I.J. Mathematical Sciences and Computing, **2018, 1, 41-55**
Published Online January 2018 in MECS (http://www.mecs-press.net)
DOI: 10.5815/ijmsc.2018.01.03

Available online at **http://www.mecs-press.net/ijmsc**

Multiobjective Artificial Bee Colony based Job Scheduling for Cloud Computing Environment

Neha Sethi[a], Dr.Surjit Singh[b], Dr.Gurvinder Singh [c]

[a,b]*IKG Punjab Technical University, Jalandhar,Punjab,India*
[c]*Guru Nanak Dev University, Amritsar, Punjab, India*

Received: 01 August 2017; Accepted: 22 September 2017; Published: 08 January 2018

Abstract

Cloud computing has become the hottest issue due to its wide range of services. Due to a large number of users, it becomes more significant to provide high availability of services to cloud users. The majority of existing scheduling techniques in the cloud environment is NP-Complete in nature. Many researchers have utilized meta-heuristic techniques to schedule the jobs in cloud data centers. The majority of existing techniques such as Genetic Algorithm, Ant colony optimization, Non-dominated Sorting Genetic Algorithm (NSGA-III), etc. suffer from poor convergence speed. Also, most of these techniques are either based upon scheduling or load balancing. Therefore, to overcome these issues, a new Variance Honey Bee Behavior with multi-objective optimization method (VHBBMO) is proposed in this paper. Extensive experiments have been conducted by considering the various set of jobs. The experimental results have shown that the proposed method provides more significant results than available methods.

Index Terms: Ant Colony Optimization, Job Scheduling, Honey Bee Colony, Particle Swarm Optimization.

1. Introduction

Cloud Computing is a design for supporting useful, on-demand network usage of any discussed no. of configurable computing assets which could be quickly provisioned as well as free with least supervision attempts or even service provider interface [1]. To provide high availability of services cloud service provider utilizes various scheduling techniques. In general, evaluating optimal schedule can be an NP-hard difficulty whereas heuristic strategies will offer next to ideal methods intended for complicated problems [2]. Cloud is not a grid computing environment only, it also provide X as a Service where X can be anything such as storage,

* Corresponding author. Neha Sethi Tel.: +918146584449
E-mail address: neha_tejpal@yahoo.com

platform, infrastructure, software etc. [3]. The job scheduling techniques are used to improve the high availability to cloud user with minor delay [4]. The complexity of scheduling situation grows along with the length of the actual lines and also becomes highly intricate to resolve it efficiently. To acquire good methods to solve this crisis new heuristic techniques that provide near to optimal solution for large grids are used [5]. Although scheduling problem is the NP-complete problem, one may evaluate the near to optimal scheduling using meta-heuristics approaches [6].

A hybrid multi-objective Particle Swarm Optimization is designed to schedule jobs for cloud data centers [7] efficiently. But this technique suffers from poor convergence speed. The agent-based prioritized dynamic round robin method is developed to efficiently schedule the jobs in cloud data centers [8]. But this technique is limited to a single objective problem only.

A parallel job scheduling method is designed to schedule the parallel workload [9]. The list scheduling based server assignment technique is developed which always evaluate an optimal server assignment that minimizes the makespan [10]. A novel multi-agent reinforcement learning method, called Ordinal Sharing Learning (OSL) method, is proposed for job scheduling problems. The OSL method can achieve the goal of load balancing efficiently [11]. A new parallel job scheduling policy based on integer linear programming is proposed. The optimization problem determines which jobs should run and at which frequency [12].

Metric-aware scheduling is proposed, which enables the scheduler to balance competing schedule goals represented by different metrics such, fairness, and system utilization [13]. A multi-objective optimization approach, i.e., maximizing the successful execution rate of jobs and minimizing the combined cost, and minimizing the fairness deviation of profits [14]. An Adaptive Scoring Job Scheduling algorithm (ASJS) is proposed. Compared to other methods, it can decrease the completion time of submitted jobs, which may consistof computing-intensive jobs and data-intensive jobs [15].

Multi-hybrid policy decision problem which is based on the primary-backup fault tolerance model theoretically show its NP-completeness. The proposed scheme confidently guarantees the fault-tolerant performance by adaptively combining jobs and resources with different rescheduling policies [16]. An agent based job scheduling algorithm for efficient and effective execution of user jobs is proposed. It also includes a statistical analysis of real workload traces to present the nature and behavior of jobs [17]. A Multiobjective Variable Neighborhood Search (MVNS) algorithm for scheduling independent jobs on the computational grid is carried out. This algorithm performs better than other metaheuristics methods [18]. A game theory based job scheduling algorithm is proposed for efficiently scheduling jobs in cloud computing**.** First of all, the game theory based scheduling can better coordinate the distribution of job and the distribution of energy. The job scheduling model for computing nodes by establishing mathematical model is proposed to deal with big data [19]. An Ant colony optimization (ACO) based scheduling technique is proposed. ACO based scheduling outperforms over the most of the metaheuristic techniques, but suffer from poor convergence [20]. Although PSO based scheduling outperforms over the available methods but suffers from the initial selection of particles. Thus, performs inconsistently every time.

Contribution: In this paper, following contributions are done.

i. In this paper, we propose a job scheduling technique considering honey bee colony optimization for cloud computing environment.
ii. First of all, the variance based honey bee colony can better coordinate the distribution of jobs and the allocation of jobs.
iii. The load between the High-end servers (HES) are also balanced to reduce the makespan further.
iv. In a word, job scheduling and load balancing technique considering variance based honey bee colony with multi-objective fitness function are designed for cloud computing environment.

Rest of paper can be represented as follows: In Section 2, mathematical model of proposed technique is demonstrated. In Section 3, proposed technique is described. The experimental set-up and results are demonstrated in Section 4. The conclusion and future work are outlined in the last Section.

2. Model Development

To handle the issue of job scheduling in cloud computing environment, a typical cloud model is designed as illustrated in Fig.1. Cloud model contains several geologically distributed high end servers associated using internet. Principally high-end servers consists of numerous computing and storing resources. These high-end servers communicates with each other using a high bandwidth intercommunication network. Thus, in designed cloud model, transmission delay does not play a significant role. In designed cloud environment, each user can utilize cloud resources with the help of internet. Cloud service provider is responsible for allocating or deallocating the resources to users. The user jobs are disseminated between several cloud data centers ($\bar{D}C_S$). Each $\bar{D}C_S$ decompose user job into sub-jobs so called jobs and allocate it between available Processing Elements $(P_{ro}E_S)$ in the respective $\bar{D}C_S$. The designed job scheduling technique is responsible for efficiently assigning user jobs into available $\bar{D}C_S$ with an objective to reduce the makespan time and average waiting time. In Figure 1, '$\bar{D}C$' shows cloud data center and 'PE' represents the sets of $P_{ro}E_S$.

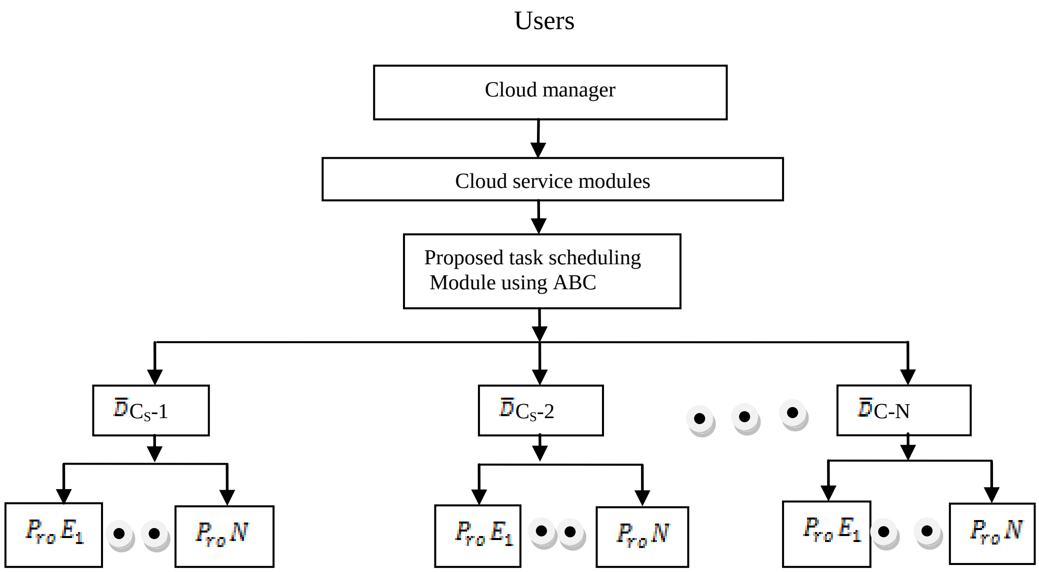

Fig.1. Cloud model

2.1. Problem Formulation

In designed model, a cloud application is taken as a group of jobs which carry out some computationally intensive jobs by considering cloud resources. Assume that Job $= (U_{S\,1}, U_{S\,2}, U_{S\,3} \dots U_S M)$ is a group of applications received in a specific period of time. Every job (U_{Sj}) is considered by a duplet $< \acute{A}_i, D_i >$. In which $\acute{A}_i$ defines arrival time of job (U_{Sj}) and D_j represents deadline of job(U_j). If a job could not finish within deadline time, then it is referred as a failed job and queued again for further processing. Throughout the scheduling procedure, jobs are allocated to data centers $(\bar{D}C'_s)$ ($\bar{D}_1$, $\bar{D}_2$, $\bar{D}_3$ … $\bar{D}_N$) , where $N \leq M$. Each $\bar{D}_j$ is associated with a duplet $< c_j, n_j >$. c_j, the cost per unit time charged by $(\bar{D}C'_s)$ to implement jobs, n_j is the number of available Processing Elements (PEs) to implement jobs. Each $(\bar{D}C'_s)$ have set of PE $\{P_{ro}E_1, P_{ro}E_2 \dots P_{ro}E_n\}$ to evaluate assigned job. Each PE is associated with a duplet < s, p > 's' and 'p' represents the burst time and energy consumption of each PEs respectively. Every Job is demonstrated as a Directed Acyclic Graph (DAG), represented as $g(v, E)$ (demonstrated in Figure 1). The set of nodes $= \{t_1 .., t_m\}$ shows jobs, and the set of arcs represents the data dependencies among jobs. An arc is in the form of $< t_j, t_i > \in E$, where t_j represent parent job and $t_j$is leaf job.The leaf jobcannot be implemented until all of its root jobs have been implemented. In a given DAG (Fig. 2), a job with no parent is referred as an *root job*, and a job without leaf node is referred as *exit job*.

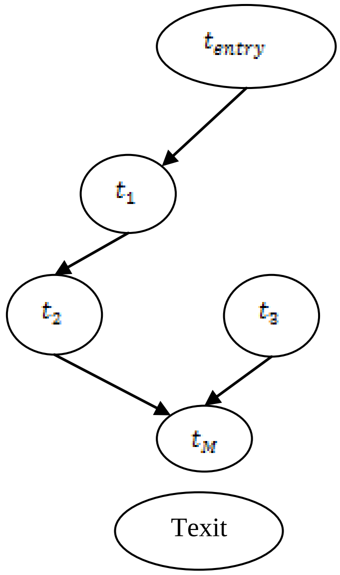

Fig.2. Layout of Directed Acyclic Graph

Each vertex E in DAG is associated with a value $< l >$, 'l' demonstrates the size of job in Million Instruction (MI).

2.2. Objective Function

Assume that user job (U_j) is allocated to data center $\overline{D}_j$. t_A define set of jobs (U_j) allocated to a PE (P_j). If the time demands executing t_A using P_i is represented by Γ_j. The deadline time of $t_i$can be evaluated as follows:

$$Finish\,(\Gamma_j) = start\,(t_A) + \Gamma_j$$

Therefore, burst time required to finish the job by $\overline{D}_i$ is represented by Makespan (MS_i) and calculated as follows:

$$MS_i = max\{Finish\,(t_A)\}$$

Where $t_{(A=1\ldots n)}$the jobs are assign to $\overline{D}_i$. The Energy consumption (E_j) to evaluate a job (U_j) by is evaluated as follows:

$$E_j = \sum_{A=1}^{m} (\Gamma_A \times P_A)$$

where P_A represent power consumed per unit time by PE (P_i) to execute given job (t_A). The cost to execute the job by $\overline{D}_i$is evaluated as follows:

$$c_j = C_j \times MS_j$$

Where c_j is the price per unit time charged by $\overline{D}_i$to implement job. The utilization (U_j) of $\overline{D}_i$ is evaluated as follows:

$$U_j = \frac{Makespan}{\max\{Makespan_k\} A = 1 \ldots . N}$$

The fitness functions of this proposed model can be represented as follows:

$$\boldsymbol{Minimize}\ MS_{j,}\ j = 1..N$$

$$\boldsymbol{Minimize}\ \mathrm{E_i}\ i = 1..N$$

$$\boldsymbol{Minimize} = \sum_{i=1}^{N} c_j$$

$$\boldsymbol{Maximize}\ U_j\ i = 1..N$$

Subject to:

1. The job must finish before deadline (D_i)
2. Every job can be assigned to only one $\overline{D}_i$.
3. Number of jobs must be less than the number of available Data.

3. Proposed Technique

In the Artificial bee colony (ABC), the colony contains three types of bees: employed, onlookers and scouts. The number of employed bees is equal to the number of food destinations towards the hive. Employed bees initially move towards its destination and return to hive back and start dancing. Any bee whose food does not lies between the desired solution space will alter its path. Onlookers bees monitor the dancing of bees and determine target based upon dance.

In ABC, a location of a food source shows a result for scheduling problem and food source is makespan of the associated schedule. Initially, schedules are generated randomly. Then, the population has repeated the iterations of search space and processes of the employed, onlooker, and scout bees, respectively. Every time used bee try to modify its schedule in such a way that the overall makespan time can be reduced. If the new schedule has lesser makespan than its old schedule, it will memorize new solution and forget the previous one. When all employed bees return their result, they start doing dancing. Then onlooker determines the best solution by evaluating the minimum makespan from the solutions provided by employed bees. This procedure keeps on going till the stopping criteria is not met.

3.1. Variance Honey Bee Behaviour Based Multi Objective Optimization Technique (VHBBMO)

Most of scheduling techniques for cloud data centers are NP-Completer problems. Many researchers have used various meta-heuristic approaches to evaluate the best schedule for the cloud environment. However, each has its benefits and limitations. Like Ant colony optimization (ACO) and Genetic Algorithm (GA) suffers from poor convergence speed. Whereas Particle swarm optimization has good speed due to velocity but limited to the initial set of particle problem. Thus, in this paper honey bee colony is improved further using the variance among the schedules. Also, the multi-objective fitness function is also designed to enhance the results further. The proposed technique initially schedules the jobs on high-end servers and then try to balance the load between these high-end servers. Table 1 represents various symbols along with their meaning, which is used in the mathematical model of VHBBMO.

Table 1. Nomenclature used

Symbols	Meaning
δ	Set of high-end servers
θ	Set of Execution Tasks
η	Non- Preemptive Tasks
β_{max}	Makespan
P	Parallel or Related Machines
∂_θ	Execution time of job
γ	Execution element
CP	Capacity of high-end servers
LD	Job on Virtual Machines
ρ	Standard Deviation of Jobs
μ	Threshold Condition Set
D	Degree of Imbalance
uHES	Supply of All high-end servers
oHES	Demand of All high-end servers
AHES	Less Loaded high-end servers
DHES	More Loaded high-end servers
BHES	Balanced Virtual Machines
σ^2	Variance

3.2. Mathematical Model

Let $\delta=\{\delta_1,\delta_2,\cdots\delta_n\}$ be the set of *'n'* HESs which should process *m* jobs represented by the set $\theta=\{\theta_1,\theta_2,\cdots\theta_m\}$. The finishing time of a job θ_a is denoted by β_a. The makespan has been denoted as $\beta_{\max}$. So the model is $P|\eta|\beta_{\max}$.

Execution time of a job θ_a on virtual HES δ_b has been denoted as ∂_{ab}. Execution time of all jobs in $\delta_b$has been defined by Eq. (1).

$$\partial_b = \sum_{a=1}^{n} \partial_{ab}\, b = 1, \dots, m \quad (1)$$

Eq. (2) is obtained by minimizingβ_{max}. Eq. (3) is derived From Eq. (1) and (2).

$$\sum_{a=1} \partial_{ab} \le \beta_{max}\, b = 1, \dots, m \quad (2)$$

$$\sum_{a=1} \quad \partial_b \le \delta_{max}\, b = 1, \dots, m \quad (3)$$

At the time of job scheduling, the job may migrate from one HES to other in order to reduce β_{max} along with response time. Execution time of a job differs from one HES to other on the basis of HES's capacity and speed. The load balancing among HESs will be done using migration of jobs. Optimally,

$$\beta_{max} = \{max_{a=1}^{n} \beta_a, max_{b=1}^{n} \sum_{a=1}^{n} \partial_{ab} \quad (4)$$

Job scheduling technique using the ABC is a dynamic technique which not merely balances the job. But, additionally, considers the priorities of jobs in the waiting queues of HESs. While balancing the load between HESs, the jobs taken from overloaded HESs behave as Honey Bees and will migrate to less loaded HESs. Therefore, migrations of jobs balance the load among HESs. Thus, it automatically minimizes the makespan time. Because all HESs are arranged in increasing order, the job eliminated will be submitted to less loaded HESs. Existing workload of available HESs may be determined by the information received from the waiting queue. The standard deviation must be calculated for jobs available in waiting for queues to measure variations of jobs on HESs.

Capacity of a HESs:

$$\mathrm{CP_a} = \gamma_{\mathrm{no.a}} \times \gamma_{\mathrm{mipsa}} + \delta_{\mathrm{bwa}} \quad (5)$$

where execution element, $\gamma_{\mathrm{no.a}}$ is the number of processors inδ_a, γ_{mipsa} is million instructions per second of processors in δ_a and δ_{bwa} is the communication bandwidth ability ofδ_a.

Capacity of all HESs:

$$\mathrm{CP} = \sum_{a=1}^{n} \mathrm{CP_a} \quad (6)$$

Summation of capacity of all HESs is the capacity of cloud data center.

Job on HESs:

Total length of jobs that are assigned to HES is called jobs available on HESs.

$$\mathrm{LD}_{\delta_a,\theta} = {}^{\mathrm{N}(\theta,\mathrm{T})}\!\big/_{\mathrm{SR}(\delta_a,\theta)} \quad (7)$$

Jobs running on HESs can be calculated by determining the number jobs at time T, on service queue of δ_a, which is divided by the service rate of δ_a at time T. Job of all HESs in a data center is calculated as follows:

$$LD = \sum_{a=1}^{n} LD_{\delta_a} \tag{8}$$

Execution time of HES:

$$pt_a = {}^{LD_{\delta_a}}\!/_{CP_a} \tag{9}$$

Execution time of all HESs:

$$pt = {}^{LD}\!/_{CP} \tag{10}$$

Standard deviation of job:

$$\rho = \sqrt{\frac{1}{n}\sum_{a=1}^{n}(pt_a - pt)^2} \tag{11}$$

JOB SCHEDULING DECISION

After applying the jobs scheduling using the proposed technique, load balancing at run time will come in action. Load balancing will migrate some jobs of heavily loaded HESs to underloaded HESs. To achieve it following criteria will be used.

1. Evaluating State of the HES group

Initially, state of the HES will be determined. The goal of this step is to determine whether the current schedule is balanced or not. If $If \rho \leq \mu$ then it is assumed that schedule is in balance state and no further load balancing is required. It can be determined as follows:

$$\begin{gathered} If \rho \leq \mu \\ System\ is\ balanced \\ Exit \end{gathered}$$

2. Evaluating the Overloaded HESs

If the current work load associated with HESs is higher than the maximum volume associated with the HESs, HESs is overloaded. Job scheduling is not possible in these circumstances.

$$\begin{array}{ll} & \text{If LD} > max.capacity \\ & \quad Job\ scheduling\ is\ not\ possible \\ Else & \\ & \quad Trigger\ job\ scheduling. \\ & \quad end \end{array}$$

3.3. Proposed Technique

If the decision would be to balance the job, the scheduler should trigger the job scheduling characteristics. To balance the load among HESs, overloaded and less-loaded HESs will be determined. To balance the load, remove one job at a time from overloaded HESs and migrate it to suitable underloaded HESs. Job will be migrated based upon their priority level. In this paper, we have given highest priority to jobs with minimum burst time. Scout bees are responsible for evaluating the over and under loaded HESs. Forager bees are responsible for migrating the jobs from one HESs to another. This Forager bee will become Hunt bee for next job. These steps remain until HESs become balanced or number of jobs finish up their tasks. HESs selection will be completed as follows:

$$\theta_H \rightarrow \delta_d | minimum\,(\sum \theta_h) \in \delta_d \tag{12}$$

$$\theta_M \rightarrow \delta_d | minimum\,(\sum \theta_h + \theta_m) \in \delta_d \tag{13}$$

$$\theta_L \rightarrow \delta_d | minimum\,(\sum \theta) \in \delta_d \tag{14}$$

here $\theta_H, \theta_M, \theta_L$ are the jobs of high, medium and low priority queues. The priorities associated with jobs are usually assembled into three queues (high, method, low).

Algorithm VHBBMO

Scheduling and balancing of load among available HESs.

1. On the basis of eq. (1), (2), (3), (4), Check the system is balanced or not. And also evaluate capacity and job of all HESs:

   $If \sigma <= \mu$

   System is balanced

   Exit

2. Job Scheduling Decision:

   $If LD > max.capacity$

   Job scheduling is not possible

   Else

   Trigger job scheduling.

3. Group HESs on the basis of Job as *AHES*, *DHES*, *BHES*
4. Job Scheduling:

   Supply of All HESs in *uHES*is

   $Supply of\, AVM_b = max.capacity - {}^{LD}/_{CP}$

   Demand of All HESs in *oHES* is

   $Demand of\, DVM_b = {}^{LD}/_{CP} - max.capacity$

   Arrange HESs in *DHES* by desc order

   Arrange HESs in *AHES* by asc order

   While $AVM \neq \emptyset and DVM \neq \emptyset$

   For s=1 to * *(DHES)* do

   Arrange jobs in HESs by selection criterion (priority)

   For all job *t* in HESs evaluate HES $vm_d \in AVM$ such as

   If (*t* is non- preemptive)

   $\theta_H \rightarrow \delta_d | \sigma^2 (\sum \theta_h) \in \delta_d and\, LD_{\delta_d} \leq Capacity_{\delta_d}$

$\theta_M \rightarrow \delta_d | \sigma^2(\sum \theta_h + \theta_m) \in \delta_d \, and \, LD_{\delta_d} \leq Capacity_{\delta_d}$
$\theta_L \rightarrow \delta_d | \sigma^2(\sum \theta) \in \delta_d \, and \, LD_{\delta_d} \leq Capacity_{\delta_d}$
If (θ is preemptive)
$\theta_H \rightarrow \delta_d | \sigma^2(\sum \theta_h) \in \delta_d$
$\theta_M \rightarrow \delta_d | \sigma^2(\sum \theta_h + \theta_m) \in \delta_d$
$\theta_L \rightarrow \delta_d | \sigma^2(\sum \theta) \in \delta_d$
Allocate the no. of jobs assigned to δ_d
Allocate the no. of priority jobs assigned to δ_d
Allocate Job on both HESsδ_d.
Allocate sets DHES, AHES, and BHES
Arrange HESs in DHES by descending order
Arrange HESs in AHES by ascending order
The three sets based on job of the HESs. They are
AHES (Less loaded HES) — the set contains the HESs of less loaded.
DHES (More loaded HES) — the set contains all more loaded HESs
BHES (Balanced HES) — remaining each HESs tends to be well-balanced and perhaps they are to be found in set.

4. Experimental Results

This section describes the experimental setup for cloud computing environment. MATLAB 2013a tool is use with the help of parallel processing toolbox to balance the load between HESs. The Dell notebook computer is used with 8 GB RAM, 2.4 GHz Intel core i5 processor with 2GB GPU built in. The proposed and other selected techniques (i.e., PSO [21], ACO [20], MVNS [18], and Game Theory [19]) are designed and implemented on the same experimental platform. 4000 jobs are tested on every technique. Following subsection describes the comparison of proposed method with existing techniques.

Table 2 and Fig. 4 demonstrate the comparison between MVNS [18], ACO [20], PSO [21], Game Theory [19] with VHBBMO on average response time (in seconds). The Table 2 and Fig. 4 have shown that the proposed technique takes lesser time compared to existing approaches. Thus, proposed method is more efficient than other techniques in terms of response time. It has been observed average response time increases whenever there is increase in number of tasks. But, in the case of proposed technique, it has been found that the proposed method has less variation (increase) in response time than earlier methods. From Table 2 and Fig.3, it has been proved that the proposed technique has significantly decreased the average response time of jobs. Compared with other methods the proposed method has considerably reduced the mean response time i.e. 2.473%. It shows that proposed technique is more suitable for real-time cloud computing environment.

Table 2. Average Response Time Analysis

No. of Tasks	MVNS [18]	ACO [20]	PSO [21]	Game Theory [19]	VHBBMO
1000	1.21±0.89	1.13±0.81	1.09±0.75	0.82±0.71	0.81±0.63
1500	1.92±0.98	1.72±0.87	1.67±0.84	1.48±0.81	1.21±0.79
2000	2.81±0.93	2.63±0.91	2.36±0.86	2.01±0.84	1.85±0.81
2500	3.59±0.89	3.11±0.84	2.73±0.83	2.28±0.77	2.10±0.72
3000	4.17±0.97	3.42±0.94	2.93±0.88	2.37±0.79	2.19±0.73
3500	5.24±1.16	5.08±1.13	4.12±0.98	3.24±0.86	2.72±0.69
4000	6.11±1.24	5.69±1.09	4.78±0.91	3.95±0.87	3.58±0.78

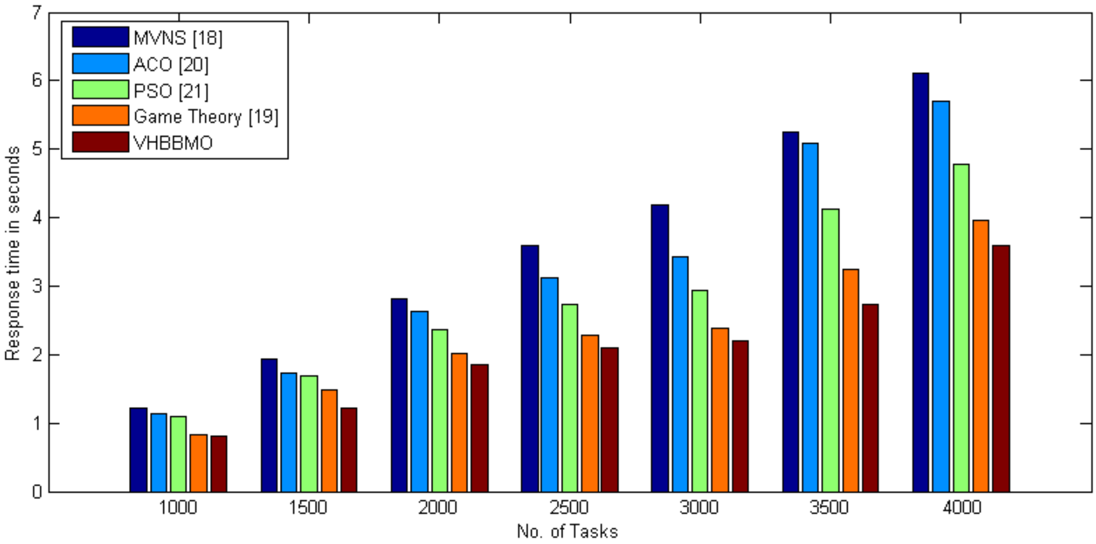

Fig.3. Comparative Analysis of Mean Response Time in Seconds

Table 3 and Fig.4 demonstrate the comparison between MVNS [18], ACO [20], PSO [21], Game Theory [19] with VHBBMO regarding makespan time (in seconds). Table 3 and Fig.4 depicts that the proposed technique has lesser makespan when compared with existing technologies. Because the mean reduction in makespan in seconds is approximately 4.7985 %. Therefore, it indicates that the VHBBMO has smaller makespan than earlier techniques. Also, when the logical analysis is considered (i.e., a range of the makespan) it has been observed that proposed method is more significant than earlier techniques. Because average variation in makespan is 129 seconds earlier which was 187, 178,169 and 149 in MVNS [18], ACO [20], PSO [21], and Game Theory [19], respectively.

Table 3. Comparison of Makespan

No. of Tasks	MVNS [18]	ACO [20]	PSO [21]	Game Theory [19]	VHBBMO
1000	15181±149	14158±136	12357±117	12126±109	11945±97
1500	21179±158	19215±149	16587±138	15473±126	14468±104
2000	32186±196	29549±184	27348±168	28981±137	27794±117
2500	37155±197	35458±192	32657±176	30146±148	29247±128
3000	43114±208	41145±178	37497±175	36784±139	34498±119
3500	48164±217	42679±204	38487±187	36498±165	35789±148
4000	58165±229	51244±226	49146±209	47657±198	47459±176

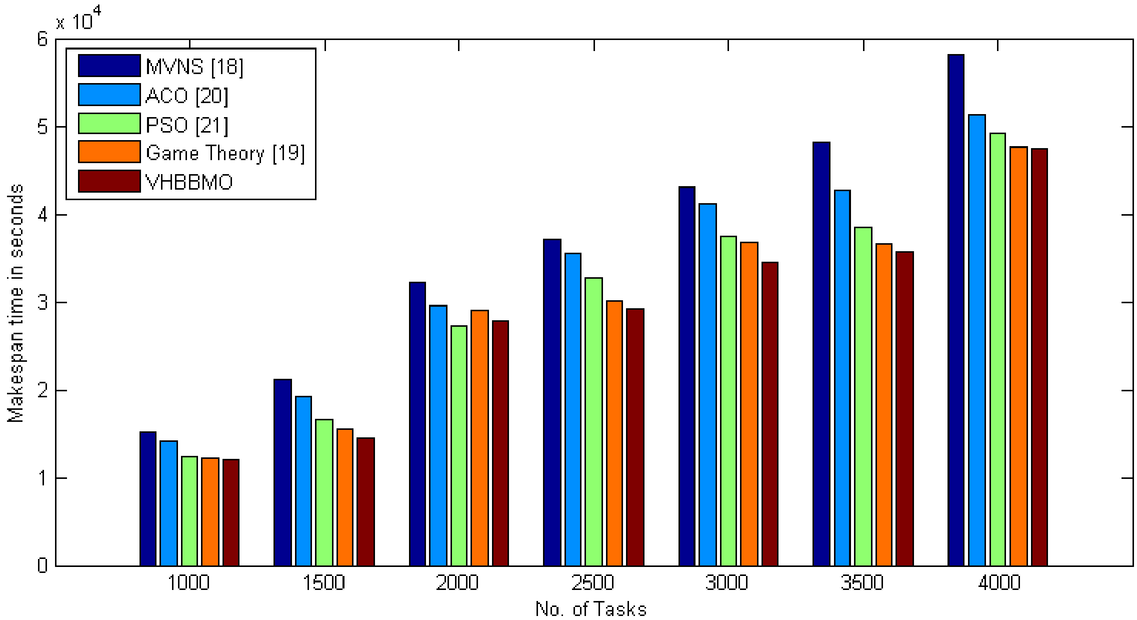

Fig.4. Comparative Analysis of Makespan time in Seconds

Degree of imbalance

$$\square = \frac{\square_{\square\square\square} - \square_{\square\square\square}}{\square_{\square\square\square}} \tag{19}$$

Where $\square_{\square\square\square}$ and $\square_{\square\square\square}$ are the maximum and minimum $\square_{\square}$ along with each HESs, $\square_{\square\square\square}$ is the average $\square_{\square}$ of HESs. Job scheduling system increases the degree of imbalance considerably.

Table 4 and Fig.5 demonstrate the comparison between MVNS [18], ACO [20], PSO [21], Game Theory [19] with VHBBMO regarding a degree of imbalance. A schedule is said to best if it is close to 0 degrees of imbalance. Therefore, from the Table 4 and Fig.5, we have proved that the proposed technique has lesser degree of imbalance. Therefore, proposed technique has balanced the load among HESs in more efficient way than earlier methods. From the Table 4 and Fig.5, it has been observed that the proposed technique has the lesser degree of imbalance compared to earlier methods. The mean reduction in the degree of imbalance is 0.978 % when proposed technique is compared with other scheduling techniques.

Table 4 Comparison Based on Degree of Imbalance.

No of Tasks	MVNS [18]	ACO [20]	PSO [21]	Game Theory [19]	VHBBMO
1000	1.31±0.48	1.61±0.39	0.81±0.37	0.73±0.36	0.71±0.31
1500	1.78±0.51	1.79±0.47	0.71±0.45	0.62±0.46	0.59±0.39
2000	1.16±0.57	1.01±0.47	0.94±0.43	0.81±0.48	0.75±0.43
2500	1.52±0.62	1.32±0.56	1.13±0.49	0.94±0.47	0.91±0.42
3000	1.44±0.59	1.38±0.68	1.19±0.58	0.83±0.53	0.81±0.49
3500	1.91±0.71	1.90±0.79	1.05±0.65	0.72±0.61	0.69±0.56
4000	1.98±0.68	1.92±0.62	1.15±0.61	0.89±0.56	0.87±0.58

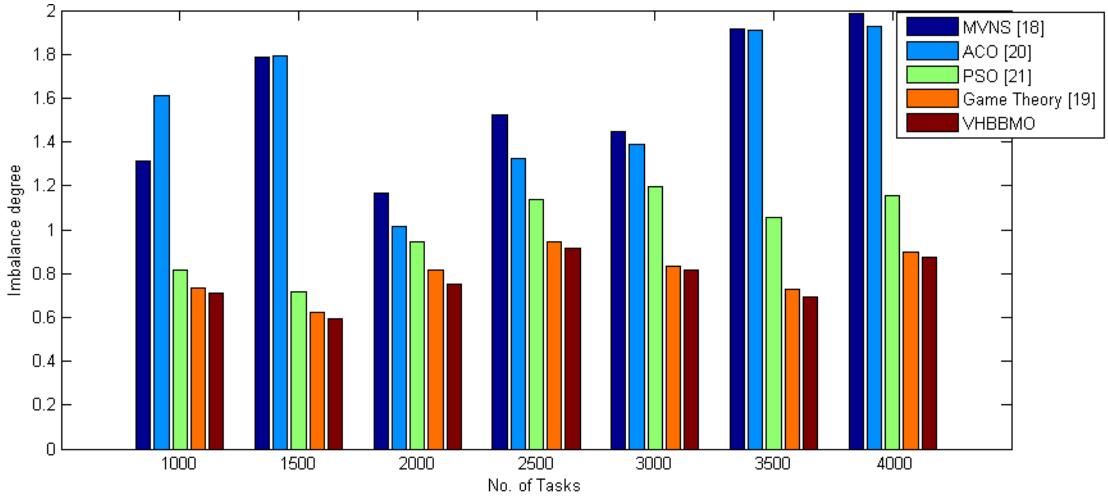

Fig.5. Comparison Based Upon Degree of Imbalance.

From experimental results and discussions, it has been observed that the proposed technique outperforms other job scheduling techniques in terms of mean response time, makespan time, and degree of load imbalancing. Therefore, the proposed technique is more efficient for real-time cloud computing scheduling techniques.

5. Conclusion

A novel VHBBMO job scheduling technique for cloud computing environment is designed by using the honey bee optimization. The proposed technique not only schedule the jobs but also balances the load among highly loaded HESs to under loaded HESs. Proposed technique has good convergence speed as compared to existing meta-heuristic techniques. First of all, proposed technique optimistically assigns the available jobs between high-end servers. The load between the high-end servers is also balanced to reduce the makespan further. The experimental framework is designed in MATAB tool with parallel processing toolbox. Comparison of proposed technique has been drawn with MVNS, PSO, ACO and Game theory based scheduling techniques. Extensive experiments indicate that the proposed technique outperforms over the available techniques.

References

[1] Ganesh, Amal, Sandhya M., Shankar S. (2014) A study on fault tolerance methods in Cloud Computing.In Advance Computing Conference (IACC). IEEE International; pp: 844-849.

[2] Mathiyalagan P., Suriya S., Sivanandam S. (2010) Modified ant colony algorithm for Grid scheduling.International Journal on Computer Science and Engineering 2. vol. no. 02; pp: 132-139.

[3] Banerjee, Soumya, Mukherjee I., Mahanti P.(2009) Cloud Computing initiative using modified ant colony framework. World academy of science, engineering and technology .56; pp: 221-224.

[4] Fidanova, Stefka, Durchova M. (2006) Ant algorithm for grid scheduling problem. In Large-Scale Scientific Computin. Springer Berlin Heidelberg; pp: 405-412..

[5] Verma, Amandeep, Kaushal S.(2017) A hybrid multi-objective Particle Swarm Optimization for scientific workflow scheduling. Parallel Computing 62; pp 1-19.

[6] Liu X., Zha Y., Yin Q., Peng Y., Qin L. Scheduling parallel jobs with tentative runs and consolidation in the cloud, Journal of Systems and Software, Volume 104; Pages 141-151.

[7] Keqin Li, Job scheduling and processor allocation for grid computing on metacomputers. Journal of Parallel and Distributed Computing. Volume 65; Pages 1406-1418

[8] Etinski M., Corbalan J., Labarta J., Valero M. Parallel job scheduling for power constrained HPC systems. Parallel Computing. Volume 38; Issue 12; Pages 615-630.

[9] Wei Tang, Dongxu Ren, Zhiling Lan, Narayan Desai, Toward balanced and sustainable job scheduling for production supercomputers, Parallel Computing, Volume 39, Issue 12, December 2013, Pages 753-768.

[10] Heyang Xu, Bo Yang, An incentive-based heuristic job scheduling algorithm for utility grids, Future Generation Computer Systems, Volume 49, August 2015, Pages 1-7.

[11] Ruay-Shiung Chang, Chih-Yuan Lin, Chun-Fu Lin, An Adaptive Scoring Job Scheduling algorithm for grid computing, Information Sciences, Volume 207, 10 November 2012, Pages 79-89.

[12] Moon Y. K., Youn C.H., Multihybrid job scheduling for fault-tolerant distributed computing in policy-constrained resource networks. Computer Networks. Volume 82, 8 May 2015; Pages 81-95.

[13] Syed Nasir Mehmood Shah, M. Nordin B. Zakaria, Ahmad KamilBin Mahmood, Anindya Jyoti Pal, NazleeniHaron, Agent Based Priority Heuristic for Job Scheduling on Computational Grids, Procedia Computer Science, Volume 9, 2012, Pages 479-488.

[14] Selvi S., Manimegalai D. Multiobjective Variable Neighborhood Search algorithm for scheduling independent jobs on computational grid. Egyptian Informatics Journal.Volume 16; Pages 199-212.

[15] Yong-Hyuk Moon, Chan-Hyun Youn, Multihybrid job scheduling for fault-tolerant distributed computing in policy-constrained resource networks, Computer Networks, Volume 82, 8 May 2015, Pages 81-95.

Authors' Profiles

Neha Sethi is assistant professor at Hindu College, Amritsar, India. She is pursuing Ph.D in computer Applications from Punjab Technical University.

Dr. Surjit Singh is professor at Punjab Technical University. He has experience of more than 30 years and his area of interest is Parallel computing and graphics programming.

Dr. Gurvinder Singh is Professor and dean Faculty of Engg. & Technology at GNDU Amritsar.He has teaching experience of 18 years. His area of Interest is Parallel and distributed Computing.

How to cite this paper: Neha Sethi, Surjit Singh, Gurvinder Singh,"Multiobjective Artificial Bee Colony based Job Scheduling for Cloud Computing Environment", International Journal of Mathematical Sciences and Computing(IJMSC), Vol.4, No.1, pp.41-55, 2018.DOI: 10.5815/ijmsc.2018.01.03