QIGA: Quantum-Inspired Genetic Algorithm for Dynamic Scheduling in Mobile Edge Computing

Sadra Galavani, Abolfazl Younesi, and Mohsen Ansari
Department of Computer Engineering,
Sharif University of Technology
Tehran, Iran
Emails: {sadragalavani79, abolfazl.yunesi, ansari}@sharif.edu

***Abstract*—This paper introduces a Quantum-Inspired Genetic Algorithm (QIGA) for efficient scheduling in a Mobile Edge Computing (MEC) environment. Inspired by the principles of quantum computing, it enhances solution diversity and convergence in the MEC landscape with heterogeneous resources and dynamic tasks. In this work, the QIGA framework is applied to an optimization problem with multiple conflicting objectives: makespan, energy consumption, and resource utilization. Extensive evaluations demonstrate the effectiveness of our proposed method, which shows improvement over three state-of-the-art scheduling methods in varied burst MEC scenarios in energy consumption about 15.7% on average and up to 39%, 20.5% in makespan, 27.5% in resource utilization, and latency up to 69.9%.**

***Keywords*— Quantum Computing, Scheduling, Mobile Edge Computing, Multi Objective Optimization**

I. Introduction

Mobile Edge Computing (MEC) has made a large impact in the landscape of distributed computing, offering unprecedented opportunities for latency-sensitive and computation-intensive applications [1][2]. By bringing computational resources closer to the network edge, MEC environments promise to reduce the load on centralized cloud infrastructure while simultaneously enhancing the quality of service for end users [2]. However, realizing the full potential of MEC critically depends on the effectiveness of task scheduling mechanisms, which must navigate the complex interaction of heterogeneous resources, diverse task requirements, and dynamic network conditions [1][8][15]. MEC represents an architectural shift where computing and storage resources are distributed across the network edge, typically in proximity to mobile devices and IoT sensors [2]. This approach aims to reduce latency, save bandwidth, and increase privacy by processing data closer to its source [1][4]. Task scheduling is extremely essential in MEC environments, as it serves as the backbone for optimizing resource utilization, energy efficiency, and latency reduction [7][20]. Effective scheduling algorithms must assign tasks to appropriate edge nodes, taking into account factors such as computational capacity, energy limitations, and network conditions to maximize overall performance.

The heterogeneous nature of tasks and resources in MEC poses a significant challenge to traditional scheduling approaches [15]. Edge nodes can vary considerably in computing capabilities, storage capacities, and energy limitations. Similarly, tasks in MEC scenarios exhibit diverse characteristics, from real-time applications to data-intensive, computation-heavy workloads [3][4][10][15]. This heterogeneity is further amplified by the dynamic nature of MEC environments, where resource availability and network conditions fluctuate rapidly [1][2][13]. Consequently, achieving efficient task scheduling in such settings requires adaptive, sophisticated algorithms capable of making real-time decisions under uncertainty [8][12][20]. While conventional scheduling methods have been effective in homogeneous computing environments [12], they often fall short in addressing the unique challenges posed by MEC [10][13][14]. Heuristic approaches, such as First Come-First Served (FCFS) or round robin, and MoHEFT [11] lack the flexibility needed to handle the dynamic and heterogeneous workload of MEC. Additionally, more advanced techniques, including traditional genetic algorithms (see Fig. 1), particle swarm optimization [15] (see Fig. 1), differential evolution [4], offer improved performance but struggle with convergence speed and solution quality in highly complex search spaces, as shown in fig. 1.

The limitations of existing approaches drive the exploration of novel scheduling paradigms [5][7][8][12][13]. In this context, Quantum-Inspired Genetic Algorithms (QIGA) emerge as a promising solution [6]. By exploiting principles of quantum computing within an evolutionary algorithm framework, QIGA offers the potential for enhanced solution space exploration, faster convergence, and improved adaptability to dynamic environments. The quantum-inspired approach enables a more precise representation of scheduling solutions, potentially resulting in higher-quality outcomes in MEC task allocation.

This paper presents a novel approach for scheduling heterogeneous tasks in MEC environments using QIGA. **Our contributions are as follows:**

- Introducing a QIGA-based scheduling framework specifically designed for the challenges of heterogeneous MEC environments, incorporating quantum-inspired operators to enhance solution diversity and convergence.
- Developing a multi-objective optimization model that simultaneously addresses key performance metrics in MEC, including task execution time, energy efficiency, and resource utilization.

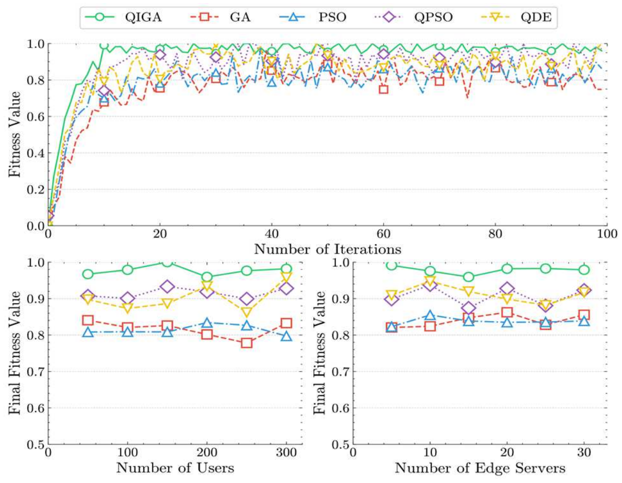


Fig. 1: Comparison of scheduling algorithms [4][7][15] across three metrics: convergence over iterations (top plot), (bottom left) performance versus the number of users and (bottom right) performance relative to the number of edge servers. In all cases QIGA has better fitness.

- Introducing a hybrid mobility model that combines Random Waypoint and Manhattan Grid models, dynamically adjusting based on the context, such as open or urban areas, to reflect real-world node mobility and network topology changes.
- Conducting extensive simulations to evaluate the effectiveness of our proposed approach (see GitHub [16]), demonstrating improvements in scheduling performance compared to state-of-the-art algorithms [4][11][12] across various MEC scenarios.

The structure of this paper is as follows: Section 2 reviews related work mobile edge computing. Section 3 introduces the system model. Section 4 presents the proposed method and followed by the experimental settings and results in Section 5. Finally, Section 6 concludes the paper.

II. Related Works

Konar et al. [6] proposed the Multi-Objective Quantum-Inspired Genetic Algorithm (Mo-QIGA) for scheduling real-time tasks in multiprocessor environments, aiming to minimize completion time and tardiness using qubit representation and rotation gates. Later, Konar et al. in 2017 [9] introduced the Hybrid Quantum-Inspired Genetic Algorithm (HQIGA), which used heuristics like EDF and SCTF to enhance scheduling efficiency and convergence. A 2020 paper [10] introduced the Binary Quantum-Inspired Gravitational Search Algorithm (BQIGSA) to improve makespan, load balance, and resource utilization. Bahrami et al. [15] presented BHNM, a hybrid algorithm for optimizing edge server placement in MEC environments. In 2024, QDE-UA was proposed [4] to optimize user allocation in edge computing, minimizing server requirements, energy consumption, and load imbalances. Zhang et al. [12] introduced chaotic quantum particle swarm optimization to reduce the energy and delay in MEC. Table 1 summarizes these related works, focusing on their key objectives such as makespan minimization, load balancing, resource utilization, latency optimization, and energy efficiency.

III. System Model

Our system model for a heterogeneous MEC environment comprises edge servers, fog nodes, and end devices. This model is structured as follows: we define a set of m edge servers as $\mathcal{E} = e_1, e_2, \ldots, e_m$. Each edge server $e_i \in \mathcal{E}$ is characterized by three key attributes: computational capacity c_i, measured in CPU cycles per second; available storage capacity s_i, measured in bytes; and energy efficiency factor ϵ_i, measured in joules per CPU cycle. Similarly, we define a set of n fog nodes as $\mathcal{F} = f_1, f_2, \ldots, f_n$. Each fog node $f_j \in \mathcal{F}$ is specified by the same attributes as edge servers: computational capacity c_j, storage capacity s_j, and energy efficiency factor ϵ_j. The network connection between edge servers and fog nodes is represented by a graph $G = (\mathcal{V}, \mathcal{L})$ where $\mathcal{V} = \mathcal{E} \cup \mathcal{F}$ represents the set of vertices (edge servers and fog nodes), and $\mathcal{L}$ represents the set of links between them. Each link $l_{ij} \in \mathcal{L}$ is characterized by bandwidth b_{ij} and latency λ_{ij}.

Table 1. Comparison of related works, MS: Makespan, LB: Load balancing, RU: Resource Utilization, L: Latency, EC: Energy Consumption, QP: Quantum principles

Ref.	**Objectives**					
	MS	**LB**	**RU**	**L**	**EC**	**QP**
[6]	✗	✗	✓	✗	✗	✓
[9]	✗	✓	✗	✗	✗	✓
[10]	✓	✓	✓	✗	✗	✓
[15]	✗	✗	✗	✓	✗	✗
[4]	✗	✓	✗	✗	✓	✓
[12]	✗	✗	✗	✓	✓	✓
QIGA	✓	✗	✓	✓	✓	✓

A. Task Model

We define the set of k tasks to be scheduled as $\mathcal{T} = t_1, t_2, \ldots, t_k$. Each $task\ t_i \in \mathcal{T}$ is characterized by four key attributes: task workload w_i, measured in CPU cycles, data size d_i measured in bytes, memory requirement μ_i, measured in bytes, and task deadline δ_i, measured in time units. Tasks are classified into different categories based on their characteristics: Large workloads, relatively low data size and normal workloads and moderate data sizes.

B. Resource Allocation

The task scheduling problem involves assigning each task $t_i \in \mathcal{T}$ to either an edge server $e_j \in \mathcal{E}$ or a fog node $f_k \in \mathcal{F}$. We introduce a binary variable x_{ij} to represent this assignment: $x_{ij} = 1$ if task t_i is assigned to resource j, otherwise, $x_{ij} = 0$. Here, resource j could be an edge server or a fog node.

C. Execution Model

Execution time measures the duration required for a task t_i to complete when assigned to a resource j (either an edge server or fog node). It consists of two primary components:

1) Computation Time (*CT*)*:* The time taken for task t_i, with workload w_i (in CPU cycles), to be processed by resource j, which has a computational capacity c_j (in CPU cycles per second).

$$CT = \frac{w_i}{c_j} \qquad (2)$$

2) Data Transfer Time (*DTT*): The time required to transfer data associated with task t_i from the device to resource j. This depends on the data size d_i (in bytes) and the bandwidth b_{ij} of the link between the device and resource j.

$$DTT = \frac{d_i}{b_{ij}} \qquad (2)$$

Thus, the execution time is the sum of these two components:

$$ET_{ij} = \frac{w_i}{c_j} + \frac{d_i}{b_{ij}} \qquad (3)$$

D. Energy Consumption Model

Energy consumption represents the total energy required by a resource j to process a task t_i. It depends primarily on the workload w_i of task t_i and the energy efficiency factor ϵ_j of resource j, which indicates the energy used per CPU cycle (measured in joules per cycle). The energy consumed by task t_i on resource j is calculated as:

$$EC_{ij} = w_i \cdot \epsilon_j \qquad (4)$$

E. Utilization Model

Resource utilization quantifies how much of a resource j's computational capacity is used when executing the scheduled tasks [20]. This metric is defined as the ratio of the total workload assigned to resource j to its computational capacity c_j. If we have k tasks, with each task t_i either assigned or not assigned to resource j (indicated by binary variable x_{ij}), the resource utilization RU_j is calculated as:

$$RU_j = \frac{\sum_{i=1}^{k} x_{ij} \cdot w_i}{c_j} \qquad (5)$$

F. Mobility Model

In our MEC environment, we employ a hybrid mobility model to simulate complex and realistic scenarios, combining elements of the Random Waypoint model (RWM) [17] and the Manhattan Grid model (MGM) [18]. The RWM applies to mobile end devices and some edge nodes in open areas, where nodes move randomly at varying speeds, pause at destinations, and then select new random destinations. In contrast, the MGM is used for edge nodes in urban environments, where nodes move in a grid pattern along streets, constrained to horizontal and vertical directions, with a probability of turning at intersections. This hybrid model dynamically switches between these models based on the node's context or location. The mobility function $M(n, t)$ is defined as follows:

$$M(n,t) = \begin{cases} \text{RWM}, & if\ \ n \ \text{is in an open area} \\ \text{MGM}, & if\ \ n \ \text{is in an urban env.} \end{cases} \tag{6}$$

Key parameters in these mobility models include node speeds $(v_{\min}, v_{\max})$, pause time $\left(t_{\text{pause}}\right)$, turn probability (p_{turn}), and position update interval $(\text{update}_{\text{interval}})$.

Mobility affects the network topology *G(t)*, link characteristics (bandwidth and delay as functions of time and distance), resource availability (the dynamic set of available edge servers and fog nodes), and task assignment $x_{ij}(t)$, which changes with node movement. Performance metrics, such as execution time, energy consumption, resource utilization, and quality of service, are all time-dependent and impacted by node mobility.

IV. Quantum-Inspired Genetic Algorithm

Genetic algorithms (GAs) are metaheuristic optimization techniques inspired by principles of natural selection [6]. In the context of MEC task scheduling, a population of potential solutions $\Pi = \{\pi_1, \pi_2, \dots, \pi_n\}$ is maintained, where each π_i represents a candidate schedule. The fitness of a schedule π_i is evaluated using a multi-objective function:

$$F(\pi_i) = w_1 f_1(\pi_i) + w_2 f_2(\pi_i) + \cdots + w_k f_k(\pi_i) \tag{7}$$

where $f_j(\pi_i)$ represents the *j*-th objective (such as execution time or energy consumption), and w_j is the corresponding weight. In a QIGA (see Algorithm 1), classical binary encoding is replaced with quantum-inspired qubit representation. A qubit $|\psi\rangle$ is defined as:

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \quad \alpha, \beta \in C, \quad \alpha^2 + \beta^2 = 1 \tag{8}$$

For a system with m tasks and n resources, each Q-individual Q is represented as:

$$Q = [q_{11}, q_{12}, \dots, q_{1n}, q_{21}, \dots, q_{mn}] \tag{9}$$

where q_{ij} is a qubit representing the allocation of task i to resource *j*.

A. Quantum Gates

Quantum gates manipulate qubits. The rotation gate $R(\theta)$ is defined as:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \tag{10}$$

The *NOT* (*X*) gate is: $R(\theta) = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$, and Quantum interference is modeled using the quantum phase θ:

$$|\psi'\rangle = \cos(\theta)\,|\psi\rangle + \sin(\theta)\,|\psi_{\text{best}}\rangle \tag{11}$$

B. QIGA Workflow

In line 1 and 2, we create an initial population $Q = \{Q_1, Q_2, \dots, Q_p\}$ of p Q-individuals. Each q_{ij} in Q_k is initialized as: $q_{ij} = \begin{bmatrix} \cos(\theta_{ij}) \\ \sin(\theta_{ij}) \end{bmatrix}$ where θ_{ij} is chosen randomly.

By observing in lines 4-7 each Q-individual to obtain a classical schedule π_i using $P(q_{ij} = 1) = \sin^2(\theta_{ij})$ (refer to Algorithm 2 for more details). Then we evaluate fitness using $F(\pi_i) = w_1\left(\frac{1}{ET(\pi_i)}\right) + w_2\left(\frac{1}{EC(\pi_i)}\right)$ where ET is execution time and EC is energy consumption. For selected pairs (Q^a, Q^b) in line 9, we apply $q_{ij}^{'a} = R(\theta_c)q_{ij}^a$, $q_{ij}^{'b} = R(-\theta_c)q_{ij}^b$ where θ_c is the crossover angle in line 10 (see Algorithm 3 for more details). In lines 11-12 we apply

Algorithm 1: Quantum Inspired Genetic Algorithm

```
    Input: Task set T, Resource set R, Population size p, Max ger
    Convergence threshold ϵ
    Output: Optimal schedule $\pi_{opt}$
1   Initialize $Q = Q_1, Q_2, \dots, Q_p$ where $Q_i = [q_{11}, q_{12}, \dots, q_{mn}]$
2   t ← 0
3   while $t < T_{max}$ and not converged do
4   $\prod \leftarrow Quantum\_Observation(Q)$
5   $F \leftarrow Evaluate\_Fitness(\prod)$
6   $Q_{best}, \pi_{best} \leftarrow$ Find_Best_Solution(Q, ∏, F)
7   $Q' \leftarrow \emptyset$
8   for $i \leftarrow 1$ to $p/2$ do
9     $Q_a, Q_b \leftarrow$ Quantum_Tournament_Selection(Q, F)
10    $Q_a', Q_b' \leftarrow$ Quantum_Crossover$(Q_a, Q_b)$
11    Quantum_Mutation$(Q_a')$
12    Quantum_Mutation$(Q_b')$
13    $Q' \leftarrow Q' \cup Q_a', Q_b'$
14  end
15  $Q' \leftarrow$ Quantum_Interference$(Q', Q_{best})$
16  $Q \leftarrow$ Quantum_Elitism_Selection$(Q \cup Q', p)$
17        if $||Q_{best} - \text{Previous\_Q}_{best}|| < \epsilon$
18          $converged \leftarrow true$
19        end
20        $t \leftarrow t + 1$
21  end
22  Return $\pi_{best}$
```

Algorithm 2: Quantum Observation Function

```
1   $\prod \leftarrow \emptyset$
2   for each $Q_i$ in $Q$ do
3     $\pi_i \leftarrow []$
4     for each $q_{ij}$ in $Q_i$ do
5       if $random() < sin^2(\theta_{ij})$ then
6         $\pi_i.append(1)$
7       else
8         $\pi_i.append(0)$
9   $\prod \leftarrow \prod \cup \{\pi_i\}$
10  Return ∏
```

Algorithm 3: Quantum Crossover Function

```
1   $Q_a', Q_b' \leftarrow Q_a, Q_b$
2   $\theta_c \leftarrow \pi/4$ // Crossover angle
3   for $i \leftarrow 1$ to length$(Q^a)$ do
4       $Q_a'[i] \leftarrow R(\theta_c)Q_a[i]$
5       $Q_b'[i] \leftarrow R(-\theta_c)Q_b[i]$
7   Return $Q'^a, Q'^b$
```

Algorithm 4: Quantum Mutation Function

```
1   for each $q_{ij}$ in $Q$ do
2     if random() < $P_m$ then
3       $q_{ij} \leftarrow Xq_{ij}$
4     end
5   end
```

Algorithm 5: Quantum Inference Function

```
1   $Q' \leftarrow \emptyset$
2   for each $Q_i$ in $Q$ do
3     $\theta_i \leftarrow \pi/2 \times \frac{F(Q_{best}) - F(Q_i)}{F(Q_{best})}$
4     $Q_i' \leftarrow cos(\theta_i)Q_i + sin(\theta_i)Q_{best}$
5     $Q' \leftarrow Q' \cup Q_i'$
6   Return $Q'$
```

quantum mutation (refer to algorithm 4 for more details) with probability P_m, the next step is to apply mutation (see Algorithm 4 for more details) using $q'_{\{ij\}} = X\, q_{\{ij\}}$. In the next step (line 15) we update Q-individuals by quantum interference (see algorithm 5 for more details) as: $Q'_i = \cos(\theta_i)\, Q_i + \sin(\theta_i)\, Q_{\text{best}}$ where θ_i is dynamically adjusted based on the fitness difference between Q_i and Q_{best}. Then in line 16-20, we retain the best elite individual (e) and select the remaining (p-e) individuals using quantum-inspired tournament selection. At the end we check convergence and terminate if $|Q_{\text{best}}(t) - Q_{\text{best}}(t-1)| <$ or if $t > T_{\max}$; otherwise, return to Step 2.

V. Performance Evaluation

To validate the effectiveness of the QIGA for task scheduling in the MEC, we conducted various simulations under different MEC scenarios. This section presents metrics, and results of our experimental analysis, comparing the QIGA approach with three other scheduling algorithms: MOHEFT [11] as multi objective heuristic algorithm, Quantum-Inspired Differential Evolution (QIDE) [4], and chaotic quantum particle swarm optimization (CQPSO) [12].

A. Simulation Setup

The simulations were conducted using a heterogeneous MEC environment comprising edge servers, fog nodes, and mobile devices through EdgeSimPy [19]. Each resource type had varying computational capacities, energy efficiency factors, and storage capacities to reflect real-world diversity [7][12][20]. We varied network parameters such as bandwidth and latency across the nodes, implementing RWM and MGM mobility models to account for realistic movement patterns of mobile and edge nodes. We used specific parameter settings for the QIGA in our experiments, as outlined in Table 2. The device configuration and network bandwidth settings is similar to [12].

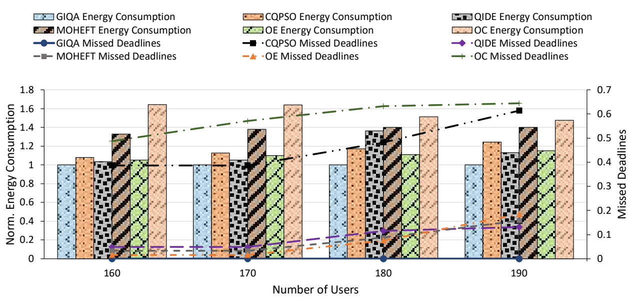
B. Energy Consumption

In Fig. 2, QIGA consistently outperforms competing methods in terms of energy efficiency and deadline adherence across varying user loads (see Fig. 2 (a)) and edge server counts (see Fig. 2 (b)). On average, QIGA shows 15.7% lower energy consumption compared to the next best method (CQPSO [12]) and up to 39.2% less than the most energy-intensive method ((Only Cloud) OC) across user counts. Similarly, when tested with increasing edge servers, QIGA achieves 20.7% lower energy consumption than CQPSO and up to 57% lower than OC, which peaks at 2.328 with 50 edge servers. This stability across both user and infrastructure changes underscores QIGA's energy efficiency.

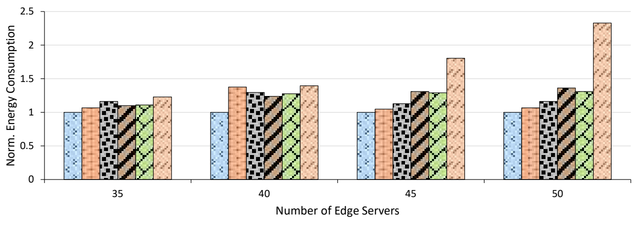
In terms of deadline adherence, QIGA achieves complete reliability with zero missed deadlines across all tested user counts, while competing methods experience notable deadline violations under higher loads. For example, CQPSO, QIDE, and MOHEFT have average missed deadline rates of 46.8%, 7.8%, and 7.7%, respectively, with CQPSO peaking at 61.4% missed deadlines. These findings confirm QIGA's robustness, scalability, and efficiency, making it a suitable solution for

Table 2. Parameters of the QIGA used for task scheduling in MEC

Parameter	Value/Range
Population Size (P)	100-200
Max Generations (T)	1000
Quantum Rotation Angle (θ)	0.05 - 0.1 radians
Mutation Probability (P_m)	0.01 - 0.05
Crossover Rate (Cr)	0.7
Elite Retention (e)	5
Convergence Threshold (ε)	1e-05
Initial Phase (θ_i)	Random [0, π/2]



(a)



(b)

Fig. 2: Comparative analysis of energy consumption and missed deadlines across SOTA scheduling methods under varying user and edge server counts

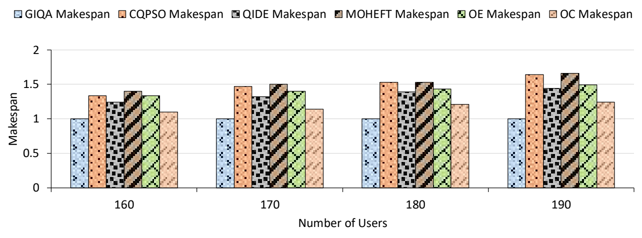
latency-sensitive applications that require low energy consumption and reliable performance, even under varying load and infrastructure conditions.

C. Makespan

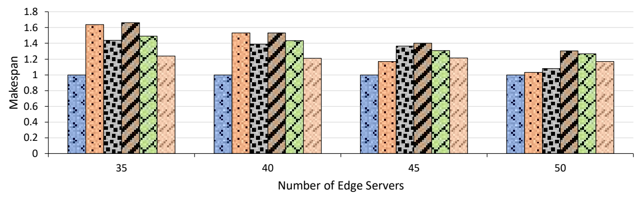
QIGA in Fig. 3 achieves shows its efficiency in the minimization of makespan and its robustness under different loads and infrastructure configurations. On average, QIGA's makespan is 20.5% smaller than that of the next best approach, OC (see Fig. 3 (a)), under user loads and 26.8% smaller under edge server configurations (see Fig. 3 (b)). It is at most 39.8% smaller than the largest makespan of the worst approach, MOHEFT. Compared with QIGA, the other approaches, such as CQPSO [12], QIDE [4], MOHEFT [11], and (Only Edge (OE)), exhibit larger makespan with a larger variance, especially under larger loads or numbers of servers, which may be less stable. OC, albeit much more efficient than the others, also trails QIGA in low makespan.

D. Resource Utilization and Latency

Regarding resource utilization (see Fig. 4 (a)), QIGA maintains near-complete utilization, averaging 98.5% across user counts, which is 27.5% higher than MOHEFT, the next best method. As shown in Fig.4, QIGA, consistently attains QoS of 1 across in all scenarios (see Fig. 4 (b)). This consistent attainment underscores QIGA's robustness and

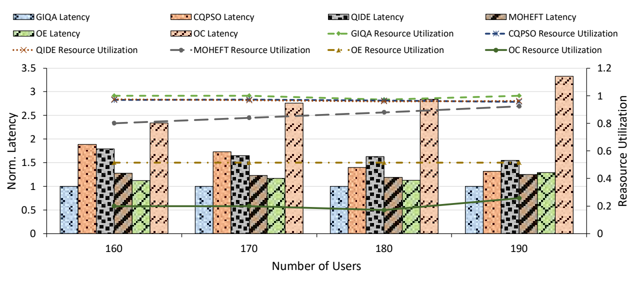


(a)

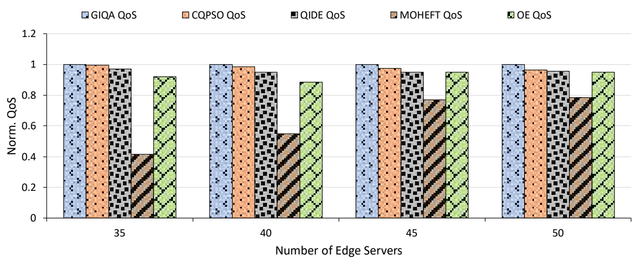


(b)

Fig. 3: Comparison of makespan across SOTA scheduling methods under varying user counts and edge server counts (Lower makespan is better)



(a)



(b)

Fig. 4: Comparison of resource utilization ((a) right), latency ((a) left), and QoS (b) across SOTA scheduling methods under varying user counts and edge server counts (Lower latency and higher QoS and utilization is better)

reliability in sustaining optimal QoS levels amid diverse infrastructure configurations. On average, QIGA's QoS is 3.5% higher than the next best method, CQPSO, and up to 58.5% higher than the least efficient method, MOHEFT, which peaks at only 78.5% of QIGA's QoS at 50 servers. MOHEFT starts with a QoS of just 41.5% of QIGA's at 35 servers, gradually improving but never matching QIGA. OC also lags behind, maintaining QoS values consistently below 70% across configurations. This consistent performance highlights QIGA's scalability and reliability in heterogeneous computing. In latency (see fig. 4 (a)), QIGA achieves a normalized value of 1 across all user counts (160-190), demonstrating its effectiveness in minimizing response times. On average, QIGA's latency is 25.2% lower than the next best method OE and up to 69.9% lower than the least efficient approach OC, which peaks at 3.325 with 190 users. This stability across user loads underscores QIGA's suitability for latency-sensitive applications.

VI. Conclusion

In this paper, we introduced a QIGA for task scheduling in MEC, addressing challenges of heterogeneity and dynamic conditions. By incorporating quantum-inspired techniques, QIGA enhances solution diversity, convergence speed, and adaptability, outperforming conventional methods in makespan, energy consumption, and resource utilization. Extensive simulations demonstrated QIGA's effectiveness in maximizing resource use and meeting task deadlines, highlighting its suitability for real-time MEC applications. Future work will focus on hybrid quantum-inspired methods and adaptive parameter tuning to further optimize scheduling in complex mobile edge computing environments.

References

[1] N. Abbas, Y. Zhang, A. Taherkordi and T. Skeie, "Mobile Edge Computing: A Survey," in *IEEE Internet of Things Journal*, vol. 5, no. 1, pp. 450-465, Feb. 2018.

[2] Y. Mao, C. You, J. Zhang, K. Huang and K. B. Letaief, "A Survey on Mobile Edge Computing: The Communication Perspective," in *IEEE Communications Surveys & Tutorials*, vol. 19, no. 4, pp. 2322-2358, Fourthquarter 2017.

[3] B. Wang, Z. Zhang, Y. Song, M. Chen, and Y. Chu, "Application of Quantum Particle Swarm Optimization for task scheduling in Device-Edge-Cloud Cooperative Computing," *Engineering Applications of Artificial Intelligence*, vol. 126, p. 107020, Nov. 2023.

[4] M. Bey, P. Kuila and B. B. Naik, "Quantum-Inspired Differential Evolution with Decoding using Hashing for Efficient User Allocation in Edge Computing Environment," in *IEEE Transactions on Emerging Topics in Computing*, 2024.

[5] M. Bey, P. Kuila, B. B. Naik, and S. Ghosh, "Quantum-inspired particle swarm optimization for efficient IoT service placement in edge computing systems," *Expert Systems with Applications*, vol. 236, p. 121270, Feb. 2024.

[6] D. Konar, K. Sharma, V. Sarogi, and S. Bhattacharyya, "A Multi-Objective Quantum-Inspired Genetic Algorithm (Mo-QIGA) for Real-Time Tasks Scheduling in Multiprocessor Environment," *Procedia Computer Science*, vol. 131, pp. 591–599, 2018.

[7] S. Dong, Y. Xia and J. Kamruzzaman, "Quantum Particle Swarm Optimization for Task Offloading in Mobile Edge Computing," in *IEEE Transactions on Industrial Informatics*, vol. 19, no. 8, pp. 9113-9122, Aug. 2023.

[8] J. Adu Ansere, D. T. Tran, O. A. Dobre, H. Shin, G. K. Karagiannidis and T. Q. Duong, "Energy-Efficient Optimization for Mobile Edge Computing With Quantum Machine Learning," in *IEEE Wireless Communications Letters*, vol. 13, no. 3, pp. 661-665, March 2024.

[9] D. Konar, S. Bhattacharyya, K. Sharma, S. Sharma, and S. R. Pradhan, "An improved Hybrid Quantum-Inspired Genetic Algorithm (HQIGA) for scheduling of real-time task in multiprocessor system," *Applied Soft Computing*, vol. 53, pp. 296–307, Apr. 2017.

[10] A. S. Thakur, T. Biswas, and P. Kuila, "Binary quantum-inspired gravitational search algorithm-based multi-criteria scheduling for multi-processor computing systems," *The Journal of Supercomputing*, vol. 77, no. 1, pp. 796–817, Apr. 2020.

[11] J. J. Durillo and R. Prodan, "Multi-objective workflow scheduling in Amazon EC2," *Cluster Computing*, vol. 17, no. 2, pp. 169–189, Nov. 2013.

[12] D. Zhang, G. Sun, J. Zhang, T. Zhang, and P. Yang, "Offloading approach for mobile edge computing based on chaotic quantum particle swarm optimization strategy," *Journal of Ambient Intelligence and Humanized Computing*, vol. 14, no. 10, pp. 14333–14347, Aug. 2023.

[13] J. A. Ansere *et al.*, "Optimal Computation Resource Allocation in Energy-Efficient Edge IoT Systems With Deep Reinforcement Learning," in *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 4, pp. 2130-2142, Dec. 2023.

[14] J. A. Ansere, E. Gyamfi, V. Sharma, H. Shin, O. A. Dobre and T. Q. Duong, "Quantum Deep Reinforcement Learning for Dynamic Resource Allocation in Mobile Edge Computing-Based IoT Systems," in *IEEE Transactions on Wireless Communications*, vol. 23, no. 6, pp. 6221-6233, June 2024.

[15] B. Bahrami, M. R. Khayyambashi and S. Mirjalili, "Multiobjective Placement of Edge Servers in MEC Environment Using a Hybrid Algorithm Based on NSGA-II and MOPSO," in *IEEE Internet of Things Journal*, vol. 11, no. 18, pp. 29819-29837, 15 Sept.15, 2024

[16] https://github.com/Anonymous0-0paper/QIGA

[17] G. Jayakumar and Gopinath Ganapathi, "Reference Point Group Mobility and Random Waypoint Models in Performance Evaluation of MANET Routing Protocols," vol. 2008, pp. 1–10, Jan. 2008.

[18] F. J. Martinez, J. . -C. Cano, C. T. Calafate and P. Manzoni, "CityMob: A Mobility Model Pattern Generator for VANETs," *ICC Workshops - 2008 IEEE International Conference on Communications Workshops*, Beijing, China, 2008, pp. 370-374.

[19] P. S. Souza, T. Ferreto, and R. N. Calheiros, "EdgeSimPy: Python-based modeling and simulation of edge computing resource management policies," *Future Generation Computer Systems*, vol. 148, pp. 446–459, Nov. 2023.

[20] A. Younesi, M. A. Fazli, and A. Ejlali, "A Novel Levy Walk-based Framework for Scheduling Power-intensive Mobile Edge Computing Tasks," *Journal of Grid Computing*, vol. 22, no. 4, pp. 1-22, 2024.