I.J. Intelligent Systems and Applications, **2018, 7, 39-47**
Published Online July 2018 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2018.07.04

Hybrid Artificial Bee Colony and Tabu Search Based Power Aware Scheduling for Cloud Computing

Priya sharma
Department of Computer Science & Engineering, Guru Nanak Dev. University, Amritsar, Punjab
E-mail: Priyashrma630@gmail.com

Kiranbir kaur
Department of Computer Science & Engineering Guru Nanak Dev. University, Amritsar, Punjab
E-mail: Kiran.dcse@gndu.ac.in

Received: 21 June 2017; Accepted: 15 September 2017; Published: 08 July 2018

***Abstract*—**Load balancing is an important task on virtual machines (VMs) and also an essential aspect of task scheduling in clouds. When some Virtual machines are overloaded with tasks and other virtual machines are under loaded, the load needs to be balanced to accomplish optimum machine utilization. This paper represents an existing technique "artificial bee colony algorithm" which shows a low convergence rate to the global minimum even at high numbers of dimensions. The objective of this paper is to propose the integration of artificial bee colony with tabu search technique for cloud computing environment to enhance energy consumption rate. The main improvement is makespan 28.4 which aim to attain a well balanced load across virtual machines. The simulation result shows that the proposed algorithm is beneficial when compared with existing algorithms.

***Index Terms*—**Cloud computing, load balancing, artificial bee colony, tabu search.

I. INTRODUCTION

Cloud computing is a technology that utilizes the web and central remote servers to offer scalable services for its consumers and all the application are managed on a cloud which includes a large number of computers interlinked together in a complicated or sophisticated way (Venugopal, 2013). Cloud offers computing resources/ method a in the proper execution of virtual machine that is an abstract that runs on physical machine (Mohamed Shameem & Shaji, 2014). There are basically three services provide by cloud: - Iaas, Paas, Saas (Zuo, Zhang, & Tan, 2014).

In cloud computing, every time a virtual machine is overloaded loaded with number of tasks, these tasks need to be eliminated and migrated to the under loaded virtual machines of exactly the similar datacenters (Venugopal, 2013). In this instances, when delete multiple task from a heavy loaded VM and when there is several VM offered to method these tasks, the tasks need to be great mixture of priorities such as number of task much wait for a long time to be able to get prepared (Science & Engineering, 2016).

According to different prioritized tasks selection of VM:-

$$T'_H \rightarrow Vm \mid \min(\sum T'_H) \in Vm \quad (1)$$

$$T'_M \rightarrow Vm \mid \min(\sum T'_H \sum T_M) \in Vm \quad (2)$$

$$T'_L \rightarrow Vm \mid \min(\sum T_M) \in Vm \quad (3)$$

Table 1. Symbols Description

Vm	Virtual machine
T'_H, T'_M, T'_L	Higher, middle and lower priority cadres of tasks.

Load balancing is performed at VM stage i.e. at intra datacenter level. In cloud environment, load balancing and scheduling is the biggest challenges/issues. Load balancing methods are generally mentioned homogenous as well as heterogeneous environment like as grid (Venugopal, 2013).

The basic architecture of load balancing process is described in fig1. Cloud information service (CIS) may be achieved which includes most of the resources obtainable in the cloud environment. It is really a register of datacenters. Each time a datacenter is establishes it's to join the CIS.

Calculate total datacenter load on each VM:-

$$LD_{DC} = \sum_{Vm=1}^{n} LD_{VM} \quad (4)$$

Calculate entire datacenter capacity:-

$$Cap_{DC} = \sum_{Vm=1}^{n} Cap_{VM} \quad (5)$$

Datacenters are diverse in character or content with unique attributes. Generally a datacenter contains a few hosts. Hosts/serves needs amount of processing components with RAM and bandwidth features (Snášel, Abraham, Krömer, Pant, & Muda, 2016).

In cloud computing these hosts are virtualized in to various quantity of virtual machines predicated on individual user request. Virtual machines can also heterogeneous features like hosts. Cloud information service (CIS) acquires detail regarding virtual machine in the datacenters that predicated on these details the cloud broker transmits these jobs to various different resources in a datacenter. In this approach, algorithm checks the overloaded VMs and then migrate tasks to under loaded VMs (Snášel et al., 2016).

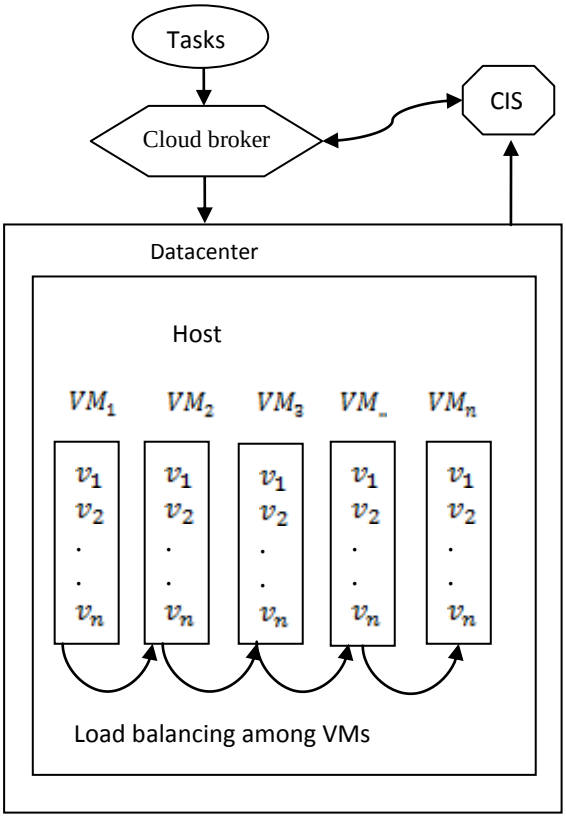

Fig.1. Architecture of Load balancing (Snášel et al., 2016).

The load balancing issue could be separated into two subscription issues:-

1. Entrance of new request for virtual machine provisioning and keeping of virtual machines on host.
2. Migration/Reallocation of virtual machines (Mohamed Shameem & Shaji, 2014).

A. Traditional Techniques in Load Balancing

Bee colony with scheduling and load balancing:-

In this algorithm is a nature inspired strategy is encouraged by the foraging behavior of honey bee colonies to resolve exact purpose optimization problem (Karaboga & Basturk, 2007).

Forging behavior of the honey bees are mimics therefore, it's a great solution for optimization problem [(Nirmala, 2013)(Babu, Mathiyalagan, & Sivanandam, 2014)].

The algorithm contains Employees bees, Onlooker, Scout (Ge, 2016). In cloud environment, scout bees load for virtual machine. Then searching a resource it delivered to cloud environment and works a dance i.e. waggle dance and then depends on this waggle dance different bees in the environment get detail regarding level of VM and distance from cloud hive [(Snášel et al., 2016), (Karaboga & Akay, 2009)].

Table 2. In cloud environment mapping parameters with enhanced bee colony algorithm (Snášel et al., 2016)

Cloud environment	**Bee hive**
'Task '(Cloudlet)	'Honey bee'
'Virtual machine'	'Food source'
Loading of a task to a Virtual machine	Honey bee foraging a some food Source
Virtual machine in overloaded condition.	Honey bee obtaining depleted at a food source
Deleted task after rescheduling to an under loaded Virtual machine having highest capacity	Foraging bee searching a newly food source

This algorithm load balancing and scheduling criteria perform in following 4 steps [(Venugopal, 2013), (Snášel et al., 2016)]:-

i. Calculate current load on virtual machine:-

$$LD_{VM} = \frac{N * len}{MIPS} \quad (6)$$

Table 3. Symbols Description

VM	Virtual machine
N	No. of tasks
Len	single task length
MIPS	Million instruction per second

a. Calculate capacity of virtual machine:-

$$Cap_{VM} = Pe'_{num} * Pe'_{mips} + Vm'_{bw} \quad (7)$$

Table 4. Symbols Description

Pe'_{num}	No. of processing elements in a specific virtual machine.
Pe'_{mips}	Processing elements in million per instructions rate.
Vm'_{bw}	band width related for a virtual machine

b. Calculate processing time (Pt') of each task:-

$$Pt' = \frac{(Current\ LD)}{cap} \quad (8)$$

Then, Standard Deviation (SD')

$$SD' = \sqrt{\frac{1}{m}\sum_{i=1}^{m}(PT_i^{'} - PT^{'})^2} \quad (9)$$

ii. Scheduling and Load Balancing Decision:-

The scheduling and load balancing is performed only when calculate Standard Deviation value larger than threshold value. Basically threshold value is set 0-1 depend upon standard deviation compared.

iii. Virtual machine Grouping:-

Virtual machines are arranging into 2 state of groups:- overloaded and under loaded condition. The virtual machines are assembled depending upon the standard deviation and limit value formerly determined on the basis of the load.

iv. Task Scheduling:-

If the decision would be stable the load, the machine has to obtain the demand to each overloaded virtual machines. The strategy chooses the task with based on priority. In which lowest priority task selects firstly from an overloaded virtual machines and then rescheduled to an under loaded virtual machines through optimum ability.

Calculated the Supply virtual machine:-

$$Supply_{VM} = LD - Cap$$

Again,

Calculated the Virtual machine demand by using the:-

$$DemandVM = LD - Cap$$

Pseudo code

1) Begin
2) for each task do
3) Determine load of virtual machine and choose balance the load or not.
4) Then grouping of the virtual machine based on OL and UL.
5) Get the way to obtain UL Virtual machine and need to OL VMs.
6) OL and UL VM sets are sort.
7) Based on the priority tasks in OL VMs are sort.
8) In the UL VMs set, find out the capacity.
9) For each task in each OL VM discovers an appropriate UL VMs predicated on capacity.
10) At last update the sets, OL and UL VM.
11) End of step 2.
12) End.

OL	Overloaded
UL	Under loaded

The specific contributions of this paper is organized as follows: a related work is given in Section 2; the gaps in existing work is discuss in detail in Section 3; a ABC and tabu search algorithm flow chart is presented in Section 4, describe the Simulation result and analysis in Section 5; and conclusion is explained in the last section.

II. Related Work

(Alla, Alla, & Ezzati, 2017) introduced an Analytic Hierarchy Process (AHP) based on Dynamic Priority-Queue (DPQ) algorithm and PSO algorithm. The proposed DPQ-PSO approach provides good performances such as make span, obtain higher resource utilization and also increased quality of service significantly. (Masdari, Salehi, Jalali, & Bidaki, 2016) discussed PSO-based scheduling approach in a on the basics of this algorithm that has been utilized in the resource accessibility, scalability and decrease the functional costs of the datacenters. Furthermore, determine more sophisticated evaluation of the way PSO algorithm has been increased and integrated in each system to resolve the workflow/task scheduling issues. (Liu, Zhang, Li, & Niu, 2015) proposed the DeMS algorithm. DeMS include basically 3 algorithms: - The first On-Demand scheduling algorithm is planned to reduce the conversation overhead between a master and slaves. Second QTM is made to balance the workload. Third STM is performed to schedule the tasks related with each other. Experimental results determine our proposed On Demand scheduling approach may considerably decrease the response time of parallel jobs. (Masdari, ValiKardan, Shahi, & Azar, 2016) analyzed the workflow scheduling approach and their numerous parameters and qualities are examined like:- ET, CT,RT,DTT. (Awad, El-Hefnawy, & Abdel_kader, 2015) considered the LBMPSO. That applied to reduce cost, decrease total time and sending time to obtain load

balancing between tasks and virtual machine, and reduce the difficulty in cloud environment and improves the consistency of cloud computing and dividing tasks on to virtual machine compared to other approaches.(Dasgupta, Mandal, Dutta, Mandal, & Dam, 2013) proposed a Genetic Algorithm (GA). This technique reduce the make span of a certain tasks set using the Cloud Analyst simulator. (Effatparvar & Garshasbi, 2014) presented algorithms in parallel heterogeneous multi-processor system which raise system utilization and minimize total time taken. Basically, to analyzed an improved solution in parallel system by considering first come first out and processing time algorithm. (Jena, 2015) focused on task scheduling based algorithm to enhanced consumption rate and running time. Thus, multi-objective PSO algorithm, could successfully resources utilization and to decrease make-span. (Domanal, Reddy, & Damanal, 2014) designed an effective algorithm which handles the load by considering the existing status of all Virtual machines. In which, proposed algorithm and existing algorithm compared on the based on resources and eliminates the inefficient usage of those Virtual machines. (Snášel et al., 2016) modified the bee colony algorithm, to perform effective and efficient load balancing within cloud environment. It tries to reduce make-span as well as the number of virtual machine migrations. (Al-maamari & Omara, 2015) proposed DAPSO to make improvements in performance from these essential (PSO) particle swarm optimization algorithm to increase a task runtime by reducing make-span associated with a specific task set and exploiting resource utilization.

III. Gaps in Existing Work

As discussed by (Snášel et al., 2016), it is analyzed that the enhanced bee colony algorithm performs better as compared to other nature inspired algorithms. The reason for this is that particle swarm optimization is limited to initial set of particles where wrongly selected particles can provide bad result. And artificial bee colony algorithm shows globally a low converging rate but integrating both scheduling algorithms, provides possibility of further improvements. In this paper, tried to implement the integration of artificial bee colony and tabu search algorithms to enhance the quality of service metrics and also improve the energy consumption rate.

IV. Methodology

In this paper, hybrid ABC with tabu search technique. This proposes technique enhance the energy consumption rate. In which consider following steps:-

- An employed bee holds the details regarding food source and gives details with a specific neighborhood.
- Based upon makespan, each onlooker bees follow employed bees.
- Then discard unoccupied food source and finally find out the best solution.
- After finding the best solutions, again start iteration and apply tabu search algorithm.
- Using previous best solutions, tabu search conclude the final result and return final result.

The main reason to hybridization of ABC algorithm with tabu search, basically ABC algorithm is an decision maker they provide best solution but using best solutions tabu search return a final solution. The methodology followed for the implementation is given in fig.2.

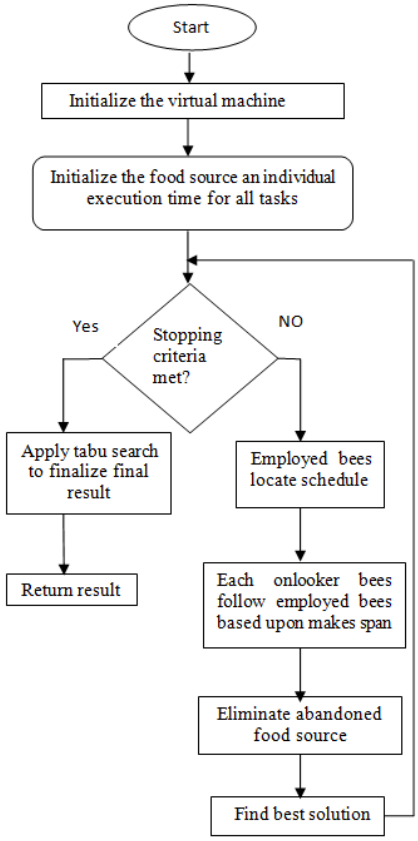

Fig.2. Flowchart of the propose technique

Pseudo code:-

Step 1: Bees_tabu_sol: - "get random bees/positions"

```
for i in range (be)
x = ran.rant(0, UB)
y = ran.rant(0, LB)
    for i in range(tabu)
t_x = ran.rant(0, UB − LB)
t_y = ran.rant(0, LB)
t_b. append((float_(tx), float_(ty)))
return t_b
```

Step 2: Develop best_bees:- "makes a best_bees from randomly selected alleles."

```
b_be = []
 1_st = [i for i in range(sf.len)]
for i in range(sf.len)
ch = ran.ch(1_st)
1_st.re(ch)
b_be.append(ch)
  return b_be
```

Step 3: rank_Scouts:-

```
lft,rgt = sf._bst_fd()
  p_1 = fd()
 p_2 = fd()
c_1  = [c for c in sf.bst_be\
        if c not in o.bst_be[lft.rgt + 1]]
p_1.bst_be = c_1 [: lft] + o.bst_be[lft,rgt + 1]\
                  = c_1 [lft:]
c_2 = [c for c in o.bst_be\
        if c not in sf.bst_be[lft : rgt + 1]]
p_2.bst_be = c_2[: lft] + sf.bst_be[lft : rgt + 1] \
                  +c_2 [lft :]
return p_1,p_2
```

Step4: employed_bees (sf):-"swap two elements"

```
lft,rgt = sf._bst_fd()
  T = sf.bst_be [lft]
sf.bst_be[lft]  = sf.bst_be[rgt]
sf.bst_be[rgt] = T
```

Step 5: food_mtrix:-

```
  for i (a_1, b_1)in enumerate (t_y)
  for j (a_2, b)in enumerate (t_y)
  d_a,d_b  = a_1  − a_2 ,b_1 − b_2
d' = sqrt (da* da + db* db)
m' [i, j] = d'
return m'
```

Step 6: exploitation_phase (matrix, queue):- """ Returns the total length of the queue """

```
tl = 0
n_be = len(que)
for i in range (n_be)
j = (i + 1)%n_be
 s_i = que[i]
 s_j = que[j]
tl+= m[s_i ,s_j]
return (to/n_be  ,0)
```

Step 7: Scouts (best_bees):-

```
 sf.sz = sz
sf.dd = sf._stp_cri()
 sf.mg = mg
    sf.nfd_sr = nfd_sr
sf.rk_scrt = rk_scrt
sf.ev_rt = ev_rt
   for fd_src in sf.dd
        fd_src.obj_fn ()
sf.g = 0
sf.ms_len = sys.mt
sf.mfd_src = none
```

Step 8: Evaluate makespan:-

```
 for i in range (1,  sf.mg + 1)
       iteration no : " +str (i)
for j in range (0,sf.sz)
       sf.dd[j].obj_fn()
csd_len = sf.dd[j]. sd_len
   if csd_len < sf.ms_len
sf.ms_len =  csd_len
sf.mfd_src = sf.dd[j]
    mks = csd_len − j
if ran.ran() < sf.rk_scrt
nfd_src = int (sf.nf_sc * sf.sz/2)
  for i in range (0,nfd_src)
       sel_1 = sf.srk()
       sel_2 = sf.srk()
      par_1 = sf.dl[sel_1]
      par_2 = sf.dl[sel_2]
cd_1,cd_2 = par_1.rk_sc(par_2)
      cd_1.obj_fn()
      cd_2.obj_fn()
   chd.append(cd_1)
  chd.append(cd_2)
for i in range (0,nfd_src)
    tls_len = 0
for l in range (0,sf.sz)
  tls_len+= sf.dl[k].s_len
  ras_len = ran.ran()
  ads_len = 0
 for j in range (0,sf.sz)
 ads_len+= (sf.dl[j].s_len/tls_len)
if ads_len ≥ ras_len
sf.dl[j] = chd[i]
 break
if ran.ran() <  sf.ev_rt
sel = sf.s()
sf.dl[sel]. emp_be()
  end loop
for i in range (0.sf.sz)
sf.dl[i].obj_fn()
csd_len = sf.dl[i].sd_len
if csd_len < sf.msd_len
sf.msd_len =  csd_len
 sf.mfd_src = sf.dl[i]
```

Step 9: select rank:-

```
sf,dl. so()
 if ran.ran() < chbst
return ran.rant (0,sf.sz * sf.nfd_sr)
else
return rand.rant(sf.sz * sf.nfd_sr\sf.sz − 1)
```

Step 10: evaluate metrics:-

```
sp = srlt/mks,2
TTT = t. t () − in_t ,2
ut = (sqrt(srlt/mks),2)
Eff = (srlt/mks, 2)
con_engy = ((mks) * Tx + engy * (t.t() − in_t ,4))
```

Experimental setup:-

Various variables along with respected value or range are shown in the table below .These variable are required to successfully simulate the proposed technique.

Table 5. Nomenclature used

Symbol	Meaning
random.randint	ran.rant
tabu_bees.append	t_b. append
tabu_bees	t_b
Lower bound , upper bound	LB,UB
best_bees	b_{be}
self.length	sf.len
random. Choice	ran.ch
Choice	ch
best_bees.append	b_{be}.append
lst.remove	1_{st}.re
left, right	lft, rgt
self._bst_food	sf._bst_{fd}
Food	Fd
self.best_bees	sf.bst_{be}
other.best_bees	o.bst_{be}
Temp	T
Matrix	M
square root	Sqrt
Distance	D
Total	Tl
Num_bees	n_{be}
Queue	Que
Slot	S
Size	sz
self.size	sf.sz
self.deadline	sf.dd
self._Stopping_criteria	sf.stp_{cri}
Maxgenerations	mg
newfood_Srcrate	nfd_{sr}
rank_Scouts_rate	$rk_{sc\,rt}$
eigenvctr_rate	ev_{rt}
food_Src.objective_fn	fd_{src}.obj_{fn}()
self.minschedule_length	sf.ms_{len}
sys.maxint	sys. mt
curschedule_length	csd_{len}
self.minfood_Src	sf.mfd_{src}
Selectrank	srk
Selected	Sel
Parent	Par
Deadline	Dl
Child	Cd
Children	Chd
randschedule_length	ras_{len}
addschedule_length	ads_{len}
employed_bees	emp_{be}
Choosebest	Chbst
Sort	So
Speedup	Sp
Serial time	Srlt
Total time taken	TTT
Initial time	in_t
Time	T
Utilization	Ut
Efficiency	Eff
Consumed energy	con_{engy}

Table 6. Experiment setup

Variable	Value/Range
Tx	0.078
UB	1000
LB	200
Len	30
Size	100
mg\ nfd_{sr}, $rk_{sc\,rt}$ \ ev_{rt}	170\0.9,0.6\0.8
Chbst	0.9

V. Simulation Result and Analysis

A. Makespan

The total period of time required to finish a several tasks is called makespan. Minimizing makespan reduces the performance time.

$$Makespan = Max(SL_i)$$

Where

$i = 1\ to\ no.\ of\ \ processor$

$SL = schedule\ length$

The above result shows that the proposed algorithm decrease the makespan as compared to the existing algorithm.

Table 7. Makespan

Iterations	Existing	proposed
1	212	184
2	231	192
3	200	188
4	229	186
5	208	199
6	220	197
7	223	195
8	230	198

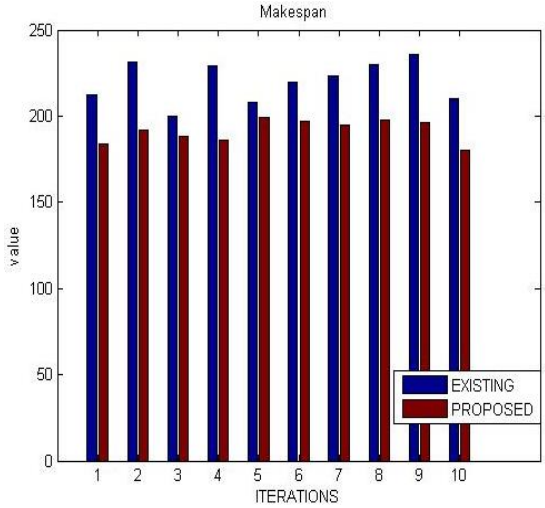

Fig.3. Makespan

B. Speed up

Speedup is an activity for raising the performance between two systems processing the same problem.

$$speedup = \frac{s_t}{m_k}$$

Where

$s_t = serial\ time$

$m_k = Makespan$

In this table, shows that the proposed technique increases the speed as compared to existing technique.

Table 8. Speedup

Iterations	Existing	proposed
1	0.94	1.04
2	0.86	0.98
3	0.83	1.01
4	0.87	1.06
5	0.86	1.08
6	0.83	0.99
7	0.85	1.03
8	0.80	1.01

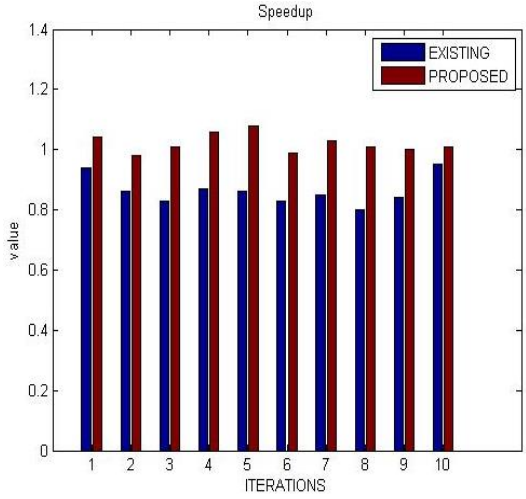

Fig.4. Speedup

C. Total time taken

The total time required for start to finish an entire tasks.

$$TTT = F_t - I_t$$

Where

$F_t = finishing\ time$

$I_t = Initial\ time$

In this experiment shows that the less time taken requires to complete the task as compare to existing technique.

Table 9. Total Time Taken

Iterations	Existing	Proposed
1	5.21	4.54
2	5.27	4.32
3	5.70	5.10
4	5.32	4.41
5	5.35	4.27
6	5.12	4.11
7	5.26	5.04
8	5.65	4.57

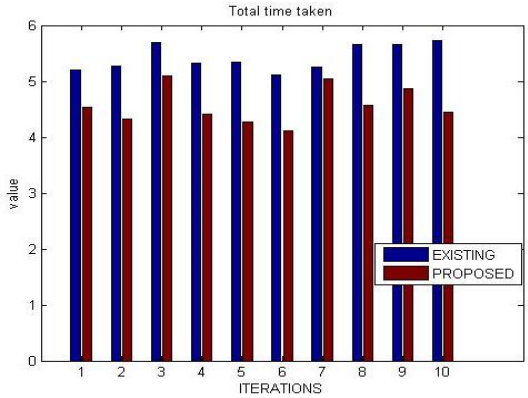

Fig.5. Total Time Taken

D. Utilization

Utilization is an action which gives the detail regarding effective use of resources.

$$Utilization = \sqrt{\frac{serial\ time}{makespan}}$$

In this table shows that the utilization improves as compare to existing technique.

Table 10. Utilization

Iterations	Existing	Proposed
1	0.55	0.61
2	0.50	0.58
3	0.49	0.59
4	0.51	0.63
5	0.54	0.63
6	0.50	0.58
7	0.50	0.6
8	0.59	0.65

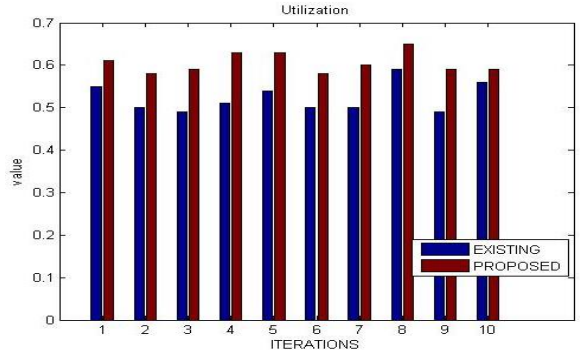

Fig.6. Utilization

E. Efficiency

Efficiency is a measurable parameter, which reduce the amount of energy required to entire system.

$$Efficiency = \frac{S_p}{N_{op}}$$

Where

$S_p = Speedup$

$N_{op} = no\ of\ \ processor$

In this experiment shows that the efficiency increases as compare to existing technique.

Table 11. Efficiency

Iterations	Existing	Proposed
1	0.31	0.34
2	0.28	0.33
3	0.27	0.32
4	0.29	0.33
5	0.25	0.30
6	0.27	0.31
7	0.28	0.30
8	0.29	0.33

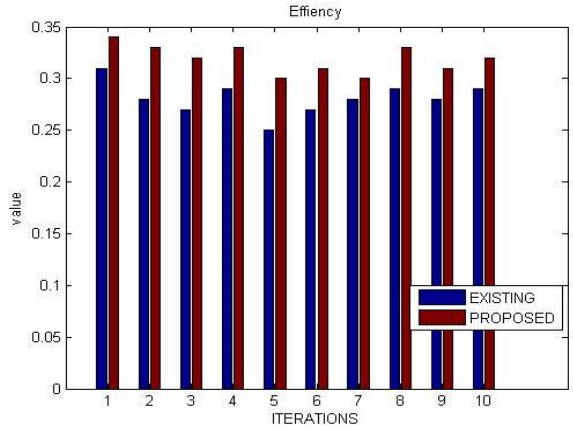

Fig.7. Efficiency

F. Consumed energy

The amount of energy or power used to complete the several tasks.

$$Energy\ consumption = m_k * Tx + F_t * I_t$$

Where

$m_k = Makespan$

$F_t = finishing\ time$

$I_t = Initial\ time$

$Tx = Transformed\ energy$

In this table shows that the less energy consumes as compare to existing technique.

Table 12. Consumed Energy

Iterations	Existing	proposed
1	20.21	18.19
2	20.24	18.97
3	21.14	18.62
4	21.61	17.79
5	20.09	17.32
6	21.06	18.66
7	21.88	18.01
8	21.20	17.27

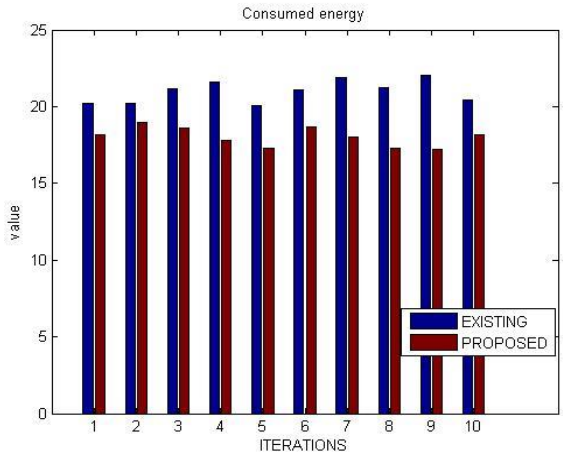

Fig.8. Consumed Energy

VI. Conclusions and Future Work

This paper proposes a hybrid ABC and tabu search scheduling algorithm for cloud environment. As a decision maker ABC is used in different areas and it takes the correct decision using the tabu search. So this technique designed and implemented it applying cloud simulator tool with help of CloudSim using python language, which goal to balance the load of virtual machines. The comparison has been drawn with the existing and proposed technique using parameters: - speedup, utilization, total time taken, efficiency, energy consumption and makespan. The experimental result has shown that it increases the convergence rate. So that the improvement between results clearly shown as the speedup is 0.158, total time taken is

0.858, utilization is 1.15, efficiency is 0.038 and energy consumed is 2.97.

This work has not focused on fault tolerance therefore in near future we will proposed fault aware meta-heuristic scheduling technique which initially monitor active nodes then try to schedule available jobs on available servers only. In case any failures of server exit during runtime then propose technique can move the jobs running or waiting on that severs to others.

REFERENCES

[1] Ajit, M., and G. Vidya, "VM level load balancing in cloud environment," Computing, Communications and Networking Technologies (ICCCNT), 2013 Fourth International Conference on. IEEE, 2013.

[2] Al-maamari, Ali, and Fatma A. Omara, "Task scheduling using PSO algorithm in cloud computing environments," *International Journal of Grid and Distributed Computing, vol:* 8, issue: 5, pp: 245-256, 2015.

[3] Awad, A. I., N. A. El-Hefnawy, and H. M. Abdel_kader, "Enhanced particle swarm optimization for task scheduling in cloud computing environments," International Conference on Communication, Management and Information Technology, *vol:* 65, pp: 920-929,2015.

[4] Babu, K. R., P. Mathiyalagan, and S. N. Sivanandam, "Pareto based hybrid Meta heuristic ABC–ACO approach for task scheduling in computational grids," *International Journal of Hybrid Intelligent Systems, vol:* 11, issue: 4, pp:241-255, 2014.

[5] Dasgupta, Kousik, et al, "A genetic algorithm based load balancing strategy for cloud computing," International Conference on Computational Intelligence: Modeling Techniques and Applications (CIMTA), vol: 10, pp: 340-347, 2013.

[6] Domanal, Shridhar G., and G. Ram Mohana Reddy, "Optimal load balancing in cloud computing by efficient utilization of virtual machines," *Communication Systems and Networks (COMSNETS), Sixth International Conference on*. IEEE, 201).

[7] Effatparvar, M., and M. S. Garshasbi, "A genetic algorithm for static load balancing in parallel heterogeneous systems," International Conference on Innovation, Management and Technology Research, pp: 358-364, 2014.

[8] Ge, Fatemeh Rastkhadiv and Kamran Zamanifar, "A Task Scheduling Algorithm Based on Load Balancing in Cloud Computing," International Journal of Advanced Biotechnology and Research, Vol: 7, Issue: 5, pp: 1058-1069,*2016.*

[9] Jena, R. K, "Multi objective task scheduling in cloud environment using nested PSO framework," *Procedia Computer Science* 57, pp: 1219-1227, (2015).

[10] Karaboga, Dervis, and Bahriye Akay, "A comparative study of artificial bee colony algorithm," *Applied mathematics and computation, vol:* 214, issue: 1, pp: 108-132, 2009.

[11] Karaboga, Dervis, and Bahriye Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of global optimization,* vol: 39, issue: 3, pp: 459-471, 2007.

[12] K.R. Remesh Babu and Philip Samue, "Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud," Innovations in Bio-Inspired Computing and Applications. Springer International Publishing, pp.:67-78, 2016.

[13] LD, Dhinesh Babu, and P. Venkata Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments," *Applied Soft Computing,* vol: 13, issue:5, pp: 2292-2303, 2013.

[14] Liu, Yu, et al, "DeMS: a hybrid Alla, Hicham Ben, et al, "An Efficient Dynamic Priority-Queue Algorithm Based on AHP and PSO for Task Scheduling in Cloud Computing," Springer International Publishing AG *International Conference on Hybrid Intelligent Systems* Springer, Cham, pp: 134-143, 2016.

[15] Masdari, Mohammad, et al, "A Survey of PSO-Based Scheduling Algorithms in Cloud Computing," *Journal of Network and Systems Management,* pp: 1-37, 2016.

[16] Masdari, Mohammad, et al, "Towards workflow scheduling in cloud computing: A comprehensive analysis," *Journal of Network and Computer Applications,* vol: 66, pp: 64-82, 2016.

[17] Science, C, & Engineering, S, "Differential Evolution Based Optimal Task Scheduling in Cloud Computing," International Journal of Advanced Research in Computer Science and Software Engineering, vol: *6, issue:* 6, pp:340–347, 2016.

[18] Shaw, Subhadra Bose, and A. K. Singh, "A survey on scheduling and load balancing techniques in cloud computing environment," *Computer and Communication Technology (ICCCT), 2014 International Conference on*. IEEE, pp: 87-95, 2014.

[19] Zuo, Xingquan, Guoxiang Zhang, and Wei Tan, "Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud," *IEEE Transactions on Automation Science and Engineering, vol*: 11, issue: 2, pp: 564-573, 2014.

Authors' Profiles

Priya Sharma is pursuing her M.tech in Computer science Engineering and Technology from Guru Nanak Dev University. Her research interests include cloud computing, grid and distributed computing. She has published various papers in the field of cloud computing.

Kiranbir kaur is Assistant Professor in Guru Nanak Dev University, Amritsar, India. She is currently pursuing her PhD in cloud computing. Her area of interest includes parallel and distributed computing, and cloud computing.

How to cite this paper: Priya sharma, Kiranbir kaur, "Hybrid Artificial Bee Colony and Tabu Search Based Power Aware Scheduling for Cloud Computing", International Journal of Intelligent Systems and Applications(IJISA), Vol.10, No.7, pp.39-47, 2018. DOI: 10.5815/ijisa.2018.07.04

