A Broker Policy for Cloud Environments using Hybrid Soft Computing Techniques

Avinash Kumar Sharma

Department of Computer Science & Engineering
Uttarakhand Technical University
Dehradun, India.
e-mail: avinashsharma2006@gmail.com

Nitin

Department of Electrical Engineering and Computer Science
College of Engineering and Applied Science
University of Cincinnati, Ohio, USA.
e-mail: nitinfu@ucmail.uc.edu

Abstract **- Broker policy is a critical decision factor for the cloud computing environment. During this process, the broker allocates the cloudlets to the different datacenters. In this paper, we have considered four strategies for the allocation of cloudlets to the datacenter. The broker allocation policies under consideration are: round robin, Bee colony optimization, genetic learning and particle swarm algorithm. Finally, the Enhanced genetic learning based Particle swarm optimization technique is designed and tested for performance. The Enhanced genetic learning based particle swarm optimization is found to be fastest in getting the job done whereas the round robin algorithm is slowest among the four algorithms. The Enhanced genetic learning based particle swarm optimization technique shows an improvement of 10% over the other soft computing techniques.**

Keywords - Broker Policy, Cloud Environment, Soft Computing Techniques.

I. Introduction

Cloud Computing has changed the perspective of doing the computation. The cloud computing has also helped the various new technologies like the internet of things and big data analytics. Mainly the cloud computing can be imagined as the vast network which helps to share the resources and computations to achieve the desired job as required by the client. In cloud computing, Infrastructure as a Service (IaaS), Software as a Service (SaaS), and Platform as a Service (PaaS) broadly classify the cloud computing services. Prime cloud computing providers like Google [3], Microsoft [2], Amazon [1], and Yahoo [4] have the customers around the globe and are successfully providing the services to their customers. Like these companies there are many other vendors available in the market, this makes the competitive environment in the market. The various vendors are exploring the different ways to reduce the implementation cost and maximize the profit to get better Return on investment.

Virtualization is the primary approach that enables the concept of the cloud environment. The various requests by the end users send to the different virtual machines so that they are mutually exclusive. The mapping of the various applications on the virtual machines residing on different data centers is a critical issue. The amount of power consumed the various data center is almost equal to that of 25000 houses[5] this is the prime issue, so our primary objective of this paper is to design the energy efficient broker policy with multiple objectives. The first objective is to load balancing and second objective is to reduce the power consumption of the system. In load balancing the various jobs are allocated to the different data center so that no data center gets overloaded. Secondly, if the data center has a load that can be executed by the other data center, then this allocation must be done on that machines to reduce the power consumption.

Figure 1 describes the block diagram of the cloud architecture. From the figure, we can see that the N end users connect to the different data center located in the different location communicating with the help of internet. The data center consists of application and data along with N number of nodes for the execution of the task. In a real application, we have multiple numbers of data centres to fulfil the request of the users. In this paper, we have studied the various broker policies like round robin, allocation using the genetic algorithm and particle swarm optimization. In this paper, we have designed the hybrid resource allocation policy using the genetic algorithm and particle swarm optimisation technique.

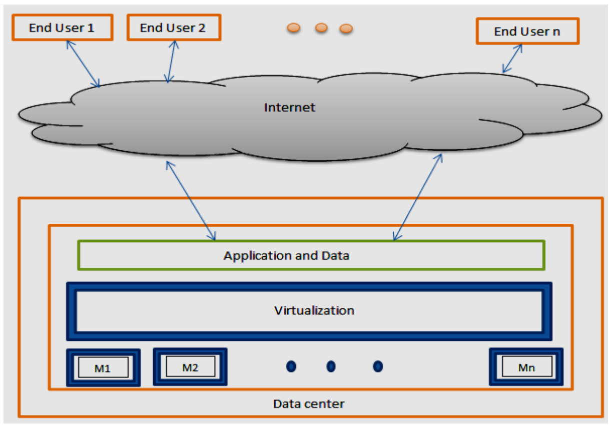


Figure 1. The organisation of cloud infrastructure

This paper divided into five sections. In section II, we are going to discuss the current load balancing like round-robin algorithm, Genetic learning and particle swarm optimization (GLPSO) technique. In section III, proposed the enhanced genetic algorithm (EGLPSO) for load balancing. In section IV, presents the simulation results and analysis with an overview of Cloud analyst simulation. At last section, VI concludes this paper.

II. Existing Approach

A. Round Robin Algorithm

The round-robin algorithm is fundamental virtual machine allocation technique. In this technique, there is no monitoring of the load on the VM. Round robin technique is independent of resource capability, the complexity of the task. This model just allocates the VMs one by one and once the cycle is completed it start the same process again. The cyclic allocation may result in the processing of high priority task to end with late response. Due to this many different version of round-robin algorithms have been suggested. Like Weighted Round Robin algorithm in which the allocation algorithm uses computational capabilities to decide the job, allocation. Still, these algorithms lag in the optimal allocation of the resources.[6]

B. BEE Colony Optimisation Technique

It is a swarm optimisation technique; here the whole cloud environment is mapped to Honey bee hive. The honey bees can are mimicking the task. The food source is like the virtual machines. The search of the food by the bees can is like the virtual machines. The exhaustion of the food by the honey bee is mimicking the overloading of resources. The search of new food is just like task migration in virtual machines. Even though the results obtained are satisfactory, but the authors do not provide the details of implementation for the bee colony optimisation. To study the details of the system authors have considered a registry (Cloud Information Services) which hold the details of the various resources available in the data center[7].

C. Genetic Learning with Particle Swarm Optimisation

The genetic algorithm is a heuristic search technique used to find the optimal solution to the particular problem. The genetic algorithm uses the objective function. This objective function is known as the fitness function. The genetic algorithm has three operator selection, crossover and mutation. The whole concept of the genetic algorithm is on Darwin's theory of survival of the fittest. The selection operator is used to select the best individuals and discard the worst solutions from the mating pool. The crossover operator operates on the individual chromosomes that consist of binary string which is capable of representing all the properties of the individual for the problem under consideration. The mutation just mimics the effect of the environment on the individuals, and the individuals change it some property accordingly.

The particle swarm optimisation has been proposed by Kennedy and Eberhart [8], [9] in 1995. It is also similar nature-inspired ideas like a bee colony, bird flocking and fish schooling. It is the technique in which a generation consists of n solutions which behaves like the particle. The best particle in the generation is the local$_{best,}$ and the best solution till now in all the solutions is known as the global best solution. The particle is in the generation tends to move towards the global best solution. There have been various applications of PSO in different fields of research. The most variants of PSO rely on the hybrid models of PSO[10-20].One such hybrid version is the mixture of the Genetic algorithm and PSO named as Genetic learning with Particle swarm optimisation GLPSO.

ALGORITHM 1. THE SUMMARISED VERSION OF GLPSO ALGORITHM.

Algorithm 1:(Best solution)**:=GLPSO**(N,Total$_{Gen}$,P$_c$, P$_m$)	
1.	Gen$_{count}$ $\leftarrow$0
2.	(Pop$_{Gen}$):=Generaterandompopulation(N); //Generate the initial population
3.	**while** Gen$_{Count}$< Total$_{Gen}$
4.	Particlevelocity(Pop$_{Gen}$);
5.	g$_{best}$:=Assignglobalbest();//Assign Global best
6.	(Temp$_{gen}$):=Crossover(P$_c$,Pop$_{gen}$);
7.	(Temp$_{gen}$):=Mutation(P$_m$, Temp$_{gen}$);
8.	Pop$_{Gen+1}$:=Selection(Temp$_{gen}$ U Pop$_{Gen}$)
9.	UpdatePositions(Pop$_{Gen+1}$)
10.	Increment the Gen$_{Con}$
11.	Display the global best as the solution to the problem

In the above algorithm, p$_c$ and p$_m$ are the probability of crossover and mutation. N is the population size and Total$_{gen}$ represents the total number of generations. The results are reported in the form of best solutions. The algorithm runs the various modules like *Generaterandompopulation(N)* that read the size of population and returns the population. Particlevelocity is are assigned using *Particlevelocity(Pop$_{Gen}$)*. In each of the iteration the global best solution is identified this identification is done by the module *Assignglobalbest()*. The three genetic operators are implemented using the 3 functions:

1. *Crossover(P$_c$,*Pop$_{gen}$*):*It accepts the probability P$_c$ as the probability of crossover and perform crossover the population.
2. Mutation(P$_m$, Temp$_{gen}$):It performs on the mutation on the population created using Crossover operator.
3. Selection (Temp$_{gen}$ U Pop$_{Gen}$): It selects the best individual from both the temporary population and current generation population.

For executing the results, we have selected the probability of crossover as 0.8 and probability of mutation as 0.3. Also, the Gen$_{count}$ is representing the total number of iteration for which both genetic algorithm and particle swarm optimisation algorithm are going to execute.

The main drawback of the algorithm mentioned above is that the particle swarm optimisation technique will not get the full opportunity to explore the solutions. Figure 2 describes the flowchart of the GLPSO.

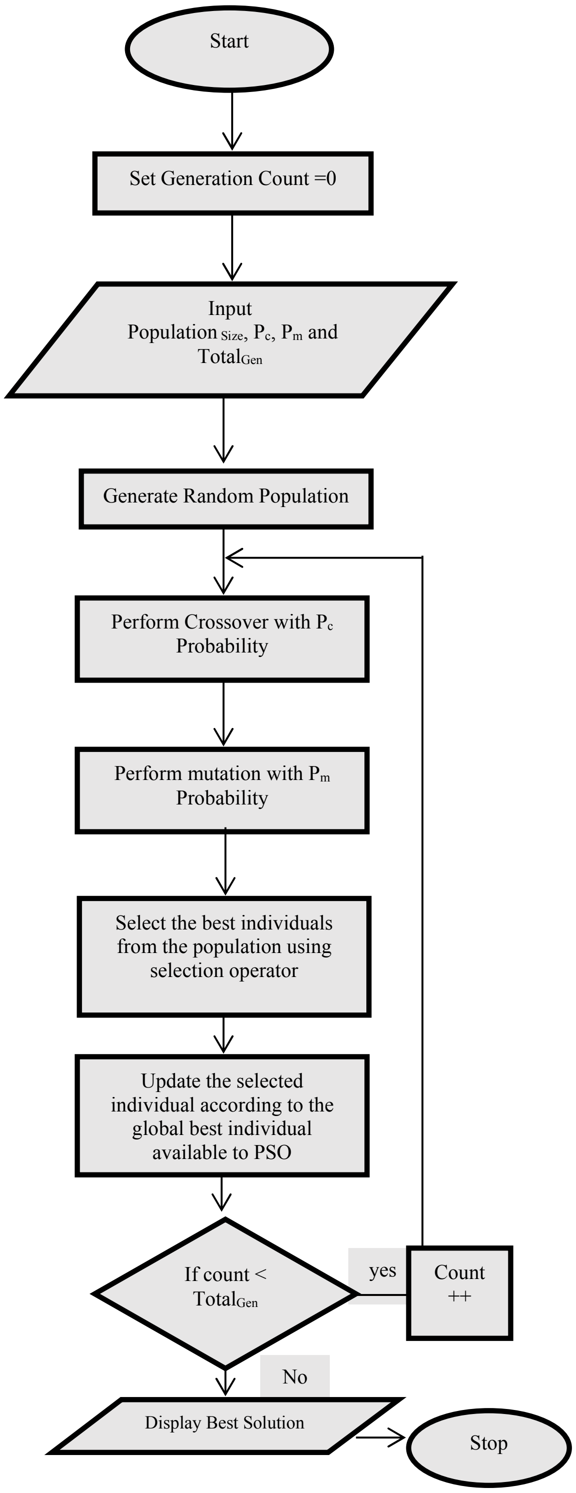

Start
Set Generation Count =0
Input Population Size, Pc, Pm and TotalGen
Generate Random Population
Perform Crossover with Pc Probability
Perform mutation with Pm Probability
Select the best individuals from the population using selection operator
Update the selected individual according to the global best individual available to PSO
If count < TotalGen
yes
Count ++
No
Display Best Solution
Stop

Figure 2. Flowchart of GLPSO

III. ENHANCED GENETIC LEARNING PARTICLE SWARM OPTIMISATION

The authors have an idea of optimizing the genetic algorithm with the help of particle swarm optimisation in[21].The authors have suggested that initially, the genetic algorithm will help the particle swarm optimisation to find the near optimal solution. Once the near optimal solution is of the genetic algorithm are collected, these results made the particles of the PSO algorithm. Both algorithms continue to work in parallel.

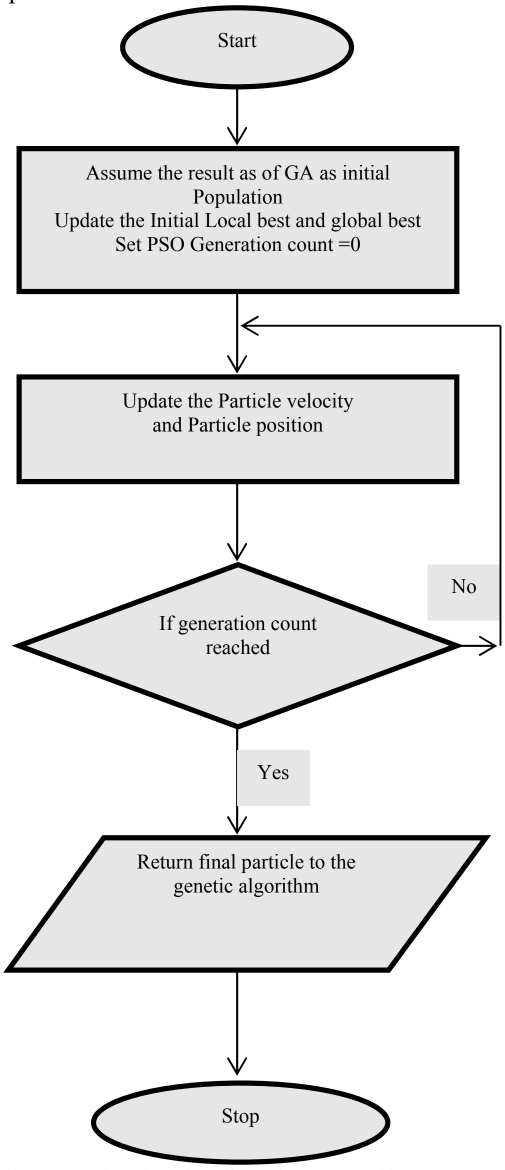

Start
Assume the result as of GA as initial Population
Update the Initial Local best and global best
Set PSO Generation count =0
Update the Particle velocity and Particle position
If generation count reached
No
Yes
Return final particle to the genetic algorithm
Stop

Figure 3.a Flowchart describing the PSO used in GLPSO

The traditional solutions obtained from PSO are precise to the nearby optimal solution. The genetic algorithm on other side generates the new solution by performing heuristic search and maintains the diversity in the solutions.

The GL-PSO algorithm has two variants proposed by the authors. In the first variant, PSO and GA are working in parallel. In the second version, PSO the part of the genetic algorithm.

In the proposed algorithm we have the PSO working on the results obtained from the genetic algorithm with a different number of iterations, Figure 3a details the flowchart of PSO algorithm.

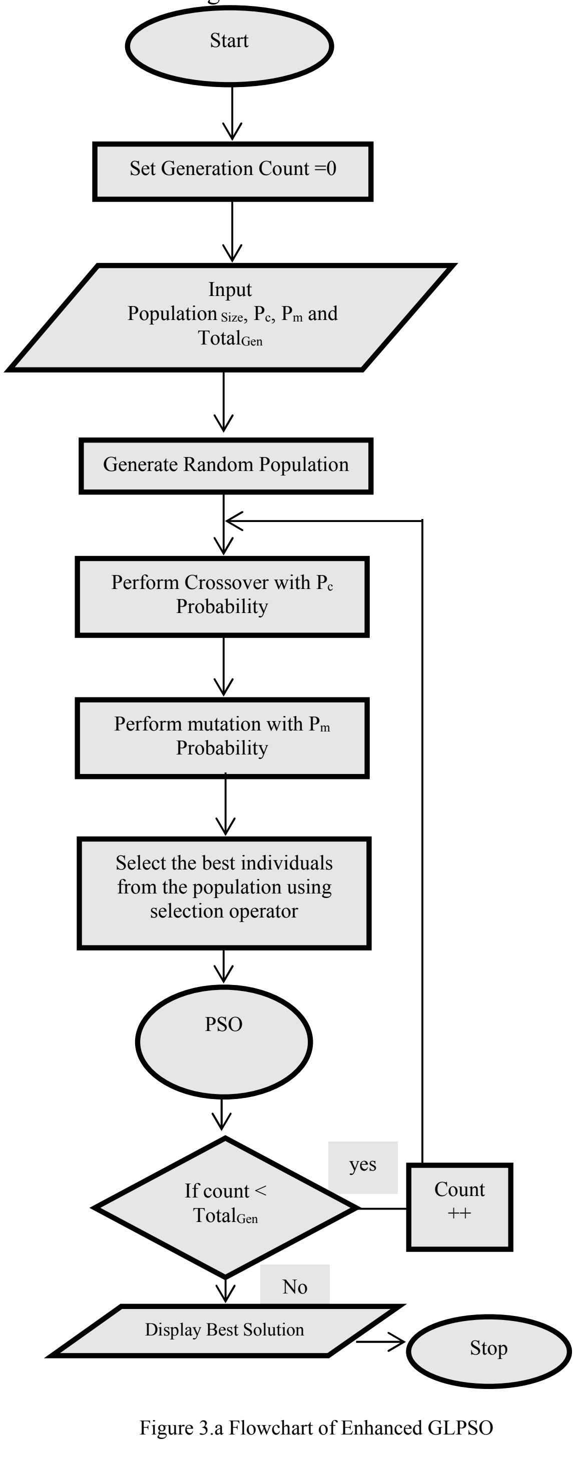

Start
Set Generation Count =0
Input Population$_{Size}$, P_c, P_m and Total$_{Gen}$
Generate Random Population
Perform Crossover with P_c Probability
Perform mutation with P_m Probability
Select the best individuals from the population using selection operator
PSO
If count < Total$_{Gen}$
yes
Count ++
No
Display Best Solution
Stop

Figure 3.a Flowchart of Enhanced GLPSO

Just like an additional operator. The major drawback of this approach is that PSO algorithm s not getting its complete power to explore the solutions as in every iteration new population is served which deteriorates the overall performance of the particle swarm optimisation. In our approach, we have modified the things that instead of using PSO as a single operator we will provide the Genetic algorithm results are behaving as the input to the PSO for *m* iterations. We know that the computational cost will increase but the solution obtained will be more diverse and precise. The detailed algorithm is as follows.

ALGORITHM 2. ENHANCED GLPSO ALGORITHM

Algorithm 2: (Best Solution) **:= EGLPSO** (N, Total$_{Gen}$, P_c, P_m, Total$_{GenPSO}$)	
1.	Gen$_{count}$ ← 0
2.	(Pop$_{Gen}$):=Generaterandompopulation(N); //Generate the initial population
3.	**while** Gen$_{Count}$ < Total$_{Gen}$
4.	g_{best}:=Assignglobalbest();//Assign Global best
5.	(Temp$_{gen}$):=Crossover(P_c,Pop$_{gen}$);
6.	(Temp$_{gen}$):=Mutation(P_m, Temp$_{gen}$);
7.	Pop$_{Gen+1}$:=Selection(Temp$_{gen}$ ∪ Pop$_{Gen}$)
8.	Gen$_{countpso}$ ← 0, assume results are initial population
9.	**while** Gen$_{Count}$ < Total$_{Gen}$
10.	Particlevelocity(Pop$_{Gen}$);
11.	g_{best}:=Assignglobalbest();
12.	UpdatePositions(Pop$_{Gen+1}$)
13.	Increment the Gen$_{countpso}$
14.	Increment the Gen$_{Con}$
15.	Display gbest as the solution

In the algorithm described above, we have combined the genetic algorithm with the particle swarm optimisation. Initially, the genetic algorithm operators select the individuals according to the fitness function. Then crossover operator with the single point crossover with probability value Pc =0.8 has been selected. The resulting offsprings then go through the mutation with the probability of Pm =0.3. The resultant generation then behaves as the particles with the velocity v and their velocity changes according to the global$_{best}$ solution. The equations for the change in the position of the particles are given by the equation 1.

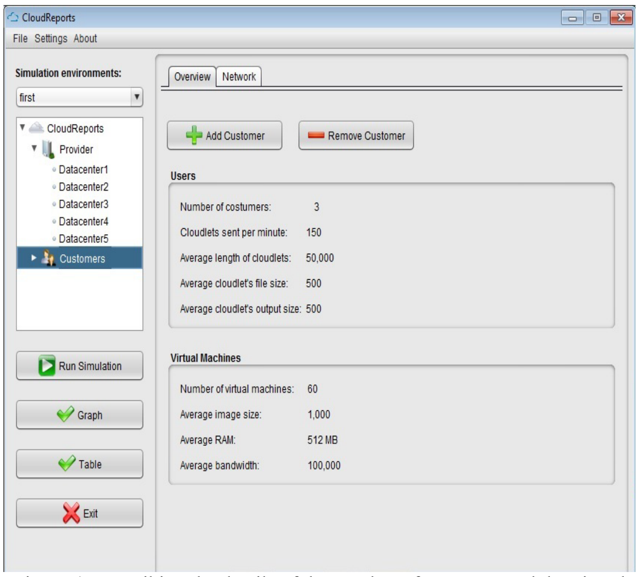
$$v = v + c_1 \times rand() \times (Local_{best} - present) + c_2 \times rand() \times (global_{best} - present) \tag{1}$$

The c_1 and c_2 are the constants with the value of c_1 and c_2 are assumed to be 2. The equation 2 represents the updates for the position of the present particle.

$$present = present + v \tag{2}$$

IV. Experimental Setup

To test the performance of the broker's algorithm we have developed the four discussed algorithms on cloud report which at its backend uses the cloud sim for the simulation. For the Study, we have considered three customers, and they have to generate variable load on each data center. Figure 4 describes the details of customers.



Users

Number of costumers: 3
Cloudlets sent per minute: 150
Average length of cloudlets: 50,000
Average cloudlet's file size: 500
Average cloudlet's output size: 500

Virtual Machines

Number of virtual machines: 60
Average image size: 1,000
Average RAM: 512 MB
Average bandwidth: 100,000

Figure 4: Describing the details of the number of customer and the virtual machines in the system

Similarly, we have considered the five data centres and Table I details the specification of each of the data center.

TABLE I. THE VARIOUS SPECIFICATION OF DATA CENTRES

S. No.	Parameters	Values
1	Number of hosts	10
2	Number of processing units	40
3	Processing capacity(MIPS)	96000
4	Storage	20TB
5	Ram	400GB

Each data center contains ten hosts and out of which 5 uses a space sharing VM Scheduling algorithm and remaining five uses the time slice based VM Scheduling algorithm.

The simulation of the system is done for one hour to observe the performance of the power consumption and the request allocation by the four algorithms.

V. Results and Discussion

The result obtained is discussed on the three-parameter one by one.

A. Average Request Completion Time

Average Request Completion Time is the mean time required by request arriving at the broker from its allocation at the datacenter and completion. To find we have used the equation three as follows:

$$ACT = \frac{\sum Time\ required\ by\ each\ request}{Total\ Number\ of\ request} \quad (3)$$

Figure 5 shows the comparison of the four broker's algorithm. From the figure it can be observed that the round robin algorithm is a most inefficient algorithm, where are the three algorithms based on soft computing approach are having better performance. Still, on observing in detail, it can be identified that the Enhance Genetic learning based Particle Swarm Optimisation approach has performed better than the other three techniques.

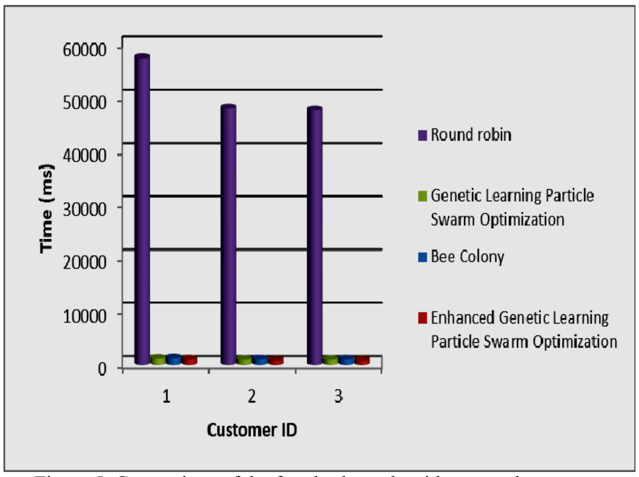


Figure 5. Comparison of the four brokers algorithm over the average request completion time.

B. Total Number of Request Completed

Another comparison of the four broker algorithm has been evaluated using the total number of request that processed by the various datacenters. The figure 6 shows that the Bee colony algorithm has performed more efficiently than the EGLPSO algorithm. Still, the EGLPSO has performed better than the other two broker algorithms.

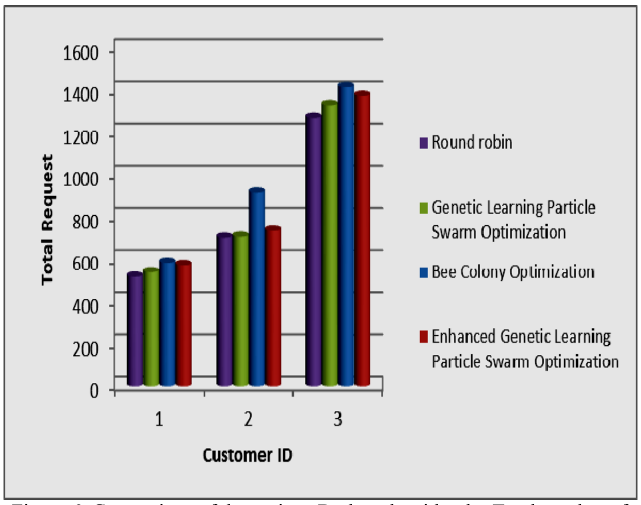


Figure 6. Comparison of the various Broker algorithm by Total number of Request.

Table II details the value obtained for the all the four broker policies regarding total request completed and average request completion time.

TABLE II. DETAILS OF TOTAL REQUEST PROCESSED AND AVERAGE COMPLETION TIME

Algorithm	Customer ID	Total Request	Time
Bee Colony Optimisation	3	585	1180.577641
	1	916	970.9030786
	2	1418	965.9107757
Enhanced Genetic Learning Particle Swarm Optimisation	3	574	958.5802251
	1	735	786.7918144
	2	1373	798.5792987
Genetic Learning Particle Swarm Optimisation	3	540	1094.647939
	1	707	899.3115743
	2	1329	909.7170521
Round robin	3	518	57516.65251
	1	702	48091.17949
	2	1271	47711.63415

C. Power Consumption

Figure 7 shows the power consumption of the various broker policies. From the study of the graph shown in figure 7 below it can be observed that the EGLPSO is consuming more power but also has completed the task earlier on another hand the genetic algorithm has tried to keep the power consumption under control but has taken more time to complete the allocated jobs. The round-robin also has taken more time to complete the request which was less than the genetic algorithm and the other broker policies. The running of the machines for more time implies that more cost has to applied by the user which will also affect the reputation of the service provider in the market.

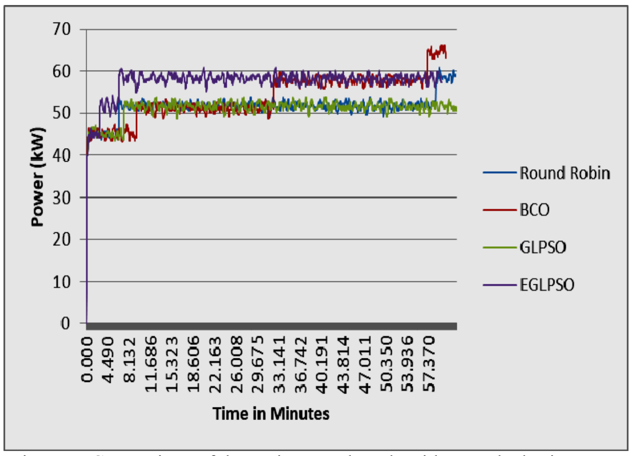


Figure 7. Comparison of the various Broker algorithm on the basic power consumption.

VI. CONCLUSION

From the results, we can observe that EGLPSO has performed better regarding request completion for the customer ID 2 but the average request for the Bee colony optimisation technique is slightly higher in comparison to that of EGLPSO. The EGLPSO is also having the fastest completion time in comparison to the other algorithms. From the results, it is clear that EGLPSO is achieving the goal 10% faster in comparison to the Bee Colony optimisation technique. The proposed algorithm is consuming 16% more power in comparison to the other broker algorithm. So this makes the tradeoff between the time and power.

REFERENCES

[1] Amazon Elastic Computing Cloud. [Online]. Available: http://aws.amazon.com/ec2/

[2] Windows Azure. [Online]. Available: http://www.microsoft.com/windows azure/

[3] Google App Engine. [Online]. Available: http://code.google.com/AppEngine

[4] Y. Chen, S. Jain, V. K. Adhikari, Z. L. Zhang, and K. Xu, “A first look at inter-data center traffic characteristics via Yahoo! datasets,” in Proc. IEEE INFOCOM, Shanghai, China, Apr. 10–15, 2011, pp. 1620–1628.

[5] J.Kaplan, W.Forrest and N.Kindler, “Revolutionizing data center energy efficiency,” McKinsey Co., New York, NY, USA, Tech. Rep., 2008.

[6] D. Chitra Devi and V. Rhymend Uthariaraj, “Load Balancing in Cloud Computing Environment Using Improved Weighted Round Robin Algorithm for Nonpreemptive Dependent Tasks,” The Scientific World Journal, vol. 2016, Article ID 3896065, 14 pages, 2016. doi:10.1155/2016/3896065

[7] K. R. R. Babu, A. A. Joy and P. Samuel, "Load Balancing of Tasks in Cloud Computing Environment Based on Bee Colony Algorithm," 2015 Fifth International Conference on Advances in Computing and Communications (ICACC), Kochi, 2015, pp. 89-93.

[8] J. Kennedy and R. Eberhart, "Particle swarm optimisation," in Proc. IEEE Int. Conf. Neural Netw., vol. 4. Perth, WA, Australia, 1995, pp. 1942–1948.

[9] R. Eberhart and J. Kennedy, "A new optimizer using particle swarm theory," in Proc. 6th Int. Symp. Micro Mach. Human Sci., Nagoya, Japan, 1995, pp. 39–43.

[10] X.-H. Shi, Y.-C. Liang, H.-P. Lee, C. Liu, and L. M. Wang, "An improved GA and a novel PSO-GA-based hybrid algorithm," Inf. Process. Lett., vol. 93, no. 5, pp. 255–261, Mar. 2005.

[11] Y.-T. Kao and E. Zahara, "A hybrid genetic algorithm and particle swarm optimisation for multimodal functions," Appl. Soft Comput., vol. 8, no. 2, pp. 849–857, Mar. 2008.

[12] F. Valdez, P. Melin, O. Castillo, and O. Montiel, "A new evolutionary method with a hybrid approach combining particle swarm optimisation and genetic algorithms using fuzzy logic for decision making," in Proc. IEEE Congr. Evol. Comput., Hong Kong, 2008, pp. 1333–1339.

[13] K. Premalatha and A. M. Natarajan, "Hybrid PSO and GA for global maximization," Int. J. Open Prob. Compt. Math., vol. 2, no. 4, pp. 597–608, Dec. 2009.

[14] C.-F. Juang, "A hybrid of genetic algorithm and particle swarm optimisation for recurrent network design," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 34, no. 2, pp. 997–1006, Apr. 2004.

[15] F. Grimaccia, M. Mussetta, and R. E. Zich, "Genetical swarm optimisation: a Self-adaptive hybrid evolutionary algorithm for electromagnetics," IEEE Trans. Antennas Propag., vol. 55, no. 3, pp. 781–785, Mar. 2007.

[16] W.-T. Li, X.-W. Shi, Y.-Q. Hei, S.-F. Liu, and J. Zhu, "A hybrid optimisation algorithm and its application for conformal array pattern synthesis," IEEE Trans. Antennas Propag., vol. 58, no. 10, pp. 3401–3406, Oct. 2010.

[17] S. Jeong, S. Hasegawa, K. Shimoyama, and S. Obayashi, "Development and investigation of efficient GA/PSO-HYBRID algorithm applicable to real-world design optimisation," IEEE Comput. Intell. Mag., vol. 4, no. 3, pp. 36–44, Aug. 2009.

[18] C.-S. Zhang, J.-X. Ning, S. Lu, D.-T. Ouyang, and T.-N. Ding, "A novel hybrid differential evolution and particle swarm optimisation algorithm for unconstrained optimisation," Oper. Res. Lett., vol. 37, no. 2, pp. 117–122, Mar. 2009.

[19] S. Li, M. Tan, I. W. Tsang, and J. T.-Y. Kwok, "A hybrid PSO-BFGS strategy for global optimisation of multimodal functions," IEEE Trans. Syst., Man, Cybern. B, Cybern., vol. 41, no. 4, pp. 1003–1014, Aug. 2011.

[20] Z.-H. Liu, J. Zhang, S.-W. Zhou, X.-H. Li, and K. Liu, "Coevolutionary particle swarm optimisation using AIS and its application in the multiparameter estimation of PMSM," IEEE Trans. Cybern., vol. 43, no. 6, pp. 1921–1935, Dec. 2013.

[21] Y. J. Gong et al., "Genetic Learning Particle Swarm Optimisation," in IEEE Transactions on Cybernetics, vol. 46, no.10, pp. 2277-2290, Oct. 2016. doi: 10.1109/TCYB.2015.2475174.

Avinash Kumar Sharma was born in Uttar Pradesh, India in 1982. He received the B.Tech. Degree from Uttar Pradesh Technical University and M.Tech degree from Uttarakhand Technical University, in 2006 and 2011, respectively and currently pursuing PhD from Uttarakhand technical university. He has the teaching experience of 11 years. He has taught various subjects to Graduates and Posts graduate students. He is the member of computer science teachers association CASA ACM. He also has the memberships of International Association of Computer Science and Information Technology (IACSIT) and International Association of Engineers (IAENG) Society of Computer Science. He has also supervised 4 M.Tech theses on the various field like cryptography, Fuzzy Logic, Software reliability and web technologies. He also has published the five articles in the various conference proceeding and reputed journals.

Prof. Nitin is an IBM certified engineer and Senior Member-IEEE and IACSIT, Life Member of IAENG, and Member-SIAM and ACIS. He has more than 160 research papers in peer reviewed International Journals and Transactions, Book Chapters, Symposium, Conferences and Position. His research interest includes Social Networks especially Computer Mediated Communications and Flaming, Interconnection Networks and Architecture, Fault-tolerance and Reliability, Networks-on-Chip, Systems-on-Chip, and Networks-in-Packages, Wireless Sensor Networks. Currently, he is working on Parallel Simulation tools, BigSim using Charm, NS-2 using TCL. He is the Co-founder of High-end Parallel Computing and Advanced Computer Architecture Lab at JUIT. He is Associate Editor of Journal of Parallel, Emergent and Distributed Systems, Taylor and Francis, UK. He is the referee for the Journal of Parallel and Distributed Computing, Computer Communications, Computers and Electrical Engineering, Mathematical and Computer Modelling, Elsevier Sciences. WSEAS Transactions, The Journal of Supercomputing, Springer and International Journal of System Science, Taylor and Francis. Till now he has guided 09 PhDs, 18 M.Tech. Theses and 60 B.Tech. Major Projects. He has been awarded the Doctor of science from Uttarakhand Technical University in 2013. He got his Doctor of Philosophy in with joint affiliation of Jaypee University of Information Technology/University of Florida India/USA, 2008 (Computer Science and Engineering). He has completed his M.Tech from Thapar University India, 2003 in software engineering and B.Tech from Dr B. R. Ambedkar University, India 2001.