SQGA: Quantum Genetic Algorithm-based Workflow Scheduling in Fog-Cloud Computing

Raouf Belmahdi* · **Djamila Mechta** * · **Saad Harous** ** · **Abdelhak Bentaleb** ***

2022 International Wireless Communications and Mobile Computing (IWCMC) | 978-1-6654-6749-0/22/$31.00 ©2022 IEEE | DOI: 10.1109/IWCMC55113.2022.9825324

Abstract Fog computing represents an extension of the Cloud infrastructure, which allows the improvement of the performance of IoT applications. The problem of task scheduling represents a challenge in this type of environment, with the aim of how to allocate the tasks to the different nodes of the Fog-Cloud infrastructure, in order to minimize makespan, cost, response time, and energy. In this paper, we propose SQGA— an algorithm to improve the workflow scheduling in Fog-Cloud environment. This algorithm is based on the quantum genetic algorithm QGA and aims to improve the makespan of applications deployed in the Fog-Cloud computing environment. The proposed SQGA scheduling algorithm is compared to the classical genetic algorithm and the First Come First Served algorithm . The experiment results show that the proposed SQGA algorithm is more efficient in makespan, and adapts better to the available resources.

Keywords Fog Computing · Cloud Computing · IoT · Tasks scheduling · Quantum genetic algorithm · Makespan

1 Introduction

Fog Computing is a paradigm that extends Cloud Computing close to the edge of the network. It provides computing, storage and networking services between end devices and traditional Cloud data centers [1], and offers several advantages for IoT applications such as [2] [3]: low latency, bandwidth saving, geographical distribution, *etc*. The total execution time or makespan is an essential QoS metric used to measure the IoT application performance. In order to improve QoS of IoT applications', the scheduling of tasks in the Fog-Cloud Computing may be optimized and consequently the system performance will be enhanced.

Task scheduling represents an NP-hard problem [4] where several types of existing algorithms have been proposed in the literature to solve it. Among them, evolutionary algorithms such as the classical genetic algorithm GA, which is widely used [5] in this field by applying selection, crossover and mutation genetic operators. The Quantum Genetic Algorithm QGA [6], has improved the performance of the classical GA, by combining the classical GA and the principle of quantum mechanics. This increases the ability of the global search, due to the diversity of a population caused by quantum representation.

In this paper, we proposed a new method for workflow tasks scheduling based on quantum approach named SQGA in the Fog-Cloud Computing. The proposed SQGA aims, primarily, to optimize the total execution time or the makespan of an IoT application.

The rest of the paper is organized as follows: in section 2, we define a formal representation of the scheduling problem, in order to obtain the fitness function to be optimized. In the next section, we explain the proposed scheduling approach, as well as the quantum representation and encoding of the solution, and also in this part we detail the different functionalities of the proposed SQGA scheduling algorithm. In section 4, we evaluate our scheduling method and discuss the results. Finally, we conclude the paper.

1.1 Related Work

Recently, several works have been proposed by researchers, in the field of tasks scheduling in the Fog-Cloud Computing

* LRSD Lab, Computer Science Department, College of Science, University of Sétif-1, Algeria
E-mail: {raouf.belmahdi,mechtadjamila}@univ-setif.dz
· ** Department of Computer Science, College of Computing and Informatics, University of Sharjah, UAE E-mail: harous@sharjah.ac.ae
· *** School of Computing, National University of Singapore E-mail: bentaleb@comp.nus.edu.sg

Authorized licensed use limited to: BRAC UNIVERSITY. Downloaded on November 01,2023 at 17:19:36 UTC from IEEE Xplore. Restrictions apply.

environment. These research works are based on different methods and algorithms, which have a common objective of improving tasks scheduling. Among these algorithms , most of the existing solutions leverage the genetic GA and the quantum genetic QGA algorithms.

In the work [7], the authors proposed a cost-aware Genetic algorithm, which allows reducing the cost of performing tasks in the Fog-Cloud layer and increasing the success rate of responding to tasks in deadlines. In [8], the authors proposed a customized genetic algorithm, for schedule IoT requests to minimize overall latency in Hybrid Fog-Cloud Computing. In [6], the authors proposed a new evolutionary computing method called a genetic quantum algorithm GQA, based on the principle of quantum computing and the genetic algorithm. This algorithm adopts a probabilistic representation of the solution which uses the principle of superposition, where this solution is represented in the form of a qubit chromosome. The experimental results demonstrated that the proposed algorithm GQA has a fast convergence and acceptable global search capability better than the classical genetic algorithm. It proposes a new approach to solve combinatorial optimization problems.

Several research works have adopted this scheduling approach based on the genetic quantum algorithm [6] . In [9], the authors have proposed a clustering algorithm based on the genetic quantum algorithm called QGCA. It selects clusters and their cluster heads in a wireless sensor network (WSN) while reducing energy consumption. In [10] and [11], the authors applied the QGA algorithm to solve the scheduling problem of a computational grid.

2 Problem Formulation

In this work, we are interested in the application of the Fog-Cloud Computing environment, where the tasks are dependent and presented in the form of a workflow. The workflow is presented by a Directed Acyclic Graph (DAG), where the nodes represent the tasks and the links represent the precedence and communication between the tasks.

A workflow consists of a set of dependant tasks, and it is formulated as follows:

$$T = \{T_1, T_2, T_3, ..., T_n\} \tag{1}$$

Where, each task is characterized by the following: (a) the computation workload W_i, (b) the list of required resources such as the required memory size SM, and (c) the size of the input and output files denoted by SI and SO, respectively.

The Fog-Cloud Computing environment consists of a set of nodes:

$$N = \{N_1, N_2, N_3, ..., N_k\}, \text{ where } N = \left\{N_{fog} \cup N_{cloud}\right\} \tag{2}$$

Each node N_i that belongs to the Fog layer or the Cloud layer of the infrastructure is characterized by the following: (a) a CPU Operating Frequency CPU_F $\left(N_j\right)$ where $N_j \in N$, (b) the transmission rate, (c) the storage capacity and, (d) the required RAM size.

The workflow tasks are allocated to the nodes of the Fog-Cloud Computing to be processed by these nodes.

The execution time ExT $\left(T_i, N_j\right)$ of a task T_i by a given node N_j is calculated by Equation 3.

$$\text{ExT}\left(T_i, N_j\right) = \frac{W_i}{\text{CPU\_F}\left(N_j\right)} \tag{3}$$

During the execution of the workflow tasks, it is necessary to transmit the data between the different tasks, this generates a transmission time TrT $\left(e_{ij}\right)$, which is calculated using Equation 4.

$$\text{TrT}\left(e_{ij}\right) = \begin{cases} \frac{We_{ij}}{r_{ij}} & \text{if} \quad \tau_i \neq \tau_j \\ 0 & \text{if} \quad \tau_i = \tau_j, \end{cases} \tag{4}$$

where e_{ij} represents the edge between two tasks T_i and T_j, τ_i represents the allocation of the task T_i to the node N_i, We_{ij} represents the quantity of data transmitted between two tasks T_i and T_j, r_{ij} represents data transfer rate between two tasks T_i and T_j. In the case of two tasks are allocated to the same node ($\tau_i = \tau_j$) then the transmission times TrT $\left(e_{ij}\right)$ is *null*.

In our work, we are interested in the improvement of total execution time or the makespan of the workflow. It is equal to the sum of the execution time of all tasks and transmission time between these tasks. To calculate the makespan we must first calculate the start time (denoted by ST) and finish time (denoted by FT) values for each node in the workflow.

The start time is calculated using Equation 5:

$$\text{ST}\left(T_i\right) = max\left\{\text{FT}\left(T_j\right) + \text{TrT}\left(e_{ij}\right)\right\} \text{ where } T_j \in pred\left(T_i\right) \tag{5}$$

where FT $\left(T_j\right)$ denotes the finish time of the Task T_j, knowing that T_j belongs to the predecessors (*pred*) of the task T_i.

We calculate the FT value of a given task using Equation 6.

$$\text{FT}\left(T_i\right) = \text{ST}\left(T_i\right) + \text{Ext}\left(T_i, N_i\right) \tag{6}$$

Then, the value of makespan is calculated using Equation 7.

$$\text{Makespan} = max\left\{FT(T_i)\right\} \tag{7}$$

This work aims to optimize the objective function (fitness) by minimizing the value of makespan given by Equation 8.

$$\text{Fitness} = min\left\{\text{Makespan}\right\} \tag{8}$$

Authorized licensed use limited to: BRAC UNIVERSITY. Downloaded on November 01,2023 at 17:19:36 UTC from IEEE Xplore. Restrictions apply.

3 Scheduling Approach

To enhance the workflow tasks scheduling process , we use an evolutionary algorithm called Quantum Genetic Algorithm QGA. This algorithm is based on the quantum principle computing, such as qubit and state superposition. While the basic unit for encoding information in quantum computing is the quantum bit (or Q-bit). Where in the classic binary bit its state is 0 or 1, in the other case, the quantum state of qubit can exist as a superposition of the two quantum states [12], a qubit may be in the 0 state, in the 1 state or in any superposition of the two states [6].

The state of a qubit can be represented as shown in Equation 9.

$$|\Psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \tag{9}$$

Where α and β are complex numbers, representing probability amplitudes [12], while $|\alpha|^2$ represents the probability that the qubit is in state 0, $|\beta|^2$ represents the probability that the qubit is in state 1 [6], where the values of α and β satisfythe normalization condition given in [12] and represented as follows:

$$|\alpha|^2 + |\beta|^2 = 1 \tag{10}$$

3.1 The proposed scheduling algorithm for Fog-Cloud Computing (SQGA)

In this work, we propose an algorithm for the scheduling of workflow tasks (SQGA). This algorithm reduces the total execution time of the tasks, which are allocated to the different nodes of the Fog and Cloud Computing layers. The proposed solution is based on the GQA algorithm [6], and adapted to solve the task scheduling problem. The SQGA algorithm is presented in Algorithm 1.

3.1.1 Encoding of the solution

In the classical evolutionary algorithms such as GA, a solution is defined by a population of individuals represented by chromosomes, which can be encoded as a string of different formats, such as bits, integers, characters, *etc.* In contrast, in algorithms based on QGA, a population is represented by a set of qubits chromosomes denoted by $Q(t)$ as highlighted in Equation 11.

$$Q(t) = \left\{q_1^t, q_2^t, ..., q_n^t\right\} \tag{11}$$

Where $Q(t)$ represents a population at generation t, n represents the size of the population, and q_j^t (see Equation 12) represents a qubit chromosome j at generation t, and $j = 1, 2, ..., n$. These chromosomes are encoded as a string of qubits or m-qubits, and the state of the qubit is defined with a pair of complex numbers α and β.

$$q_j^t = \begin{bmatrix} \alpha_1^t & \alpha_2^t & \dots & \alpha_m^t \\ \beta_1^t & \beta_2^t & \dots & \beta_m^t \end{bmatrix}, \tag{12}$$

where m is the string length of the qubit chromosome.

Algorithm 1 SQGA for Fog-Cloud Computing infrastructure

```
begin
    t ← 0
    initializeQ(t)
    makeP(t)
    allocateP(t)
    evaluateP(t)
    storeBestP(t)
    while MaxGeneration do
        begin
            t ← t + 1
            makeP(t)
            allocateP(t)
            evaluateP(t)
            updateQ(t)
            storeBestP(t)
        end
    end while
end
```

3.1.2 Description of the algorithm

The proposed SQGA algorithm consists of five steps as described below.

- In the first step "*initialize(t)*", we initialize the α and β coefficients of each qubit of the population $Q(t)$ at $t = 0$ or initial generation, by the same value of probabilities which is equal to $\frac{1}{\sqrt{2}}$, see Equation (13).

$$q_j^0 = \begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} & \cdots & \frac{1}{\sqrt{2}} \end{bmatrix}, \tag{13}$$

  for $j = 1, 2, ..., m$.
- In the second step "*makeP(t)*", first of all the algorithm must measure the values of the qubits of the population $Q(t)$ at a generation t, and then on the basis of these measured values, the procedure *makeP(t)* generates a binary representation called binary string X_j^t, of the set of solutions at a generation t, we denote by $P(t)$, see Figure 1.

$$P(t) = \left\{X_1^t, X_2^t, ..., X_n^t\right\}, \tag{14}$$

  where X_j^t represents a binary solution, and $j = 1, 2, \ldots .., n$.

  The *makeP(t)* procedure is based on a function called "*measure(q)*", see Algorithm 2. This function allows us to measure the state of the qubits in order to generate a

Authorized licensed use limited to: BRAC UNIVERSITY. Downloaded on November 01,2023 at 17:19:36 UTC from IEEE Xplore. Restrictions apply.

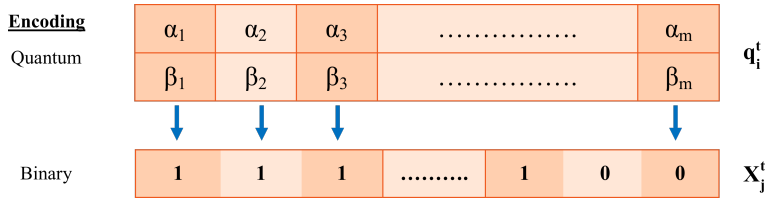

Fig. 1 Qubit chromosome measurement

Algorithm 2 Measure function

```
function MEASURE(q)
    if RANDOM(0, 1) > α² then
        return 1
    else
        return 0
    end if
end function
```

binary representation, it starts by generating a random value r in the range of [0..1], and it compares with the value of α^2 of qubit of a given solution q_j^t, in the case where $r > \alpha^2$ we set the bit of the binary string to 1, otherwise we set it to 0.

– The third step is the allocation procedure named by "*allocateP(t)*". This procedure converts a binary string from the X_j^t solution to a decimal representation, the value obtained by the conversion represents the identifiers of all nodes N_i in the Fog-Cloud infrastructure, see Equation 2.

To calculate the size m of the binary string of the solution X_j^t, we use Formula: $m = number\ of\ bits\ per\ node \times number\ of\ tasks$, and the value of m represents also the number of qubits in a quantum representation. To determine the number of bits to present a node N_i of the infrastructure, we use Equation 15.

$$\text{Number of bits per node} = \log_2 (\text{number of nodes}) + 1 \quad (15)$$

To allocate the tasks to the infrastructure nodes, we must assign in the same order, each node obtained by the conversation to the task (Equation 1) that suits the workflow, see Figure 2.

– After the allocation of tasks T_i to the appropriate nodes N_i, SQGA algorithm uses the procedure called "*evaluateP(t)*", to evaluate each solution of the population at a generation t. This procedure calculates the value of the objective function described by Equation 8 for each solution, in order to have a minimum makespan value. All values of the Makespan are calculated by the evaluation of the objective function, and sorted using the function called "*storeBestP(t)*", in order to find the best solution (minimum makespan) at generation t.

– In the while loop block of Algorithm 1, each time we generate a new generation, and before passing from one generation to another, we update the qubit chromosomes of the population $Q(t)$ using the rotation gate denoted by $U(\theta)$. This update is calculated using Equation 16.

$$\begin{bmatrix} {\alpha_i}^{t+1} \\ {\beta_i}^{t+1} \end{bmatrix} = \begin{bmatrix} \cos(\theta_i) & -\sin(\theta_i) \\ \sin(\theta_i) & \cos(\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i^t \\ \beta_i^t \end{bmatrix}, \quad (16)$$

where the new value of the angle of rotation θ_i for quantum qubit gate is calculated as follows: $\theta_i = S(\alpha_i, \beta_i) \cdot \Delta\theta_i$, where $S(\alpha_i, \beta_i)$ determines the direction of the angle, and the value of $\Delta\theta_i$ is determined from the Lookup table shown in Table 1, which ensures the convergence of the algorithm.

Table 1 Lookup table of rotation angle.

x_i	b_i	$f(x) \geq f(b)$	$\Delta\theta_i$	s($\alpha_i\beta_i$)			
				$\alpha_i\beta_i > 0$	$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\beta_i = 0$
0	0	false	0	0	0	0	0
0	0	true	0	0	0	0	0
0	1	false	δ	+1	-1	0	±1
0	1	true	δ	-1	+1	±1	0
1	0	false	δ	-1	+1	±1	0
1	0	true	δ	+1	-1	0	±1
1	1	false	0	0	0	0	0
1	1	true	0	0	0	0	0

Here, x_i is the *i-th* bits of binary solution x, b_i is the *i-th* bits of the best solution b, $f(.)$ is the fitness function, $s(\alpha_i\beta_i)$ is the sign of the rotation angle θ_i.
The values shown in the lookup table (Table 1), are based on the strategy proposed in [13]. It presents a universal strategy in the form of an update lookup table, that we can use in various optimization problems.

In lookup table, the case where the value of $f(x) > f(b)$, we adjust the qubit of the corresponding bit of $(x_i \neq b)$, by changing the probability amplitude towards the direction favoring the appearance of x_i. Otherwise, $f(x) < f(b)$, we adjust the qubit of the corresponding bit to change the probability amplitude towards the direction favoring the appearance of b_i.

The angle δ, shown in Table 1, presents the update value of $\Delta\theta_i$, this value must be small for better convergeence

Authorized licensed use limited to: BRAC UNIVERSITY. Downloaded on November 01,2023 at 17:19:36 UTC from IEEE Xplore. Restrictions apply.

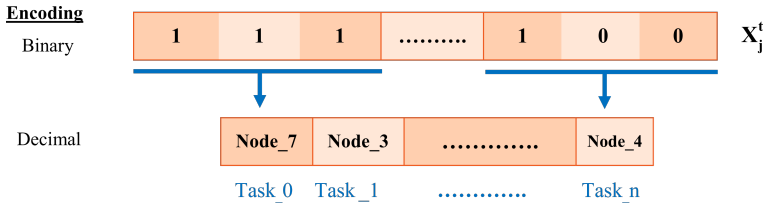

Fig. 2 Allocation of tasks to infrastructure nodes

towards the optimal solution. If the value of δ is very large, the solution may diverge or converge towards a local optimum [13].

4 Performance Evaluation

We conducted a series of simulation to evaluate the performance of the proposed scheduling approach and compared it to some existing algorithms.

4.1 Experimental Settings

The environment consists of a set of heterogeneous nodes, distributed in Fog-Cloud layers corresponding to the considered scenario. In the first scenario, the nodes are deployed in Fog-Cloud layers. In the second scenario, we use only the cloud layer. Each node has its own characteristics such as: CPU operating frequency, memory size and bandwidth. These nodes are responsible for processing the various end user requests. These requests are divided into tasks. Each task is characterized by the number of instructions (measured in million instructions), and the size of input and output files. The number of nodes varies from 5 to 25 nodes (second scenario) where the number of tasks varies from 50 to 250. We repeated each simulation case 100 times, and took the average value of the results obtained from each considered algorithm. Table 2 summarizes the values of the parameters used in the experiments.

Table 2 The parameters of the experiment.

Entity	Parameter	Values
Cloud nodes	CPU rate	[3000 - 4000] MIPS
	RAM size	[4000- 16000] MB
	Bandwidth	1000 Mbps
Fog nodes	CPU rate	[1000 - 2000] MIPS
	RAM size	[500 - 2000] MB
	Bandwidth	[100-1000] Mbps
Tasks	Length	[100 -1000] MI
	I/O file size	[10 -1000] MB
SQGA Algorithm	Rotation angle δ	0.001π

4.2 Results and discussion

4.2.1 Evaluation of the fitness function

To evaluate the efficiency of the proposed scheduling algorithm, we measured the makespan QoS metric for the three tasks scheduling strategies: the proposed SQCA, FCFS and GA. First, we generate a list of workflows of varying sizes, starting with 50 tasks per workflow up to 250 tasks per workflow. These workflows are sent to the nodes of the Fog-Cloud computing infrastructure to be processed. This infrastructure consists of 20 Cloud nodes and 10 Fog nodes. . For each experiment, the average value of the fitness function (shown in Equation 8) is calculated for the three algorithms SQGA, GA, FCFS. The results obtained are shown in Figure 3.

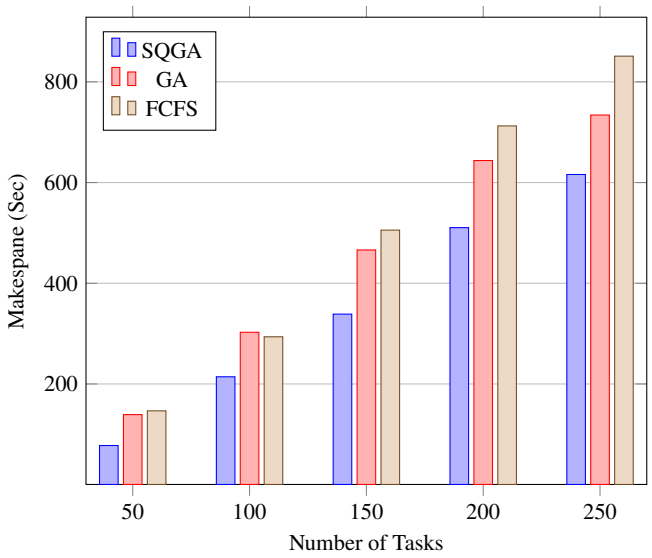

Fig. 3 Comparison of the fitness function of SQGA algorithm

Figure 3 shows that the proposed SQGA scheduling algorithm outperforms both GA and FCFS algorithms by 40% and 50% in makespan, respectively. This improvement is due to the fact that the proposed scheme is based on a quantum representation which allows a diversity of the population. In addition, it has a better ability of the global search compared to the classical GA.

Authorized licensed use limited to: BRAC UNIVERSITY. Downloaded on November 01,2023 at 17:19:36 UTC from IEEE Xplore. Restrictions apply.

4.2.2 Evaluation of the impact of using cloud nodes

We perform experiments where we set the workflow's size to 250 tasks, and varying the number of Cloud nodes used in the Fog-Cloud infrastructure. We varied the number of Cloud nodes from 5 to 25 nodes. We measure the average fitness value. The results are shown in Figure 4.

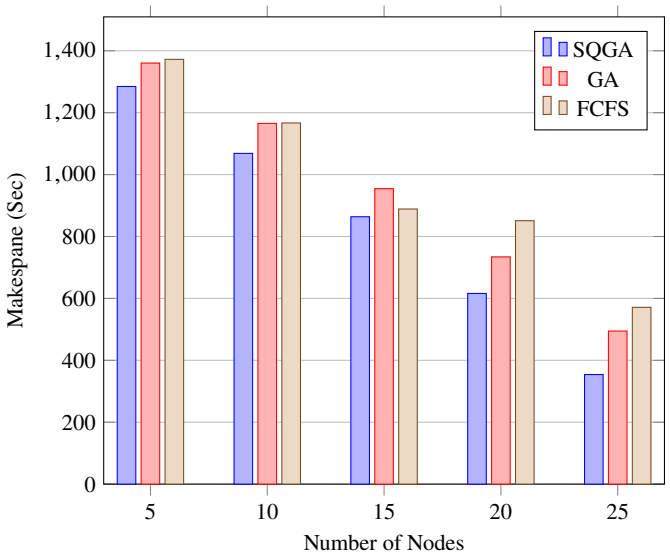

Fig. 4 Impact of using Cloud Nodes

According to the results obtained in this scenario, the number of nodes of the Cloud layer used in the scheduling of the tasks, has a direct impact on the performance of the scheduling. Figure 4 shows that when the number cloud nodes increases, the makespan decreases.

This is linked to the nature of the nodes of the Cloud layer, because they are more sophisticated, and have a large capacity relative to the Fog layer nodes. This case is very useful for the critical IoT applications where the response time must be reduced without affecting the cost of the operation.

Figure 4 shows that our proposed scheduling algorithm SQGA adapts better to the available resources. Based on the number of Cloud nodes provided, it always gives the minimum makespan compared to GA and FCFS algorithms.

5 Conclusion

In order to improve the scheduling in the Fog-Cloud Computing environment, and to minimize the total execution time of a workflow, a new makespan-aware quantum computing-based workload tasks scheduling algorithm is proposed. The various simulation results demonstrated that our scheduling approach based on the SQGA algorithm outperforms both GA and FCFS algorithms in term of makespan by 40% and 50%, respectively. SQGA algorithm adapts better to the resources available on the Fog-Cloud computing infrastructure. In the feature work, we plan to consider other QoS metrics in the scheduling approach, in order to improve the performance of task scheduling in this type of infrastructure. In addition, for efficient performance, we may use distributed simulation [14] [15] [16].

References

1. J. Zhu F. Bonomi, R. Milito and S. Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing*, MCC '12, page 13–16, New York, NY, USA, 2012. ACM.
2. H. Ning P. Hu, D. Sahraoui and T. Qiu. Survey on fog computing: architecture, key technologies, applications and open issues. *Journal of Network and Computer Applications*, 98:27–42, 2017.
3. M. Zia AK. Jadoon U. Akram S. Raza MR. Anawar, S. Wang. Fog computing: An overview of big iot data analytics. *Wireless Communications and Mobile Computing*, 2018, 2018.
4. Michael Pinedo and Khosrow Hadavi. Scheduling: theory, algorithms and systems development. In *Operations Research Proceedings 1991*, pages 35–42. Springer, 1992.
5. R. Belmahdi, D. Mechta, and S. Harous. A survey on various methods and algorithms of scheduling in fog computing. *Ingénierie des systèmes d'information*, 26:211–224, 04 2021.
6. KH. Han and JH. Kim. Genetic quantum algorithm and its application to combinatorial optimization problem. In *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No.00TH8512)*, volume 2, pages 1354–1360 vol.2, 2000.
7. A. M. Rahmani Z. Bakhshi T. S. Nikoui, A. Balador. Cost-aware task scheduling in fog-cloud environment. In *2020 CSI/CPSSI International Symposium on Real-Time and Embedded Systems and Technologies (RTEST)*, pages 1–8, 2020.
8. T. Landolsi R. O. Aburukba, M. AliKarrar and K. El-Fakih. Scheduling internet of things requests to minimize latency in hybrid fog–cloud computing. *Future Generation Computer Systems*, 111:539–551, 2020.
9. D. Mechta and S. Harous. Qgac: Quantum genetic based-clustering algorithm for wsns. In *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, pages 82–88, 2018.
10. Y. He Yunhui Z. Mo, G. Wu and L. Hongwei. Quantum genetic algorithm for scheduling jobs on computational grids. *Measuring Technology and Mechatronics Automation, International Conference on*, 2:964–967, 03 2010.
11. S. Prakash and D. P. Vidyarthi. A novel scheduling model for computational grid using quantum genetic algorithm. *J. Supercomput.*, 65(2):742–770, aug 2013.
12. R. LaPierre. *Qubits*, pages 57–72. Springer International Publishing, 2021.
13. B. Li J. Yang and Z. Zhuang. Research of quantum genetic algorith and its application in blind source separation. *Journal of Electronics (China)*, 20:62–68, 09 2003.
14. D. Kumar and S. Harous. Distributed simulation of timed petri nets: basic problems and their resolution. *IEEE transactions on systems, man, and cybernetics*, 24:1498–1510, 1994.
15. D. Kumar and S. Harous. An approach towards distributed simulation of timed petri nets. In *1990 Winter Simulation Conference Proceedings*, pages 428 – 435, 1990.
16. D. Kumar and S. Harous. A study of achievable speedup in distributed simulation via null messages. *IEEE Transactions on Parallel and Distributed Systems*, 4:347–354, 1993.

Authorized licensed use limited to: BRAC UNIVERSITY. Downloaded on November 01,2023 at 17:19:36 UTC from IEEE Xplore. Restrictions apply.