Advances and Applications in Mathematical Sciences
Volume 20, Issue 10, August 2021, Pages 2045-2060

QUANTUM INSPIRED GENETIC ALGORITHM TO SOLVE MULTIPROCESSOR JOB SCHEDULING PROBLEM

RASHIKA BANGROO, KUSHAL GUPTA and ANIL KUMAR DAHIYA

DIT University
Dehradun, Uttarakhand, India
E-mail- dahiyaanil@yahoo.com

Abstract

The dynamic NP-Hard mixture problem formulation in the domain of simultaneous processing is the challenge of multiprocessor programming. Numerous heuristics and metaheuristics have been widely used by researchers to tackle this issue in an optimal way. Even so, the proposed method is among the many commonly used methods to solve such problems. But genetic algorithm has known to show certain drawbacks which limits its use in such problems. This paper therefore presents a modified rendition of the evolutionary algorithm, i.e. the Quantum Genetic Algorithm, to address the problem of multi-processor route optimization. The results collected were also contrasted with the other meta-heuristics to show their validity.

I. Introduction

The advantages of multiple heuristics and meta-heuristics approaches have been used by researchers to address multi-processor job scheduling problem. Literature comprises of many such heuristics. In 2008, Jin Shiyuan introduces a comparative analysis of various heuristics on the fundamental issue of multiprocessor job scheduling [1];Min-Min conceptual model by C. Kim and O. Ibarra [2], A^* selection strategy by Kcafilet [3], simulation model deliberation by Elperin and R. T. Eliasi [4]; fastest pace with first approximated time (HLFET) by K. Chandy as well as T. Adam [5]; intubation planning algorithmic (ISH) by Kruatrachue [6, 7] and evolutionary

2010 Mathematics Subject Classification: 03G12, 35J10, 60J70, 68Q12, 68M20.
Keywords: Multi-processor DAG Scheduling problem, Quantum Genetic Algorithm, Gauss Jorden Eradication, LU Decomposition.
Received October 7, 2020; Accepted January 15, 2021

algorithms by E. Hou and N. Ansari [8].

Genetic algorithm draws on the mathematical principle of competition and aims to construct a variety of natural events and to make them perform in an environment where even the best of the best can thrive. Every element in the package shall be equal to a genetic material. The quantum genetic algorithm was developed to create a sufficiently persuasive of a simulated annealing that combines the properties of each fundamental and GA computation. Initially, QGA was applied to solve a multi - objective optimization problem, such as the linear constraints, the traveling sales representative problem [10], etc. and has concluded that it increases the velocity and effectiveness of a standard GA. QGA offers many benefits among several meta-heuristic methods, along with excellent global search capability, as well as the scale of the local population does not influence the algorithm. As there is no empirical research done so far, this study thus introduces a modified technique for solving Most processed staff preparation challenges use Quantum-inspired optimization technique to minimize the overall time required in the execution of job It also enables a quantitative analysis of all out comes collected in conjunction with new meta-heuristic strategies. Other segment briefs about the concepts related to Multiprocessor DAG scheduling problem. Section III outlines the concepts of the Quantum Genetic Algorithm. This also briefs up on its associated work. Section IV, outlines the methodology proposed. Section *V* includes the experimental setup and performance parameters. Finally, the conclusive remarks and acknowledgments are given in section VI.

II. Multi-Processor Dag Scheduling Problem

Multi-Processor Job scheduling problem implies the scheduling of '*N*' number of jobs on '*M*' amount of Devices with these predictability and information sharing limitations to minimize overall execution time. It thus plays a potent role in productive application of the as sets. The activities are shown as DAG (direct acyclic graph) as $G(N, E)$, in which "*N*" symbolizes the series of all vertices and "*E*" symbolizes the series of all corners. [9]

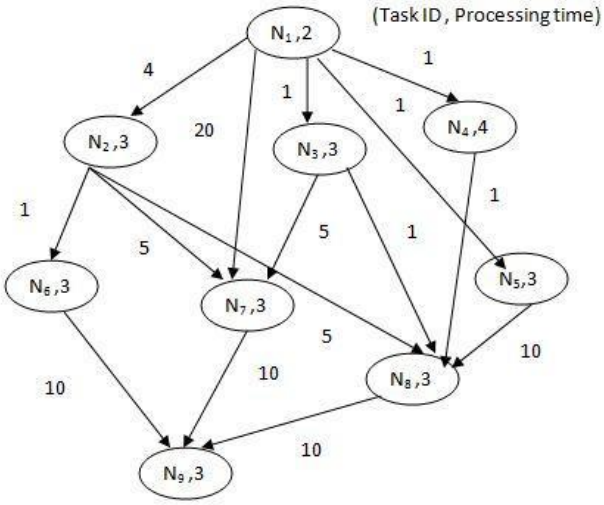

Figure 1. Directed Acyclic Graph.

All node is characterized by $N(n_t, P\text{ time})$ wherever n_t denotes the node id and the' P-time denotes the execution time of node n_i. An edge $(n_i, n_j) \in E$ symbolizes the message and precedence between the nodes n_i and n_j. It also indicates that the job n_j cannot start its execution prior to job n_i. [9, 11]. Thus, our objective is to assign these '*N*' nodes to the '*M*' homogeneous processors so that the make-span of the DAG is minimum. Symbol 1 describes a DAG with 9 nodes (jobs). Each node is associated with a node number and its processing time. The weights on the edges determine the communication delay. In case the two jobs are being allocated to the same processor, the communication cost in that case becomes zero. Figure 2 depicts a feasible schedule on two homogeneous processors corresponding to DAG in figure 1 which gives a total minimum make-span of 25 time units.

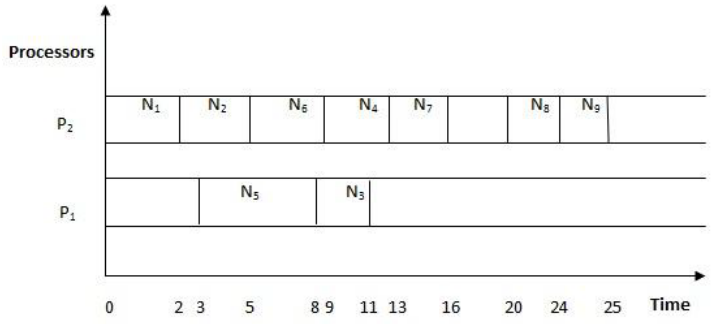

Figure 2. A Feasible Schedule for Figure 1.

III. Quantum Genetic Algorithm

Quantum genetics algorithms arise in the field of quantum processing. The quantum genetic algorithm is primarily centered on the principles of subatomic particles and the state of the considerable superposition. [12]. Information is stored as qubits in quantum computers. Qubit is a dual quanta scheme that works as a physical medium to stock the anticipated info.

Quantum computing will be in either of the domains $0>, |1>$ or the permutation of the different sides. The $0>$ and $>$ states are respectively referred to as the spin-down and spin-up qubit states. A quantum computing contains all spin up and spin down recorded data. The quantum computing status described this way:

$$|\phi> = \alpha|0> + \beta|> . \tag{1}$$

In which α and β are involved metrics recognized as the respective qubit state's probability amplitude. The following equation must be satisfied by the α and β

$$|\alpha|^2 + |\beta|^2 = 1. \tag{2}$$

A. Qubit Programming

Genetic algorithms that use binary coding use bits to represent each chromosome gene. In the case of QGA, a quantum bit [13] represents each gene of a chromosome. Qubit is typically described in terms of its probability

amplitude as $\begin{bmatrix} \alpha \\ \beta \end{bmatrix}$ and the multi-qubit encoding is defined as follows for n parameters:

$$\begin{bmatrix} \propto_1 & \alpha_2 & \cdots & \alpha_n \\ \beta_1 & \beta_2 & & \beta_n \end{bmatrix}$$

where n denotes chromosome gene counts.

B. Quantum Rotating Gates

The Quantum genetic model utilizes qubit's probability amplitude [14] to Embed the set of genes and constantly moving qubit gates to perform the chromosome update procedure. In such cases, the chromosome thus produced from the parent chromosome does not depend on the traits of the parent chromosome but is measured by means of the probabilistic magnitude from every case and the optimal parent chromosome solution. The primary distinction between standard GA and QGA is the use of Quantum Rotating gates to alter the amplitude beliefs of the possibility. Therefore, QGA's primary operation is Quantum Rotating Gates, which significantly impacts the efficiency of the algorithm and is generally described as

$$U(\theta_i) = \begin{bmatrix} \cos\ \theta_i & -\sin\ \theta_i \\ \sin\ \theta_i & \cos\ \theta_i \end{bmatrix}. \tag{3}$$

And the chromosome values are updated as:

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = U(\theta_i) \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = \begin{bmatrix} \cos\ \theta_i & -\sin\ \theta_i \\ \sin\ \theta_i & \cos\ \theta_i \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{4}$$

where $\begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix}$ and $\begin{bmatrix} \propto_i' \\ \beta_i' \end{bmatrix}$ denotes the probability amplitudes of i^{th} qubit chromosome before and after implementation of the rotating gates and θ_i indicates the rotating angle.

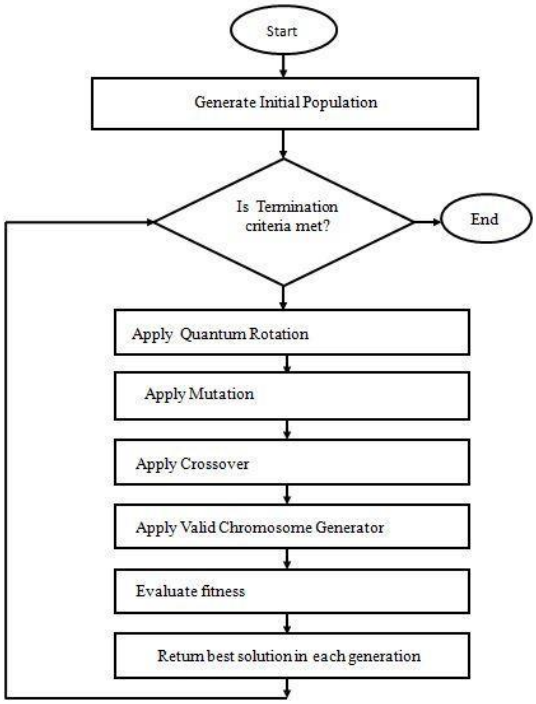

Figure 3. Flowchart of QGA.

This study focuses on the scheduling of jobs with the aim of minimizing the make-span of the jobs. Make-span is defined as the duration for all jobs to execute in a given schedule. There are some disadvantages of the standard QGA, such as that it only uses the fitness feature to determine the realistic alternative. This induces genetics even for chromosomes with a higher objective function to easily spread in the solution, leading to premature loss of diversity. The standard QGA is also suitable for universal quest capabilities, and while it is inadequate for specific check efficiency, resulting in slower consolidation in subsequent phases, leading to a globally optimized non-convergence solution. Therefore, some new operators are introduced in order to alter the suggested QGA to provide more effective outcomes at certain points. Figure 3 depicts the flow chart for the suggested QGA.

A. Fitness Function

Fitness functionality could be defined as an analytical process that allows the original parameter values to be accessed according to the problem and provides the most practicable resolution for the input results calculated. The

fitness function should always be competent adequate to ensure that a specified problem is solved with the least amount of resources. The fitness of the chromosomes is assessed constantly, so that the fitness function is sufficiently capable of producing rapid outcomes. The key purpose of the stated research is to increase the total schedule time for multiprocessor jobs. The solution can perhaps be attained through the use of make-span, that specifies the net scheduling span of the jobs as described below:

$$make \quad span \quad = \max \quad (FT \ (n)_{i,\ j}). \tag{5}$$

$FT\ (n)_{i,\ j}$ refers to an effecting interval of the job n_i on the appropriate processor p_i and $1 \le i \le N, 1 \le j \le N$. The fitness gathering taken into consideration in this work relies on the serial schedule scheme (SSS). The SSS sequentially assigns all jobs until a viable solution is reached. When the chromosomes or solution created is incorrect, the SSS model is used.

B. Valid Chromosome Generator

In some instances, the chromosomes generated may not be valid because they contravene certain precedence constraints. Therefore the valid generator of chromosomes (VCG) ensures the generation of a valid chromosome. Consequently, the VCG monitors and exchanges suspicious activities that can lead to an invalid chromosome generation. The in degree is calculated initially for each activity. The variable i is used to store the in degrees having null values. After that each solution's out degree is being calculated which belongs to I. Now every solution is being examined whether or not it belongs to i. If each individual's degree out is unique and which do not relate to I after which calculate the highest extent apart and interact with the occurrence. On the other hand, when all the out-degrees are related to I, then there is hardly any need for any switching.

C. Self Adaptive Rotation Angle

The rotating gate operator utilizes a fixed angle for standard QGA, were as in this work we have considered using a self-adaptive rotation angle approach. The value of the rotation angle in this approach is focused on an intrinsic method and is dynamically adapted. The adjustment approach comprises of comparison between the fitness of current use, $f(y)$ of the

separable with the current best possible individual. On the other hand, if $f(y) < f$ (best) then alter the quantum computing so that their probability amplitude (α_i, β_i) converges to the direction promising to *g*(best). The cost of the approach of rotation is accustomed in accordance with the following formulation:

$$\theta_i = \theta_{\max} - \left(\frac{\theta_{\max} - \theta_{\min}}{it_{\max}} \right) * iter , \tag{6}$$

where $it_{\max}$ symbolizes the absolute number of iterations and 'iter' denotes the current iteration. Selecting the appropriate position of alternation shows a significant role in determining the amount of iterations required to achieve an ideal solution. It too aids to retain a suitable equilibrium among confined optimal and comprehensive ideals. Consequently, the rotation angle value is changed by means of the self-adaptive turning position strategy for each iteration.

D. Mutation and Crossover Operator

After the rotation gate approach is implemented, the mutation operator is being used. It aims to distinguish certain chromosomes significantly from the present evolutionary path and thus prohibits them from drifting into the local optimum. The mutation operator replaces the values of the probability amplitude (α, β) which completely converses the evolutionary strategies of the chromosomes. This reinforces the algorithm's local search capacity and avoids cost of statistics from the chromosome. It additionally increases the population variety and decreases the likelihood of premature convergence. Figure 4 depicts the operation of a mutation operator.

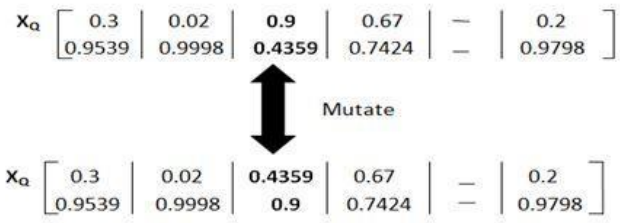

Figure 4. Quantum Mutation Operator.

The cross over operator helps in maintaining the diversification in the solution. Several variants of cross-over operators are available in literature. In this study, we contemplated the multi-point cross-operator that is identical to the standard genetic algorithms. This method involves the random selection of two parent chromosomes. The chosen parents are then coupled by a predefined rate of genetic defect and is the proportion of the complete offspring produced to the population density as a whole. Figure 5 illustrates the working of a cross over operator.

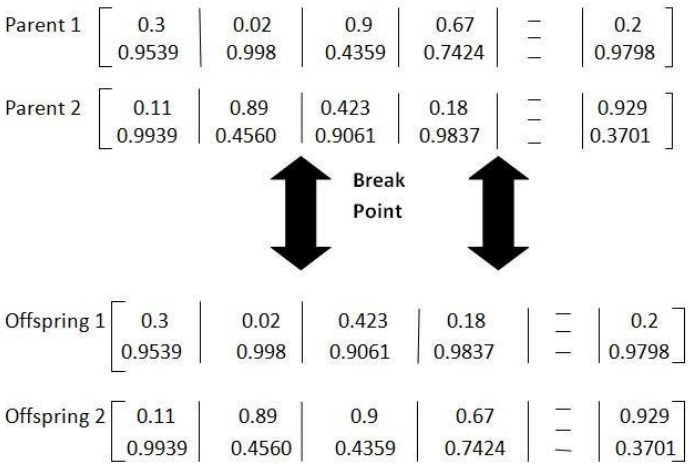

Figure 5. Crossover Operator.

D. Algorithm

The following steps are being performed in QGA:

1. At time $t = 0$, randomly generate the initial population using the equation (3) as follows: $P_q(t) = \{p_1(t), p_2(t), p_3(t), \ldots, p_n(t)\}$ by transforming Q-bit representation to chromosomes.

2. Calculate the fitness values of each individual chromosome and reorder them on the basis of their fitness values in descending order.

3. Using selection operator, select the optimum initial result by substituting the bottom 1/6th individuals by best 1/6th individuals

4. Repeat steps 5-12 until $n = 0$,

5. At $t = t + 1$, Update $P_n(t)$ by $P_n(t + 1)$ using Quantum gate rotating operator.

6. Using certain value of mutation probability, implement the mutation operation.

7. Execute the cross over operation.

8. Implement the effective chromosome generator (VCG).

9. Append the created solution to the best original answer.

10. Calculate the fitness values of newly generated individual chromosomes and reorder them on the basis of their fitness values in plunging direction.

11. Using selection operator, select the optimum initial result by substituting the bottom 1/6th individuals by best 1/6th individuals.

12. Keep record of the optimal solution in every origination.

The procedure is explained in detail below. Initially, all variables including number of jobs, communication delay, precedence constraints, size of the jobs, etc. are delivered as feedback. The first sample is then randomly generated using equation (3), by resetting both the values of α and β as $1/\sqrt{2}$ and $1/\sqrt{2}$ in the chromosome respectively.

Fitness is then used to calculate the optimal feasible solution in the population using equation (5) and the chromosomes are arranged by descending sequence according to their fitness scores. The optimal initial population is created by substituting $1/6$ of the produced population by the best $1/5$ th population. Increase t to $t + 1$ value and apply the quantum rotating gate strategy with the help of equation (4). The rotational angle value, i.e θ_i, is tuned by setting the values of $\theta_{\min}$ and $\theta_{\max}$ as $0.01 * \pi$ and $0.05 * \pi$ respectively. The crossover and mutation operators help to preserve the diversity of the method. Now, once again calculate the fitness of the generated population after implementation of the mutation method. Then the valid chromosome generator. Is being used to ensure that the chromosomes generated are valid. The chromosomes are then further arranged according to

their fitness scores in reducing sequence. Select the finest initial population by combining $1/6$ of the population produced with the finest $1/6$ population produced. Replay the process till the state of completion is attained, which in this situation is the proportion of chromosomes

Results and Conclusion

A. Experimental Evaluation

No such criteria are accessible in works for analyzing the presentation of job arranging difficulties [15]. Investigators frequently habit random graphs for implementation analysis. DAG is formulated using two fundamental problems of linear algebra i.e., LU decomposition and Gauss Jorden Elimination (GJE). [16] The suggested QGA algorithm for multiprocessor job scheduling problem is tested in MATLAB by using DAG generation method of Gauss Jorden Elimination (GJE) and LU decomposition [17]. Figure 6 and 7 depicts the graphical representation of GJE and LU decomposition problem respectively. Table 5.1 depicts the experimental set up details for the GJE job graph.

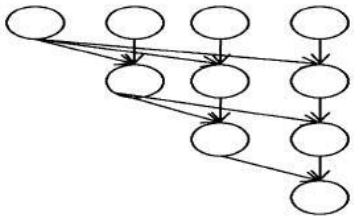

Figure 6. GJE job graph with 10 nodes.

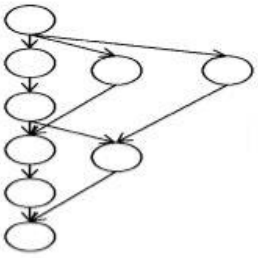

Figure 7. LU decomposition graph with 9 nodes.

Table 5.1. Experimental details of GJE job graph.

Number of Jobs(N)	15	21	28	36
Processing Time			40s	
Communication Cost	100	100	100	100
Processors(M)	4	4	4	4
QGA Iterations	10	10	10	10

To ensure a realistic comparison [16, 18, 9], all parameters of assessment are drawn from accessible meta-heuristics. The suggested model starts with the initialization of the quantum bit updating vector with random values ranging from 1 to n. Also, the chromosomes initially produced are 1000.

Table 5.2. LU decomposition job graph experimental.

Number of Jobs(N)	14	20	27	35
Processing Time			10s bottom layer job plus 10s for every layer	
Communication Cost	80/edge	80/edge	80/edge	80/edge
Processors(M)	4	4	4	4
QGA Iterations	10	10	10	10

Figure 8 and Figure 10 depicts the comparison of QGA with other meta-heuristics using the GJE job graph and the LU decomposition process graphically.

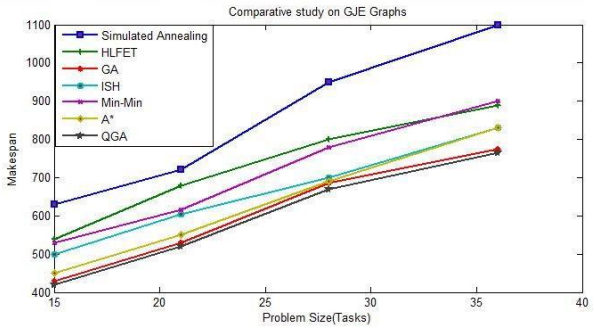

Figure 8. Comparison of 7 heuristics on GJE graph.

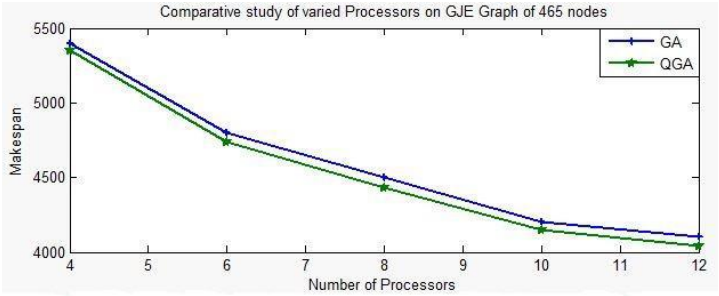

Figure 9. Relative analysis of two meta-heuristics using GJE work graph with varying processor numbers.

Figure 8 and 10 show that the make-span acquired with QGA is superior to any other meta-heuristic on different problem sizes, using values provided in table 5.1 and in table 5.2. The graph thus demonstrates QGA's efficacy against other methods. The findings also indicate that the heuristics GA and QGA are more encouraging than other equivalents. Therefore, we only take these two methods into account for further assessment. Figure 9 demonstrates the comparative analysis of GA and QGA by erratic GJE job graph of 465 nodes in the number of processors. Illustration 11 displays the evaluation on LU graphs of 464 nodes with respect to the varied processors. QGA therefore surpasses all other meta heuristics and is therefore a promising alternative to schedule multi-processor jobs. It is evident from Figure 9 and 11 that when the quantity of processors varies in accordance to size of GJE and LU graphs, the QGA functions better than other heuristics.

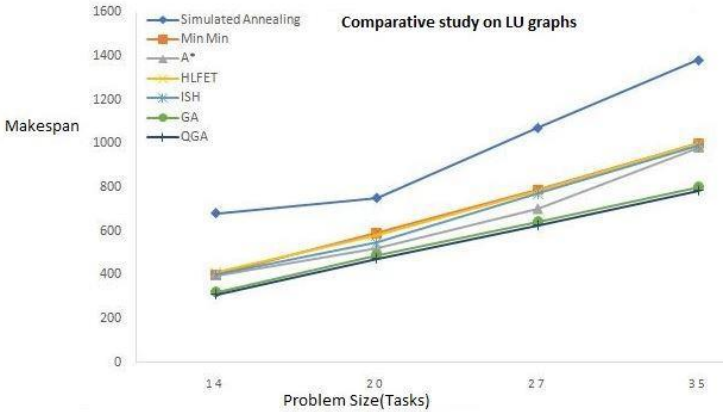

Figure 10. Comparison of 7 heuristics on LU job graph.

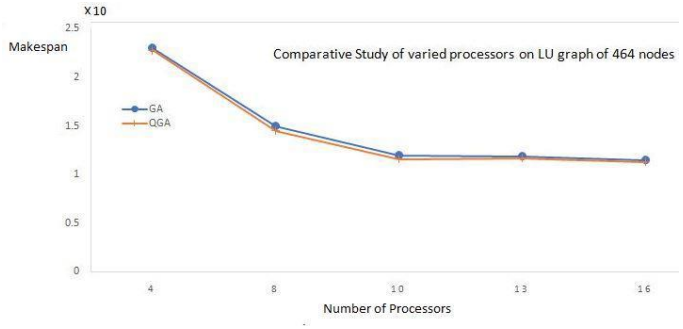

Figure 11. Relative analysis of two meta-heuristics using LU work graph with varying processor numbers.

V. Conclusion

This research has addressed the problem of scheduling multiprocessor jobs by means of the Quantum Genetic Algorithm. To ensure compatibility with the same problem, the suggested QGA has been amended at some stages using some advanced operations. Instead of using a steady value of about 0.01π, the strategy for self-adaptive variation is actuality used in which the alternation angle varies as per the requirement of the problem. Additional operators, like mutation and crossovers, are also used in order to diversify and converge the solution to the optimal global value. In addition, comparisons are made with outcomes acquired from other meta-heuristics and the quantum genetic algorithm on job scheduling problem and QGA is found to be more efficient than its counterparts. Two common glitches of rectilinear algebra i.e. Gauss Jordan Elimination (GJE) as well as LU decomposition, are used to verify the efficiency of the suggested QGA. Thus, the amalgam meta-heuristic QGA offers an efficient alternative to the solution of multiprocessor scheduling problem and can thus also be used to efficiently address other major complex issues of combinatorial optimization. The QGA can be used in future work for other complex combinatorial problems.

References

[1] Jin, Shiyuan, Guy Schiavone and Damla Turgut, A performance study of multiprocessor job scheduling algorithms, The Journal of Supercomputing 43(1) (2008), 77-97.

[2] Oscar H. Ibarra and Chul E. Kim, Heuristic algorithms for scheduling independent jobs on non-identical processors, Journal of the ACM (JACM) 24(2) (1977), 280-289.

[3] Kafil, Muhammad and Ishfaq Ahmad, Optimal job assignment in heterogeneous computing systems, Heterogeneous Computing Workshop, 1997. (HCW'97) Proceedings., Sixth. IEEE, 1997.

[4] R. Eliasi, T. Elperin and A. Bar-Cohen, Monte Carlo thermal optimization of populated printed circuit board, IEEE transactions on components, hybrids, and manufacturing technology 13(4) (1990), 953-960.

[5] Thomas L. Adam, K. Mani Chandy and J. R. Dickson, A comparison of list schedules for parallel processing systems, Communications of the ACM 17(12) (1974), 685-690.

[6] B. Kruatrachue and T. G. Lewis, Duplication scheduling heuristics (dsh): A new precedence job scheduler for parallel processor systems, Oregon State University, Corvallis, OR (1987)

[7] Kruatrachue, Boontee and Ted Lewis, Grain size determination for parallel processing, IEEE software 5(1) (1988), 23-32.

[8] Hou, S. H. Edwin, Nirwan Ansari and Hong Ren, A genetic algorithm for multiprocessor scheduling, IEEE Transactions on Parallel and Distributed systems 5(2) (1994), 113-120.

[9] Neetesh Kumar and Deo Prakash Vidyarthi, A novel hybrid PSOGA meta-heuristic for scheduling of DAG with communication on multiprocessor systems, Engineering with Computers 32(1) (2016), 35-47.

[10] Talbi, Hichem, Amer Draa and Mohamed Batouche, A new quantum-inspired genetic algorithm for solving the travelling salesman problem, Industrial Technology, 2004. IEEE ICIT'04. 2004 IEEE International Conference on. Vol. 3. IEEE, 2004.

[11] Hwang, Reakook, Mitsuo Gen and Hiroshi Katayama, A comparison of multiprocessor job scheduling algorithms with communication costs, Computers Operations Research 35(3) (2008), 976-993.

[12] Han, Kuk-Hyun and Jong-Hwan Kim, Quantuminspired evolutionary algorithm for a class of combinatorial optimization, IEEE transactions on evolutionary computation 6(6) (2002), 580-593.

[13] Lahoz-Beltra, Rafael. Quantum genetic algorithms for computer scientists, Computers 5(4) (2016), 24.

[14] Zhang, Gexiang, Weidong Jin and Laizhao Hu, A novel parallel quantum genetic algorithm, Parallel and Distributed Computing, Applications and Technologies, 2003. PDCAT'2003. Proceedings of the Fourth International Conference on, IEEE, 2003.

[15] Y-K. Kwok and Ishfaq Ahmad, Benchmarking the job graph scheduling algorithms, Parallel Processing Symposium, 1998. IPPS/SPDP 1998. Proceedings of the First Merged International... and Symposium on Parallel and Distributed Processing 1998. IEEE, 1998.

[16] Jin, Shiyuan, Guy Schiavone and Damla Turgut, A performance study of multiprocessor job scheduling algorithms, The Journal of Supercomputing 43(1) (2008), 77-97.

[17] Gerasoulis, Apostolos and Tao Yang, Performance bounds for column-block partitioning of parallel Gaussian elimination and Gauss-Jordan methods, Applied numerical mathematics 16(1-2) (1994), 283-297.

[18] R. Bangroo, N. Kumar and R. Sharma, A Model for Multi-processor Job Scheduling Problem Using Quantum Genetic Algorithm, In: A. Abraham, P. Muhuri, A. Muda, N. Gandhi (eds) Hybrid Intelligent Systems, HIS 2017. Advances in Intelligent Systems and Computing 734. Springer, Cham (2018).