International Journal of Recent Technology and Engineering (IJRTE)
ISSN: 2277-3878 (Online), Volume-8 Issue-4, November 2019

Independent Task Scheduling in Heterogeneous System

Sarthak Srivastava,Santanu Kumar Misra

Abstract*:*Recently, the rapid development in processing speeds, fast storage devices and better network connectivity, hasaccelerated the popularization of cloud computing. Cloud computing is an on-demand-servicewhich provides users with high end servers,storage and processing capabilities where the user need not be concerned with its infrastructure.Although, there are abundant resources in the cloud infrastructure, for the efficient working and execution of tasks, task scheduling plays a crucial role. Task scheduling results in better performance (throughput) of the system along with better resource utilization which ultimately results inreduced energy consumption. At any given time, a processor should never be in idle state, as it still consumes some amount of energy. In this paper, the use of Quantum Genetic Algorithm has led to the reduction in energy consumption. The objective is to find a scheduling sequencewhich can be implemented ina cloud computing environment. Along with minimizing energy consumption, the algorithm helps reduce makespan time of a processor as well.The results show a decrease in energy consumption by 10-15% under different test scenarios involving a variable number of tasks, processors, and the number of iterations (generations) for which the algorithm was run. The algorithm converges to the desired result within 10-15 iterations, as can be seen from the results published in this paper.

Keywords*:*Energy Consumption, Makespan, Throughput, Quantum Genetic Algorithm.

I. INTRODUCTION

Cloud computing is the delivery of computing services, main among which are servers, storage and processing, over the Internet, offering flexible resources on pay as you use model. The services offered by Cloud Computing areSoftware as a Service (SaaS),Platform as a Service (PaaS), Infrastructure as a Service (IaaS)[1]. For optimal performance and efficient utilization of the resources in a cloud system, an efficient scheduling algorithm is of utmost importance. The performance in terms of throughput can be measured by the number of tasks that can be executed in a given time frame. Scheduling of tasks plays a very crucial role in it.

Manuscript published on November 30, 2019.
* Correspondence Author
Sarthak Srivastava*, pursuing B.Tech, Computer Science and Engineering, Sikkim Manipal Institute of Technology, Sikkim India.
Mr. Santanu Kumar Misra, department of Computer Science & Engineering, at Sikkim Manipal Institute of Technology, Sikkim. India.

Broadly, there are 3 classes of scheduling algorithms used in cloud computing [2] – namely Resource Scheduling, Task Scheduling and Workflow Management.
The above classes involve use of various techniques such as Greedy Scheduling, Genetic Scheduling, Round Robin, Priority based, Particle Swarm Optimization and Min Min algorithm. Although, there are virtually infinite resources in a cloud computing system, managing their usage is still very crucial. Within a virtual machine, to optimize the workflow (i.e. performing tasks in best possible sequence), we need scheduling so that an optimum throughput is achieved. The challenges that are faced in task scheduling in a cloud computing system can be categorized into 5 categories:

- **Varying Workloads**: The type of workload is unforeseeable. If we know that a particular type of job is to be done, then suitable measures can be taken such that maximum efficiency is achieved.
- **Resource Scheduling**: There is an abundance of resources in a cloud computing system. It can be utilized by any person/industry whose work involves usage of computers. Since, there are a large number of users of the cloud, the management of resources has to be the utmost priority. There should be no wastage of resources and no delay in the allocation process. Users should have access to these resources readily.
- **Distributed Systems**: There are many warehouses that house hundreds of processing units. So, there often arises a scenario in which data is stored in different systems so, it has to be transferred to a single location for the processing to be done. There should be no delay in the transfer phase.
- **Heterogeneous Systems**: The processing units in a cloud computing system do not have the same specifications.
- **Load Balancing**: Optimal utilization of cloud resources requiresproper load distributions among different virtual machines so that they do not suffer from underutilization and overutilization scenarios.

Genetic Algorithm also has several drawbacks:

- It is difficult to get the correct genetic representation of the problem along witha proper objective function and the different operations.
- The selection of the initial population is very crucial.
- The whole process is very computationally expensive.

In this paper, we are tackling the issue of task scheduling in a heterogeneous system by implementing a technique known as Quantum Genetic Algorithm. A Genetic Algorithm is a type of Evolutionary Algorithm, which works on the principle of “Survival of the fittest”.

A set of phenotypes (i.e. candidate solutions) are worked upon towards a better solution. Each phenotype consists of chromosomes-i.e. an encoding representing the properties of the candidate solution, which is then mutated repeatedly. In the beginning, a random population set is generated whichis then evaluated iteratively on the basis of its fitness value. The chromosomes having a higher fitness value are chosen and a new population set is derived from them. Genetic Algorithms have found their way in Automotive Design, Robotics, Computer-Aided Molecular Design etcetera.

Quantum Computing:

The computers we use today are known as classical computers. They have memory which is represented by binary digits (i.e. 0 and 1).

These bits are used to represent the information. The bits are passed to complex logic gates as inputs and operations ranging from addition to video rendering are done. A single bit can represent either 0 or 1 at a time.

In a quantum computer on the other hand, instead of bits we have Quantum Bits or simply Qubits. The quantum objects used today are photons, electrons or nucleus. Each of these particles have a property called spin. A bit can either be 1 (Spin Up) or 0 (Spin Down), whereas, a qubit can be in both Spin Up or Spin Down at the same time. This special property of a qubit is called Quantum Superposition, a property fundamental to the Quantum Mechanics and Quantum Computing where the values α^2 and β^2 represent the probability of seeing a bit to be 0 or 1 respectively when the value of the qubit is measured. As such, equation $\alpha^2 + \beta^2 = 1$ must be satisfied [3]. The spins can be measured by using quantum logical gates. Hadamard Gate is a quantum logical gate used to map a qubit's state to either 0 or 1.

For example, 2 bits (or N) can represent 4 (2^N) different pieces of information but will represent only one of them, but 2 qubits will represent all 4 pieces of information at the same time. We use the superposition property of a quantum particle inQuantum Genetic Algorithm. A classical register with N bits can store one value out of 2^N, whereas, a quantum register is capable of storing all 2^N values. Similarly,operations applied to a quantum register generates a superposition of all possible results. This is what is meant by the term "quantum parallelism." [2].

II. MOTIVATION

The motivation for this research paper comes from the fact that cloud computing is the future. There have been advancements in technology but cloud is not popular because of that, but rather, because people are now more aware of these technologies. Cloud enables people to access state-of-the-art machines in a pay as you use model. It offers a robust and secure environment for the customers. Cloud is allowing start-ups to save money on infrastructure and computing devices.

In order for cloud computing services to be usable, advancements have to be done rapidly so that the end user gets a seamless and hassle-free experience. Task scheduling is a big cog in the cloud computing system. The better the scheduling is, better the experience will be.

The paper is organized as follows; Section IIand III are focused on the motivation behind this research and the author's contribution. Section IV describes the previous work/research that has been done in the field of task scheduling in a cloud computing environment. Section V mentions the shortcomings of the related work done and how the proposed methodology overcomes them. Section VI defines the various terminologies and notations used in this paper. Section VII describes the problem definition of task scheduling in a heterogeneous system. The proposed approach is explained step by step in section VIII. The results obtained during experimentation are published in section IX followed by conclusion and future scope in section X.

III. AUTHORS CONTRIBUTION

In this paper, we tackle the task scheduling issue in a Heterogeneous Computing System for independent tasks. The proposed algorithm uses Quantum Genetic Algorithm (QGA) for the representation of chromosomes and GA operations. QGA has been used over GA as it uses quantum memory registers which allow us to store all the superpositions of N bits (i.e. N^2) and an operation applied to the quantum register produces a superposition of all possible results. A lot of work in recent times involves the use of PSO and ACO. Proper implementation of Quantum Algorithms in task scheduling will produce excellent results, even for dependent task scheduling.

IV. PREVIOUS WORK DONE

- In [4], the authors have compared the performance of 3 techniques, namely, ACO (Ant Colony Optimisation), GA (Genetic Algorithm) and the proposed GAACO technique which delivers a better makespan than the other two given the system is heterogeneous. The ACO algorithm has a fast rate of convergence and also has a good feedback mechanism so it finds the optimum path. GA further refines the solution given by ACO algorithm.
- In [5], the authors proposed a new dynamic task scheduling algorithm by using improved genetic algorithm (IGA). IGA reduces the throughput greatly,even when the number of tasks is large.
- In [6], the authors propose an algorithm to optimize the bi-objective makespan and cost using Integer PSO, a variant of the continuous PSO algorithm.
- In [7], a dynamic rule for modifying the heuristic function used in ACO had been suggested in to carry out the process of task assignment for a Resource constrained project scheduling problem (RCPSP). A modified ACO approach named Dynamic and Delay Ant Colony System (DDACS) is used which provides an efficient technique to come to an optimal solution.
- In [8], the authors compared the ACO algorithm and the Cuckoo algorithm individually against the proposed hybrid algorithm consisting of the best traits of both the algorithm. The proposed algorithm fails to converge to global optima since it only searches in local domain.
- In [9], priorities are assigned to the tasks using Heterogeneous Earliest Finish Time (HEFT). The solution can be further optimised by incorporating more evolutionary algorithms, as suggested by the authors.

- In [10], to tackle the multiprocessor task-graph scheduling problem the authors proposed a Max–Min Ant System (MMAS) which providesbest sequence of task which is assigned to the processor based on Early Start Time.
- In order to decrease the total turnaround time of the execution and the waiting time in ready queue, the author Pradhan et al. [11]proposed Modified Round Robin Algorithm.
- In [12], V.M.A. Xavier and S. Annadurai proposed Chaotic Social Spyder Algorithm (CSSA) for load balance aware task scheduling. The makespan was reduced by ~15% when compared to GA, PSO, ABC and HFKCS, thereby reducing the overall throughput of the system.

CSSA is well suited for global convergence as compared to GA and PSO which often converge locally. The paper, however, failed to make the proposed algorithm suitable for independent tasks.

- In [13], J.G. de Matos et al. proposed Genetic and Static Algorithm for Scheduling Tasks in Cloud Computing (GSASTCC)which improves processing time of tasks and also reduce the number of virtual machines via Maximum Resource algorithm. The proposed technique works fine for small number of tasks, but is not suitable for a large task set.

V. ADVANTAGES OF PROPOSED WORK OVER OTHER RELATED WORK

- Most of the work that has been done in task scheduling usually involves the use of classical genetic algorithm, only a small amount of work has been done to explore the potential of Quantum Algorithms. The speed-up offered by Quantum Algorithms is exponentially more when compared to the classical algorithms.
- Most of the previous work has not taken into account the sporadic tasks, which means the independent tasks which comes irregularly but inthe proposed methodology, we have focused on the execution of sporadic tasks.
- Also, in our proposed methodology we have considered heterogeneity in processing units with varying processing frequencies and voltages.In this heterogeneous environment, we have validated our proposed methodology by showing its behaviour in different scenarios.

VI. NOTATIONS AND TERMINOLOGIES

1. **Quantum Genetic Algorithm:** Refer the introduction section of the paper.
2. **Chromosome:** It is the encoded representation of the solution set of a problem.
3. **Population:** It is a set of multiple chromosomes.
4. **Genome:** It is the length of a chromosome.
5. **Fitness Function:** A fitness function is a way of evaluating a given set of chromosomes and determine how far the current solution is from the optimal solution.
6. **Rotation:** Technique used to generate a new population using the chromosomes in the mating pool from the last generation.
7. **Makespan:** The time at which the last job leaves from a processing unit.

VII. PROBLEM DEFINITION

The aim of this paper is to find a satisfactory scheduling algorithm which minimizes the energy consumption of a cloud computing system by scheduling tasks for execution in an efficient manner given that there are independent tasks in the queue using Quantum Genetic Algorithm. Independent tasks/programs are tasks/programs that can start without the need for some other program to invoke them. The finish time is calculated after the task allotment is done to each processor. After that, the energy is calculated from this finish time. The maximum energy consumption is taken from each individual in a population and the individual with the least energy consumption is chosen for the mating pool, i.e., to make the next set of chromosomes.

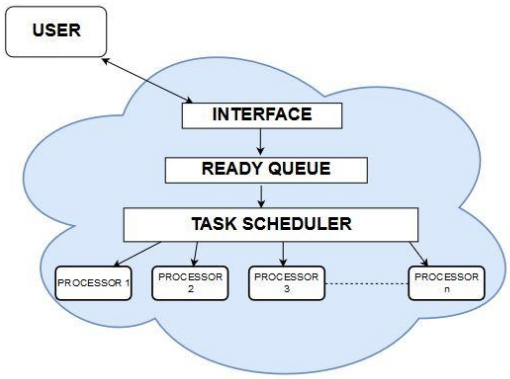

Fig. 1: System's View

VIII. PROPOSED APPROACH

The main steps of a Quantum Genetic algorithm are [14]:

Algorithm: Quantum Genetic Algorithm

1. Begin
2. Initialize a quantum population Q(0) randomly
3. Measure each chromosome to obtain a Classical Population P(0)
4. Evaluate P(0)
5. while (termination condition not met)
   - 5.1. Begin
   - 5.2. t ←t + 1
   - 5.3. Update Q(t) applying Q-gates: Q(t+1) = U(t).Q(t)
   - 5.4. Obtain classical population P(t)
   - 5.5. Evaluate P(t)
6. End

A: *Initializing Population*

A.1: *Initializing Quantum Population*

Population Q(0) of size *m* is generated in which a chromosome comprises of $\alpha^2$and β^2 values where $\alpha^2$and β^2 are the probabilities of a qubit being 0 or 1.

$$\begin{pmatrix} \alpha_1, & \alpha_2, & \alpha_3, & \alpha_4, & \alpha_5, & \alpha_6 \dots & \alpha_n \\ \beta_1, & \beta_2, & \beta_3, & \beta_4, & \beta_5, & \beta_6 \dots & \beta_n \end{pmatrix}_1$$

$$\begin{pmatrix} \alpha_1, & \alpha_2, & \alpha_3, & \alpha_4, & \alpha_5, & \alpha_6 \dots & \alpha_n \\ \beta_1, & \beta_2, & \beta_3, & \beta_4, & \beta_5, & \beta_6 \dots & \beta_n \end{pmatrix}_2$$

.
.
.

$$\begin{pmatrix} \alpha_1, \alpha_2, \alpha_3, \alpha_4, \alpha_5, \alpha_6 \dots \alpha_n \\ \beta_1, \beta_2, \beta_3, \beta_4, \beta_5, \beta_6 \dots \beta_n \end{pmatrix}_m$$

A.2: ***Generating Classical Population (Measuring)***

Next, we generate a classical population P(0) by comparing each α^2 value against a threshold value θ. Ifα^2 is more than θ, then the genome in P(0) is equal to 1, else 0. After measurement, the population obtained assumes the form:

$$(x_1, x_2, x_3, x_4, x_5 \dots x_n)_i$$

(Where 0<=i<=Population Size)

B: ***Fitness Evaluation***

The fitness function is the main part of a GA. It determines whether a given chromosome is useful or not. The fitness function used is divided into different phases.

B.1: ***Decoding the chromosome***

The given solution is decoded to the appropriate solution set. For that, the number of bits, *b*, required to represent the number of processors is calculated. Thereafter, *b* bits from the chromosome are taken at a time and converted to decimal number. The absolute value of the difference between the decimal value obtained and the number of processors is taken if the decimal number obtained is greater than the number of processors. For example, the decoded population obtained looks like:

(3, 5, 1, 4, 4, 2, 1)

When the number of processors = *5*

B.2: ***Processor Assignment***

The population obtained is then assigned to the processors according to the decimal number.

The task number t is sent to the $(d-1)^{th}$ processor if the t^{th} task is *d.*

For example, for the decoded population *(3, 5, 1, 4, 4, 2, 1),* the population assignment will be as follows:

PROCESSOR NUMBER	TASK NUMBER
0	3, 7
1	6
2	1
3	4, 5
4	2

B.3: ***Calculating Finish Time***

Initially, all the processors start at time t = 0.

$$\text{Finish time} = \sum_{i=0}^{n} (t_{ex})_i \qquad (1)$$

where, n is the total number of tasks in a processor;

$(t_{ex})_i$ is the execution time of task number i;

The execution time of the processes is imported from a file which has 10,000 values randomly generated between 5 and 20.

B.4: ***Calculating Energy Consumed***

The energy consumption of each processor is calculated by the formula [15]:

$$E = \alpha v^2 F t_{finish} \qquad (2)$$

Where,

α is the capacitance load;

v is the supply voltage;

F is the average of the maximum and minimum frequency at the which the processor operates;

t_{finish} is the time at which a processor finishes executing all the processes assigned to it.

B.5: ***Selection***

There are many selection techniques which are used to select the fittest candidate in the population. The roulette wheel is one of the most commonly used techniques in scheduling problems. However, in this paper, the selection method used can be called as the min-max method. The energy consumed by each processor is calculated and then the processor with the maximum energy consumption is selected and compared with the maximum energy consumptions of other chromosomes. The chromosome which gives the least maximum energy consumption is selected for the gene pool.

C: ***Rotation***

This step of the QGA is performed on the encoded chromosomes. The best chromosome *bestChrom*is selected from the current generation. All the remaining chromosomes are then compared with *bestChrom*. If the chromosome[i, j] = 0 and the bestChrom[j] = 1, then all the qubits of that chromosome in the quantum population are rotated clockwise by an angle of delta_theta. If chromosome[i, j] = 1 and bestChrom[j] = 0, then all the qubits of that chromosome in the quantum population are rotated counter-clockwise by an angle of delta_theta.

D: ***Mutation***

The mutation operation is a small change at a random position in the chromosome. The main motive of this operation is to counter the local minima problem.

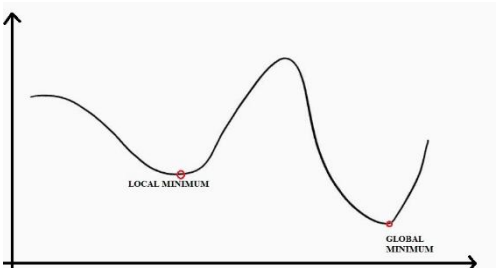

Fig 2 :Local and Global Minimum

As visible in the above figure number 2, the optimization algorithms can often fail to converge to the global minimum. This small random change is added to introduce new properties to the chromosomes (individuals). This is done to improve the existing fitness but it can also result in the degradation of the chromosome, i.e., negative impact on its fitness.

The phases mentioned above are repeated till a desired fitness level is attained.

IX. EXPERIMENTATION AND RESULTS

Inputs to the above algorithm were experimented with and several observations were made.

Inputs:

Number of Tasks: 200, 400, 600, 800 and 1000.

Number of Processors: 20, 60, 100, 140, 180, 250, 350, 400 and 500.

Number of Generations: 10 and 50.

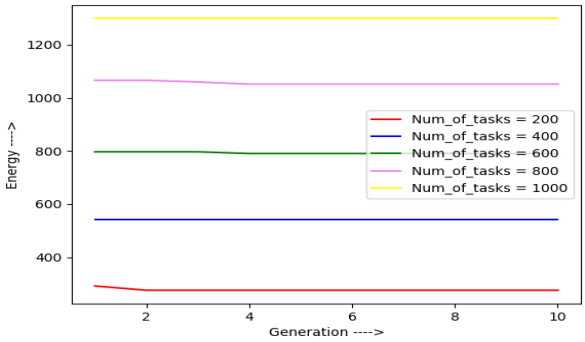

Fig 3

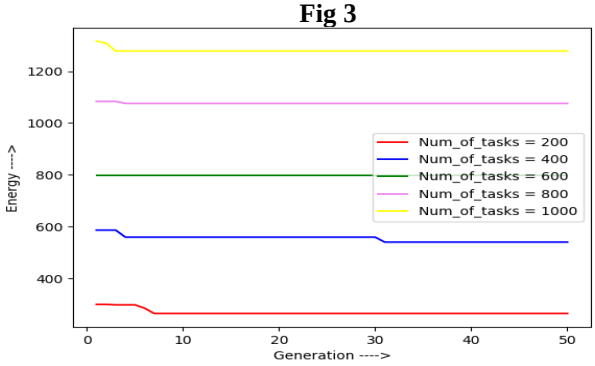

Fig4

In the above figures 3 and 4, the number of processors is 20 and the number of generations is 10 and 50 respectively. The makespan times obtained for number of tasks 200, 400, 600, 800 and 1000 was 173, 340, 496, 660 and 815 and 166, 339, 500, 675 and 802 respectively for number of generations as 10 and 50.

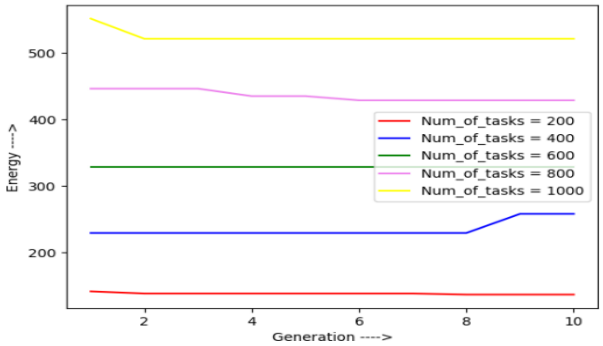

Fig 5

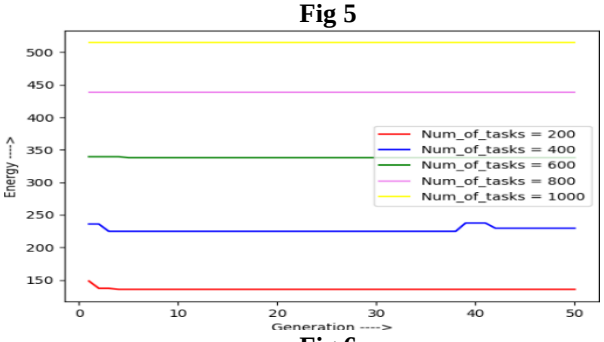

Fig 6

In the above figures 5 and 6, the number of processors is 60 and the number of generations is 10 and 50 respectively. The makespan times obtained for number of tasks as 200, 400, 600, 800 and 1000 was 86, 144, 206, 269 and 327 and 85, 141, 212, 275, 323 respectively for number of generations as 10 and 50.

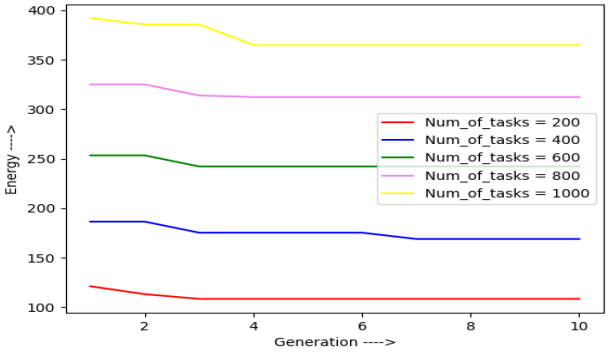

Fig7

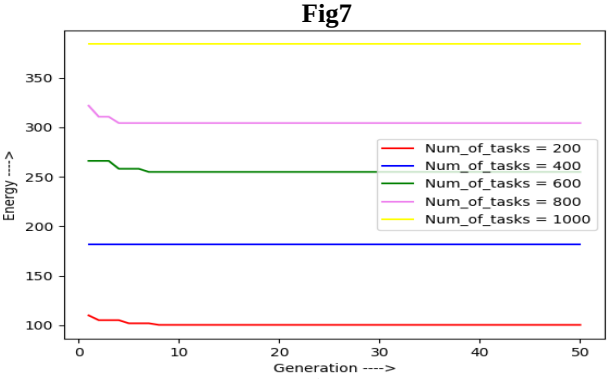

Fig 8

In the above figures 7 and 8, the number of processors is 100 and the number of generations is 10 and 50 respectively. The makespan times obtained for number of tasks as 200, 400, 600, 800 and 1000 was 68, 106, 152, 196 and 229 and 63, 114, 160, 191 and 241 respectively for number of generations as 10 and 50.

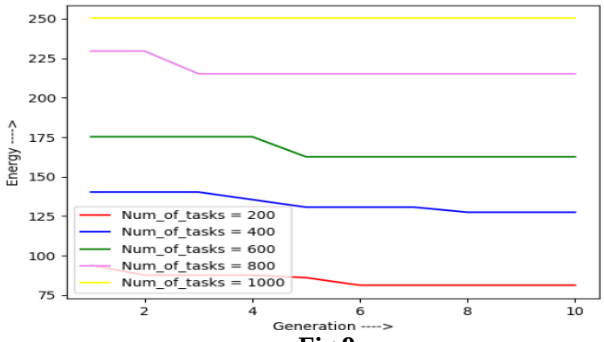

Fig 9

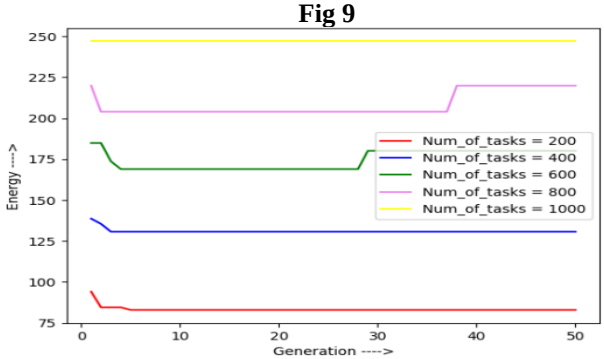

Fig 10

In the above figures 9 and 10, the number of processors is 140 and the number of generations is 10 and 50 respectively. The makespan times obtained for number of tasks as 200, 400, 600, 800 and 1000 was 51, 80, 102, 135 and 157 and 52, 82, 106, 128 and 155 respectively for number of generations as 10 and 50.

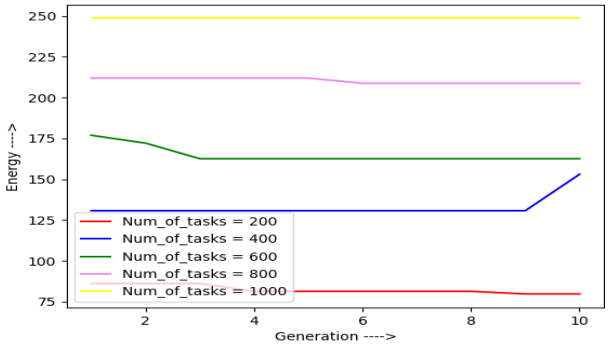

Fig 11

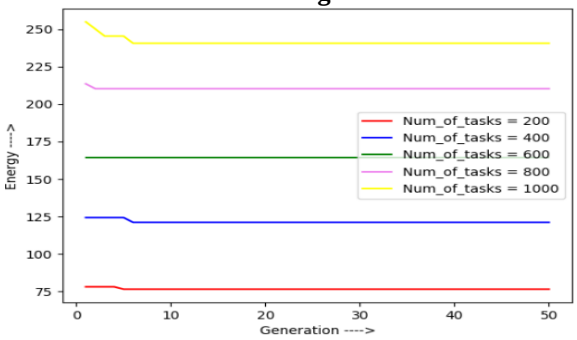

Fig12

In the above figures 11 and 12, the number of processors is 180 and the number of generations is 10 and 50 respectively. The makespan times obtained for number of tasks as 200, 400, 600, 800 and 1000 was 50, 82, 102, 131 and 156 and 48, 76, 103, 132 and 151 respectively for number of generations as 10 and 50.

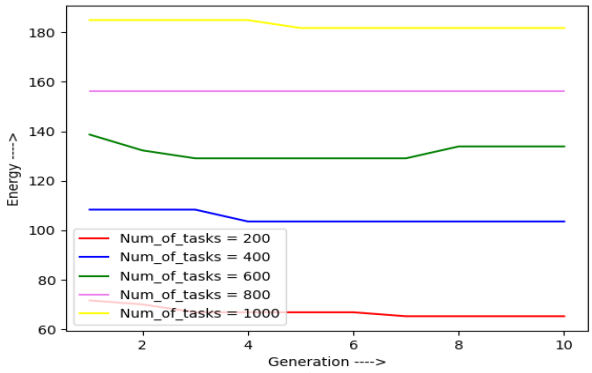

Fig 13

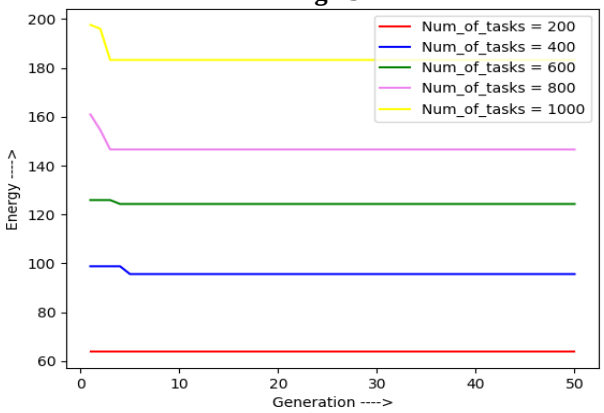

Fig 14

In the above figures 13 and 14, the number of processors is 250 and the number of generations is 10 and 50 respectively. The makespan times obtained for number of tasks as 200, 400, 600, 800 and 1000 was 41, 65, 81, 98 and 114 and 40, 60, 78, 92 and 115 respectively for number of generations as 10 and 50.

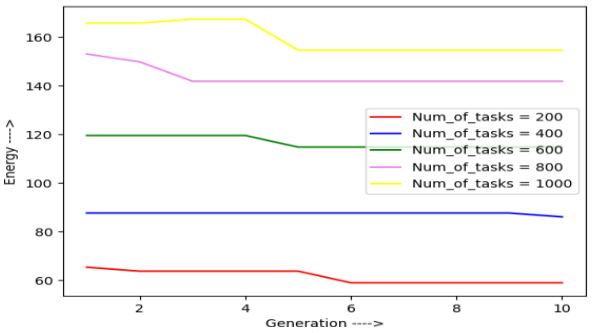

Fig 15

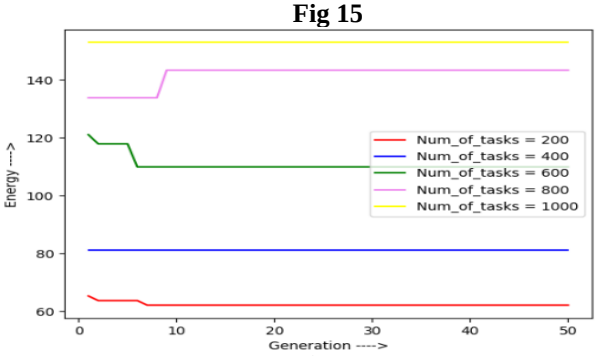

Fig 16

In the above figures 15 and 16, the number of processors is 350 and the number of generations is 10 and 50 respectively. The makespan times obtained for number of tasks as 200, 400, 600, 800 and 1000 was 37, 54, 72, 89 and 97 and 39, 51, 69, 84 and 96 respectively for number of generations as 10 and 50.

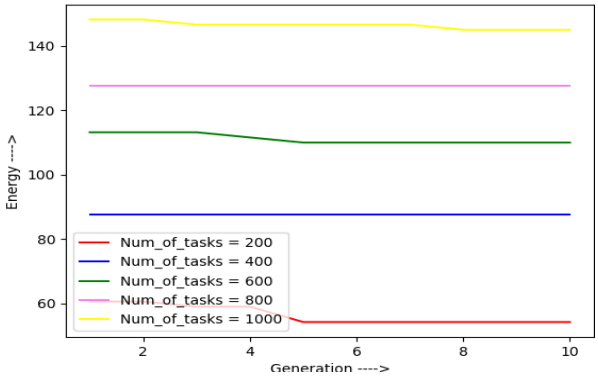

Fig 17

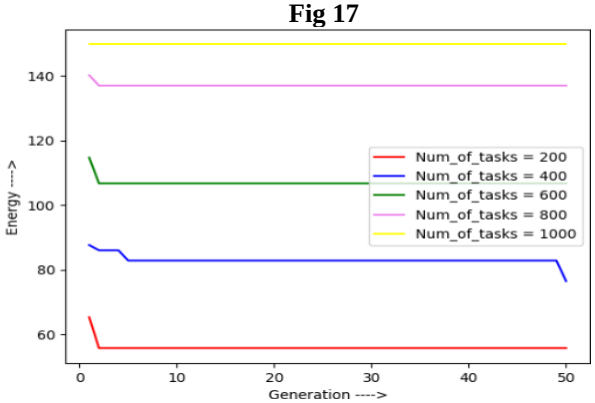

Fig 18

In the above figures 17 and 18, the number of processors is 400 and the number of generations is 10 and 50 respectively. The makespan times obtained for number of tasks as 200, 400, 600, 800 and 1000 was 34, 55, 69, 80 and 91 and 35, 48, 67, 86 and 94 respectively for number of generations as 10 and 50.

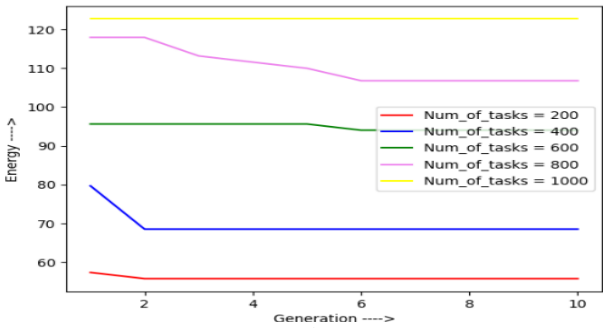

Fig 19

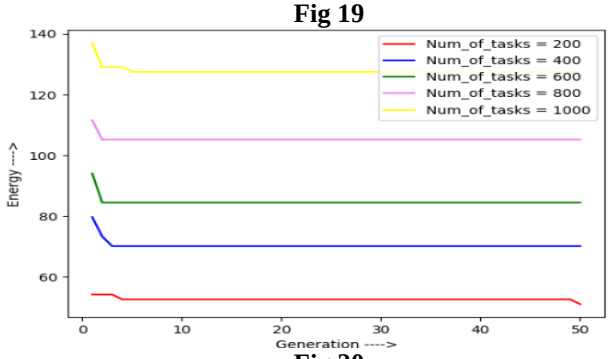

Fig 20

In the above figures 19 and 20, the number of processors is 500 and the number of generations is 10 and 50 respectively. The makespan times obtained for number of tasks as 200, 400, 600, 800 and 1000 was 35, 43, 59, 67 and 77 and 32, 44, 53, 66 and 80 respectively for number of generations as 10 and 50.

X. CONCLUSION

In this paper, we propose a new scheduling technique based on QGA. It is a combination of Genetic Algorithm with the idea of Quantum Computing. The QGA takes advantage of the properties of Qubits and Quantum Registers, thereby, increasing the speed of execution considerably as a result but causes the LOC to increase in the program and consequently to its complexity. Quantum Algorithms implemented on a classical computer will result in slowing down the execution time but will give the intended performance once implemented on Quantum Computers. The mutation step helps in avoiding the local minimum but as the number of generations increase, it often produces undesirable results. The optimal results are obtained by 10-15 generations. The proposed methodology helps reduce the energy consumption of a processor by 10-15% in almost all the test cases by balancing the load on each processor thereby, also reducing the makespan time greatly.

The future prospects of this work include incorporating techniques such as PSO and ACO to QGA to reach the global minimum faster as the stand-alone implementation of this algorithm is very time consuming. The scheduling of dependent tasks can also be worked upon by making suitable changes to the proposed algorithm.

REFERENCES

1. Hicham Ben Alla, Said Ben Alla, AbdellahEzzati AhmedMouhsen, "A Novel Architecture with Dynamic Queues Based on Fuzzy Logic and Particle Swarm Optimization Algorithm for Task Scheduling in Cloud Computing", Advances in Ubiquitous Networking 2, Lecture Notes in Electrical Engineering 397, DOI 10.1007/978-981-10-1627-1_16
2. Babur Hayat Malik, Mehwashma Amir, Bilal Mazhar, Shehzad Ali, Rabiya Jalil, Javaria Khalid, "Comparison of Task Scheduling Algorithms in Cloud Environment" (IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 9, No. 5, 2018
3. Bart Rylander, Terry Soule, James Foster & Jim Alves-Foss, "Quantum Genetic Algorithms", Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00), Las Vegas, Nevada, USA, July 8-12, 2000
4. SayantaniBasu, Marimuthu Karuppiah, K. Selvakumar, Kuan-Ching Li, S.K. Hafizul Islam, Mohammad Mehedi Hassan, Md. ZakirulAlam Bhuiyan, "An intelligent/cognitive model of task scheduling for IoT applications in cloud computing environment ", Future Generation Computer Systems · June 2018 DOI: 10.1016/j.future.2018.05.056
5. Juntao Ma, Weitao Li, Tian Fu, Lili Yan and Guojie Hu, "A novel dynamic task scheduling algorithm based on improved genetic algorithm in cloud computing", Wireless Communications, Networking and Applications pp 829-835
6. S. AjeenaBeegom, M. S. Rajasree, "A particle swarm optimization based pareto optimal task scheduling in cloud computing.", ICSI 2014: Advances in Swarm Intelligence pp 79-86
7. Shih-Tang Lo, Ruey-Maw Chen, Yueh-Min Huang & Chung-Lu Wu. "Multiprocessor system scheduling with precedence andresource constraints using an enhanced ant colony system" ,Expert Systems with Applications 34(3):2071-2081 · April 2008, DOI: 10.1016/j.eswa.2007.02.022
8. Babukarthik RG, P Dhavachelvan, 'Hybrid Algorithm using the advantage of ACO and Cuckoo Search for Job Scheduling', International Journal of Information Technology Convergence and Services (IJITCS) Vol.2, No.4, August 2012 DOI : 10.5121/ijitcs.2012.2403
9. Bahman Keshanchi, Alireza Souri&Nima Jafari Navimpour, "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(February 2017):1–21 · July 2016, DOI: 10.1016/j.jss.2016.07.006
10. Hamid Reza Boveiri, Raouf Khayami, MohamedElhoseny, M. Gunasekaran, "An efficient Swarm-Intelligence approach for task scheduling in cloudbased internet of things applications", Journal of Ambient Intelligence and Humanized Computing, September 2019, Volume 10, Issue 9, pp 3469–3479
11. Pandaba Pradhan, Prafulla Ku.Behera&B.N.B.Ray, "Modified Round Robin Algorithm for Resource Allocation in Cloud Computing", Procedia Computer Science 85:878-890 · December 2016
12. V. M. Arul Xavier, S. Annadurai, "Chaotic social spider algorithm for load balance aware task scheduling in cloud computing ", Cluster Computing 22(9) · January 2019, DOI: 10.1007/s10586-018-1823-x
13. Jocksam G. De Matos, Carla K. De M. Marques & Carlos Heitor Pereira Liberalino ,Genetic and static algorithm for task scheduling in cloud computing, IJCC 2019 DOI:10.1504/ijcc.2019.097891
14. Rafael Lahoz-Beltra, "Quantum Genetic Algorithms for Computer Scientists" , MDPI
15. Lizhe Wang, Samee U. Khan, Dan Chen, Joanna Kolodziej, Rajiv Ranjan, Cheng-zhong Xu, Albert Zomaya, "Energy aware parallel task scheduling in a cluster", Future Generation Computer Systems 29(7):1661-1670 · September 2013, DOI: 10.1016/j.future.2013.02.010

AUTHORS PROFILE

Sarthak Srivastavais currently pursuing his B.Tech. in Computer Science and Engineering from Sikkim Manipal Institute of Technology, Sikkim. He will complete his course in May 2020. His areas of interest are Machine Learning, Theoretical Computer Science and Computer Vision.

Mr. Santanu Kumar Misra is currently working in the department of Computer Science & Engineering, at Sikkim Manipal Institute of Technology, Sikkim. He received B.E in Computer Science & Engineering from WBUT, M. Tech in Computer Science & Engineering from SMU. He is currently pursuing his Ph. D in Computer Science & Engineering at National Institute of Technology, Sikkim. He has around twelve years of teaching experience. His area of research includes Operating Systems, Cloud Computing, Soft Computing etc.