Received: July 22, 2020. Revised: August 30, 2020.

INASS

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Hybrid Max-Min Genetic Algorithm for Load Balancing and Task Scheduling in Cloud Environment

Shilpa Kodli[1]* **Sujata Terdal[1]**

[1]*Department of Computer Science and Engineering, PDA College of Engineering, Kalaburagi, Karnataka, India*
* Corresponding author's Email: shilpakodli@gmail.com

Abstract: In recent decades, task scheduling and load balancing in the cloud is a growing research area, due to the vast amount of data stored in the server highly increases the load. In order to address this concern, Hybrid Max-Min Genetic Algorithm (HMMGA) is proposed for task scheduling and load balancing in the cloud environment. At first, the load is evaluated for every Virtual Machine (VM), if the load is high, then HMMGA is used for balancing the load. HMMGA selects the best VMs to assign the tasks and migrates the over-loaded VMs tasks to the under-loaded VMs. HMMGA significantly avoids the imbalanced workload performance in the cloud environment. In this research paper, the proposed HMMGA performance is compared to Max-Min algorithm, Low time complexity and low cost binary Particle Swarm Optimizer (IBPSO-LBS) and PSO with Technique of Order Preference by Similarity to Ideal Solution (TOPSIS) algorithm to examine the efficacy of HMMGA. From the experimental simulation, the result shows that HMMGA averagely delivers 1.63 and 3.88 seconds less make span compared to the Max-Min and TOPSIS-PSO algorithm for five VMs. In addition, HMMGA averagely enhances 10% to 40% of resource utilization than the Max-Min and TOPSIS-PSO algorithm. In another experiment, the HMMGA approximately showed 1.7 to 25.99 seconds less average waiting time compared to the Max-Min and IBPSO-LBS.

Keywords: Cloud computing, Genetic algorithm, Load balancing, Max-Min algorithm, Task scheduling, Virtual machine.

1. Introduction

In recent decades, cloud computing is a growing technology which has the ability of sharing cloud host and access the distributed environment and virtualize the technologies [1]. The task scheduling and load balancing are two major challenges issued in cloud resource management in order to meet the cloud providers and user's requirements [2]. Therefore, the number of tasks is increasing highly by increasing the number of cloud users, but the VMs remain stagnant. However, the number of VMs is limited to the capacity of physical machines, due to the constraint of energy consumption [3, 4]. The task scheduling and load balancing are used for distributing the work-load to multiple nodes for ensuring that no node is either under-loaded or over-loaded [5-7]. For achieving good performance in task scheduling and load balancing, many heuristic and meta-heuristic methodologies are developed by the researchers such as bee colony optimization algorithm [8], dragonfly optimization algorithm [9] and Jaya algorithm [10]. Most of the heuristic algorithms use only the priority methods such as VM priority, task priority, and sorting methods for increasing the resource utilization and minimizing the execution time. Still, the meta-heuristic and heuristic algorithms have high time complexity in realistic computational infrastructures.

To overcome the above mentioned issues, a new fast meta-heuristic algorithm; Hybrid Max-Min Genetic Algorithm (HMMGA) is proposed for load balancing and task scheduling in a cloud environment. HMMGA focuses on reducing the completion time between the heterogeneous VMs and complex scheduling decision. HMMGA defines two constraints like earliest finish time and optimal completion time with resource utilization to attain

effective performance in task scheduling and load balancing. The contribution of the research paper is as given below:

- The proposed model Hybrid Max-Min Genetic Algorithm (HMMGA) meets the requirements of cloud providers and users simultaneously with limited computational time.
- The proposed model optimizes based on two constraints such as earliest finish time and optimal completion time with respect to resource utilization for effective performance.
- HMMGA optimally schedules the tasks and balances the user tasks to different VMs in the cloud environment with limited cost.
- By using HMMGA, cloud providers reduce the makespan, wait time, load unbalancing, time complexity and running time and improves the resource utilization to increase the number of user tasks to gain more profit.

A few existing research papers are surveyed in the Section 2. Section 3 explains about HMMGA with pseudo code and mathematical expressions. In Section 4, the performance of the proposed and existing models is analyzed with performance measures. Finally, the conclusion is made in Section 5.

2. Literature survey

Chaudhary and Kumar [11] implemented a new load scheduling algorithm; Hybrid Genetic Gravitational Search Algorithm (HG-GSA) to decrease the power of computation cost. The developed algorithm finds the optimal position of the particle in the search space that was used to calculate the force. In load scheduling, the developed algorithm (HG-GSA) dramatically reduces the total computation cost and delivers higher user satisfaction. The experimental outcome shows that the developed algorithm attained superior performance in load balancing by means of total cost analysis, normalized total cost of cloudlets and mean of the normalized pairwise distance. In the condition of a large number of VMs and cloudlets, the developed HG-GSA algorithm minimized the mean of pairwise distance between the iterations that results in poor performance.

Kaur and Kaur [12] used Hybrid Heterogeneous earliest finish time heuristic with Ant colony optimizer (HHA) and Hybrid Predict earliest finish time heuristic with ant colony optimizer (HPA) for load balancing in a cloud environment. The developed hybrid optimization algorithms utilized genome, ligo and cybershake workflows for execution. In this literature work, the performance of developed algorithm was verified based on cost and makespan. The developed optimization algorithms have less ability for executing the direct acyclic graph file (tasks), while the smaller number of VM is available for execution. In addition, Kong [13] developed a fast heuristic load balancing algorithm on the basis of the zero imbalance method in a cloud environment. In order to attain effective task scheduling and load balancing, the developed algorithm defines two constraints like earliest finish time and optimal completion time. In this research study, the developed algorithm performance was analyzed in light of the degree of imbalance, scheduling time, resource usage, makespan, monetary cost and waiting time. In the data-center, power consumption remains one of the main factors that have great impact on load balancing, where the developed work failed to focus.

Mapetu [14] developed IBPSO-LBS algorithm with minimum cost and time complexity of balancing and scheduling the tasks. The developed optimization algorithm significantly reduces the waiting time of a user request and also helps in achieving better task scheduling and load balancing in a cloud environment. In addition, Panwar [15] developed PSO with TOPSIS method for scheduling the tasks in a cloud environment. Initially, the TOPSIS methodology was utilized to obtain the relative tasks on the basis of scheduling criteria. Then, PSO algorithm was used to compute the relative closeness of the given criteria for all the tasks in all VMs. In this scenario, the performance of load balancing was influenced by energy consumption and live migration for satisfying both the cloud providers and users, which was a major concern in these studies.

Priya [16] developed a novel fuzzy based resource scheduling algorithm in cloud infrastructure. The developed multi-dimensional resource scheduling algorithm and queuing network achieves better data sharing in a cloud environment with low computation complexity. In this research study, the developed algorithm performance was validated by means of response time, the average success rate and resource scheduling efficiency. In contrast, the developed fuzzy based algorithm was not concentrating on the data privacy, where privacy preserving was one of the essential factors in data or information sharing. In addition, some of the major concerns faced by the researchers in load balancing and task scheduling are listed as follows.

- For load balancing, most of the prior research works did not concentrate on the resource ability and user requirements.

- Due to the high number of tasks, the existing techniques faced the challenges like high makespan and low resource utilization.

To address the above mentioned issues, HMMGA is proposed to attain better performance in task scheduling and load balancing in the cloud.

3. Proposed methodology

Due to the increasing of users, cloud computing meets several issues in resource sharing such as task scheduling and load balancing. Recently, several heuristic and meat-heuristic algorithms are developed by the researchers to achieve better performance, whereas the time complexity is high in realistic computational infrastructure that is considered as a major problem. Thus, hybrid algorithms are presented to exploit the benefits of each algorithm. In this study, HMMGA model is used for load balancing and task scheduling in the cloud. The explanations about the undertaken algorithms are given as follows.

3.1 Max–Min load balancing algorithm

In this research work, the max-min algorithm is used for maintaining a VM status table and an executing task status table inside a load balancer. The VM status table contains existing tasks in VM, VM life cycle status, total execution time of task and the last updated time. In addition, the task status table contains completion time, task execution time and the latest updated status. The following steps are to be followed when the allocation tasks arriving in the same batch are:

- Choose the task with higher execution time (Max).
- Evaluate the estimated time of the tasks in every VM with the help of VM table.
- Choose the VM with lower completion time (Min).
- Finally, allocate the task to the relevant VM. In addition, it is compulsory to update the total number of tasks and the total execution time of the VM in the VM status table [17].

3.2 Genetic algorithm for task scheduling

Genetic algorithm is a global optimization technique, which imitates the behavior of natural genetic evolution process. In addition, genetic algorithm is a probabilistic optimization technique that mainly comprises of three processes such as selection, crossover and mutation [18]. Hence, the selection process is utilized for obtaining a suitable solution and the crossover and mutation processes are utilized for maintaining the population diversity. Step by step procedure of a genetic algorithm is given below.

Step 1: Initialize the parameters such as probability of cross over P_c, probability of mutation P_m, size of the population $Pop - size$, and maximum evolution number $E - num$.
Step 2: Initialize the population in terms of chromosome to obtain better solutions in the optimization problem.
Step 3: Estimate the population on the basis of the objective function of every chromosome and then evaluate the fitness of every chromosome.
Step 4: Choose the chromosomes on the basis of fitness of every chromosome by using roulette.
Step 5: Crossover genetic operation is performed on the basis of P_c.
S**tep 6:** Variation genetic operation is performed on the basis of P_m.
Step 7: If the evolution number reaches$E - num$, consider it as a best chromosome, otherwise go to **step 3.**

In the genetic algorithm, P_m and P_c plays an essential role in identifying the optimal solution. If the P_c is high, the search procedure is fast and easy to generate a new genetic pattern structure. In addition, if the P_m is large, the genetic algorithm is equivalent to pure random search. Therefore, the mathematical equation of P_c and P_m are denoted in the Eq. (1) and (2).

$$P_c = \begin{cases} \frac{k_1(f_{max} - f')}{f_{max} - f_{avg}} & f' \geq f_{avg} \\ k_3 & f' < f_{avg} \end{cases} \quad (1)$$

$$P_m = \begin{cases} \frac{k_2(f_{max} - f)}{f_{max} - f_{avg}} & f \geq f_{avg} \\ k_4 & f < f_{avg} \end{cases} \quad (2)$$

Where, f' is indicated as fitness function of two intersected individuals, f_{max} is represented as maximum fitness in the population, f is denoted as fitness value of the individuals, f_{avg} is stated as the average fitness in the population, and $k_{1,2,3,and\ 4}$ are stated as constant values that ranges from (0 to 1). Therefore, the step by step procedure of HMMGA is detailed as follows.

3.3 Step by step procedure of HMMGA

Step 1: Initialize the cloudlets and VMs in the cloud data-center.

Step 2: Calculate the completion time for every VM and execution time for every task.
Step 3: Choose the tasks with maximum execution time and assign it to the VMs with optimal completion time. Here, the optimal completion time refers to the expected time taken to complete the task without overloading the corresponding VM.
Step 4: Initialize the gene population, which should be equal to the number of tasks.
Step 5: Estimate the load for each task in a VM using max-min scheduler, as given in Eq. (3).

$$Load = Amountofinstructionsintask/ processingcapacityofVM \quad (3)$$

Where, cloudlet length is equal to the number of instructions and the VM's processing capacity is the product of processors. Hence, the processing speed is in the form of Million Instructions Per Seconds (MIPS).
Step 6: Perform mutation and crossover between the genes for comparing their load and transfer the information about the VM selection in order to change the VM assignment if the optimal completion time is obtained.
Step 7: After every iteration, update the status of genes, tasks and VMs.
Step 8: As a result, best order of task assignment is obtained with the optimal completion time that significantly decreases the degree of imbalance.

In addition, the pseudo code of HMMGA is detailed as follows.

Pseudo code of HMMGA

- Initialize the cloudlets and VMs in the data-center;
- **For** all the submitted tasks (t_i);
- **For** all the resources (R) in Meta-Task (MT);
- Estimate the completion time **(**$completiontime_i = execution_i + R_i$)
- **End**
- **End**
- **For** every task in MT;
- Identify the tasks $t_i$with maximum execution time, and assign the resources R;
- Eliminate the tasks t_i with high completion time with respect to VM;
- Then, update the resources R from MT;
- Update the completion time of all un-mapped tasks in MT;
- Repeat, until all the tasks in MT is mapped;
- **End**
- Initialize generation $0: k := 0$; $Pk \rightarrow$ population of randomly generated individuals;
- Compute Pk: Evaluate fitness(i);
- **For** each $i \in Pk$;**do** {
- Create generation $k + 1$:

Copy: Choose $(1 - Ti) \times n$ members of$Pk$and insert into$Pk + 1$;

Crossover: Choose $(Ti \times n)$ members of Pk ; pair them; generate off-spring; insert the off-spring into$Pk + 1$;

Mutation: Choose $\mu \times n$ members of$Pk + 1$; invert a randomly-selected bit; $n \rightarrow$ number of tasks and $\mu \rightarrow$mutation constant.

- Compute $Pk + 1$: evaluate fitness (i) for every $i \in Pk$;
- Increment: $k := k + 1$; }
- **While** fitness of individual in Pk is not high;
- For Pk, get the fittest order of genes;
- Perform cloud simulation on the basis of assigned order of optimal completion time of VMs;
- Calculate the results finally;
- **End**
- **End**

4. Experimental investigation and discussion

For experimental simulation, cloud-sim Net beans (8.2 version) software is utilized with the operating system: windows 10 (64 bit) and commercial licensed processor Intel(R) Core(TM) i9-9980XE CPU @ 3.00GHz, 3000 MHz, 18 Core(s), 36 logical processor(s) installed physical memory (RAM) DDR4 1200MHz 128 GB GPU NVIDIA RTX 2080 Ti GDDR6 22GB. In this research article, HMMGA performance is compared with two existing research studies such as IBPSO-LBS [14], and TOPSIS–PSO [15] for evaluating the performance of HMMGA. In this research, HMMGA performance is analyzed in light of makespan, transmission time, resource utilization, average waiting time, and degree of imbalance. The mathematical expressions about the undertaken performance measures are listed below.

Degree of imbalance: It measures the load distribution among the VMs, where the lower value of degree of imbalance denotes that the load is effectively balanced. Mathematically, it is stated in the Eq. (4).

$$Degreeofimbalance = \frac{T_{max} - T_{min}}{T_{avg}} \quad (4)$$

Where, T_{avg} is indicated as average total execution time, T_{max} is stated as maximum execution time, and T_{min} is denoted as minimum execution time.

Makespan: It is defined as the overall completion time, which is required to execute all the tasks. The lower makespan delivers efficient and good task planning to the resources. Hence, the makespan is mathematically represented in Eq. (5).

$$Makespan = \max_{1 \le i \le m} \{Completiontime_i\} \quad (5)$$

Where, m is represented as the number of VMs.

Resource utilization: It is applied for measuring the utilization of resources. The cloud provider earns maximum profit with higher resource utilization rate. The resource utilization is defined in the Eq. (6).

$$Resourceutilization = \frac{\sum_{i=1}^{m} Completiontime_i}{Makespan \times m} \quad (6)$$

Transmission time: The time consumed to transfer the task on a specific VM is represented as transmission time that is mathematically estimated by utilizing the Eq. (7).

$$Transmissiontime = \frac{Sizeofthetask}{BandwidthoftheVMs} \quad (7)$$

4.1 Experiment-1

In the Tables 1, 2 and 3, the performance of the HMMGA is compared with max-min algorithm and TOPSIS-PSO [15] by means of resource utilization, makespan and transmission time. The parameter setting of this experiment is defined as follows; Task description: number of tasks is 10-40, length is 100-2500 and MIPS is 300-4000. Host description: utilization model is full utilization, bandwidth ranges from 500-2024 and the number of processing elements are five. Graphically, the performance analysis of the proposed and existing models with VMs 5 are indicated in the Figs. 1 and 2.

The main aim of this research work is to maximize the resource utilization and to minimize the transmission time and makespan. The resource utilization, the transmission time and makespan are individually observed with different number of tasks (10 to 40) which having 5, 6 and 10 numbers of VMs, respectively. The makespan, resource utilization and transmission time is calculated by using the Eqs. (5) - (7). Tables 1, 2 and 3 denote the estimated values of makespan, resource utilization and transmission time for 5, 6 and 10 VMs. By inspecting the Tables 1, 2 and 3, HMMGA averagely delivers 1.63 and 3.88 seconds less makespan than the Max-Min algorithm and TOPSIS-PSO [15] for five VMs. Likewise, HMMGA improved resource utilization up to 40% related to the existing models; Max-Min, and TOPSIS PSO [15]. Similarly, HMMGA delivers 1.61 and 3.84 seconds less makespan for ten VMs. Additionally, HMMGA achieves good performance in load balancing and task scheduling by means of transmission time. Graphically, the performance analysis of the proposed and existing models with VMs 6 and 10 is indicated in the Figs. 3, 4 and 5.

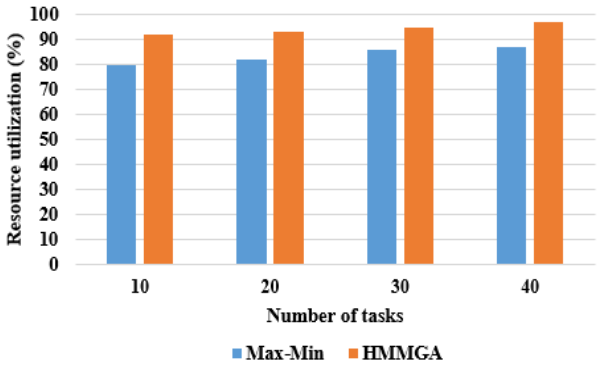

Figure. 1 Graphical evaluation of the proposed and existing models with VMs 5 in terms of resource utilization

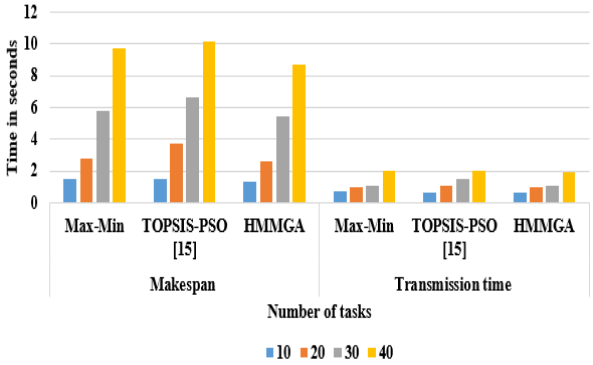

Figure. 2 Graphical evaluation of the proposed and existing models with VMs 5 in terms of makespan and transmission time

Table 1. Performance analysis of the proposed and existing models with VMs 5

Number of VMs is 5								
Number of Tasks	Resource utilization (%)		Makespan (seconds)			Transmission time (seconds)		
	Max-Min	HMMGA	Max-Min	TOPSIS-PSO [15]	HMMGA	Max-Min	TOPSIS-PSO [15]	HMMGA
10	80	92	1.48	1.490	1.37	0.7	0.664	0.65
20	82	93	2.78	3.710	2.63	1.0	1.104	1.0
30	86	95	5.79	6.680	5.47	1.12	1.524	1.10
40	87	97	9.73	10.15	8.68	2.0	2.0	1.98

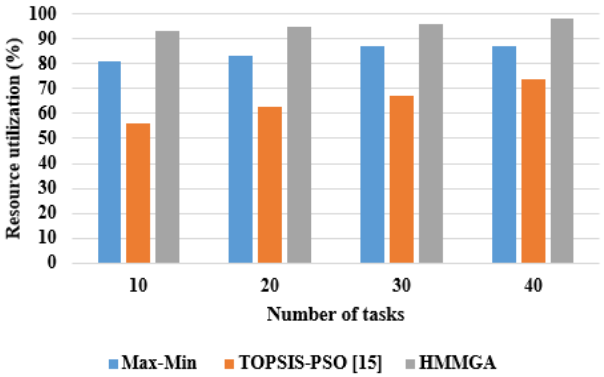

Figure. 3 Graphical evaluation of the proposed and existing models with VMs 6 in terms of resource utilization

Table 2. Performance analysis of the proposed and existing models with VMs 6

Number of VMs is 6			
Number of Tasks	Resource utilization (%)		
	Max-Min	TOPSIS-PSO [15]	HMMGA
10	81	56	93
20	83	63	95
30	87	67	96
40	87	74	98

Table 3. Performance analysis of the proposed and existing models with VMs 10

Number of Tasks	Resource utilization (%)		Makespan (seconds)			Transmission time (seconds)	
	Max-Min	HMMGA	Max-Min	TOPSIS-PSO [15]	HMMGA	Max-Min	HMMGA
10	81	93	0.61	0.77	0.53	0.8	0.71
20	83	95	1.23	1.84	0.98	1.23	1.18
30	87	96	2.38	3.46	2.08	1.58	1.47
40	87	98	4.76	5.13	3.78	2.16	2.05

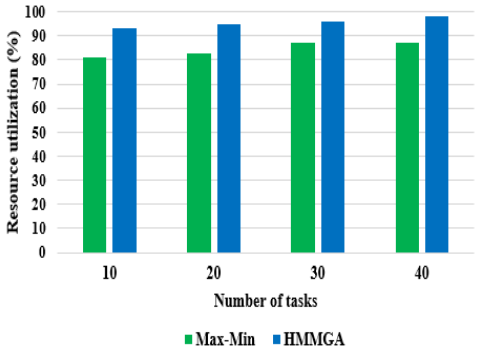

Figure.4 Graphical evaluation of the proposed and existing models with VMs 10 in terms of resource utilization

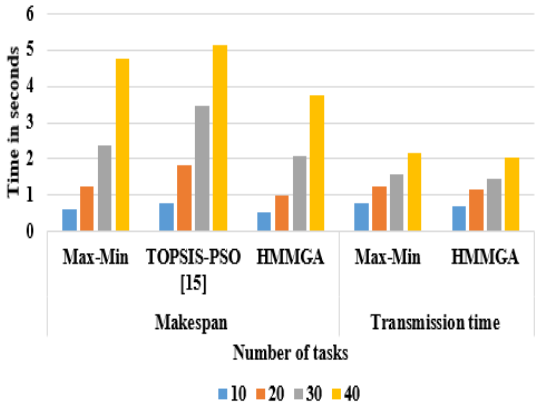

Figure.5 Graphical evaluation of the proposed and existing models with VMs 10 in terms of makespan and transmission time

Table 4. Performance analysis of the proposed and existing models with VMs 150 in light of average waiting time

Number of VMs is 150			
Number of Tasks	Average waiting time (seconds)		
	Max-Min	IBPSO-LBS [14]	HMMGA
2000	7.73	28.9	6.03
3500	21.38	44.87	18.88
5000	43.99	59.89	38.23
6500	69.56	76.32	56.39
8000	108.92	91.39	88.18

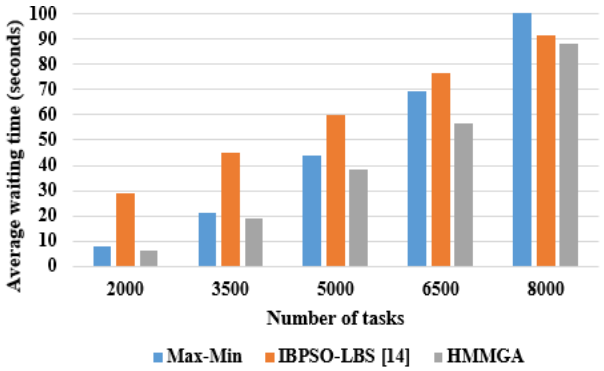

Figure.6 Graphical evaluation of the proposed and existing models with VMs 150 in terms of average waiting time

Table 5. Performance analysis of the proposed and existing models with VMs 150 in light of degree of imbalance and makespan

Number of Tasks	Degree of Imbalance			Makespan (seconds)		
	Max-Min	IBPSO-LBS [14]	HMMGA	Max-Min	IBPSO-LBS [14]	HMMGA
2000	0.72	0.26	0.67	20.00	100	18.44
3500	0.45	0.189	0.41	28.91	132	23.18
5000	0.31	0.1872	0.28	43.77	167	35.09
6500	0.26	0.185	0.23	55.66	200	49.16
8000	0.24	0.1825	0.16	63.83	237	56.79

4.2 Experiment-2

In Tables 4 and 5, the performance of HMMGA is analyzed in terms of average waiting time, makespan and degree of imbalance. Here, the number of tasks ranges from 2000 to 8000 for 150 VMs. By inspecting table 4, the proposed model; HMMGA achieved better performance than the conventional Max-Min and IBPSO-LBS [14] algorithms in light of average waiting time. The parameter setting of this experiment is given as follows; Datacenter: host bandwidth (Mbps) is 100,000 host storage (MB) is 10,000,000 the number of hosts is 3-15, policy type is time shared, and number of datacenter is one. Independent tasks: length of task is 100,000-600,000 and total number of tasks is 2000-8000. VM: number of VMs is 150, number of processing elements per VM are 5 and the policy type is space shared. By investigating Table 4 and Fig. 6 it is clear that HMMGA showed significant performance in task scheduling and load balancing by minimizing Average waiting time related to the comparative algorithms.

In Table 5, the degree of imbalance, and makespan is observed individually with a dissimilar number of tasks (2000-8000) which has 150 numbers of VMs. By observing Table 5, HMMGA significantly diminishes the makespan as the number of VM increases due to the increasing of processing power for executing the assigned tasks. From the analysis, HMMGA out-performs the existing techniques; Max-Min algorithm, and IBPSO-LBS [14] with minimum makespan time. In addition, HMMGA minimizes the degree of imbalance than the comparative techniques. It means that HMMGA delivers high load balancing level.

4.3 Discussion

As discussed in the experimental section, HMMGA is proposed in this paper for load balancing and task scheduling in the cloud environment. In most of the existing works, max-min algorithm is used only for balancing the load in a cloud environment, whereas the combination of max-min and a genetic algorithm is adaptable for multi-objectives such as minimization of transmission time, average waiting time and makespan by considering the factors like load, task length and VM resources. The effect of HMMGA is indicated in Tables 1 to 5 and the performance analysis is verified using the performance metrics like makespan, transmission time, resource utilization, average waiting time, and degree of imbalance. Under such condition, HMMGA averagely enhanced 10% to 40% of resource utilization related to the existing models; Max-Min, TOPSIS-PSO algorithms. In this research work, HMMGA achieves better performance in load balancing and task scheduling compared to Max-Min, TOPSIS-PSO and IBPSO-LBS algorithms in light of

makespan, transmission time, resource utilization, average waiting time, and degree of imbalance through multi-objective optimization of completion time through factors like load and VM resources.

5. Conclusion

In this research paper, several research gaps in the prior literature for load balancing and task scheduling based on heuristic and meta-heuristic algorithms are presented. HMMGA is implemented on the basis of hybrid algorithms; Max-Min and genetic algorithm for balancing the load in the VMs and to schedule the task with limited computational complexity. From the experimental simulation, HMMGA attained significant performance in task scheduling and load balancing in the cloud environment in light of makespan, transmission time, resource utilization, average waiting time, and degree of imbalance as compared to Max-Min, TOPSIS-PSO and IBPSO-LBS algorithms. For instance, HMMGA averagely delivers 1.63 and 3.88 seconds less makespan related to the Max-Min algorithm and TOPSIS-PSO algorithm for five VMs. In addition, HMMGA enhanced 10% to 40% of resource utilization compared to the Max-Min, TOPSIS-PSO algorithms. In another experiment, the HMMGA averagely showed 1.7 to 25.99 seconds less average waiting time related to the comparative models; Max-Min and IBPSO-LBS. In future work, a new improved optimization technique can be proposed for load balancing and task scheduling in the cloud environment.

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

The paper conceptualization, methodology, software, validation, formal analysis, investigation, resources, data curation, writing—original draft preparation, writing—review and editing, visualization, have been done by 1st author. The supervision and project administration, have been done by 2nd author.

References

[1] L. Tang, Z. Li, P. Ren, J. Pan, Z. Lu, J. Su, and Z. Meng, "Online and offline based load balance algorithm in cloud computing", *Knowledge-Based Systems*, Vol. 138, pp. 91-104, 2017.

[2] N. Leontiou, D. Dechouniotis, S. Denazis, and S. Papavassiliou, "A hierarchical control framework of load balancing and resource allocation of cloud computing services", *Computers & Electrical Engineering*, Vol. 67, pp. 235-251, 2018.

[3] D. B. LD and P. V. Krishna, "Honey bee behavior inspired load balancing of tasks in cloud computing environments", *Applied Soft Computing*, Vol. 13, No. 5, pp. 2292-2303, 2013.

[4] M. Adhikari, S. Nandy, and T. Amgoth, "Meta heuristic-based task deployment mechanism for load balancing in IaaS cloud", *Journal of Network and Computer Applications*, Vol. 128, pp. 64-77, 2019.

[5] S. L. Chen, Y. Y. Chen, and S. H. Kuo, "CLB: A novel load balancing architecture and algorithm for cloud services", *Computers & Electrical Engineering*, Vol. 58, pp. 154-160, 2017.

[6] X. Shao, M. Jibiki, Y. Teranishi, and N. Nishinaga, "An efficient load-balancing mechanism for heterogeneous range-queriable cloud storage", *Future Generation Computer Systems*, Vol. 78, pp. 920-930, 2018.

[7] S. Sotiriadis, N. Bessis, C. Amza, and R. Buyya, "Vertical and horizontal elasticity for dynamic virtual machine reconfiguration", *IEEE Transactions on Services Computing*, Vol. 99, pp. 1-1, 2016.

[8] K. R. Babu and P. Samuel, "Enhanced bee colony algorithm for efficient load balancing and scheduling in cloud", *Innovations in Bio-Inspired Computing and Applications*, Springer, Cham, pp. 67-78, 2016.

[9] V. Polepally and K. S. Chatrapati, "Dragonfly optimization and constraint measure-based load balancing in cloud computing", *Cluster Computing*, pp. 1-13, 2017.

[10] S. Mohanty, P. K. Patra, M. Ray, and S. Mohapatra, "An Approach for Load Balancing in Cloud Computing Using JAYA Algorithm", *International Journal of Information Technology and Web Engineering*, Vol. 14, No.1, pp. 27-41, 2019.

[11] D. Chaudhary, and B. Kumar, "Cost optimized Hybrid Genetic-Gravitational Search Algorithm for load scheduling in Cloud Computing", *Applied Soft Computing*, Vol. 83, pp. 105627, 2019.

[12] A. Kaur and B. Kaur, "Load balancing optimization based on hybrid Heuristic-Metaheuristic techniques in cloud environment", *Journal of King Saud University-Computer and Information Sciences*, 2019.

[13] L. Kong, J. P. B. Mapetu, and Z. Chen, "Heuristic Load Balancing Based Zero

Imbalance Mechanism in Cloud Computing", *Journal of Grid Computing*, pp .1-26, 2019.

[14] J. P. B. Mapetu, Z. Chen, and L. Kong, "Low-time complexity and low-cost binary particle swarm optimization algorithm for task scheduling and load balancing in cloud computing", *Applied Intelligence*, Vol. 49, No. 9, pp. 3308-3330,2019.

[15] N. Panwar, S. Negi, M. M. S. Rauthan, and K. S. Vaisla, "TOPSIS–PSO inspired non-preemptive tasks scheduling algorithm in cloud environment", *Cluster Computing*, Vol. 22, No. 4, pp. 1379-1396, 2019.

[16] V. Priya, C. S. Kumar, and R. Kannan, "Resource scheduling algorithm with load balancing for cloud service provisioning", *Applied Soft Computing*, Vol. 76, pp. 416-424, 2019.

[17] Y. Mao, X. Chen, and X. Li, "Max–min task scheduling algorithm for load balance in cloud computing", In: *Proc. of International Conf. on Computer Science and Information Technology, Springer*, New Delhi, pp. 457-465, 2014.

[18] Y. Zhang, and Y. Zhou, "Distributed coordination control of traffic network flow using adaptive genetic algorithm based on cloud computing", *Journal of Network and Computer Applications*, Vol. 119, pp. 110-120, 2018.