International Journal on Science and Technology (IJSAT)
E-ISSN: 2229-7677 ● Website: www.ijsat.org ● Email: editor@ijsat.org

Load Balancing in Cloud using improved GWO(IGWO)

Ms. Vatsala Johari[1], Dr. Rohitashwa Pandey[2]

[1]Research Scholar, M. Tech. CSE, Bansal Institute of Engineering and Technology, Lucknow.
[2]Associate Professor, HOD-CSE, Bansal Institute of Engineering and Technology, Lucknow.

Abstract:
The rapid increase of data computation and storage in cloud results in uneven distribution of workload on its heterogeneous resources, which may violate SLAs and degrades system performance. Distributing a balanced workload over the available hosts is a key challenge in a cloud computing environment. The main problem of using population-based meta-heuristic algorithms (like PSO, ABC, HS, etc.) is to properly tune their algorithm specific control parameters. The control parameters are sometimes problem specific and hence the performance of such algorithms is heavily dependent on their parameters. To alleviate the tuning of parameters, GWO is used to minimize the imbalance of cloud load. GWO is less dependent on their control parameters and powerful in terms of convergence, exploration, exploitation and local optima avoidance. In this Paper, meta-heuristic and based on Grey Wolf Optimization (GWO), which is an optimization technique inspired by hunting behavior of grey wolves. All the approaches optimize the degree of imbalance of cloud datacenter to improve system performance. The other approach is based on classical GWO. The fourth approach is the hybrid Grey Wolf Optimization-Particle Swarm Optimization (GWO-PSO) approach with the benefits to avoid trapping into local optima and to achieve global optima. The fifth approach is based on GWO with improvements (iGWO) to further improve the convergence using a right balance between exploration and exploitation phases.

Keywords: GWO, iGWO, cloud computing, Distributing load.

1. Introduction

In minimization algorithms, finding global minima is a challenging task. The popular way to converge near optimal solution in population-based optimization algorithms is to divide the algorithm into two different phases. The first phase is exploration, where search agents try to discover the entire search space rather than making cluster near local minima. The second phase is exploitation, in which search agents try to exploit the gathered information to converge towards the global minimum. Exploration promotes candidate solutions to change rapidly and randomly, which improves the variety of solutions and increases the exploration rate. Exploitation improves the solutions' quality by locally searching close to the found potential solutions during the exploration phase.

Exploration and exploitation are two different and conflicting phases, where encouraging one makes other worst. The only exploration prohibits an algorithm to find a globally optimal solution and only exploitation makes it stuck in local optima. A perfect balance between two guarantees near-optimal solution.

In the classical GWO, 50% of the iterations are reserved for exploration and others for exploitation. The perfect balance between exploration and exploitation is overlooked in GWO. The impact of perfect balance between two guarantees a near optimal solution. An improved GWO (iGWO) is proposed in this chapter, which focuses on the required meaningful balance between exploration and exploitation. This leads to the optimal performance of the algorithm. Unimodal and multimodal benchmark test functions are used to check the quality and performance of the proposed iGWO variant and compared to other well-known optimization methods. The proposed iGWO is used to optimize the Degree of Imbalance (DoI) of chapter 4 to achieve load balancing and simulated for a different number of iterations, PMs, search agents, and runs. To show the effectiveness of iGWO, results were compared with the existing techniques.

2. Problem Identification:

The following points have motivated to design and implement a load balancing in the cloud using iGWO.

- In classical GWO, the switching between exploration and exploitation is performed by the modified values of a and A. In that, 50% of the iterations are reserved for exploration and others are reserved for exploitation. Relatively more exploration is related to too much of randomness and too much of exploitation is similar to too little randomness, which will probably not give optimized results. Thus, there must be a logical balance between both the phases.
- The position vector of a grey wolf in classical GWO is guided equally by the positions of α, β and δ wolves even though the most dominating wolf among the group is α followed by β and δ.
- Existing improved and modified GWO algorithms [53-55] are for different domain and objective functions and are not effective to our objective function.

3. Proposed improved Grey Wolf Optimization(iGWO)

In classical GWO, the value of a decreased from 2 to 0 linearly using the following equation:

$$a = 2\left(1 - \frac{t}{T}\right)$$

where T indicates the total number of iterations and t is the current ongoing iteration. An iGWO employs relatively less exponential value for the decay of a over the course of iterations as mentioned below:

$$a = 2\left(1 - \frac{t^{0.95}}{T^{0.95}}\right)$$

Using this proposed function of a, the number of iterations used for exploration and exploitation are 48% and 52%, respectively. Also, the position vector of a grey wolf in classical GWO is guided equally by the positions of α, β and δ wolves as follows:

$$X_g(t+1) = \frac{X_\alpha + X_\beta + X\gamma}{3}$$

But, the most dominating wolf among the group is α followed by β and δ. Therefore, in the proposed iGWO, more weight is given to the α followed by β and δ wolves. Position vector of a grey wolf can be found by:

$$X_g(t+1) = \frac{2 * X_\alpha + 1.5 * X_\beta + X_\gamma}{4.5}$$

4. Load Balancing in Cloud using iGWO

Figure proposed the implementation of iGWO to achieve load balancing in the cloud. The objective of load balancing in the cloud computing environment is to minimize the Degree of Imbalance(DoI) of chapter 4. An iGWO takes the utilization vector of resources

(CPU, RAM, NW bandwidth) of PMs as input and adjusts iteratively to minimize the objective function.

optimized values of resources, objective function's best value

Input

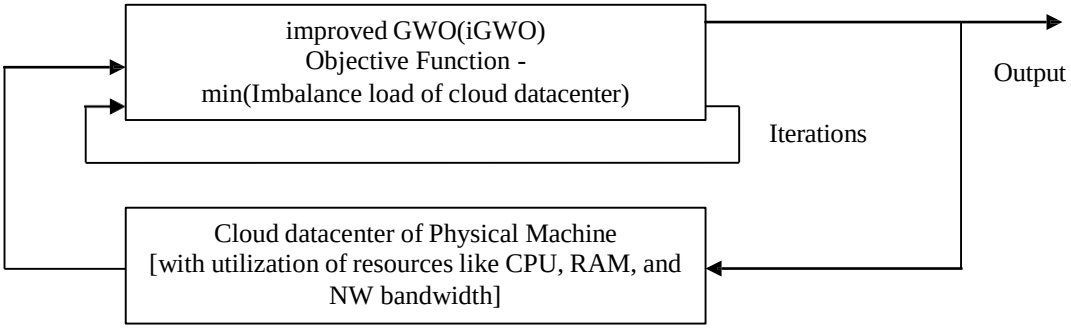

Fig. 4.1: Load Balancing in Cloud using iGWO

To find the value of the objective function, each position of the grey wolf is mapped into the utilization of resources like CPU, RAM, and NW bandwidth of PMs. The minimum value of the objective function (best fitness) is stored as an alpha score and the corresponding value of variables are stored as alpha position. Similarly, second best fitness and positions are stored as beta score and beta position followed by third best fitness and position are stored as delta score and delta position. In the next step, the search agents update their positions as per objective function. This process continues up to a maximum number

of iterations.

The Degree of Imbalance(DoI) was measured against the different number of iterations, search agents, PMs, and runs. To measure the degree of imbalance against the different number of iterations for all the algorithms, a simulation environment was set up with 60 PMs in the datacenter, 100 SAs in search space and a different number of iterations with 10 different runs. Fig.4.2 shows the graph of degree of imbalance against the number of iterations.

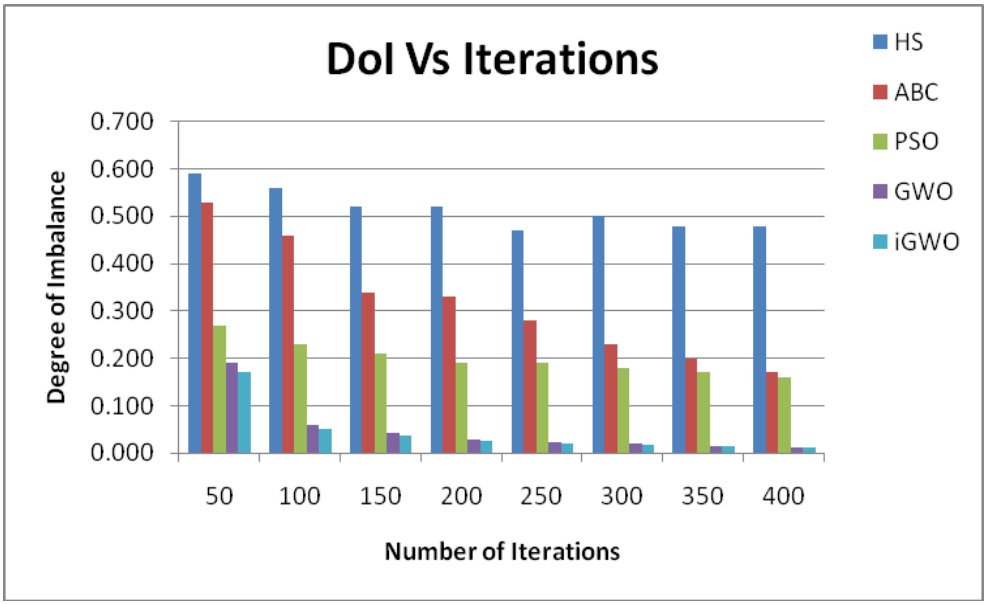

Fig.4.2: Degree of Imbalance against Number of Iterations

Table 4.3 shows the load imbalance value of fitness function against the different number of iterations for HS, ABC, PSO, GWO, and iGWO. From the results of fig.4.2 and Table 4.3, it is observed that GWO outperformed other algorithms against all the iterations. It has also been observed that the iGWO converges near optimal as the number of iterations increases.

Figure 4.4 shows the graph of the Degree of Imbalance against the different number of PMs. To measure the degree of imbalance against the different number of PMs for all the algorithms, a simulation environment was set up with different number of PMs, 100 search agents in search space and 100 iterations with 10 different runs.

Table 4.3: Degree of Imbalance against Number of Iterations

No of Iterations	**Algorithms**				
	HS	**ABC**	**PSO**	**GWO**	**iGWO**
50	0.590	0.530	0.270	0.190	**0.170**
100	0.560	0.460	0.230	0.060	**0.050**
150	0.520	0.340	0.210	0.043	**0.036**
200	0.520	0.330	0.190	0.028	**0.026**

250	0.470	0.280	0.190	0.023	**0.021**
300	0.500	0.230	0.180	0.019	**0.017**
350	0.480	0.200	0.170	0.015	**0.013**
400	0.480	0.170	0.160	0.012	**0.011**

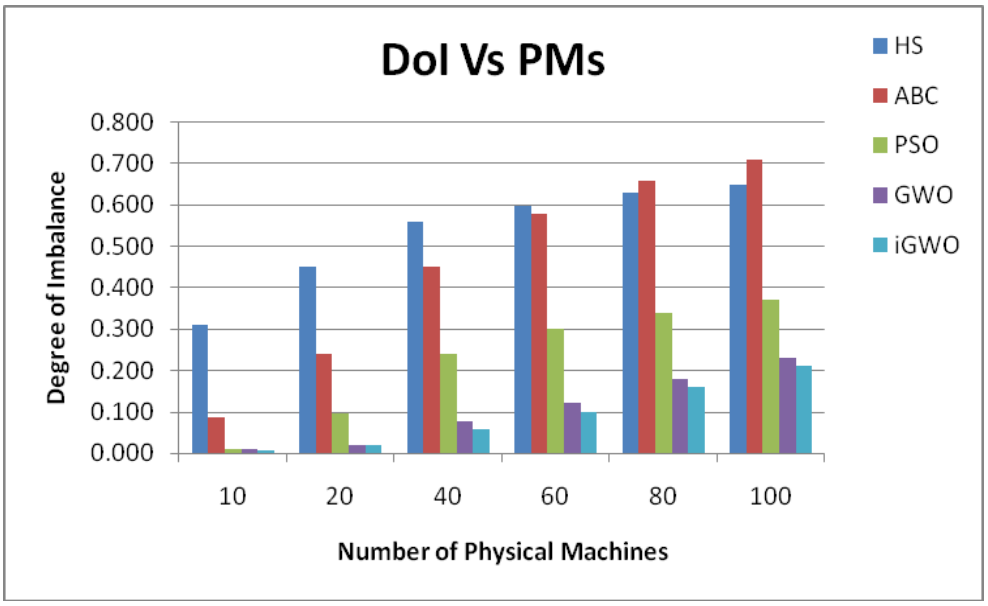

Fig.4.4: Degree of Imbalance against Number of PMs

Table 4.5 shows the load imbalance value of fitness function against the different number of PMs for PSO, HS, ABC, GWO, and iGWO. From the results of fig. 4.4 and Table 4.5, it is observed that iGWO outperformed other algorithms for all the PMs.

Table 4.5: Degree of Imbalance against Number of PMs

No of PMs	**Algorithms**				
	PSO	**HS**	**ABC**	**GWO**	**iGWO**
10	0.310	0.088	0.010	0.011	**0.006**
20	0.450	0.240	0.097	0.020	**0.019**
40	0.560	0.450	0.240	0.077	**0.057**
60	0.600	0.580	0.300	0.120	**0.100**
80	0.630	0.660	0.340	0.180	**0.160**
100	0.650	0.710	0.370	0.230	**0.210**

Figure 4.6 shows the graph of the Degree of Imbalance against the different number of search agents. To measure the degree of imbalance against the different number of search agents in search space for all the

algorithms, a simulation environment was set up with 60 PMs in the datacenter, a different number of search agents in search space and 100 iterations with 10 different runs.

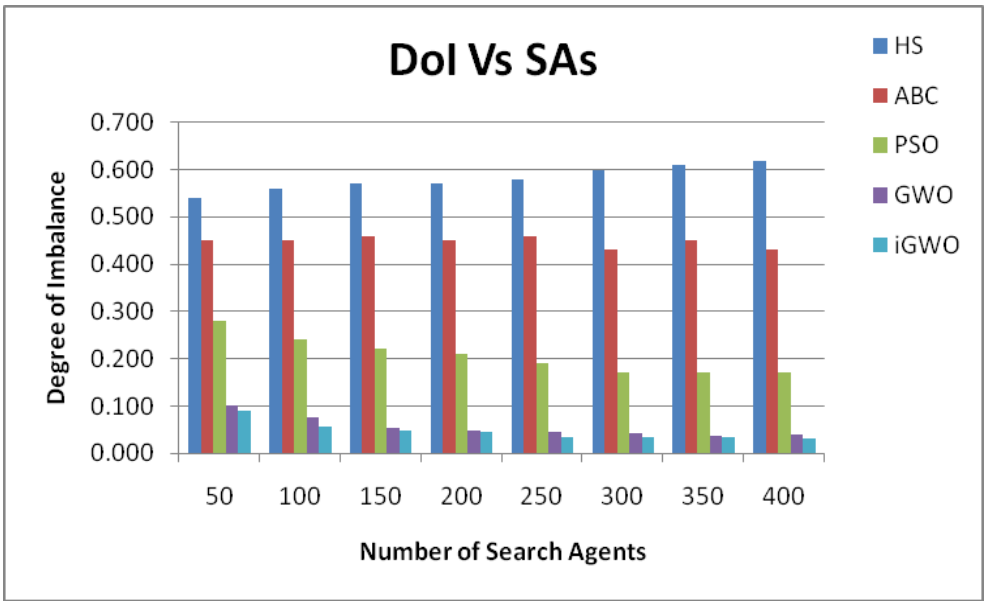

Fig.4.6: Degree of Imbalance against Number of Search Agents

Table 4.7 shows the load imbalance value of fitness function against the different number of search agents for PSO, HS, ABC, GWO, and iGWO. From the results of Fig. 4.6 and Table

4.7, it is observed that iGWO outperformed other algorithms for all the PMs. It has also been observed that the iGWO converges near optimal as the number of search agents increases.

Table 4.7: Degree of Imbalance against Number of Search Agents

No of Search Agents	**Algorithms**				
	PSO	**HS**	**ABC**	**GWO**	**iGWO**
50	0.540	0.450	0.280	0.100	**0.090**
100	0.560	0.450	0.240	0.077	**0.057**
150	0.570	0.460	0.220	0.052	**0.047**
200	0.570	0.450	0.210	0.047	**0.045**
250	0.580	0.460	0.190	0.044	**0.035**
300	0.600	0.430	0.170	0.041	**0.035**
350	0.610	0.450	0.170	0.036	**0.033**
400	0.620	0.430	0.170	0.038	**0.031**

Figure 4.8 shows the graph of the Degree of Imbalance against the different number of runs. To measure the degree of imbalance against the different number of runs for all the algorithms, a simulation

environment was set up with 60 PMs in the datacenter, 100 search agents in search space and 100 iterations with a different number of runs.

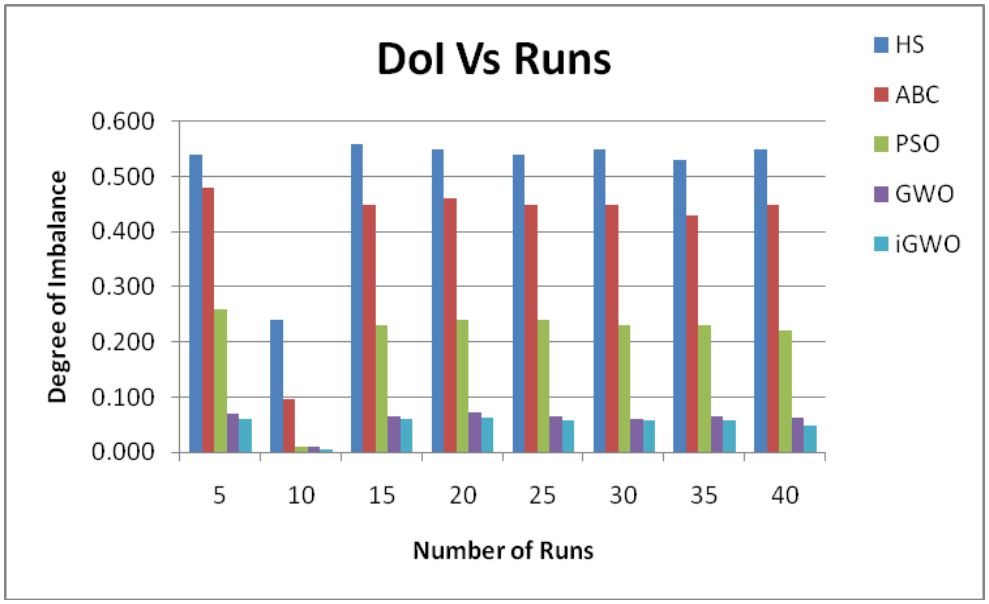

Fig.4.8: Degree of Imbalance against Number of Runs

Table 4.9 shows the load imbalance value of fitness function against the different number of runs for PSO, HS, ABC, GWO, and iGWO. From the results of fig.4.8 and Table 4.9, it is observed that iGWO outperformed other algorithms for all the PMs. It has also been observed that the iGWO converges near optimal as the number of runs increases.

Table 4.9: Degree of Imbalance against Number of Runs

No of Runs	**Algorithms**				
	PSO	**HS**	**ABC**	**GWO**	**iGWO**
5	0.540	0.480	0.260	0.070	**0.060**
10	0.240	0.096	0.009	0.010	**0.006**
15	0.560	0.450	0.230	0.064	**0.059**
20	0.550	0.460	0.240	0.073	**0.062**
25	0.540	0.450	0.240	0.065	**0.057**
30	0.550	0.450	0.230	0.060	**0.057**
35	0.530	0.430	0.230	0.064	**0.057**
40	0.550	0.450	0.220	0.062	**0.048**

Summary:

This paper proposed an improvement to the Grey Wolf Optimizer named iGWO and applied it for load

balancing in the cloud. Improvement in convergence is due to more iterations for exploiting the search space already explored and assigning the more weight to alpha and beta wolves compared to delta wolve due to their distance from the prey. Employing more iterations to exploit the explored search space and giving more weight to alpha and beta wolves compared to delta wolve, Different experiments carried out in terms of average value of an objective function(DoI) against the different number of PMs, search agents, iterations, and runs for HS, ABC, PSO, classical GWO and iGWO with their best-identified control parameters. The results prove that the proposed algorithm is found to be improved due to near global optimal and fewer chances in local minima stagnation. An iGWO is simple to implement because it does not require to tune algorithm-specific parameters like other meta- heuristics algorithms viz. HS, ABC, PSO, etc.Simulation results based on exploitation and exploration benchmark functions and the problem of load balancing in cloud demonstrate the effectiveness, efficiency, and stability of iGWO compared with the classical GWO, HS, ABC and PSO algorithms for solving real- world optimization issues.

References:

1. V. S. Rathor, R. Pateriya, and R. K. Gupta, “An efficient virtual machine scheduling technique in cloud computing environment”, International Journal of Modern Education and Computer Science, Vol. 7, No. 3, pp. 39, 2015.
2. S. Mirjalili, S. M. Mirjalili, and A. Lewis, “Grey wolf optimizer”, Advances in Engineering Software", Vol.69, pp. 46-61, 2014.
3. NIST Definition of Cloud Computing v15, csrc.nist.gov/groups/SNS/cloud- computing/cloud-def-v15.doc.
4. Q. Zhang, L. Cheng, and R. Boutaba, “Cloud computing: state-of-the-art and research challenges,” Journal of Internet Services and Applications, vol. 1, no. 1, pp. 7-18, April 2010.
5. A. T. Velte, T. J. Velte, and R. Elsenpeter, “Cloud Computing A Practical Approach” ,TATA McGRAW-HILL Edition 2010.
6. A. M. Alakeel, “A Guide to Dynamic Load Balancing in Distributed Computer Systems”, International Journal of Computer Science and Network Security(IJCSNS), Vol.10 No.6, June 2010.
7. V. Sreenivas, M. Prathap, and M. Kemal, “Load Balancing Techniques: Major Challenge In Cloud Computing - A Systematic Review", IEEE International Conference on Electronics and Communication Systems (ICECS), 2014.
8. M. Xu, W. Tian, and R. Buyya, “A survey on load balancing algorithms for virtual machines placement in cloud computing”, Concurrency and Computation: Practice and Experience 29(12), 2017.
9. A. N. Klaithem, M. Nader, A. N. Mariam, and A. Jameela, “A survey of load balancing in Cloud Computing:Challenges and Algorithms”, IEEE Second Symposium on Network Cloud Computing and applications, 2012.
10. W. Tian, Y. Zhao, Y. Zhong, M. Xu, and C. Jing, “A dynamic and integrated load balancing scheduling algorithm for cloud datacenters”, 2011 IEEE International Conference on Cloud Computing and Intelligence Systems (CCIS), pp. 311–315, 2011.
11. X. Li, Z. Qian, R. Chi, B. Zhang, and S. Lu, “Balancing resource utilization for continuous virtual machine requests in clouds”, 2012 Sixth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), pp. 266–273, 2012.