MODIFIED GENETIC ALGORITHM BASED SOLUTION FOR TASK SCHEDULING IN CLOUD COMPUTING ENVIRONMENT

Sumandeep Kaur*
Computer Science and Engineering, University Institute of Engineering and Technology,
Panjab University, Chandigarh, 160014, India
sumandeepkaur2094@gmail.com

Ravreet Kaur
Department of Computer Science and Engineering,
University Institute of Engineering and Technology, Panjab University, Chandigarh, India
ravreet@yahoo.com

Nirmal Kaur
Department of Computer Science and Engineering,
University Institute of Engineering and Technology, Panjab University, Chandigarh, India
nirmaljul19@gmail.com

Abstract - Cloud computing is a growing technology that provides on demand shared pool of resources over the internet. Sharing of resources amongst the number of cloud users makes task scheduling a challenging issue. Task scheduling issue in many cases resolved by meta-heuristic approaches. This paper proposes a solution for task scheduling in a cloud computing environment based on the meta-heuristic, Genetic Algorithm. The proposed solution i.e. Modified Genetic algorithm (MGA) uses a hybrid solution based on Genetic Algorithm along with Predict Earliest Finish Time (PEFT) scheduling on Directed Acyclic Graph (DAG). Simulated results of the Modified Genetic Algorithm are compared with basic GA and with hybrid GA with HEFT (Heterogeneous Finish Time First) scheduling algorithms. Further, comparative analysis has been performed based on makespan, average processor utilization, processing cost metrics. It is observed that MGA gives optimal results in terms of processing cost and processor utilization for the unbounded number of processors.

Keywords: Cloud computing, Task scheduling, DAG, Genetic algorithm, HEFT, PEFT

1. Introduction

Cloud is used as a metaphor for the internet and computing through internet called cloud computing [2]. Cloud computing environment provides three basic services namely Infrastructure as a Service (IaaS) that provides required operating environment, Platform as a Service (PaaS) that provides required tools and services to the application developer to run user's application, Software as a Service (SaaS) that distributes the services to the end users [13]. Cloud contains large numbers of users from different geographical area who need cloud resources according to their requirements to execute their tasks. For this Virtualization technique is used that creates virtual version of the resources by dividing them into more than one computing environment called Virtual Machines (VMs). As users are more than the available cloud resources, it emerges need of scheduling.

Task scheduling [22] allocates the tasks to the available resources or virtual machines (VMs) in a manner that reduces the total execution time of all the tasks. Further load balancing [2] distributes the load between the available VM in a way that no machine should be idle or overloaded; if it is, then tasks are transferred from overloaded VM to under-loaded VM.

This paper integrates the GA with PEFT[1] scheduling algorithm with an objective to manage the total completion time of all the tasks, increase processor utilization, and to reduce processing cost using less number of VMs. This paper formulates as: Section 2 provides the brief survey of existing meta-heuristic GA based task scheduling with their performances. Section 3 describes the methodology of the proposed algorithm. In section 4 performances are evaluated using performance evaluation metrics and at the end, conclusion and future scope are compiled under section 5.

2. Related Work

Shaminder Kaur et al. (2012) [12] have developed a task scheduling algorithm for cloud computing environment. The author used Shortest Cloudlet to Fastest Processor (SCFP) and Longest Cloudlet to Fastest Processor (LCFA) algorithm to initialize the population of GA. Their algorithm takes variable power processors and variable length tasks to represent a real-time scenario but considers single user job. Sung Ho Jang et al. [8] have implemented an algorithm in 2012 considering Quality of Service parameter documented in the SLA. Their approach uses restart and stop condition to find out a best fit solution that prevents GA to fall in Local Optimum and explore various search spaces. But the results are evaluated using static numbers of tasks. In 2013, GE Junwei et al. [3] have suggested an algorithm that takes the budget and the time constrained into consideration. It does scheduling of the dependent task by dividing the large task into the subtasks and manages the dependency between them. But they don't consider any standard dataset and also does not give priority to the tasks. Kousik Dasgupta et al. (2013) [2] have presented an approach that handles the load on processors as well as reduces the makespan, but takes equal priority tasks. In 2014, Rajveer Kaur et al. [11] have developed a scheduling algorithm that assigns a priority to the tasks according to Role Based Access Control method. Roles are defined according to Job competency, Authorization, and Responsibility. It reduces the execution time of all the tasks but considered limited number of tasks. Tingting Wang et al. [20] have presented a Job spanning time Load balancing Genetic Algorithm (JLGA) in 2014. It uses a Greedy approach to initialize the population. JLGA overcomes the local optimum problem and gives a better result than a simple genetic algorithm. Their algorithm reduces the makespan and balances the load, but they had not given any priority to the tasks. Yuming Xu et al. (2014) [22] have developed a strategy that handles the computing system of heterogeneous nature. Heterogeneous Earliest Finish Time (HEFT) technique is used to assign the priority to tasks. It covers a larger search space without incurring a high computational cost and gives a higher speedup of subtask execution. It gives a better result than standard HEFT algorithms, but considered limited number of tasks and number of processors. In 2016, Xiaodong Sheng et al. [21] have presented a template-based method for an independent task. In their paper, the template is made with a number of tasks according to the maximal size of tasks. It focuses on QoS constrained and user requirement except for cost constraints. Safwat A. Hamad et al. [5] in 2016 has implemented an algorithm to reduce the total completion time and processing cost and increasing throughput. Special crossover operation is used that produces four offspring, from them, the best of two is chosen to produce new offspring. Their algorithm gives improved results than Round-Robin and basic GA, but considered limited number of independent tasks.

3. Modified Genetic Algorithm (MGA)

This paper proposed an algorithm called MGA that integrates Genetic Algorithm (GA) with existing Predict Earliest Finish Time (PEFT) scheduling. The main highlights of the algorithm are as shown in the flowchart in Fig 1.

1) *Input the DAG workflow* to the cloud simulator.
2) Make the *Topological sorting* list of DAG.
3) Generate the *initial population* of dependent tasks. The output of this step gives the number of chromosome sets.
4) Calculate the *Fitness* of each chromosome set.
5) While (!*termination condition*) do
    5.1 *Select* the fittest parent.
    5.2 *Crossover* between the selected parent offspring using uniform crossover.
    5.3 *Mutate* the parent offspring by bit flip mutation.

   End While
6) Output the *scheduling list*.

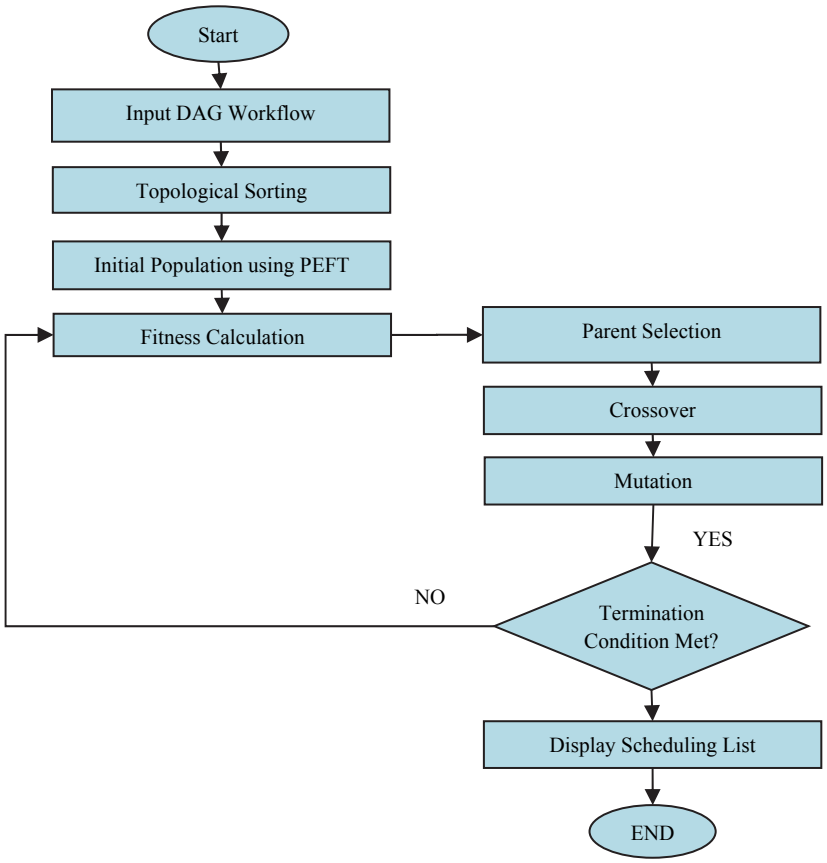


Fig 1 Modified Genetic Algorithm

The steps of MGA algorithm is given as under:-

3.1 ***Input DAG Workflow*****:** The DAG [10] is input to the MGA algorithm at compile time. This DAG file contains the tasks, instruction size, name and dependency in XML form. The DAGParser parses the DAG file into file size in bytes, task size in seconds, parent-child relationship, and number of required input tasks and defines output of the tasks.

3.2 ***Topological Sorting:*** It provides the sorted list of all nodes of the DAG by taking care of the precedence dependencies among them.

3.3 ***Initial Population*****:** It is generated using the PEFT scheduling algorithm. The output of this step gives the number of chromosome sets. In the proposed algorithm chromosome defines the scheduling list based on minimum execution time of a task (Ti) and its successor task (Ti+1) as shown below, where n is a total number of tasks.

$$\sum_{i=1}^{n}(\boldsymbol{size(Ti \,\&\, Ti+1) < size(Ti+1 \& Ti+2)}) \quad (1)$$

The task of low execution time has a high priority than the task of high execution time.

3.4 ***Fitness Calculation:*** Each solution (chromosome) as obtained in a previous step is checked against the task execution time and processing cost constraints [15]. The solution that has a lower total execution time and execution cost have a higher fitness value to survive**.** Fitness is calculated using equation 2, where ET is an execution time of a task.

$$\boldsymbol{F = min\left(max(\sum_{i=1}^{n} ET(Ti)), max(number\ of\ VMs)\right)} \quad (2)$$

3.5 ***Termination Condition:*** Termination condition tells when to stop the algorithm. In this paper, 100 iteration of GA is used as termination condition. While the termination condition is not met the following steps will be executed.

3.5.1 *Tournament Selection (TS):* It initially selects the number of parents randomly. Amongst them the solution which has highest fitness value is selected that acts as a parent offspring for the next generation. TS with tournament size equal to 5 is used to select parent offspring.

3.6 ***Crossover*****:** It is a convergence operation that generates new solution by merging some of the parent bits for the creation of child offspring. Uniform Crossover with 0.5 probability value is used here to generate new offspring. It selects the two parent offspring and the bits of parent offspring that has a lower value than probability value is selected for child offspring and other one is discarded.

3.7 ***Mutation:*** It is a divergence operation that is used to preserve the Genetic diversity from one generation to the next generation. Bit Flip Mutation will flip the bit from 0 to 1 and from 1 to 0 for the one that has a lower value than the given probability i.e. 0.015.

3.8 ***Scheduling List:*** It will display the final scheduling list with their execution time and processing cost.

4. Performance Comparison

The performance of proposed Modified Genetic Algorithm is compared and analyzed with well known scheduling algorithms, namely GA and GA with HEFT using performance calculation metrics.

I. ***Makespan:*** It defines the total time taken by all the incoming tasks from task submission to completion of a last task as given in equation 3.

$$Makespan = LastTaskFinishTime \quad (3)$$

where LastTaskFinishTime is an exit time of last task.

II. ***Processor Cost:*** Processor or VM cost represents the processing cost in term of MIPS (million instructions per second). Cost depends on the size of tasks and number of VMs as calculated in the equation 4.

$$Processor\ Cost = \sum_{i=1}^{m} \left(\frac{\mathrm{VMi_{Runtime}}}{\mathrm{BillingTime_{inSeconds}}} \right) * \mathrm{BillingPriceUnits} \quad (4)$$

Where m is a number of processors (VMs), VMi_Runtime describes ith VM's runtime, BillingTime_inSeconds and BillingPriceUnits define the predefined time on which cost will be applied, i.e. 1.0 Price Unit for 3600 Time Units.

III. ***Number of Virtual Machines:*** The number of VMs directly affects the performance of an algorithm. A lower number of VMs will increase the makespan whereas the larger number will reduce the makespan, but it will increase the processing cost. It is calculated using equation 5.

$$NumberOfVMs = \sum_{i=1}^{m} i \quad (5)$$

IV. ***Average Processor Utilization*:** Processor utilization defines how efficiently resources are utilized by the number of available tasks. It is calculated using equation 6.

$$Processor\ Utilization = \frac{VMs_{AvailableTime}}{\#VMs * makespan} * 100 \quad (6)$$

where VMsAvailableTime is VM's remaining runtime, #VMs defines the number of virtual machines.

V. ***Speedup:*** This metric show reduced parallel scheduling time related to sequential scheduling time. Speedup is calculated using equation 7 where #VM is a number of Virtual Machines.

$$Speedup = \frac{Total\ Execution\ Time\ of\ Tasks\ on\ 1VM}{Total\ Execution\ Time\ of\ Tasks\ on\ \#VM} \quad (7)$$

VI. ***Efficiency:*** It is calculated from the speedup. It is a ratio of speedup and number of used resources or processors. The efficiency value lies between 0 to 1. Higher value of efficiency represents the algorithm gives higher throughput and productivity. It is calculated using equation 8.

$$Efficiency = \frac{Speedup}{\#\,VM} \quad (8)$$

4.1 ***Simulation Results:*** The simulation results are implemented on Intel core i3 machine, with 280 GB HDD, 4GB RAM on Windows 7 OS, Eclipse with JAVA with WorkflowSim tool. The performance metrics (makespan, processor cost, number of virtual machines, average processor utilization, speedup and efficiency) of proposed MGA and two other existing GA and GA+HEFT are calculated shown in Table 1 to 6.

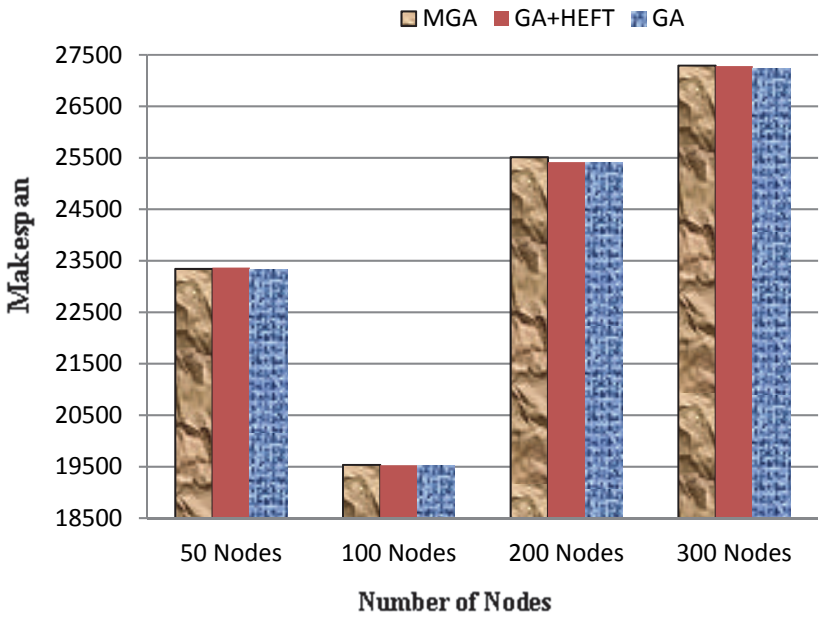
Table 1 Calculated makespan with respect to varying number of nodes

	No. of Nodes	MGA	GA+HEFT	GA
Makespan	50 Nodes	23340.83	23366.29	23332.9
	100 Nodes	19533.79	19526.28	19526.28
	200 Nodes	25512.02	25414.6	25414.6
	300 Nodes	27291.3	27280.45	27240.4

Table 2 Calculated processor cost with respect to varying number of nodes

	No. of Nodes	MGA	GA+HEFT	GA
Processor Cost	50 Nodes	38	44	62
	100 Nodes	80	126	130
	200 Nodes	173	263	272
	300 Nodes	272	310	389

Above table values of makespan and processing cost depicts that the MGA reduces the processing cost than GA and GA+HEFT strategy with slight increase in makespan for unbounded number of VMs.



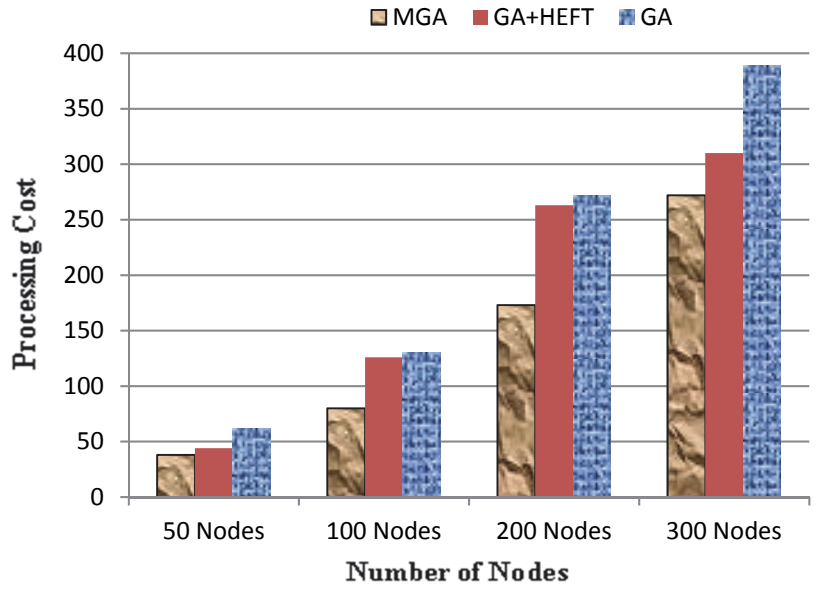


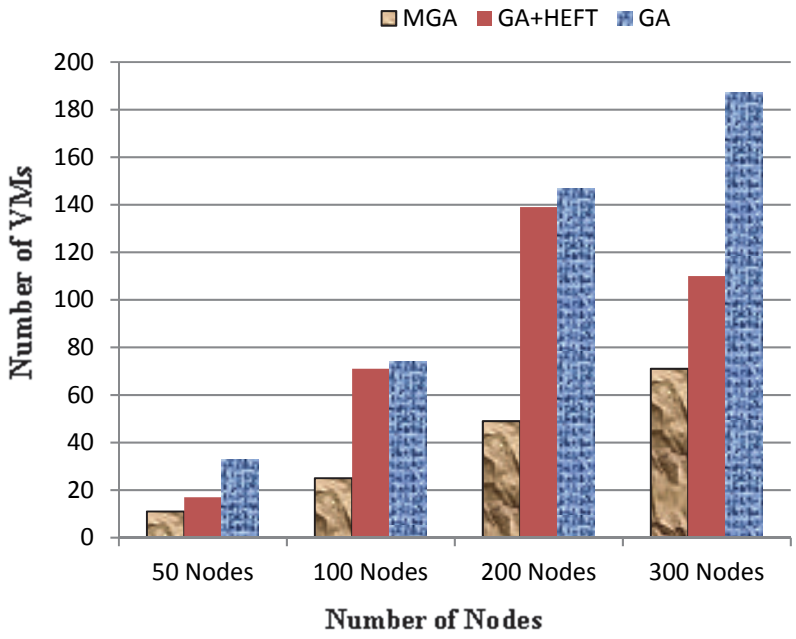
Table 3 Calculated numbers of VMs with respect to varying number of nodes

	No. of Nodes	MGA	GA+HEFT	GA
Number of VMs	50 Nodes	11	17	33
	100 Nodes	25	71	74
	200 Nodes	49	139	147
	300 Nodes	71	110	187

Table.4 Calculated average processor utilization with respect to varying number of nodes

	No. of Nodes	MGA	GA+HEFT	GA
Average Processor Utilization	50 Nodes	9.09	5.88	3.03
	100 Nodes	4	1.4	1.35
	200 Nodes	10.83	0.7	0.68
	300 Nodes	1.45	0.9	0.53

Above table values shows that the proposed algorithm uses the less number of virtual machines with larger average processor utilization rate than GA and GA+HEFT solutions. With 200 nodes, MGA uses only 49 VMs whereas GA uses 147 VMs also GA and GA+HEFT utilizes 0.6% to 0.7% of processor whereas MGA has 10.83% of processor utilization value.



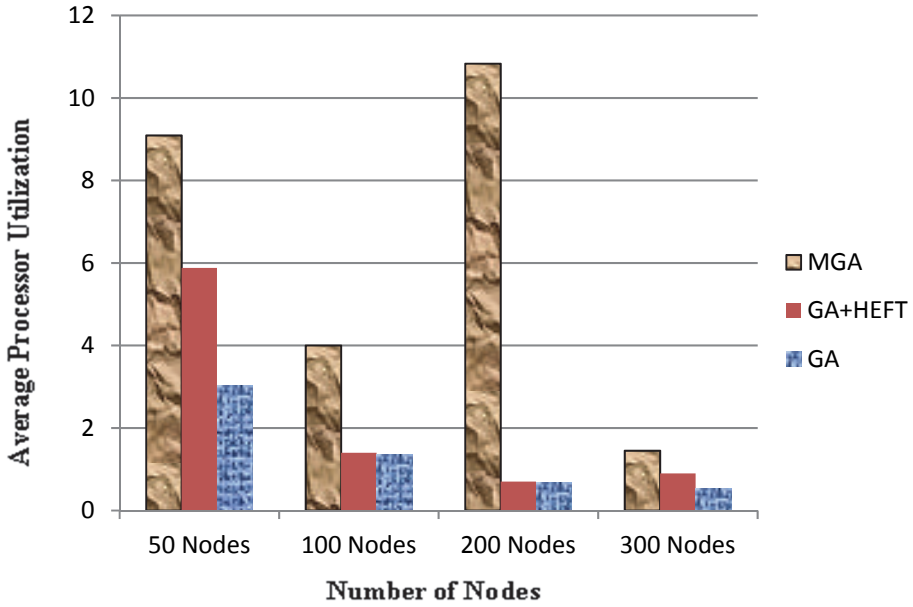


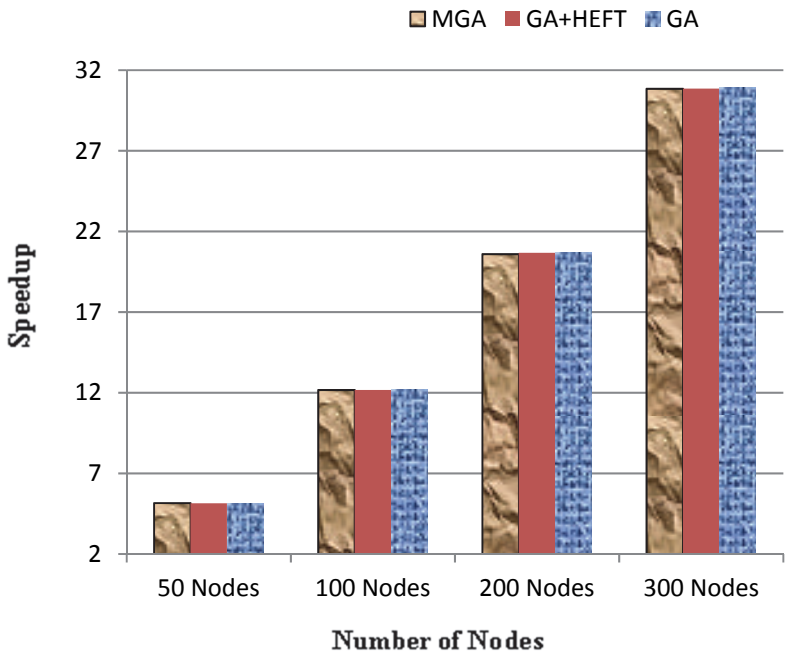
Table 5 Calculated speedup with respect to varying number of nodes

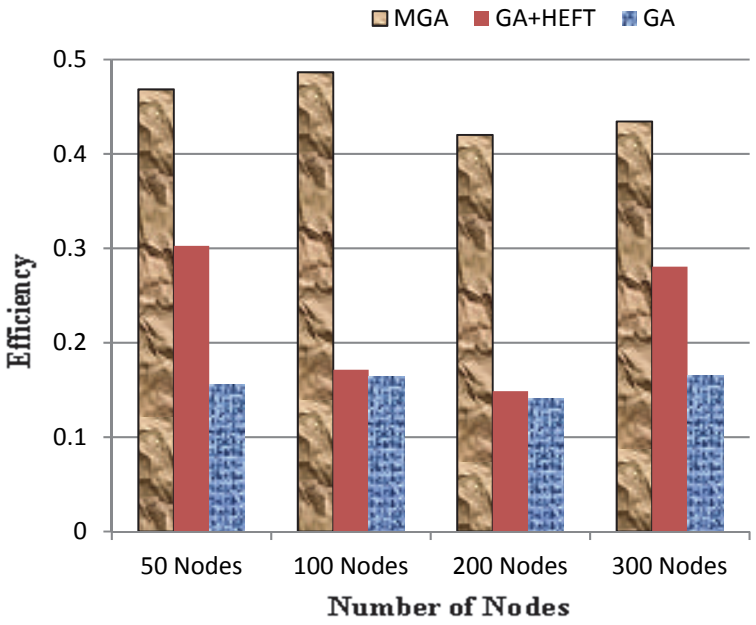
	No. of Nodes	MGA	GA+HEFT	GA
Speedup	50 Nodes	5.152	5.142	5.154
	100 Nodes	12.163	12.168	12.168
	200 Nodes	20.589	20.668	20.668
	300 Nodes	30.844	30.856	30.902

Table 6 Calculated efficiency of the algorithms with respect to varying number of nodes

	No. of Nodes	MGA	GA+HEFT	GA
Efficiency	50 Nodes	0.4684	0.3025	0.1561
	100 Nodes	0.4865	0.1713	0.1644
	200 Nodes	0.4202	0.1486	0.1406
	300 Nodes	0.4344	0.2805	0.1652

Speedup reflects the total completion time of the algorithm. The minimum total completion time will increase the speedup. As shown in the above table, in case of 50 nodes speedup of GA is 5.154 whereas speedup of MGA is 5.152, which is slightly lower but has higher efficiency.





5. Conclusion and Future Scope

This paper proposes a hybrid approach based solution for task scheduling. Proposed solution, Modified Genetic Algorithm (MGA) assigns a priority to the tasks based on Predict Earliest Finish Time and optimizes the scheduling results using the meta – heuristic, Genetic Algorithm. Experimental results have been recorded based on metrics like makespan, processing cost, speedup, efficiency, number of used processors and processor utilization. Further, comparison with basic GA and GA with HEFT clearly shows that proposed MGA gives better results. MGA reduces the processing cost, number of virtual machines and increases the average processor utilization. But this improvement is achieved with a slight increase in makespan.

For future work, an optimized algorithm can be implemented that further reduces the total completion time. Dynamic nature of virtual machines can be addressed and accordingly adaptive solutions can be proposed. This algorithm used only computation cost. In future, this algorithm can be extended by integrating computation as well as communication cost for scheduling. To represent the real-time environment, dynamic tasks can be used with varying incoming time.

References

[1] Arabnejad, H., & Barbosa, J. G. (2014). List scheduling algorithm for heterogeneous systems by an optimistic cost table. IEEE Transactions on Parallel and Distributed Systems, 25(3), 682-694.
[2] Dasgupta, K., Mandal, B., Dutta, P., Mandal, J. K., & Dam, S. (2013). A genetic algorithm (ga) based load balancing strategy for cloud computing. Procedia Technology, 10, 340-347.
[3] Ge, J. W., & Yuan, Y. S. (2013). Research of cloud computing task scheduling algorithm based on improved genetic algorithm. In Applied Mechanics and Materials (Vol. 347, pp. 2426-2429). Trans Tech Publications.
[4] Goyal, P., & Kaur, N. (2015). An optimizing technique based on genetic algorithm for power management in heterogeneous multi-tier web clusters. International Journal of Computer Applications, 115(17).
[5] Hamad, S. A., & Omara, F. A. (2016). Genetic-Based Task Scheduling Algorithm in Cloud Computing Environment. International Journal of Advanced computer Science and Applications, 7(4), 550-556.
[6] https://download.pegasus.isi.edu/misc/SyntheticWorkflows.tar.gz
[7] Hu, J., Gu, J., Sun, G., & Zhao, T. (2010, December). A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on (pp. 89-96). IEEE.
[8] Jang, S. H., Kim, T. Y., Kim, J. K., & Lee, J. S. (2012). The study of genetic algorithm-based task scheduling for cloud computing. International Journal of Control and Automation, 5(4), 157-162.
[9] Kashyap, D., & Viradiya, J. (2014). A survey of various load balancing algorithms in cloud computing. International Journal of Scientific and Technology Research, 3(11), 115-19.
[10] Kaur, G. (2016). A DAG based Task Scheduling Algorithms for Multiprocessor System-A Survey. International Journal of Grid and Distributed Computing, 9(9), 103-114.
[11] Kaur, R., & Kinger, S. (2014). Enhanced genetic algorithm based task scheduling in cloud computing. International Journal of Computer Applications, 101(14).
[12] Kaur, S., & Verma, A. (2012). An efficient approach to genetic algorithm for task scheduling in cloud computing environment. International Journal of Information Technology and Computer Science (IJITCS), 4(10), 74.
[13] Keshanchi, B., Souri, A., & Navimipour, N. J. (2017). An improved genetic algorithm for task scheduling in the cloud environments using the priority queues: formal verification, simulation, and statistical testing. Journal of Systems and Software, 124, 1-21.
[14] Lu, X., Zhou, J., & Liu, D. (2012). A method of cloud resource load balancing scheduling based on improved adaptive genetic algorithm. JOURNAL OF INFORMATION &COMPUTATIONAL SCIENCE, 9(16), 4801-4809.
[15] Maciej, M. Cost-and deadline-constrained provisioning for scientific workflow ensembles in IaaS clouds. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. IEEE Computer Society Press.
[16] Maduravoyal, C. LIST BASED SCHEDULING ALGORITHM FOR HETEROGENEOUS SYSYTEM. International Journal of Applied Environmental Sciences (IJAES), 10(1), 2015.
[17] Ravichandran, S., & Naganathan, E. R. (2013). Dynamic scheduling of data using genetic algorithm in cloud computing. International Journal of Computing Algorithm, 2(01), 127-133.
[18] Sheng, X., & Li, Q. (2016, August). Template-Based Genetic Algorithm for QoS-Aware Task Scheduling in Cloud Computing. In Advanced Cloud and Big Data (CBD), 2016 International Conference on (pp. 25-30). IEEE.
[19] Vanitha, M., & Marikkannu, P. (2017). Effective resource utilization in cloud environment through a dynamic well-organized load balancing algorithm for virtual machines. Computers & Electrical Engineering, 57, 199-208.
[20] Wang, T., Liu, Z., Chen, Y., Xu, Y., & Dai, X. (2014, August). Load balancing task scheduling based on genetic algorithm in cloud computing. In Dependable, Autonomic and Secure Computing (DASC), 2014 IEEE 12th International Conference on (pp. 146-152). IEEE.
[21] Xiao, Z., Song, W., & Chen, Q. (2013). Dynamic resource allocation using virtual machines for cloud computing environment. IEEE transactions on parallel and distributed systems, 24(6), 1107-1117.
[22] Xu, Y., Li, K., Hu, J., & Li, K. (2014). A genetic algorithm for task scheduling on heterogeneous computing systems using multiple priority queues. Information Sciences, 270, 255-287.