80	0.77	0.74	0.68
50	0.80	0.77	0.76	0.68
PM = 10 and VM = 30 and Task = 75				
Iterations	FA	DA	ADA	QODA-LB
10	0.19	0.12	0.08	0.07
20	0.22	0.13	0.08	0.07
30	0.26	0.14	0.08	0.07
40	0.28	0.15	0.09	0.08
50	0.28	0.16	0.13	0.09
PM = 20 and VM = 50 and Task = 100				
Iterations	FA	DA	ADA	QODA-LB
10	0.25	0.15	0.11	0.09
20	0.28	0.16	0.12	0.10
30	0.29	0.17	0.13	0.11
40	0.32	0.18	0.13	0.12
50	0.33	0.19	0.14	0.12

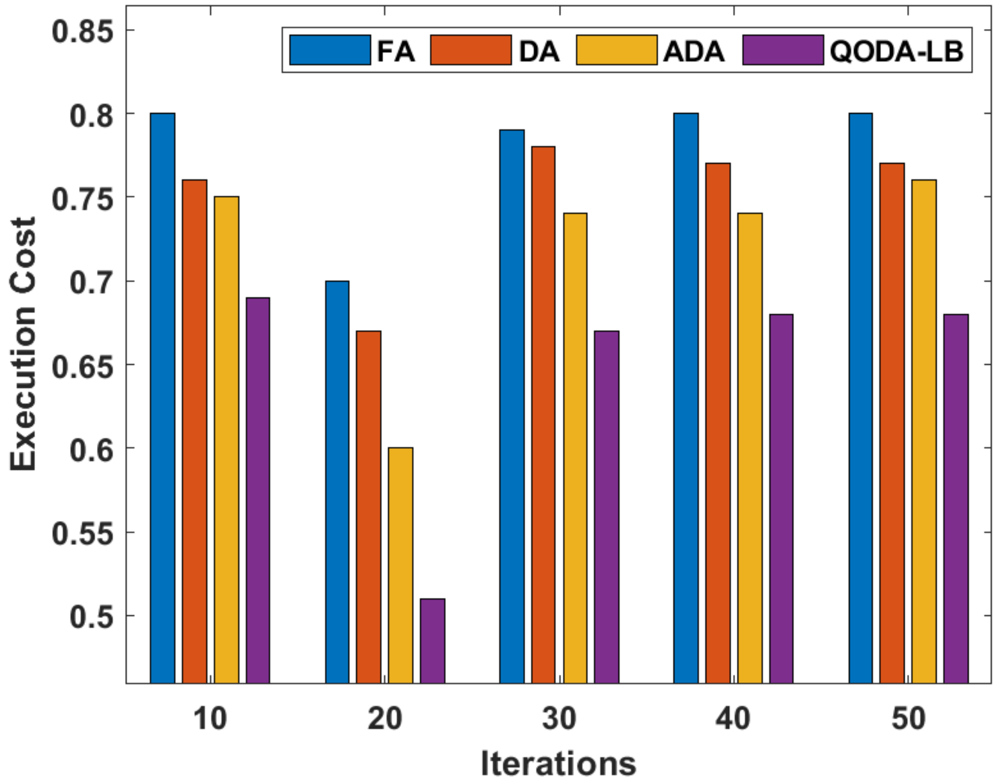

Figure7. Execution cost analysis of QODA-LB model under 5 PMs, 15 VMs and 50 tasks

Fig. 8 implies the execution cost analysis of the QODA-LB technology under 10 PMs, 30 VMs and 75 tasks. The experimental results have shown that FA model has depicted inferior results by showing higher execution cost. Meantime, the DA and ADA has attempted to represent better results by showing minimal execution cost. Hence, the QODA-LB model has implied optimal performance by accomplishing low execution cost. For sample, under the iteration of 10, the QODA-LB approach has depicted to a lower execution cost of 0.07 while maximum execution cost of 0.19, 0.12 and 0.08 are acquired by the FA, DA and ADA techniques correspondingly. In line with this, under the iteration of 20, the QODA-LB framework has showcased least execution cost of 0.07 whereas higher execution cost of 0.22, 0.13 and 0.08 are attained by the FA, DA and ADA models correspondingly. In addition, under the iteration of 30, the QODA-LB approach has provided to a lower execution cost of 0.07 while higher execution cost of 0.26, 0.14 and 0.08 are achieved by the FA, DA and ADA methodologies respectively. Then, under the iteration of 40, the QODA-LB technology has offered to a lower execution cost of 0.08 while maximum execution cost of 0.28, 0.15 and 0.09 are accomplished by the FA, DA and ADA technologies correspondingly. Moreover, under the iteration of 50, the QODA-LB algorithm has demonstrated to a least execution cost of 0.09 while maximum execution cost of 0.28, 0.16 and 0.13 are incurred by the FA, DA and ADA techniques respectively.

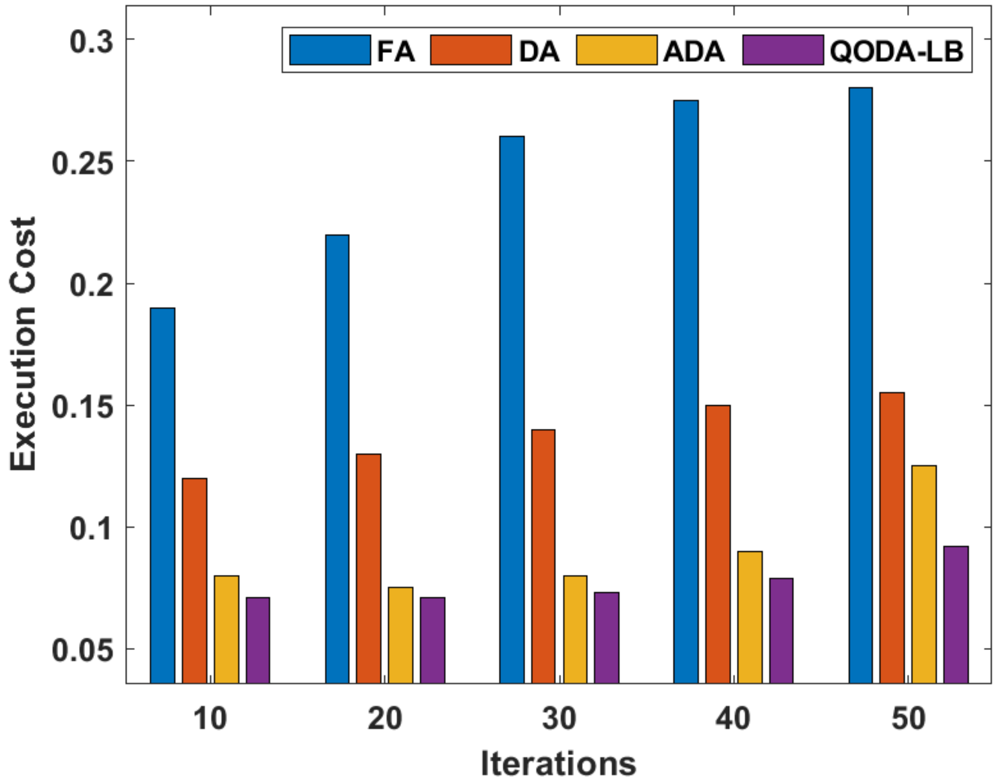

Figure. 8. Execution cost analysis of QODA-LB model under 10 PMs, 30 VMs and 75 tasks

Fig. 9 illustrated the execution cost analysis of the QODA-LB algorithm under 20 PMs, 50 VMs and 100 tasks. The experimental results have implied that FA algorithm has depicted worst results by showing higher execution cost. Concurrently, the DA and ADA has tried to depict moderate results by accomplishing better execution cost. Thus, the QODA-LB model has showcased considerable performance by reaching lower execution cost. For instance, under the iteration of 10, the QODA-LB method has illustrated to lower execution cost of 0.09 while maximum execution cost of 0.25, 0.15 and 0.11 are achieved by the FA, DA and ADA techniques respectively. Along with that, under the iteration of 20, the QODA-LB algorithm has provided to lower execution cost of 0.10 whereas maximum execution cost of 0.28, 0.16 and 0.12 are obtained by the FA, DA and ADA methodologies correspondingly. In addition, under the iteration of 30, the QODA-LB scheme has resulted to a lower execution cost of 0.11 while high execution cost of 0.29, 0.17 and 0.13 are obtained by the FA, DA and ADA models respectively.

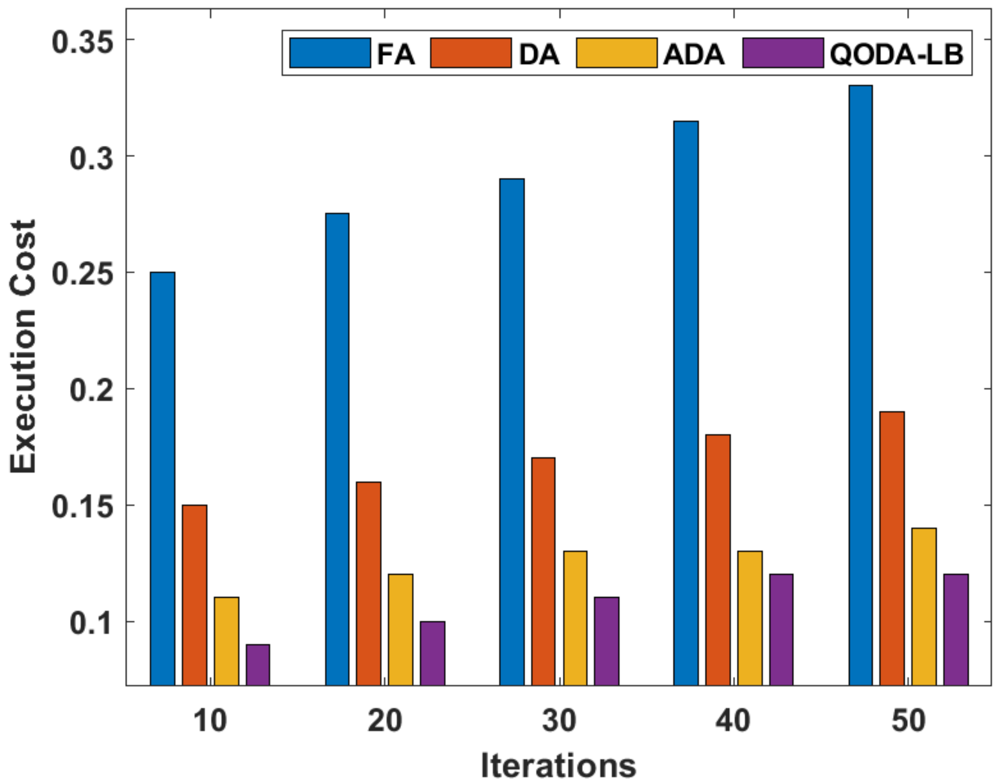

Figure. 8. Execution cost analysis of QODA-LB model under 10 PMs, 30 VMs and 75 tasks

Fig. 9 illustrated the execution cost analysis of the QODA-LB algorithm under 20 PMs, 50 VMs and 100 tasks. The experimental results have implied that FA algorithm has depicted worst results by showing higher execution cost. Concurrently, the DA and ADA has tried to depict moderate results by accomplishing better execution cost. Thus, the QODA-LB model has showcased considerable performance by reaching lower execution cost. For instance, under the iteration of 10, the QODA-LB method has illustrated to lower execution cost of 0.09 while maximum execution cost of 0.25, 0.15 and 0.11 are achieved by the FA, DA and ADA techniques respectively. Along with that, under the iteration of 20, the QODA-LB algorithm has provided to lower execution cost of 0.10 whereas maximum execution cost of 0.28, 0.16 and 0.12 are obtained by the FA, DA and ADA methodologies correspondingly. In addition, under the iteration of 30, the QODA-LB scheme has resulted to a lower execution cost of 0.11 while high execution cost of 0.29, 0.17 and 0.13 are obtained by the FA, DA and ADA models respectively.

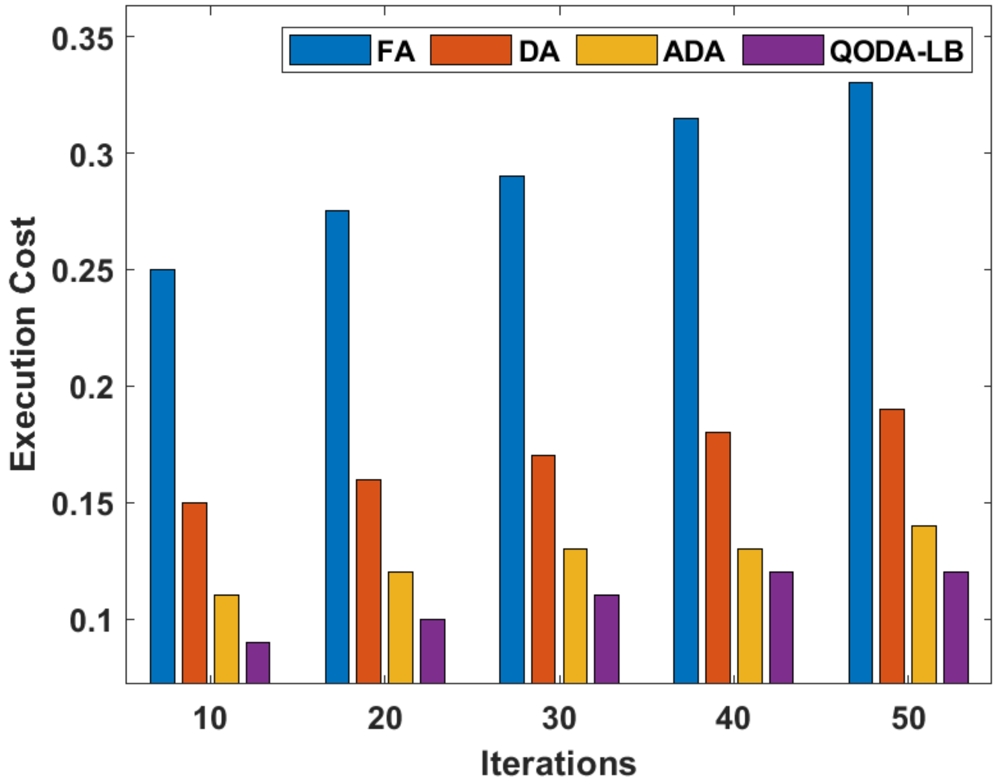

Figure. 9. Execution cost analysis of QODA-LB model under 20 PMs, 50 VMs and 100 tasks

Followed by, under the iteration of 40, the QODA-LB approach has provided to lower execution cost of 0.12 and high execution cost of 0.32, 0.18 and 0.13 are attained by the FA, DA and ADA techniques. Moreover, under the iteration of 50, the QODA-LB algorithm has offered to a lower execution cost of 0.12 whereas higher execution cost of 0.33, 0.19 and 0.14 are incurred by the FA, DA and ADA schemes correspondingly.

4. Conclusion

This paper has presented an effective QODA-LB algorithm in CC environment to achieve optimal resource scheduling. The main theme of this model is to limit the execution time and expense of the task and to accomplish a balanced load over all VMs in CC system.The proposed QODA-LB algorithm derives an objective function using three variables namely execution time, execution cost, and load. According to the derived objective function, the QODA-LB algorithm allocates tasks to VM with respect to its capacity. The simulation outcome has depicted optimal load balancing performance and demonstrated better results compared to state of art methods. A set of simulations were carried out to examine the execution cost and execution time analysis of the QODA-LB algorithm under varying number of PMs, VMs and tasks. In future, the performance of the QODA-LB algorithm is further enhanced by the use of deep learning models.

References

1. Alharbi, F., Rabigh, K.S.A., (2012). *Simple scheduling algorithm with load balancing for grid computing. Asian Trans. Comput. 2 (2), 8–15.*
2. Deng, X., Wu, D., Shen, J., He, J., (2016). *Eco-aware online power management and load scheduling for green cloud datacenters. IEEE Syst. J. 10 (1), 78–87.*
3. Deng, Y., Lau, R.W., (2014). *Dynamic load balancing in distributed virtual environments using heat diffusion. ACM Trans. Multimedia Comput. Commun. Appl. (TOMM) 10 (2), 16*
4. Devi, D. Chitra, RhymendUthariaraj, V., (2016.) *Load balancing in cloud computing environment using improved weighted round robin algorithm for nonpreemptive dependent tasks. Scientific World J.*

5. Ezumalai, R., Aghila, G., Rajalakshmi, R., (2010). *Design and architecture for efficient load balancing with security using mobile agents. Int. J. Eng. Technol. (IACSIT) 2 (1), 149–160 [Online].*
6. Guha, D., Roy, P. and Banerjee, S., (2017.) *Quasi-oppositional symbiotic organism search algorithm applied to load frequency control. Swarm and Evolutionary Computation, 33, pp.46-67.*
7. Hsiao, H.C., Chung, H.Y., Shen, H., Chao, Y.C., (2013). *Load rebalancing for distributed file systems in clouds. IEEE Trans. Parallel Distrib. Syst. 24 (5), 951–962.*
8. Hwang, Eunji, KyongHoon Kim, (2012). *Minimizing cost of virtual machines for deadline-constrained mapreduce applications in the cloud. In: Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing. IEEE Computer Society.*
9. Krishna, P. Venkata, (2013.) *Honey bee behavior inspired load balancing of tasks in cloud computing environments. Appl. Soft Comput. 13 (5), 2292–2303.*
10. Kanakala, V.R.T., Reddy, V.K., (2015.) *Performance analysis of load balancing techniques in cloud computing environment. TELKOMNIKAIndones. J. Electr. Eng. 13 (3), 568–573.*
11. Larumbe, F., Sanso, B., (2013.) *A tabu search algorithm for the location of data centers and software components in green cloud computing networks. IEEE Trans. Cloud Comput. 1 (1), 22–35.*
12. Mishra, Sambit Kumar, Sahoo, Bibhudatta, Parida, PritiParamita, (2018*). Load balancing in cloud computing: a big picture. J. King Saud Univ.-Comput. Information Sci.*
13. Mondal, B., Choudhury, A., (2015.) *Simulated annealing (SA) based load balancing strategy for cloud computing. (IJCSIT) Int. J. Comput. Sci. Information Technol. 6 (4), 3307–3312.*
14. Neelima, P. and Reddy, A.R.M., (2020). *An efficient load balancing system using adaptive dragonfly algorithm in cloud computing. Cluster Computing, pp.1-9.*
15. Selvi, R.T., Aruna, R., (2016). *Longest approximate time to end scheduling algorithm in Hadoop environment. Int. J. Adv. Res. Manag. Archit. Technol. Eng. 2 (6).*
16. Singha, A., Juneja, D., Malhotra, M., (2015). *Autonomous Agent Based Load-balancing algorithm in Cloud Computing. International Conference on Advanced ComputingTechnologies and Applications (ICACTA), 45, 832–841.*
17. Song, S., Lv, T., Chen, X., Feb. (2014). *Load balancing for future internet: an approach based on game theory. J. Appl. Math. 2014, (2014) 959782.*
18. Vasudevan, S.K., Anandaram, S., Menon, A.J., Aravinth, A., (2016). *A novel improved honey bee based load balancing technique in cloud computing environment. Asian J. Information Technol. 15 (9), 1425–1430.*
19. Yang, S.J., Chen, Y.R., (2015). *Design adaptive task allocation scheduler to improve MapReduce performance in heterogeneous clouds. J. Netw. Comput. Appl. 57, 61–70.*
20. Zhu, Y., Zhao, D., Wang, W., and He, H. (2016) *'A Novel Load 460 Balancing Algorithm Based on Improved Particle Swarm Optimization in Cloud Computing Environment', In International Conference on Human Centered Computing, Springer, pp. 634–645.*