ISSN: 0976-3104
SUPPLEMENT ISSUE
E.sharifi et al.

ARTICLE OPEN ACCESS

THE PROPOSED APPROACH FOR LOAD BALANCINGOF NODES IN THE CLOUD COMPUTING, USING A COMBINATION OF IMPERIALIST COMPETITIVE ALGORITHM AND GENETICS

E. Sharifi[1], R. Shamsi[2]

[1]*Department of Engineering, Qeshm International Branch, Islamic Azad University, Qeshm, IRAN*

[2]*Department of Mathematics, Minab Branch, Islamic Azad University, Minab, IRAN*

ABSTRACT

Internet has had a lot of progress from the beginning of its career so far, and has been causing changes in human life. Cloud computing service is one of the recent changes in the way the Internet works. Due to its features, this new technology has progressed quickly because in cloud computing all kinds of facilities, are provided for the users as a service. Naturally, any change and any new concept in the world of technology, has problems and complexities of its own. The use of cloud computing is no exception. It has caused enormous challenges for the experts in this field. Load balancing, security and reliability of the data are some of these challenges. In this study a combination of Imperialist Competitive optimization Algorithm and genetics is used for scheduling jobs and resources in cloud computing. The proposed method uses a combinatorial algorithm, which makes it easy to schedule and plan, causes the jobs to be processed in the minimum possible time, and makes the resources of the network to be balanced. Given the importance of load balancing process in the cloud computing, the aim of this thesis is to assess the process and compare the discussed methods in this field.

Published on: 25th– Sept-2016

KEY WORDS

cloud computing, optimization, network resource management, genetic optimization algorithm

***Corresponding author: Email:** shamsi.razii@gmail.com**Tel.:**00989173820318

INTRODUCTION

Using resources, cloud computing, provides the data, software, and equipment subscribing to a service requested by the client at a certain time. Of course, the length of time generally used in the Internet. The Internet can be considered as a cloud; operating and capital costs can be reduced by using cloud computing [1].

Each node in the cloud can be loaded in an inconsistent way of jobs being determined by the amount of work required by the customer. This phenomenon can significantly reduce the cloud performance; for some nodes that are loaded too much, spend more time to complete the job when compared with a node with less load under the same cloud. This problem is not only limited to one cloud but is related to any network such as a table, etc. as well. In The study, to have a better load distribution among the nodes we use a combination of genetic algorithm and Imperialist Competitive Algorithm. The recent cloud computing combines many nodes and provides services to users. There are many cloud computing service providers (CCSP) such as Google, Amazon and so on. Because they store cloud services as programs in their nodes, they can charge the customer with the primary information [2]. Therefore, the main defect of the cloud is in the fact that if the service providers have a slow speed because of traffic load or any other factors, the customers tend to refer to other service providers. A cloud is composed of different nodes that perform calculations based on customers' needs. The customers' requests can be attributed to random nodes so they can change their size and the load of the nodes can vary. So each node in the cloud can be loaded in an inconsistent way of jobs being determined by the amount of work required by the customer. This phenomenon can significantly reduce the cloud performance; for some nodes that are loaded too much, spend more time to complete the job when compared with a node with less load under the same cloud. This problem is not only limited to one cloud but is related to any network such as a table, etc. as well [3].

The load balancing of a network is one of the most difficult issues that must be done in the cloud computing system, the purpose of load balancing is to distribute the workload evenly across two or more computers, network links, CPUs, hard drives, and other resources. This is in order to achieve optimal utilization of resources, maximize throughput, minimize response time, and avoid overloads. The balance of resources is defined as a non-deterministic polynomial complete problem. Network resource management is a very challenging job because the shared resources are distributed and heterogeneous [4].

The main research question is;

Is load balancing of nodes in cloud computing done dynamically and intelligently and are good results achieved?

In this study, we sought to solve this question.

In this article we follow four hypotheses to be able to control the balancing of the load between different nodes in cloud computing; this will increase the popularity and efficiency of cloud computing technology.

1. Load balancing of nodes in cloud computing is done dynamically and intelligently and good results are achieved?
2. Genetic optimization algorithm and imperialist competitive algorithm are used to perform load balancing of nodes in cloud computing.
3. Balancing load of nodes is optimized in terms of speed and time when combining the Genetic optimization algorithm and imperialist competitive algorithm.
4. The comparison of the results of nodes' load balancing in cloud computing is done by combining imperialistic competitive algorithm and Genetics, and other certain algorithms are also used for optimization.

The imperialistic competitive and genetic optimization algorithms are combined here to achieve the above hypotheses successfully. There are different types of algorithms that can be used to balance resources in the grid computing system. Colorni, Dorigo, and Maniezzo (1991) used genetic algorithm and Tabu 9 to solve the problem of load balancing of network offers in a dynamic environment[2]. A study conducted by Humo and Saeed (2013) have studied on Towards a Reference Model for Surveying a Load Balancing [5]. The strategy of load balancing combination uses the successive jobs combined by static and dynamic load balancing strategies, which are offered in the study of Pavani and Waldman, 2006 when combining GA with FCFS. Lorpunmanee et al. (2007) presents ant colony optimization for job's dynamic programming in-network environment and aims to minimize the total time delay[6]. The ant colony optimization is used in another study on network's load balancing [3]. In the paper of Kumar and colleagues, the load balancing of nodes in the cloud is conducted by the use of ant colony optimization and has had good results [2].

MATERIALS AND METHODS

This method focuses on the reduction of the calculation time of each job and at the same time on the balance of all available resources in a network environment. This method, selects the resources based on the number of countries. A matrix is used which contains the number of countries in each resources in order to facilitate the selection of appropriate resources for processing jobs. The method contains four main components of the network information server, network resources server, jobs and resources; it works as described below:

1. User sends a request for the processing of work. The Information about the total number of jobs, the size of each job, and CPU's time required per jobs are included in each request.
2. Network resources' server, begin to calculate parameters related to the program works after receiving messages from users. In addition, the Information Server provides resource information for the network resources' server.

The largest amount of matrix' input (PV), will be selected through the proposed method to supply the proposed work process. The local updating of countries is done after assigning a job to a resource.

The global updating of the countries takes place after the resource completed a job. Results will be sent to the user. In the proposed method, a country (particle) represents a network system. Network resource' server obtains the available resources from the server of the network information. Particles move randomly in the network system and check the status of each resource. The value of particles at the source, show the capacity of each resource in the network system. The value of initial particles (countries) of each resources for each job is calculated based on an estimated transfer time and runtime of a jobassigned to that resource. The estimated transit time can be achieved by

$$\frac{S_j}{bandwidth_r}$$

In the above equation S_j is the given job's size, $bandwidth_r$ is the available bandwidth between the server of the network resource and the resource. The initial particle is determined by the following formula:

$$PV_{ij} = \left[\frac{S_j}{bandwidth_r} + \frac{C_j}{MIPS_r * (1 - load_r)}\right]^{-1}$$

THE IIOAB JOURNAL ISSN: 0976-3104

In the above equation, PV_{ij} is the amount of particles for the job of j assigned to the resource r. C_j is the amount of cpu required for the job j. $MIPS_r$is the r resource's processor speed, and 1- load is the current load of r resource. Time, processor's speed and bandwidth can be obtained from the server of network information.

Suppose there are n jobs and m resources in PV Matrix:

$$PV = \begin{array}{c} \\ r_1 \\ r_2 \\ .. \\ r_m \end{array} \begin{array}{c} \begin{array}{cccc} j_1 & j_2 & .. & j_n \end{array} \\ \begin{bmatrix} PV_{11} & PV_{12} & .. & PV_{1n} \\ .. & .. & .. & .. \\ .. & .. & .. & .. \\ PV_{m1} & PV_{m2} & .. & PV_{mn} \end{bmatrix} \end{array}$$

When fully processed, the local update and then the global update of particles (countries) is conducted to recalculate the entire PV matrix. After a solution was found for all particles, the routs of countries are updated according to the following formula.

$$\tau_{ir}(t+1) = (1-\rho)\tau_{ir} + \rho\Delta\tau_{ir}^{best}$$

In the above equation $\Delta \ \tau_{jr}^{best} = 1/L^{best}$ is the permitted countries added to the particles that may be the best duplicate solution or best global solution. If a particular resource is often used as the best solution, it will receive a larger amount of particles and recession will occur. Therefore, to avoid recession, upper and lower limits of each resource are applied on the possible strength point of the particles. The limitations of the imposed routs have an effect on the limitations of possible $P_{iu}$of the selected u resource, when the particles have a distance of [pmax, pmin] in the I resource; $0<p_{min}\leq p_{ij}\leq p_{max}\leq I$. A resource with smaller limits of a sequence is of less interest for being selected for jobs.Because the resources with more limits of a sequence are preferred [3].

In this experiment three jobs (J_1,J_2, J_3) are processed by three resources (R_1, R_2, R_3). The volume of each job is 15MB, 10MB and is 5MB; the information needed for the resources will be specified as well. It will be conducted for the jobs with higher volumes and the results will be compared with pso, Fa and ANT COLONY algorithms [3].

In the present study, we have used the idea of Husna Jamal, Ku Ruhana (2010) which has implemented the ant colony algorithm in the mesh processes. Using a combination of imperialistic competitive algorithm and the Genetics, we have presented a new idea for the balancing of the node's loads in cloud computing.

As described in the previous section we have used ICA and genetics which are two separate algorithms. in this section, we attempt to combine these two algorithms and categorize the new algorithms' jobs to have a better performance in terms of efficiency. According to researches of Lee, Leu, Chang (2011), some resources form a subfolder, and the combination of these several subfolders form a larger cluster. For each of these large clusters there is a local scheduler[7]. Jobs should be divided between them in a balanced way; there is an interface port for users to provide jobs. Information Server discovers registered resource's nodes. Global scheduler receives job, applies load balancing for each node (cluster), and Selects a larger cluster. Now the local scheduler selects a resource with the most powerful computing power among these sub-clusters to perform a given job. Upon completion of the implementation of new resources' jobs, the result of the implementation will be sent to information server. The global scheduler collects data from the information server, uses them to calculate the clusters' weight, and applied balance of loads. The weight of each large cluster is stored in the scheduler and scheduler uses it as a parameter for the new algorithm. The initial weight of each major cluster can be achieved for each job. The difference is that instead of using the CPU's speed, the average speed of the resources' processor or the average load of each major cluster's sub-clusters are used. In each replication, we select the major input of matrix, assuming that PI_{ij} is selected. Job j selects several sub clusters with the fastest average of computing power to be implemented; it is obtained through the following equation [7].

$$\mathrm{ACPi} = \frac{\sum_{k=1}^{n} CPU_{Speed\,K^k} * 1}{\mathrm{N}}$$

CPU_Speedk is the processor's speed of resource k in the sub cluster i. CPU_K is the productivity of the current processor of the resource k in the sub cluster of i. n is the number of resources in the sub cluster I. Then the scheduler in each selected cluster selects a resource with the fastest-average computing power and the resource that runs faster sends the result. After the work is assigned to a resource, the local update (row) is applied in the matrix pi. The global updating reflects the changes in network conditions and the status of resources after a job completion. In the proposed approach, the scheduling of jobs is implemented optimally and it will not exceed the considered time for the job. The optimization of scheduling is done through the combination of imperialist competitive algorithm and genetics. The differences in our methods and others' can be studied from two fundamental aspects:

- Time, speed and accuracy of the job
- Obtaining answers in the initial performances

The advantage of the combination of these two algorithms is that the scheduling of the jobs is done with the speed and accuracy. It should be noted that it can also be implemented with other programming languages such as C #, C ++, Clod Sim and other languages. Here we have used the programming language of MATLAB. The simulation results are given in the next section. In the proposed procedure, 50 percent of the countries have used the policy of assimilation to move towards the imperial power and the remaining 50% have used the combining method in the algorithm. Details of the procedure are as follows.

1. Establishment of the countries, initialization and evaluation of the countries
2. The determination of colonizers and the allocation of colonies to them in order to form an Empire
3. The Move of colonies toward the colonizers

3.1. fifty percent of the countries have used the policy of assimilation to move towards the imperial power

3.2. Fifty percent have used the combining method shown in **[Figure- 1].**

C_{ij}	0	1	1	0	0	1	1	0	0
Im_j	0	0	1	1	1	0	1	0	1
G_imp	1	1	1	0	0	1	0	0	1

New-con1	0	1	1	0	0	1	1	0	1
New-con2	0	0	1	0	0	1	0	0	1
New-con3	1	1	1	0	0	1	1	0	0

Fig: 1. Create new country with combining method

....

Here, $C_{ij}$is the ith country in the Jth Empire

Im_j: Jth Empire's clonizer

G_imp: the colonizer of the best Empire

New- con 1,2,3: the new established countries

Each of the three new countries that are less expensive than C_{ij} would be replaced with it.

4. Revolution
5. In- group competition
6. Out-group competition

RESULTS AND DISCUSSION

In this section, we implement our method. The implementation involves the fact that the load-balancing problem in MATLAB is simulated and modeled. To demonstrate the combination of imperialist competitive algorithm and genetic algorithm, we have considered two small, medium and large systems. The existing issues are implemented by a combination of genetic algorithm and imperialist competitive algorithm. At the end of the simulation results are analyzed.

Implementation on the small system

Husna Jamal, Ku Ruhana (2010) have implemented the ant colony algorithm in the mesh processes to balance the nodes. In this experiment three jobs (J_1,J_2, J_3) are processed by three resources (R_1, R_2, R_3). The volume of each jobis 15MB, 10MB and is 5MB; the information needed for the resources will be shown in **Table- 1**.

Table 1: Information on resources

Status	R1	R2	R3
Processor speed(MIPS)	250	540	600
Bandwidth(Megabits/s)	15.40	35.50	42.37

According to the simulation results of 5, 10, 15 MB jobs, the Cpu Time assigned to each of the many different resources is shown in **Table- 1**. Then in the **table- 3.5** we have shown the amount of bandwidth of each load and in Table 1 we have indicated the percentage of each resources' capacity allocated to different loads..

Table 2: Cpu Time assigned to each load

		J1	J2	J3
Cpu Time	1	0.3160	0.5949	0.4894
	2	0.6740	0.4485	1.4906
	3	0.1994	1.3634	1.7244

The servers of network resources in matrix PV select the maximum amount of countries PV22. So R2 processes J2. After the assignment of J2 to R2, the local updates of the countries occur in the second row of R2. The third column is no longer needed because it is assigned to J3. The new PV matrix is as follows:

Table 3: The amount of bandwidth of each load

		J1	J2	J3
PV	BND	1.5983	6.1992	7.6025
		3.4372	4.5950	27.4678
		1.0003	14.0948	27.2749

Table 4: The resources' capacity allocated to loads by resources

		J1	J2	J3
load	1	44.3037	41.1517	14.5446
	2	28.3941	21.7003	49.9056
	3	52.9694	30.5203	16.5103

Based on 10 times of implementation of the program by different algorithms, the convergence results were extracted in **Table- 5**.

Table 5: Comparison of different algorithms in medium system

'algorithm	convergence results
ICA-GA	7.6851
PSO	9.0633
ANT COLONY	10.49581
Bee Colony	11.1226

The convergence of the algorithms is presented in **Figure- 2**.

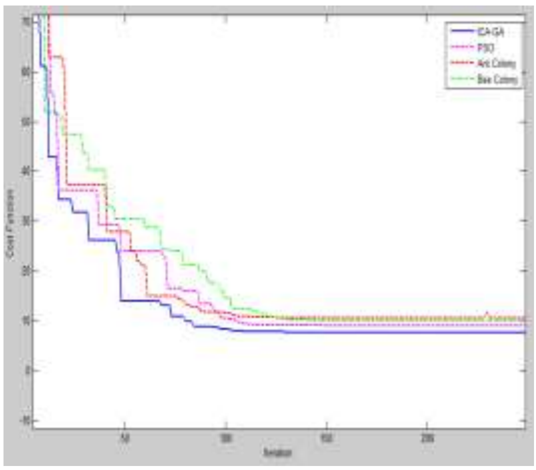

Fig: 2. Comparison of objective function's convergence chart using different algorithms of small systems.

……..

The objective function's separated convergence Chart is presented in **Figure- 2** using the combined genetic algorithm and imperialist competitive algorithm;

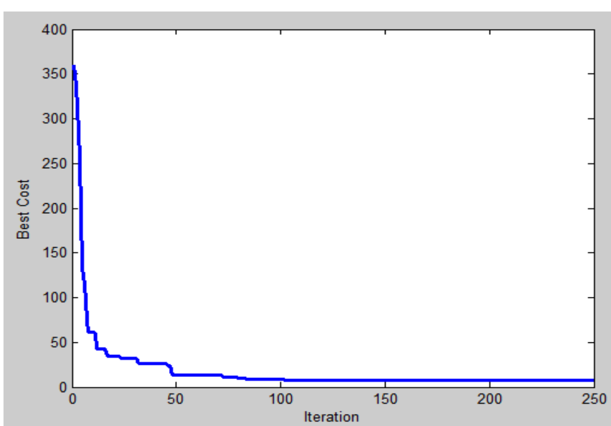

Fig: 3. The objective function's convergence chart using the combined genetic algorithm and imperialist competitive algorithm in the small system

…….

Implementation on the medium system

In the experiment conducted on the medium systems, the sizes of jobs are increased. According to the existing conditions, the ability of the combined algorithm is assessed. In this experiment five jobs (j1, j2, j3, j4 , j5) are evaluated by (R1, R2, R3, R4, R5) resources. The volume of each jobs is 150.60 Mb, 210.40 MB, 230.80Mb, 260.75 MB, 290.40 MB. The data is shown in **Table- 6**.

Table 6. Information on resources

status	1	2	3	4	5
Processor speed(MIPS)	20	50	10	67	70
Bandwidth(Mega bits/s)	8.56	3.20	1.22	7.36	2.47

Like the previous sections, the Cpu Time assigned to each of the many different resources is shown in **Table- 7**. Then in the table 8 we have shown the amount of bandwidth of each load and in **Table- 9** **we have indicated the percentage of each resources' capacity allocated to different loads.**

Table 7. Cpu Time assigned to each load

		J1	J2	J3	J4	J5
Cpu Time	R1	0.212	0.0356	0.0340	0.0324	0.0356
	R2	0.0317	0.0278	0.0690	0.0094	0.0430
	R3	0.0141	0.0125	0.0105	0.0092	0.1050
	R4	0.0289	0.0175	0.0703	0.0830	0.0319
	R5	0.0138	0.0951	0.0034	0.0460	0.0803

Table: 8. The amount of bandwidth of each load

		J1	J2	J3	J4	J5
BND	R1	3.2263	7.6893	8.1223	8.8096	10.7125
	R2	4.8447	6.0538	16.8464	2.4688	12.9864
	R3	2.1417	2.6470	2.4508	2.4215	41.5589
	R4	4.5353	3.7242	16.8213	22.8362	9.4430
	R5	2.0928	20.7547	0.7854	12.3345	26.5026

Table :9. The resources' capacity allocated to loads by resources

		J1	J2	J3	J4	J5
load	R1	17.8631	8.4333	26.2854	36.6571	10.7612
	R2	2.2688	49.2235	16.8371	26.9520	4.7185
	R3	3.0346	3.8843	15.3693	6.1689	72.0430
	R4	76.2703	9.6197	0.4309	11.4461	2.2330
	R5	3.8351	2.5599	5.3386	23.2043	66.0622

Based on 10 times of implementation of the program by different algorithms, the convergence results were extracted in **Table -10**.

Table 10. Comparison of different algorithms in medium system

algorithm	convergence results
ICA-GA	0.961445
PSO	0.987957
ANT COLONY	1.066721
Bee Colony	1.090439

The convergence chart of algorithms is shown in **figure- 4**.

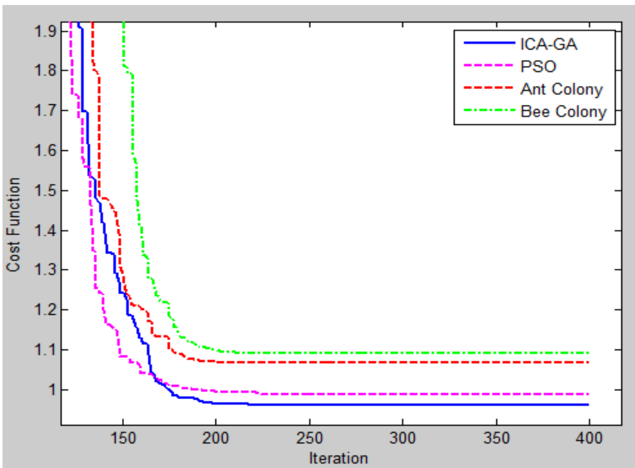

Fig: 4. Comparison of objective function's convergence chart using different algorithms of medium systems.

………

The objective function's separated convergence Chart is presented in Figure 5 using the combined genetic algorithm and imperialist competitive algorithm;

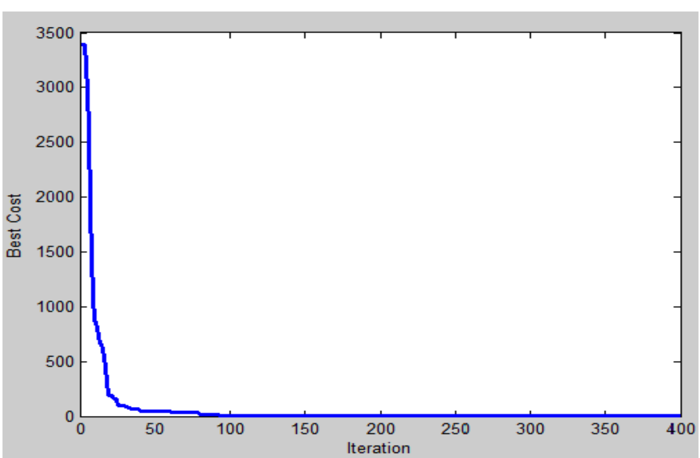

Fig:5. The objective function's convergence chart using the combined genetic algorithm and imperialist competitive algorithm in the medium system

………

Implementation on the large system

In the experiment conducted on the large systems, the sizes of jobs are increased significantly. According to the existing conditions, the ability of the combined algorithm is assessed. In this experiment fifteen jobs (j1, j2, j3, j4 ,

THE IIOAB JOURNAL ISSN: 0976-3104

j5, j 6, j7, j8, j9, j10, j11, j12, j 13, j14, j15) are evaluated by (R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R11, R12, R13, R14, R15) resources. The volume of each jobs is 150.60 MB, 210.40 MB, 230.8MB, 260.75 MB, 290.40 MB, 300.10 MB, 320.28 MB, 350.24 MB, 370.36MB, 385. 89MB, 400.27 MB, 420.45MB, 450.37MB, 500.87MB, 510.26MB. The data is shown in **Table- 11**.

Table 11. Information on resources

Status	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12	R13	R14	R15
Processor speed(MIPS)	470	490	505	520	560	590	605	620	680	715	745	810	850	920	940
Bandwidth (Megabits/s)	40.35	48.32	50.44	53.21	57.17	58.44	60.25	64.22	68.45	73.17	77.26	84.25	86.39	91.50	95.36

Like the previous sections, the Cpu Time assigned to each of the many different resources is shown in **Table- 12**. Then in the **Table -13** we have shown the amount of bandwidth of each load and in Table 14 we have indicated the percentage of each resources' capacity allocated to different loads.

Table :12. Cpu Time assigned to each load

	JJ1	JJ2	JJ3	JJ4	JJ5	JJ6	JJ7	JJ8	JJ9	JJ10	JJ11	JJ12	JJ13	JJ14	JJ15
RR 1	0.0440	0.0027	0.0045	0.0231	0.0118	0.0270	0.0050	0.0381	0.0324	0.0053	0.0174	0.2387	0.1695	0.0257	0.0853
RR 2	0.0250	0.0119	0.0224	0.0323	0.0343	0.0009	0.0993	0.0305	0.1054	0.1550	0.0369	0.0346	0.0024	0.0474	0.1365
RR 3	0.0481	0.0021	0.0003	0.0063	0.0329	0.1352	0.1820	0.0262	0.0798	0.1102	0.0294	0.0580	0.1622	0.0611	0.0030
RR 4	0.0128	0.0332	0.0027	0.0048	0.0005	0.0569	0.0695	0.0525	0.0544	0.1257	0.0181	0.0445	0.1628	0.0927	0.0964
RR 5	0.0591	0.0020	0.0106	0.0051	0.0215	0.0562	0.0672	0.0171	0.0181	0.0172	0.0240	0.0412	0.0935	0.0100	0.1069
RR 6	0.0221	0.0151	0.0088	0.0004	0.0449	0.1208	0.0257	0.1184	0.0376	0.0086	0.0588	0.0065	0.0860	0.0559	0.0950
RR 7	0.0003	0.0042	0.0067	0.0073	0.0129	0.0592	0.1565	0.0196	0.0190	0.0034	0.0351	0.0089	0.0411	0.1144	0.0782
RR 8	0.0016	0.0524	0.0060	0.0208	0.0118	0.0836	0.0436	0.0338	0.0210	0.0493	0.0562	0.1266	0.0331	0.0655	0.0312
RR 9	0.0290	0.0871	0.0876	0.0032	0.0036	0.0046	0.0275	0.0377	0.0552	0.0010	0.0991	0.0748	0.0955	0.1188	0.0133
RR 10	0.0019	0.0186	0.0017	0.0018	0.0386	0.0483	0.0200	0.0397	0.0005	0.0675	0.0926	0.0436	0.0772	0.0480	0.0168
RR 11	0.0017	0.0041	0.0012	0.0379	0.0005	0.0874	0.0734	0.0426	0.0697	0.0832	0.0381	0.0434	0.0128	0.0082	0.0056
RR 12	0.0074	0.0000	0.0595	0.0678	0.0127	0.0747	0.0060	0.0547	0.0410	0.0068	0.0150	0.0332	0.0200	0.0391	0.0152
RR 13	0.0306	0.0008	0.0040	0.0050	0.0061	0.0403	0.0033	0.0330	0.0361	0.0715	0.0276	0.0399	0.0155	0.0809	0.0005
RR 14	0.0213	0.0012	0.0000	0.0167	0.0036	0.0527	0.0061	0.0361	0.0614	0.0039	0.0188	0.0040	0.0572	0.0046	0.0693
RR 15	0.0062	0.0188	0.0120	0.0023	0.0093	0.0459	0.0330	0.0330	0.0573	0.0047	0.0113	0.0180	0.0471	0.0169	0.0171

Table:13. The amount of bandwidth of each load

	JJ1	JJ2	JJ3	JJ4	JJ5	JJ6	JJ7	JJ8	JJ9	JJ10	JJ11	JJ12	JJ13	JJ14	JJ15
RR1	6.7008	0.4103	0.6762	3.5087	1.7910	4.3127	0.7474	86.0294	5.0915	0.8024	2.6556	52.6924	28.9365	12.9899	24.4018
RR2	5.3025	2.5079	4.7519	6.8769	7.3323	0.1843	23.0863	6.6475	23.8110	55.8892	8.1319	7.5848	0.4994	12.8544	72.5551
RR3	11.3055	0.4841	0.0656	1.4587	7.7190	45.8637	85.3845	6.2008	37.9771	30.8815	7.2407	15.2647	54.1270	15.6991	0.6899
RR4	3.3553	8.7945	0.7044	1.2669	0.1349	20.4973	18.9956	14.5473	20.7743	84.2416	4.8806	41.6379	53.2575	40.1177	29.4406
RR5	17.7151	0.5943	3.0891	1.4999	6.3283	17.8928	20.5530	77.0436	5.4253	5.1440	10.7886	56.5153	30.7789	2.9912	68.7951
RR6	6.7107	4.5620	2.6629	0.1160	13.8561	40.1649	39.7263	58.6426	11.7655	2.5997	18.7000	1.9851	38.8454	21.6355	35.7309
RR7	0.0882	1.3445	2.1407	2.3524	4.1589	22.1683	67.0177	10.5059	10.6493	15.5330	12.5421	2.9266	16.5639	53.8381	31.0132
RR8	0.5604	18.9771	2.1153	7.3940	4.1836	31.9630	15.9661	12.2786	50.0215	18.6353	24.6947	61.9763	33.0540	27.3159	21.7825
RR9	10.9292	34.1249	34.6832	1.1841	1.3510	1.7982	55.7822	15.1004	22.5570	0.3735	44.1987	39.8748	43.3742	53.9561	8.9925
RR10	0.7150	7.3141	0.6643	0.6960	15.4423	19.8362	8.5600	16.0867	0.1911	30.7526	41.7835	40.1171	50.3858	20.3452	43.1821
RR11	0.6744	1.6474	0.4967	15.6133	0.1820	44.3453	52.4773	17.7882	33.6821	44.1450	22.2139	23.8796	5.2827	3.3191	28.7590
RR12	3.1082	0.0131	26.3544	30.3260	5.3860	41.2056	2.5513	33.6982	20.1077	18.6340	6.4425	20.2579	9.4532	17.6271	6.6603
RR13	14.1234	0.3625	1.7995	2.2565	2.7860	19.5149	1.4946	15.6708	17.0613	38.0188	13.9500	22.8710	21.9080	42.8395	0.2131
RR14	10.8759	0.6010	0.0117	8.5211	1.8075	28.8494	3.1012	19.4618	35.4440	1.9744	9.7167	26.1054	32.0745	2.5494	46.4209
RR15	3.1936	9.7577	6.1741	1.1781	4.8019	40.3329	17.5608	28.3050	31.3292	18.7702	7.8291	9.8201	27.2473	12.7487	11.2705

Table 14. The resources' capacity allocated to loads by resources

	JJ1	JJ2	JJ3	JJ4	JJ5	JJ6	JJ7	JJ8	JJ9	JJ10	JJ11	JJ12	JJ13	JJ14	JJ15
RR1	5.4097	6.9926	41.2861	3.3916	0.6603	85.0141	15.3758	98.6915	73.7368	61.5674	35.8290	62.2924	22.4469	98.0231	90.2337
RR2	4.5428	9.5298	3.8282	0.8042	0.0128	73.2897	51.3154	60.1258	26.6601	73.8903	50.4620	40.1320	14.7691	84.0359	82.9017
RR3	0.6324	2.5743	6.8184	0.8024	0.3484	79.2074	81.6251	46.9339	91.8629	51.9589	69.6455	63.8710	56.6400	50.2213	61.3505
RR4	1.8644	5.0297	0.1827	2.5175	5.2988	88.5722	12.4873	50.2655	89.8316	82.1167	59.2009	94.6434	25.3598	78.2108	38.0879
RR5	6.0240	10.3703	9.5780	13.1658	16.1082	60.5282	13.1765	98.8705	59.4351	38.9202	94.3686	94.9470	18.0456	57.2288	79.6603
RR6	5.6510	1.6681	12.6523	1.7567	5.5275	20.6277	97.8723	80.8189	38.8156	15.7213	9.2768	47.3888	72.5051	73.2271	49.1528
RR7	0.4302	1.6483	0.9308	30.2104	1.8441	72.0040	55.8628	96.6830	96.7703	99.6076	69.4240	60.0895	77.1791	58.8536	53.3306
RR8	7.9172	2.6854	2.5115	11.1990	3.5633	25.4471	21.9333	28.0278	98.0367	22.6786	72.3280	48.0516	93.6158	48.4717	92.0923
RR9	5.7375	0.2028	8.0665	0.4537	2.6498	91.9848	97.2335	60.8610	50.7926	23.0077	38.3480	69.7251	31.4925	14.6086	96.0913
RR10	16.9452	27.4852	0.7465	2.5339	16.7469	34.6038	82.4211	32.6753	55.3497	43.2251	29.7322	90.3258	73.9485	26.4948	97.2593
RR11	2.1972	0.5474	0.1776	0.6162	22.8488	64.6773	85.1802	12.2760	63.1904	54.7811	86.0834	78.7264	40.2819	24.1498	99.1243
RR12	10.2035	8.3351	8.9450	8.1551	8.0315	71.9260	25.7436	84.5931	72.5418	98.8743	15.5106	85.0037	72.4046	11.2195	41.1056

Based on 10 times of implementation of the program by different algorithms, the convergence results were extracted in Table 15.

Table 15. Comparison of different algorithms in the large system

algorithm	convergence results
ICA-GA	9.19689
PSO	9.399799
ANT COLONY	11.42051
Bee Colony	11.43052

The convergence chart of algorithms is shown in figure 6.

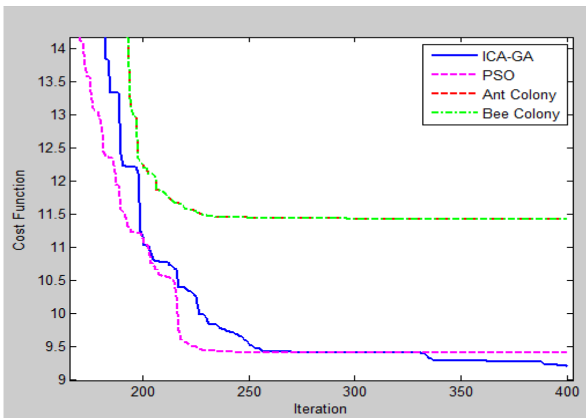

Figure 6. Comparison of objective function's convergence chart using different algorithms of large systems.

The objective function's separated convergence Chart is presented in Figure 7 using the combined genetic algorithm and imperialist competitive algorithm;

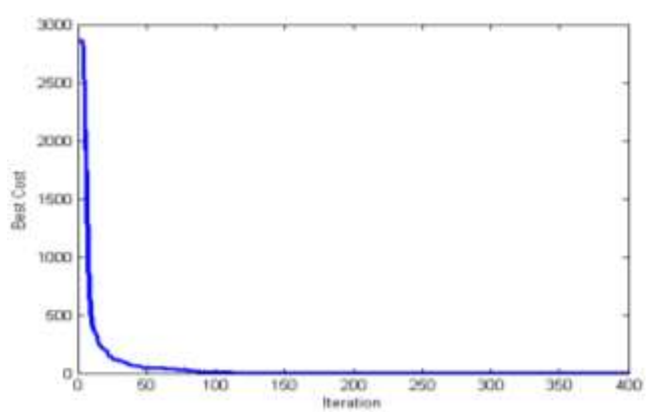

Figure 7. The objective function's convergence chart using the combined genetic algorithm and imperialist competitive algorithm in the large system

In this study, the combined algorithm of genetic and imperialist competitive was used to achieve load balancing in and network scheduling tasks. We implemented the combined algorithm on a series of hypothetical sets and examined the number of various tasks. The results show that the combination of genetic algorithm and imperial competitive algorithm remarkably balances the tasks between different nodes and achieves good results. Load balancing process in the proposed method is based on local updating of the countries. Local Updating of the countries reduces the value of the country in the dedicated resources. To ensure that the resource, in the limitation of the sequence, is less desirable for other countries, the permitted range of countries is limited to maximum and minimum power sequence.

CONCLUSION

This technique is used for controlling the amount of updated countries in each resource. The proposed method, is simply implemented because of the information from each resource and each job. By using this method, the load on each balanced resources and time of implementation on any job can be minimized. In this chapter we have introduced our methodology and resources needed for simulation are expressed. Then adjusting the parameters of the algorithm and various algorithms, we have shown the efficiency of our combined genetic and imperial competitive algorithm. Drawing convergence charts, we indicated the desirable efficiency of our combined genetic and imperial competitive algorithm. The results of the simulations showed that the combined genetic and imperialist competitive algorithm had a better performance than other algorithms.

Further studies are required to implement load balancing of nodes in combined cloud environments and examine the priority of the tasks policies of countries. Alternatively, other optimization algorithms can be used to implement this policy or e high speed and precision policies can also be added to this task.

CONFLICT OF INTEREST
Authors declare no conflict of interest

ACKNOWLEDGEMENTS
None

FINANCIAL DISCLOSURE
None

REFERENCES

[1] Swain R, Padhy L.N and Rao R.B[2011]. "*BENEFICIATION STUDIES ON BAUXITE MINING WASTE: A VALUE ADDITION FOR REFRACTORYINDUSTRIES*", *Iranian Journal of Materials Science & Engineering Vol. 8, NO 3,* 37-49.

[2] Colorni A, Dorigo M, and Maniezzo V.[1991] *Distributed optimizationby ant colonies*" presented at Proceedings of the First EuropeanConference on Artificial Life, Paris, France, Amsterdam: ElsevierPublishing, pp. 134-142.

[3] Jamal H, Nasir A, Ruhana K, Mahamud K and Din A.M.[2010] "*Load Balancing Using Enhanced Ant Algorithm in Grid Computing*",Proceedings of the Second International Conferenceon Computational Intelligence, Modelling and Simulation, pp.160-165.

[4] Pavani G and Waldman H.[2006] *Grid resource management by means of ant colony optimization*," 3rd International Conference on Broadband Communications, Networks and Systems (BROADNETS).

[5] Hamo A and Saeed A.[2013] *Towards a Reference Model for Surveying a Load Balancing*",IJCSNS International Journal of Computer Science and Network Security , Vol.13.

[6] Lorpunmanee S, Sap M, Abdullah A, and Chompoo-inwai C.[2007] *An ant colony optimization for dynamic job scheduling in grid environment*", International Journal of Computer and Information Science and Engineering, vol. 1(4), pp. 207-214.

[7] Chiang ML, Luo.J.N, Lin.CB and Wang.SS.[2011] *High-Reliable Dispatching Mechanisms for Tasks in Cloud Computing*". Department of Information and Communication Engineering, Chaoyang University of Technology.